# A Reduction Theorem for Store Buffers 

Ernie Cohen ${ }^{1}$, Norbert Schirmer ${ }^{2, \star}$<br>${ }^{1}$ Microsoft Corp., Redmond, WA, USA<br>${ }^{2}$ German Research Center for Artificial Intelligence (DFKI) Saarbrücken, Germany<br>ecohen@amazon.com, norbert.schirmer@web.de


#### Abstract

When verifying a concurrent program, it is usual to assume that memory is sequentially consistent. However, most modern multiprocessors depend on store buffering for efficiency, and provide native sequential consistency only at a substantial performance penalty. To regain sequential consistency, a programmer has to follow an appropriate programming discipline. However, naïve disciplines, such as protecting all shared accesses with locks, are not flexible enough for building high-performance multiprocessor software. We present a new discipline for concurrent programming under TSO (total store order, with store buffer forwarding). It does not depend on concurrency primitives, such as locks. Instead, threads use ghost operations to acquire and release ownership of memory addresses. A thread can write to an address only if no other thread owns it, and can read from an address only if it owns it or it is shared and the thread has flushed its store buffer since it last wrote to an address it did not own. This discipline covers both coarse-grained concurrency (where data is protected by locks) as well as fine-grained concurrency (where atomic operations race to memory). We formalize this discipline in Isabelle/HOL, and prove that if every execution of a program in a system without store buffers follows the discipline, then every execution of the program with store buffers is sequentially consistent. Thus, we can show sequential consistency under TSO by ordinary assertional reasoning about the program, without having to consider store buffers at all.


[^0]
## Table of Contents

A Reduction Theorem for Store Buffers ..... 1
Ernie Cohen, Norbert Schirmer
1 Introduction ..... 2
2 Preliminaries ..... 5
3 Programming discipline ..... 6
4 Formalization ..... 8
4.1 Store buffer machine ..... 9
4.2 Virtual machine ..... 10
4.3 Reduction ..... 13
5 Building blocks of the proof ..... 13
5.1 Intermediate models ..... 15
5.2 Coupling relation ..... 18
5.3 Simulation ..... 20
6 PIMP ..... 26
7 Conclusion ..... 29
A Appendix ..... 31
A. 1 Memory Instructions ..... 31
A. 2 Abstract Program Semantics ..... 32
A. 3 Memory Transitions ..... 37
A. 4 Safe Configurations of Virtual Machines ..... 39
A. 5 Simulation of Store Buffer Machine with History by Virtual Machine with Delayed Releases ..... 428
A. 6 Simulation of Store Buffer Machine without History by Store Buffer Machine with History ..... 699
A. 7 Plug Together the Two Simulations ..... 722
A. 8 PIMP ..... 724

## 1 Introduction

When verifying a shared-memory concurrent program, it is usual to assume that each memory operation works directly on a shared memory state, a model sometimes called atomic memory. A memory implementation that provides this abstraction for programs that communicate only through shared memory is said to be sequentially consistent. Concurrent algorithms in the computing literature tacitly assume sequential consistency, as do most application programmers.

However, modern computing platforms typically do not guarantee sequential consistency for arbitrary programs, for two reasons. First, optimizing compilers are typically incorrect unless the program is appropriately annotated to indicate which program locations might be concurrently accessed by other threads; this issue is addressed only cursorily in this report. Second, modern processors buffer stores of retired instructions. To make such buffering transparent to single-processor programs, subsequent reads of the processor read from these buffers in preference to the cache. (Otherwise, a program could write a new value to an address but later read an older value.) However, in a multiprocessor system, processors do not snoop the store buffers of other processors, so a store is visible to the storing processor before it is visible to other processors. This can result in executions that are not sequentially consistent.

The simplest example illustrating such an inconsistency is the following program, consisting of two threads T 0 and T 1 , where x and y are shared memory variables (initially 0 ) and r 0 and r 1 are registers:

| T0 | T1 |
| :--- | :--- |
| $\mathrm{x}=1 ;$ | $\mathrm{y}=1 ;$ |
| $\mathrm{r} 0=\mathrm{y} ;$ | $\mathrm{r} 1=\mathrm{x} ;$ |

In a sequentially consistent execution, it is impossible for both $r 0$ and $r 1$ to be assigned 0 . This is because the assignments to x and y must be executed in some order; if x (resp. y) is assigned first, then r 1 (resp. r0) will be set to 1 . However, in the presence of store buffers, the assignments to r 0 and r 1 might be performed while the writes to x and y are still in their respective store buffers, resulting in both $r 0$ and $r 1$ being assigned 0 .

One way to cope with store buffers is make them an explicit part of the programming model. However, this is a substantial programming concession. First, because store buffers are FIFO, it ratchets up the complexity of program reasoning considerably; for example, the reachability problem for a finite set of concurrent finite-state programs over a finite set of finite-valued locations is in PSPACE without store buffers, but undecidable (even for two threads) with store buffers. Second, because writes from function calls might still be buffered when a function returns, making the store buffers explicit would break modular program reasoning.

In practice, the usual remedy for store buffering is adherence to a programming discipline that provides sequential consistency for a suitable class of architectures. In this report, we describe and prove the correctness of such a discipline suitable for the memory model provided by existing x86/x64 machines, where each write emerging from a store buffer hits a global cache visible to all processors. Because each processor sees the same global ordering of writes, this model is sometimes called total store order (TSO) [2] ${ }^{3}$

The concurrency discipline most familiar to concurrent programs is one where each variable is protected by a lock, and a thread must hold the corresponding lock to access the variable. (It is possible to generalize this to allow shared locks, as well as variants such as split semaphores.) Such lock-based techniques are typically referred to as coarse-grained concurrency control, and suffice for most concurrent application programming. However, these techniques do not suffice for low-level system programming (e.g., the construction of OS kernels), for several reasons. First, in kernel programming efficiency is paramount, and atomic memory operations are more efficient for many problems. Second, lock-free concurrency control can sometimes guarantee stronger correctness (e.g., wait-free algorithms can provide bounds on execution time). Third, kernel programming requires taking into account the implicit concurrency of concurrent hardware activities (e.g., a hardware TLB racing to use page tables while the kernel is trying to access them), and hardware cannot be forced to follow a locking discipline.

A more refined concurrency control discipline, one that is much closer to expert practice, is to classify memory addresses as lock-protected or shared. Lock-protected addresses are used in the usual way, but shared addresses can be accessed using atomic operations provided by hardware (e.g., on x 86 class architectures, most reads and writes are atomic ${ }^{4}$ ). The main restriction on these accesses is that if a processor does a shared write and a

[^1]subsequent shared read (possibly from a different address), the processor must flush the store buffer somewhere in between. For example, in the example above, both x and y would be shared addresses, so each processor would have to flush its store buffer between its first and second operations.

However, even this discipline is not very satisfactory. First, we would need even more rules to allow locks to be created or destroyed, or to change memory between shared and protected, and so on. Second, there are many interesting concurrency control primitives, and many algorithms, that allow a thread to obtain exclusive ownership of a memory address; why should we treat locking as special?

In this report, we consider a much more general and powerful discipline that also guarantees sequential consistency. The basic rule for shared addresses is similar to the discipline above, but there are no locking primitives. Instead, we treat ownership as fundamental. The difference is that ownership is manipulated by nonblocking ghost updates, rather than an operation like locking that have runtime overhead. Informally the rules of the discipline are as follows:

- In any state, each memory address is either shared or unshared. Each memory address is also either owned by a unique thread or unowned. Every unowned address must be shared. Each address is also either read-only or read-write. Every read-only address is unowned.
- A thread can (autonomously) acquire ownership of an unowned address, or release ownership of a address that it owns. It can also change whether an address it owns is shared or not. Upon release of an address it can mark it as read-only.
- Each memory access is marked as volatile or non-volatile.
- A thread can perform a write if it is sound. It can perform a read if it is sound and clean.
- A non-volatile write is sound if the thread owns the address and the address is unshared.
- A non-volatile read is sound if the thread owns the address or the address is read-only.
- A volatile write is sound if no other thread owns the address and the address is not marked as read-only.
- A volatile read is sound if the address is shared or the thread owns it.
- A volatile read is clean if the store buffer has been flushed since the last volatile write. Moreover, every non-volatile read is clean.
- For interlocked operations (like compare and swap), which have the side effect of the store buffer getting flushed, the rules for volatile accesses apply.

Note first that these conditions are not thread-local, because some actions are allowed only when an address is unowned, marked read-only, or not marked read-only. A thread can ascertain such conditions only through system-wide invariants, respected by all threads, along with data it reads. By imposing suitable global invariants, various thread-local disciplines (such as one where addresses are protected by locks, conditional critical reasons, or monitors) can be derived as lemmas by ordinary program reasoning, without need for meta-theory.

Second, note that these rules can be checked in the context of a concurrent program without store buffers, by introducing ghost state to keep track of ownership and sharing and whether the thread has performed a volatile write since the last flush. Our main result is that if a program obeys the rules above, then the program is sequentially consistent when executed on a TSO machine.

Consider our first example program. If we choose to leave both $x$ and $y$ unowned (and hence shared), then all accesses must be volatile. This would force each thread to flush the store buffer between their first and second operations. In practice, on an x86/x64 machine,
this would be done by making the writes interlocked, which flushes store buffers as a side effect. Whichever thread flushes its store buffer second is guaranteed to see the write of the other thread, making the execution violating sequential consistency impossible.

However, couldn't the first thread try to take ownership of x before writing it, so that its write could be non-volatile? The answer is that it could, but then the second thread would be unable to read $x$ volatile (or take ownership of $x$ and read it non-volatile), because we would be unable to prove that $x$ is unowned at that point. In other words, a thread can take ownership of an address only if it is not racing to do so.

Ultimately, the races allowed by the discipline involve volatile access to a shared address, which brings us back to locks. A spinlock is typically implemented with an interlocked read-modify-write on an address (the interlocking providing the required flushing of the store buffer). If the locking succeeds, we can prove (using for example a ghost variable giving the ID of the thread taking the lock) that no other thread holds the lock, and can therefore safely take ownership of an address "protected" by the lock (using the global invariant that only the lock owner can own the protected address). Thus, our discipline subsumes the better-known disciplines governing coarse-grained concurrency control.

To summarize, our motivations for using ownership as our core notion of a practical programming discipline are the following:

1. the distinction between global (volatile) and local (non-volatile) accesses is a practical requirement to reduce the performance penalty due to necessary flushes and to allow important compiler optimizations (such as moving a local write ahead of a global read),
2. coarse-grained concurrency control like locking is nothing special but only a derived concept which is used for ownership transfer (any other concurrency control that guarantees exclusive access is also fine), and
3. we want that the conditions to check for the programming discipline can be discharged by ordinary state-based program reasoning on a sequentially consistent memory model (without having to talk about histories or complete executions).

Overview In Section 2 we introduce preliminaries of Isabelle/HOL, the theorem prover in which we mechanized our work. In Section 3 we informally describe the programming discipline and basic ideas of the formalization, which is detailed in Section 4 where we introduce the formal models and the reduction theorem. In Section 5 we give some details of important building blocks for the proof of the reduction theorem. To illustrate the connection between a programming language semantics and our reduction theorem, we instantiate our framework with a simple semantics for a parallel WHILE language in Section 6. Finally we conclude in Section 7.

## 2 Preliminaries

The formalization presented in this papaer is mechanized and checked within the generic interactive theorem prover Isabelle [16]. Isabelle is called generic as it provides a framework to formalize various object logics declared via natural deduction style inference rules. The object logic that we employ for our formalization is the higher order logic of Isabelle/HOL [12].

This article is written using Isabelle's document generation facilities, which guarantees a close correspondence between the presentation and the actual theory files. We distinguish formal entities typographically from other text. We use a sans serif font for types and constants (including functions and predicates), e.g., map, a slanted serif font for free variables, e.g., $x$, and a slanted sans serif font for bound variables, e.g., $x$. Small capitals
are used for data type constructors, e.g., Foo, and type variables have a leading tick, e.g., 'a. HOL keywords are typeset in type-writer font, e.g., let.

To group common premises and to support modular reasoning Isabelle provides $l o-$ cales $[4,5]$. A locale provides a name for a context of fixed parameters and premises, together with an elaborate infrastructure to define new locales by inheriting and extending other locales, prove theorems within locales and interpret (instantiate) locales. In our formalization we employ this infrastructure to separate the memory system from the programming language semantics.

The logical and mathematical notions follow the standard notational conventions with a bias towards functional programming. We only present the more unconventional parts here. We prefer curried function application, e.g., $f$ a $b$ instead of $f(a, b)$. In this setting the latter becomes a function application to one argument, which happens to be a pair.

Isabelle/HOL provides a library of standard types like Booleans, natural numbers, integers, total functions, pairs, lists, and sets. Moreover, there are packages to define new data types and records. Isabelle allows polymorphic types, e.g., 'a list is the list type with type variable 'a. In HOL all functions are total, e.g., nat $\Rightarrow$ nat is a total function on natural numbers. A function update is $f(y:=v)=(\lambda x$. if $x=y$ then $v$ else $f x)$. To formalize partial functions the type 'a option is used. It is a data type with two constructors, one to inject values of the base type, e.g., $\lfloor x\rfloor$, and the additional element $\perp$. A base value can be projected with the function the, which is defined by the sole equation the $\lfloor x\rfloor=x$. Since HOL is a total logic the term the $\perp$ is still a well-defined yet un(der)specified value. Partial functions are usually represented by the type ' $a \Rightarrow$ 'b option, abbreviated as 'a $' b$. They are commonly used as maps. We denote the domain of map $m$ by dom $m$. A map update is written as $m(a \mapsto v)$. We can restrict the domain of a map $m$ to a set $A$ by $m \upharpoonright_{A}$.

The syntax and the operations for lists are similar to functional programming languages like ML or Haskell. The empty list is [], with $\mathrm{x} \# \mathrm{xs}$ the element x is 'consed' to the list $x s$.With xs @ ys list ys is appended to list xs. With the term map $f$ xs the function $f$ is applied to all elements in xs. The length of a list is $|x s|$, the $n$-th element of a list can be selected with $\mathrm{xs}_{[n]}$ and can be updated via $x s[n:=v]$. With dropWhile $P$ xs the prefix for which all elements satisfy predicate $P$ are dropped from list xs.

Sets come along with the standard operations like union, i.e., $A \cup B$, membership, i.e., $x \in A$ and set inversion, i.e., $-A$.

Tuples with more than two components are pairs nested to the right.

## 3 Programming discipline

For sequential code on a single processor the store buffer is invisible, since reads respect outstanding writes in the buffer. This argument can be extended to thread local memory in the context of a multiprocessor architecture. Memory typically becomes temporarily thread local by means of locking. The C-idiom to identify shared portions of the memory is the volatile tag on variables and type declarations. Thread local memory can be accessed non-volatilely, whereas accesses to shared memory are tagged as volatile. This prevents the compiler from applying certain optimizations to those accesses which could cause undesired behavior, e.g., to store intermediate values in registers instead of writing them to the memory.

The basic idea behind the programming discipline is, that before gathering new information about the shared state (via reading) the thread has to make its outstanding changes to the shared state visible to others (by flushing the store buffer). This allows to sequentialize the reads and writes to obtain a sequentially consistent execution of the global system. In this sequentialization a write to shared memory happens when the write
instruction exits the store buffer, and a read from the shared memory happens when all preceding writes have exited.

We distinguish thread local and shared memory by an ownership model. Ownership is maintained in ghost state and can be transferred as side effect of write operations and by a dedicated ghost operation. Every thread has a set of owned addresses. Owned addresses of different threads are disjoint. Moreover, there is a global set of shared addresses which can additionally be marked as read-only. Unowned addresses - addresses owned by no thread - can be accessed concurrently by all threads. They are a subset of the shared addresses. The read-only addresses are a subset of the unowned addresses (and thus of the shared addresses). We only allow a thread to write to owned addresses and unowned, read-write addresses. We only allow a thread to read from owned addresses and from shared addresses (even if they are owned by another thread).

All writes to shared memory have to be volatile. Reads from shared addresses also have to be volatile, except if the address is owned (i.e., single writer, multiple readers) or if the address is read-only. Moreover, non-volatile writes are restricted to owned, unshared memory. As long as a thread owns an address it is guaranteed that it is the only one writing to that address. Hence this thread can safely perform non-volatile reads to that address without missing any write. Similar it is safe for any thread to access read-only memory via non-volatile reads since there are no outstanding writes at all.

Recall that a volatile read is clean if it is guaranteed that there is no outstanding volatile write (to any address) in the store buffer. Moreover every non-volatile read is clean. To regain sequential consistency under the presence of store buffers every thread has to make sure that every read is clean, by flushing the store buffer when necessary. To check the flushing policy of a thread, we keep track of clean reads by means of ghost state. For every thread we maintain a dirty flag. It is reset as the store buffer gets flushed. Upon a volatile write the dirty flag is set. The dirty flag is considered to guarantee that a volatile read is clean.

Table 1a summarizes the access policy and Table 1 b the associated flushing policy of the programming discipline. The key motivation is to improve performance by minimizing the number of store buffer flushes, while staying sequentially consistent. The need for flushing the store buffer decreases from interlocked accesses (where flushing is a side-effect) over volatile accesses to non-volatile accesses. From the viewpoint of access rights there is no difference between interlocked and volatile accesses. However, keep in mind that some interlocked operations can read from, modify and write to an address in a single atomic step of the underlying hardware and are typically used in lock-free algorithms or for the implementation of locks.

Table 1: Programming discipline.
(a) Access policy

|  | shared <br> (read-write) | shared <br> (read-only) | unshared |
| :--- | :--- | :--- | :--- |
| un- <br> owned <br> owned <br> owned <br> by other | $\mathrm{vR}, \mathrm{vR}, \mathrm{vW}$ | $\mathrm{vR}, \mathrm{R}$ | unreachable |

(v)olatile, (R)ead, (W)rite
all reads have to be clean
(b) Flushing policy

|  | flush (before) |
| :--- | :---: |
| interlocked | as side effect |
| vR | if not clean |
| R, vW, W | never |

## 4 Formalization

In this section we go into the details of our formalization. In our model, we distinguish the plain 'memory system' from the 'programming language semantics' which we both describe as a small-step transition relation. During a computation the programming language issues memory instructions (read / write) to the memory system, which itself returns the results in temporary registers. This clean interface allows us to parameterize the program semantics over the memory system. Our main theorem allows us to simulate a computation step in the semantics based on a memory system with store buffers by $n$ steps in the semantics based on a sequentially consistent memory system. We refer to the former one as store buffer machine and to the latter one as virtual machine. The simulation theorem is independent of the programming language.

We continue with introducing the common parts of both machines. In Section 4.1 we describe the store buffer machine and in Section 4.2 we then describe the virtual machine. The main reduction theorem is presented in 4.3.

Addresses a, values $v$ and temporaries $t$ are natural numbers. Ghost annotations for manipulating the ownership information are the following sets of addresses: the acquired addresses $A$, the unshared (local) fraction $L$ of the acquired addresses, the released addresses $R$ and the writable fraction $W$ of the released addresses (the remaining addresses are considered read-only). These ownership annotations are considered as side-effects on volatile writes and interlocked operations (in case a write is performed). Moreover, a special ghost instruction allows to transfer ownership. The possible status changes of an address due to these ownership transfer operations are depicted in Figure 1. Note that ownership of an address is not directly transferred between threads, but is first released by one thread and then can be acquired by another thread. A memory instruction is a datatype with the


Fig. 1: Ownership transfer
following constructors:

- Read volatile a $t$ for reading from address a to temporary $t$, where the Boolean volatile determines whether the access is volatile or not.
- Write volatile a sop A L $R W$ to write the result of evaluating the store operation sop at address a. A store operation is a pair $(D, f)$, with the domain $D$ and the function $f$. The function $f$ takes temporaries $\vartheta$ as a parameter, which maps a temporary to a value. The subset of temporaries that is considered by function $f$ is specified by the domain $D$. We consider store operations as valid when they only depend on their domain:

$$
\text { valid-sop sop } \equiv \forall D f \theta \text {. sop }=(D, f) \wedge D \subseteq \operatorname{dom} \theta \longrightarrow f \theta=f\left(\left.\theta\right|_{D}\right)
$$

Again the Boolean volatile specifies the kind of memory access.

- RMW a $t$ sop cond ret $A L R W$, for atomic interlocked 'read-modify-write' instructions (flushing the store buffer). First the value at address a is loaded to temporary $t$, and then the condition cond on the temporaries is considered to decide whether a store operation is also executed. In case of a store the function ret, depending on both the old value at address a and the new value (according to store operation sop), specifies the final result stored in temporary $t$. With a trivial condition cond this instruction also covers interlocked reads and writes.
- Fence, a memory fence that flushes the store buffer.
- Ghost $A L R W$ for ownership transfer.


### 4.1 Store buffer machine

For the store buffer machine the configuration of a single thread is a tuple ( $p$, is, $\vartheta, s b$ ) consisting of the program state $p$, a memory instruction list is, the map of temporaries $\vartheta$ and the store buffer $s b$. A global configuration of the store buffer machine ( $t s, m$ ) consists of a list of thread configurations ts and the memory $m$, which is a function from addresses to values.

We describe the computation of the global system by the non-deterministic transition relation $(t s, m) \stackrel{\text { sb }}{\Rightarrow}\left(t s^{\prime}, m^{\prime}\right)$ defined in Figure 2 .

$$
\begin{gathered}
\frac{i<|t s| \quad t s_{[i]}=(p, i s, \vartheta, s b) \quad \vartheta \vdash p \rightarrow_{\mathrm{p}}\left(p^{\prime}, i s^{\prime}\right)}{(t s, m) \stackrel{\text { sb }}{\Rightarrow}\left(t s\left[i:=\left(p^{\prime}, \text { is @ is }, \vartheta, s b\right)\right], m\right)} \\
\frac{i<|t s| \quad t s_{[i]}=(p, \text { is, } \vartheta, s b) \quad(i s, \vartheta, s b, m) \stackrel{\text { sb }}{\mathrm{m}}\left(i s^{\prime}, \vartheta^{\prime}, s b^{\prime}, m^{\prime}\right)}{(t s, m) \stackrel{\text { sb }}{\Rightarrow}\left(t s\left[i:=\left(p, i s^{\prime}, \vartheta^{\prime}, s b^{\prime}\right)\right], m^{\prime}\right)} \\
\frac{i<|t s| \quad t s_{[i]}=(p, i s, \vartheta, s b) \quad(m, s b) \rightarrow_{\mathrm{sb}}\left(m^{\prime}, s b^{\prime}\right)}{(t s, m) \stackrel{\text { sb }}{\Rightarrow}\left(t s\left[i:=\left(p, i s, \vartheta, s b^{\prime}\right)\right], m^{\prime}\right)}
\end{gathered}
$$

Fig. 2: Global transitions of store buffer machine

A transition selects a thread $t s_{[i]}=(p, i s, \vartheta, s b)$ and either the 'program' the 'memory' or the 'store buffer' makes a step defined by sub-relations.

The program step relation is a parameter to the global transition relation. A program step $\vartheta \vdash p \rightarrow_{\mathrm{p}}\left(p^{\prime}\right.$, is $\left.{ }^{\prime}\right)$ takes the temporaries $\vartheta$ and the current program state $p$ and makes a step by returning a new program state $p^{\prime}$ and an instruction list is ${ }^{\prime}$ which is appended to the remaining instructions.

A memory step $(i s, \vartheta, s b, m) \xrightarrow{s b}_{m}\left(i^{\prime}, \vartheta^{\prime}, s b^{\prime}, m^{\prime}\right)$ of a machine with store buffer may only fill its store buffer with new writes.

In a store buffer step $(m, s b) \rightarrow_{s b}\left(m^{\prime}, s b^{\prime}\right)$ the store buffer may release outstanding writes to the memory.

The store buffer maintains the list of outstanding memory writes. Write instructions are appended to the end of the store buffer and emerge to memory from the front of the list. A read instructions is satisfied from the store buffer if possible. An entry in the store buffer is of the form $\mathrm{WritE}_{\text {sb }}$ volatile a sop $v$ for an outstanding write (keeping the volatile flag), where operation sop evaluated to value $v$.

As defined in Figure 3 a write updates the memory when it exits the store buffer.

$$
\overline{\left(m, \text { Write }_{\mathrm{sb}} \text { volatile a sop } v A L R W \# s b\right) \rightarrow_{\mathrm{sb}}(m(\mathrm{a}:=v), \mathrm{sb})}
$$

Fig. 3: Store buffer transition

$$
\begin{aligned}
& \frac{v=\left(\text { case buffered-val } s b \text { a of } \perp \Rightarrow m \text { a } \mid\left\lfloor v^{\prime}\right\rfloor \Rightarrow v^{\prime}\right)}{(\text { READ volatile a } t \# \text { is, } \vartheta, s b, m)_{\rightarrow_{m}(\text { is, } \vartheta(t \mapsto v), s b, m)}} \\
& \frac{s b^{\prime}=s b @\left[\mathrm{Write}_{\mathrm{sb}} \text { volatile a }(D, f)(f \vartheta) A L R W\right]}{(\text { Write volatile a }(D, f) A L R W \# \text { is, } \vartheta, \mathrm{sb}, \mathrm{~m}) \xrightarrow{\mathrm{sb}}_{\mathrm{m}}\left(\text { is, } \vartheta, \mathrm{sb} b^{\prime}, \mathrm{m}\right)} \\
& \frac{\neg \operatorname{cond}(\vartheta(t \mapsto m \mathrm{a})) \quad \vartheta^{\prime}=\vartheta(t \mapsto \mathrm{ma})}{(\text { RMW a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta,[], m) \xrightarrow{\mathrm{sb}}_{\mathrm{m}}\left(\text { is, } \vartheta^{\prime},[], \mathrm{m}\right)} \\
& \frac{\operatorname{cond}(\vartheta(t \mapsto m \mathrm{a})) \quad \vartheta^{\prime}=\vartheta(t \mapsto \operatorname{ret}(m \mathrm{a})(f(\vartheta(t \mapsto m \mathrm{a})))) \quad m^{\prime}=m(\mathrm{a}:=f(\vartheta(t \mapsto \mathrm{ma})))}{(\mathrm{RMW} \text { a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta,[], m) \xrightarrow{\text { sb }} \mathrm{m}\left(\mathrm{is}, \vartheta^{\prime},[], \mathrm{m}^{\prime}\right)} \\
& \overline{(\text { Fence } \# \text { is }, \vartheta,[], m) \xrightarrow{\mathrm{sb}}_{\mathrm{m}}(\text { is }, \vartheta,[], m)} \\
& \overline{(\operatorname{Ghost} A L R W \# \text { is, } \vartheta, s b, m) \xrightarrow{\text { sb }}_{\mathrm{m}}(i s, \vartheta, \mathrm{sb}, \mathrm{~m})}
\end{aligned}
$$

Fig. 4: Memory transitions of store buffer machine

The memory transition are defined in Figure 4 . With buffered-val $s b$ a we obtain the value of the last write to address a which is still pending in the store buffer. In case no outstanding write is in the store buffer we read from the memory. Store operations have no immediate effect on the memory but are queued in the store buffer instead. Interlocked operations and the fence operation require an empty store buffer, which means that it has to be flushed before the action can take place. The read-modify-write instruction first adds the current value at address a to temporary $t$ and then checks the store condition cond on the temporaries. If it fails this read is the final result of the operation. Otherwise the store is performed. The resulting value of the temporary $t$ is specified by the function ret which considers both the old and new value as input. The fence and the ghost instruction are just skipped.

### 4.2 Virtual machine

The virtual machine is a sequentially consistent machine without store buffers, maintaining additional ghost state to check for the programming discipline. A thread configuration is a tuple ( $p$, is $, \vartheta, \mathcal{D}, \mathcal{O}$ ), with a dirty flag $\mathcal{D}$ indicating whether there may be an outstanding volatile write in the store buffer and the set of owned addresses $\mathcal{O}$. The dirty flag $\mathcal{D}$ is considered to specify if a read is clean: for all volatile reads the dirty flag must not be set. The global configuration of the virtual machine $(t s, m, \mathcal{S})$ maintains a Boolean map of shared addresses $\mathcal{S}$ (indicating write permission). Addresses in the domain of mapping $\mathcal{S}$ are considered shared and the set of read-only addresses is obtained from $\mathcal{S}$ by: read-only $\mathcal{S} \equiv\{a . \mathcal{S} a=\lfloor$ False $\rfloor\}$

According to the rules in Fig 5 a global transition of the virtual machine (ts, m, $\mathcal{S}$ ) $\stackrel{\vee}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)$ is either a program or a memory step. The transition rules for its memory system are defined in Figure 6. In addition to the transition rules for the virtual machine we introduce the safety judgment $\mathcal{O} s, i \vdash($ is, $\vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$ in Figure 7, where $\mathcal{O} s$ is the list of ownership sets obtained from the thread list $t s$ and $i$ is the index of the current

$$
\begin{gathered}
\frac{i<|t s| \quad t s_{[i]}=(p, i s, \vartheta, \mathcal{D}, \mathcal{O}) \quad \vartheta \vdash p \rightarrow_{\mathrm{p}}\left(p^{\prime}, \text { is }\right)}{(t s, m, \mathcal{S}) \stackrel{\rightharpoonup}{\Rightarrow}\left(t s\left[i:=\left(p^{\prime}, \text { is @ is } s^{\prime}, \vartheta, \mathcal{D}, \mathcal{O}\right)\right], m, \mathcal{S}\right)} \\
\frac{i<|t s| \quad t s_{[i]}=(p, \text { is, } \vartheta, \mathcal{D}, \mathcal{O}) \quad(\text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \xrightarrow{\vee}_{\mathrm{m}}\left(i s^{\prime}, \vartheta^{\prime}, m^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{S}^{\prime}\right)}{(t s, m, \mathcal{S}) \stackrel{\vee}{\Rightarrow}\left(t s\left[i:=\left(p, i s^{\prime}, \vartheta^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}\right)\right], m^{\prime}, \mathcal{S}^{\prime}\right)}
\end{gathered}
$$

Fig. 5: Global transitions of virtual machine

$$
\begin{aligned}
& \overline{(R E A D} \text { volatile a } t \# \text { is, } \vartheta, x, m, g h s t) \stackrel{\rightharpoonup}{\rightarrow}_{\mathrm{m}}(\text { is, } \vartheta(t \mapsto m \text { a) }, \mathrm{x}, \mathrm{~m}, \mathrm{ghst}) \\
& \text { (Write False a }(D, f) A L R W \# \text { is, } \vartheta, x, m, g h s t) \stackrel{\rightharpoonup}{\rightarrow}_{m}(i s, \vartheta, x, m(a:=f \vartheta) \text {, ghst) } \\
& \frac{g h s t=(\mathcal{D}, \mathcal{O}, \mathcal{S}) \quad \text { ghst }^{\prime}=\left(\text { True, } \mathcal{O} \cup A-R, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)}{(\text { Write True a }(D, f) A L R W \# \text { is, } \vartheta, \mathrm{x}, \mathrm{~m}, \text { ghst }) \stackrel{\rightharpoonup}{\rightarrow}_{\mathrm{m}}(\text { is, } \vartheta, \mathrm{x}, \mathrm{~m}(\mathrm{a}:=\mathrm{f} \vartheta) \text {, ghst') }} \\
& \frac{\neg \operatorname{cond}(\vartheta(t \mapsto m \text { a })) \quad \text { ghst }=(\mathcal{D}, \mathcal{O}, \mathcal{S}) \quad \text { ghst }^{\prime}=(\text { False, } \mathcal{O}, \mathcal{S})}{(\text { RMW a } t(D, f) \text { cond ret A L R W \# is, } \vartheta, x, m, g h s t) \xrightarrow{\bullet}_{\mathrm{m}}\left(\text { is, } \vartheta(t \mapsto m a), x, m, g h s t^{\prime}\right)} \\
& \text { cond }(\vartheta(t \mapsto m a)) \quad \vartheta^{\prime}=\vartheta(t \mapsto r e t(m a)(f(\vartheta(t \mapsto m a))))
\end{aligned}
$$

$$
\begin{aligned}
& \frac{\text { ghst }=(\mathcal{D}, \mathcal{O}, \mathcal{S}) \quad \text { ghst }^{\prime}=(\text { False, } \mathcal{O}, \mathcal{S})}{(\text { Fence } \# \text { is, } \vartheta, x, m, g h s t) \xrightarrow{\stackrel{v}{m}}\left(\text { is, } \vartheta, x, m, g h s t^{\prime}\right)} \\
& \frac{g h s t=(\mathcal{D}, \mathcal{O}, \mathcal{S}) \quad \text { ghst }^{\prime}=\left(\mathcal{D}, \mathcal{O} \cup A-R, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)}{(\text { Ghost } A L R W \# \text { is, } \vartheta, \mathrm{x}, \mathrm{~m}, \text { ghst }) \stackrel{\rightharpoonup}{\rightarrow}_{\mathrm{m}}\left(\text { is, } \vartheta, \mathrm{x}, \mathrm{~m}, \mathrm{ghst} t^{\prime}\right)}
\end{aligned}
$$

Fig. 6: Memory transitions of the virtual machine
thread. Safety of all reachable states of the virtual machine ensures that the programming discipline is obeyed by the program and is our formal prerequisite for the simulation theorem. It is left as a proof obligation to be discharged by means of a proper program logic for sequentially consistent executions. In the following we elaborate on the rules of

$$
\begin{aligned}
& \frac{a \in \mathcal{O} \vee a \in \text { read-only } \mathcal{S} \vee \text { volatile } \wedge a \in \operatorname{dom} \mathcal{S} \quad \text { volatile } \longrightarrow \neg \mathcal{D}}{\mathcal{O} s, i \vdash(\operatorname{ReAD} \text { volatile a } t \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \frac{a \in \mathcal{O} \quad a \notin \operatorname{dom} \mathcal{S}}{\mathcal{O} s, i \vdash(\text { Write False } a(D, f) A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \forall j<|\mathcal{O s}| . i \neq j \longrightarrow a \notin \mathcal{O}_{[j]} \quad \text { a } \notin \text { read-only } \mathcal{S} \\
& \frac{\forall j<|\mathcal{O} s| . i \neq j \longrightarrow A \cap \mathcal{O}_{[j]}=\emptyset \quad A \subseteq \mathcal{O} \cup \operatorname{dom} \mathcal{S} \quad L \subseteq A \quad R \subseteq \mathcal{O} \quad A \cap R=\emptyset}{\mathcal{O} s, i \vdash(\text { Write True a }(D, f) A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \neg \operatorname{cond}(\vartheta(t \mapsto m a)) \quad a \in \operatorname{dom} \mathcal{S} \cup \mathcal{O} \\
& \overline{\mathcal{O} s, i \vdash(\mathrm{RMW}} \text { a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ } \\
& \text { cond }(\vartheta(t \mapsto m a)) \quad \forall j<\left|\mathcal{O s}_{s}\right| . i \neq j \longrightarrow a \notin \mathcal{O}_{[j]} \quad a \notin \text { read-only } \mathcal{S} \\
& \frac{\forall j<|\mathcal{O} s| . i \neq j \longrightarrow A \cap \mathcal{O}_{[j]}=\emptyset \quad A \subseteq \mathcal{O} \cup \operatorname{dom} \mathcal{S} \quad L \subseteq A \quad R \subseteq \mathcal{O} \quad A \cap R=\emptyset}{\mathcal{O} s, i \vdash(\text { RMW a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \overline{\mathcal{O} s, i \vdash(\text { Fence } \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \begin{array}{c}
A \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} \quad L \subseteq A \quad R \subseteq \mathcal{O} \quad A \cap R=\emptyset \quad \forall j<|\mathcal{O} s| . i \neq j \longrightarrow A \cap \mathcal{O}_{S_{[j]}}=\emptyset \\
\mathcal{O} s, i \vdash(\operatorname{Grost} A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }
\end{array}
\end{aligned}
$$

Fig. 7: Safe configurations of a virtual machine

Figures 6 and 7 in parallel. To read from an address it either has to be owned or read-only or it has to be volatile and shared. Moreover the read has to be clean. The memory content of address a is stored in temporary $t$. Non-volatile writes are only allowed to owned and unshared addresses. The result is written directly into the memory. A volatile write is only allowed when no other thread owns the address and the address is not marked as read-only. Simultaneously with the volatile write we can transfer ownership as specified by the annotations $A, L, R$ and $W$. The acquired addresses $A$ must not be owned by any other thread and stem from the shared addresses or are already owned. Reacquiring owned addresses can be used to change the shared-status via the set of local addresses $L$ which have to be a subset of $A$. The released addresses $R$ have to be owned and distinct from the acquired addresses $A$. After the write the new ownership set of the thread is obtained by adding the acquired addresses $A$ and releasing the addresses $R$ : $\mathcal{O} \cup A-R$. The released addresses $R$ are augmented to the shared addresses $S$ and the local addresses $L$ are removed. We also take care about the write permissions in the shared state: the released addresses in set $W$ as well as the acquired addresses are marked writable: $\mathcal{S} \oplus_{W}$ $R \ominus_{A} L$. The auxiliary ternary operators to augment and subtract addresses from the sharing map are defined as follows:

```
\(\mathcal{S} \oplus_{W} R \equiv \lambda a\). if \(a \in R\) then \(\lfloor a \in W\rfloor\) else \(\mathcal{S} a\)
\(\mathcal{S} \ominus_{A} L \equiv\)
\(\lambda a\). if \(a \in L\) then \(\perp\) else case \(\mathcal{S}\) a of \(\perp \Rightarrow \perp \mid\lfloor\) writeable \(\rfloor \Rightarrow\lfloor a \in A \vee\) writeable \(\rfloor\)
```

The read-modify-write instruction first adds the current value at address a to temporary $t$ and then checks the store condition cond on the temporaries. If it fails this read is
the final result of the operation. Otherwise the store is performed. The resulting value of the temporary $t$ is specified by the function ret which considers both the old and new value as input. As the read-modify-write instruction is an interlocked operation which flushes the store buffer as a side effect the dirty flag $\mathcal{D}$ is reset. The other effects on the ghost state and the safety sideconditions are the same as for the volatile read and volatile write, respectively.

The only effect of the fence instruction in the system without store buffer is to reset the dirty flag.

The ghost instruction Ghost $A L R W$ allows to transfer ownership when no write is involved i.e., when merely reading from memory. It has the same safety requirements as the corresponding parts in the write instructions.

### 4.3 Reduction

The reduction theorem we aim at reduces a computation of a machine with store buffers to a sequential consistent computation of the virtual machine. We formulate this as a simulation theorem which states that a computation of the store buffer machine ( $t \mathrm{~s}_{\mathrm{sb}}$, $m) \stackrel{\text { sb }}{\Rightarrow}\left(t_{\mathrm{s}_{\mathrm{sb}}}{ }^{\prime}, \mathrm{m}^{\prime}\right)$ can be simulated by a computation of the virtual machine $(t s, m, \mathcal{S})$ $\stackrel{v^{*}}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)$. The main theorem only considers computations that start in an initial configuration where all store buffers are empty and end in a configuration where all store buffers are empty again. A configuration of the store buffer machine is obtained from a virtual configuration by removing all ghost components and assuming empty store buffers. This coupling relation between the thread configurations is written as $t s_{\mathbf{s b}} \sim t s$. Moreover, the precondition initialv ${ }^{\mathrm{v}}$ 顷 $\mathcal{S}$ ensures that the ghost state of the initial configuration of the virtual machine is properly initialized: the ownership sets of the threads are distinct, an address marked as read-only (according to $\mathcal{S}$ ) is unowned and every unowned address is shared. Finally with safe-reach $(t s, m, S)$ we ensure conformance to the programming discipline by assuming that all reachable configuration in the virtual machine are safe (according to the rules in Figure 7).

## Theorem 1 (Reduction).

```
\(\left(t s_{\mathrm{sb}}, m\right) \stackrel{\text { sb }}{\Rightarrow}\left(t s_{\mathrm{sb}}{ }^{\prime}, m^{\prime}\right) \wedge\) empty-store-buffers \(t s_{\mathrm{sb}}{ }^{\prime} \wedge t s_{\mathrm{sb}} \sim t s \wedge\) initialv \(t s \mathcal{S} \wedge\)
    safe-reach \((t s, m, \mathcal{S}) \longrightarrow\)
        \(\exists t s^{\prime} \mathcal{S}^{\prime} .(t s, m, \mathcal{S}) \stackrel{v}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right) \wedge t s_{\mathrm{sb}}{ }^{\prime} \sim t s^{\prime}\)
```

This theorem captures our intiution that every result that can be obtained from a computation of the store buffer machine can also be obtained by a sequentially consistent computation. However, to prove it we need some generalizations that we sketch in the following sections. First of all the theorem is not inductive as we do not consider arbitrary intermediate configurations but only those where all store buffers are empty. For intermediate confiugrations the coupling relation becomes more involved. The major obstacle is that a volatile read (from memory) can overtake non-volatile writes that are still in the store-buffer and have not yet emerged to memory. Keep in mind that our programming discipline only ensures that no volatile writes can be in the store buffer the moment we do a volatile read, outstanding non-volatile writes are allowed. This reordering of operations is reflected in the coupling relation for intermediate configurations as discussed in the following section.

## 5 Building blocks of the proof

A corner stone of the proof is a proper coupling relation between an intermediate configuration of a machine with store buffers and the virtual machine without store buffers. It
allows us to simulate every computation step of the store buffer machine by a sequence of steps (potentially empty) on the virtual machine. This transformation is essentially a sequentialization of the trace of the store buffer machine. When a thread of the store buffer machine executes a non-volatile operation, it only accesses memory which is not modified by any other thread (it is either owned or read-only). Although a non-volatile store is buffered, we can immediately execute it on the virtual machine, as there is no competing store of another thread. However, with volatile writes we have to be careful, since concurrent threads may also compete with some volatile write to the same address. At the moment the volatile write enters the store buffer we do not yet know when it will be issued to memory and how it is ordered relatively to other outstanding writes of other threads. We therefore have to suspend the write on the virtual machine from the moment it enters the store buffer to the moment it is issued to memory. For volatile reads our programming discipline guarantees that there is no volatile write in the store buffer by flushing the store buffer if necessary. So there are at most some outstanding non-volatile writes in the store buffer, which are already executed on the virtual machine, as described before. One simple coupling relation one may think of is to suspend the whole store buffer as not yet executed intructions of the virtual machine. However, consider the following scenario. A thread is reading from a volatile address. It can still have non-volatile writes in its store buffer. Hence the read would be suspended in the virutal machine, and other writes to the address (e.g. interlocked or volatile writes of another thread) could invalidate the value. Altogether this suggests the following refined coupling relation: the state of the virtual machine is obtained from the state of the store buffer machine, by executing each store buffer until we reach the first volatile write. The remaining store buffer entries are suspended as instructions. As we only execute non volatile writes the order in which we execute the store buffers should be irrelevant. This coupling relation allows a volatile read to be simulated immediately on the virtual machine as it happens on the store buffer machine.

From the viewpoint of the memory the virtual machine is ahead of the store buffer machine, as leading non-volatile writes already took effect on the memory of the virtual machine while they are still pending in the store buffer. However, if there is a volatile write in the store buffer the corresponding thread in the virtual machine is suspended until the write leaves the store buffer. So from the viewpoint of the already executed instructions the store buffer machine is ahead of the virtual machine. To keep track of this delay we introduce a variant of the store buffer machine below, which maintains the history of executed instructions in the store buffer (including reads and program steps). Moreover, the intermediate machine also maintains the ghost state of the virtual machine to support the coupling relation. We also introduce a refined version of the virutal machine below, which we try to motivate now. Esentially the programming discipline only allows races between volatile (or interlocked) operations. By race we mean two competing memory accesses of different threads of which at least one is a write. For example the discipline guarantees that a volatile read may not be invalidated by a non-volatile write of another thread. While proving the simulation theorem this manifests in the argument that a read of the store-buffer machine and the virtual machine sees the same value in both machines: the value seen by a read in the store buffer machine stays valid as long as it has not yet made its way out in the virtual machine. To rule out certain races from the execution traces we make use of the programming discipline, which is formalized in the safety of all reachable configurations of the virtual machine. Some races can be ruled out by continuing the computation of the virtual machine until we reach a safety violation. However, some cannot be ruled out by the future computation of the current trace, but can be invalidated by a safety violation of another trace that deviated from the current one at some point
in the past. Consider two threads. Thread 1 attempts to do a volatile read from address a which is currently owned (and not shared) by thread 2, which attempts to do a nonvolatile write on a with value 42 and then release the address. In this configuration there is already a safety violation. Thread 1 is not allowed to perform a volatile read from an address that is not shared. However, when Thread 2 has executed his update and has released ownership (both are non-volatile operations) there is no safety violation anymore. Unfortunately this is the state of the virtual machine when we consider the instructions of Thread 2 to be in the store buffer. The store buffer machine and the virtual machine are out of sync. Whereas in the virtual machine Thread 1 will already read 42 (all non-volatile writes are already executed in the virtual machine), the non-volatile write may still be pending in the store buffer of Thread 2 and hence Thread 1 reads the old value in the store buffer machine. This kind of issues arise when a thread has released ownership in the middle of non-volatile operations of the virtual machine, but the next volatile write of this thread has not yet made its way out of the store buffer. When another thread races for the released address in this situation there is always another scheduling of the virtual machine where the release has not yet taken place and we get a safety violation. To make these safety violations visible until the next volatile write we introduce another ghost component that keeps track of the released addresses. It is augmented when an ghost operation releases an address and is reset as the next volatile write is reached. Moreover, we refine our rules for safety to take these released addresses into account. For example, a write to an released address of another thread is forbidden. We refer to these refined model as delayed releases (as no other thread can acquire the address as long as it is still in the set of released addresses) and to our original model as free flowing releases (as the effect of a release immediate takes place at the point of the ghost instruction). Note that this only affects ownership transfer due to the Ghost instruction. Ownership transfer together with volatile (or interlocked) writes happen simultaneously in both models.

Note that the refined rules for delayed releases are just an intermediate step in our proof. They do not have to be considered for the final programming discipline. As sketched above we can show in a separate theorem that a safety violation in a trace with respect to delayed releases implies a safety violation of a (potenitally other) trace with respect to free flowing releases. Both notions of safety collaps in all configurations where there are no released addresses, like the initial state. So if all reachable configurations are safe with respect to free flowing releases they are also safe with respect to delayed releases. This allows us to use the stricter policy of delayed releases for the simulation proof. Before continuing with the coupling relation, we introduce the refined intermediate models for delayed releases and store buffers with history information.

### 5.1 Intermediate models

We begin with the virtual machine with delayed releases, for which the memory transitions $($ is $, \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}){\xrightarrow{\mathrm{v}_{\mathrm{d}}}}_{\mathrm{m}}\left(\right.$ is $\left.^{\prime}, \vartheta^{\prime}, m^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}\right)$ are defined Figure 8. The additional ghost component $\mathcal{R}$ is a mapping from addresses to a Boolean flag. If an address is in the domain of $\mathcal{R}$ it was released. The boolean flag is considered to figure out if the released address was previously shared or not. In case the flag is True it was shared otherwise not. This subtle distinction is necessary to properly handle volatile reads. A volatile read from an address owned by another thread is fine as long as it is marked as shared. The released addresses $\mathcal{R}$ are reset at every volatile write as well as interlocked operations and the fence instruction. They are augmented at the ghost instruction taking the sharing information into account:
$\operatorname{aug}(\operatorname{dom} \mathcal{S}) R \mathcal{R}=$
(READ volatile a $t \#$ is, $\vartheta, m$, ghst) ${\stackrel{v_{d}}{m}}_{m}(i s, \vartheta(t \mapsto m$ a), $m$, ghst)

```
            \(\overline{\text { (Write False a }(D, f) A L R W \# \text { is, } \vartheta, m, g h s t) \xrightarrow{v_{\mathrm{c}}}{ }_{\mathrm{m}}(i s, \vartheta, m(a:=f \vartheta) \text {, ghst) }) ~}\)
            \(\frac{g h s t=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad \text { ghst }{ }^{\prime}=\left(\text { True, } \mathcal{O} \cup A-R, \lambda x . \perp, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)}{(\text { Write True a }(D, f) A L R W \# \text { is, } \vartheta, m, \text { ghst }){\xrightarrow[G]{\mathrm{v}_{\mathrm{f}}}}^{m}(\text { is, } \vartheta, m(\mathrm{a}:=f \vartheta), \text { ghst' })}\)
    \(\frac{\neg \operatorname{cond}(\vartheta(t \mapsto m \mathrm{a})) \quad \text { ghst }=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad \text { ghst }^{\prime}=(\text { False, } \mathcal{O}, \lambda x . \perp, \mathcal{S})}{(\mathrm{RMW} \text { a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta, m, \text { ghst }) \xrightarrow[\mathrm{c}_{\mathrm{m}}]{ }\left(\text { is, } \vartheta(t \mapsto m \text { a }), m, \text { ghst }^{\prime}\right)}\)
\(\operatorname{cond}(\vartheta(t \mapsto m a)) \quad \vartheta^{\prime}=\vartheta(t \mapsto r e t(m a)(f(\vartheta(t \mapsto m a)))) \quad m^{\prime}=m(a:=f(\vartheta(t \mapsto m a)))\)
            ghst \(=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad\) ghst \({ }^{\prime}=\left(\right.\) False, \(\left.\mathcal{O} \cup A-R, \lambda x . \perp, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)\)
            (RMW a \(t(D, f)\) cond ret \(A L R W \#\) is, \(\vartheta, m\), ghst) \({\xrightarrow{v_{d}}}_{m}\left(i s, \vartheta^{\prime}, m^{\prime}\right.\), ghst')
            \(\overline{(F e n c e ~ \#}\) is, \(\vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \xrightarrow[\mathrm{v}_{\mathrm{f}}]{\mathrm{v}_{\mathrm{m}}}\) (is, \(\vartheta, m\), False, \(\left.\mathcal{O}, \lambda x . \perp, \mathcal{S}\right)\)
\(\frac{\text { ghst }=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad \operatorname{ghst}^{\prime}=\left(\mathcal{D}, \mathcal{O} \cup A-R, \operatorname{aug}(\operatorname{dom} \mathcal{S}) R \mathcal{R}, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)}{(\text { Ghost } A L R W \# \text { is, } \vartheta, m, \text { ghst }){ }^{v_{\mathrm{J}}}{ }_{\mathrm{m}}\left(\text { is, } \vartheta, m, \text { ghst }{ }^{\prime}\right)}\)
```

Fig. 8: Memory transitions of the virtual machine with delayed releases
( $\lambda$ a. if $a \in R$ then case $\mathcal{R}$ a of $\perp \Rightarrow\lfloor a \in \operatorname{dom} \mathcal{S}\rfloor \mid\lfloor s\rfloor \Rightarrow\lfloor s \wedge a \in \operatorname{dom} \mathcal{S}\rfloor$ else $\mathcal{R}$ a)

If an address is freshly released $(a \in R$ and $\mathcal{R} a=\perp)$ the flag is set according to dom $\mathcal{S}$. Otherwise the flag becomes 【False」 in case the released address is currently unshared. Note that with this definition $\mathcal{R} \mathrm{a}=\lfloor$ False $\rfloor$ stays stable upon every new release and we do not loose information about a release of an unshared address.

The global transition $(t s, m, s) \stackrel{v_{s}}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, s^{\prime}\right)$ are analogous to the rules in Figure 5 replacing the memory transtions with the refined version for delayed releases.

The safety judgment for delayed releases $\mathcal{O}_{s}, \mathcal{R}_{s}, i \vdash($ is, $\vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$ is defined in Figure 9. Note the additional component $\mathcal{R} s$ which is the list of release maps of all threads. The rules are strict extensions of the rules in Figure 7: writing or acquiring an address $a$ is only allowed if the address is not in the release set of another thread (a $\notin$ dom $\left.\mathcal{R S}_{[j]}\right)$; reading from an address is only allowed if it is not released by another thread while it was unshared $\left(\mathcal{R}_{[j]}\right.$ a $\neq\lfloor$ False $\left.\rfloor\right)$.

For the store buffer machine with history information we not only put writes into the store buffer but also record reads, program steps and ghost operations. This allows us to restore the necessary computation history of the store buffer machine and relate it to the virtual machine which may fall behind the store buffer machine during execution. Altogether an entry in the store buffer is either a

- $\mathrm{READ}_{\text {sb }}$ volatile a $t v$, recording a corresponding read from address a which loaded the value $v$ to temporary $t$, or a
- Write ${ }_{\text {sb }}$ volatile a sop $v$ for an outstanding write, where operation sop evaluated to value $v$, or of the form
- $\mathrm{Prog}_{\text {sb }} p p^{\prime}$ is ${ }^{\prime}$, recording a program transition from $p$ to $p^{\prime}$ which issued instructions is ${ }^{\prime}$, or of the form
- Ghost $_{\text {sb }} A L R W$, recording a corresponding ghost operation.

As defined in Figure 10 a write updates the memory when it exits the store buffer, all other store buffer entries may only have an effect on the ghost state. The effect on the ownership

$$
\begin{aligned}
& a \in \mathcal{O} \vee a \in \text { read-only } \mathcal{S} \vee \text { volatile } \wedge a \in \operatorname{dom} \mathcal{S} \quad \forall j<|\mathcal{O} s| . i \neq j \longrightarrow \mathcal{R}_{[j]} \text { a } \neq\lfloor\text { False }\rfloor \\
& \neg \text { volatile } \longrightarrow\left(\forall j<\left|\mathcal{O}_{s}\right| . i \neq j \longrightarrow a \notin \operatorname{dom} \mathcal{R}_{[j]}\right) \quad \text { volatile } \longrightarrow \neg \mathcal{D}
\end{aligned}
$$

$$
\begin{aligned}
& \frac{a \in \mathcal{O} \quad a \notin \operatorname{dom} \mathcal{S} \quad \forall j<|\mathcal{O} s| . i \neq j \longrightarrow a \notin \operatorname{dom} \mathcal{R}_{s_{[j]}}}{\mathcal{O} s, \mathcal{R}_{s}, i \vdash(\text { Write False a }(D, f) A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \forall j<|\mathcal{O} s| . i \neq j \longrightarrow a \notin \mathcal{O} s_{[j]} \cup \operatorname{dom} \mathcal{R}_{[j]} \\
& \text { a } \notin \text { read-only } \mathcal{S} \quad \forall j<\left|\mathcal{O}_{s}\right| . i \neq j \longrightarrow A \cap\left(\mathcal{O}_{[j]} \cup \operatorname{dom} \mathcal{R}_{S_{[j]}}\right)=\emptyset \\
& A \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} \quad L \subseteq A \quad R \subseteq \mathcal{O} \quad A \cap R=\emptyset \\
& \mathcal{O}_{s}, \mathcal{R}_{s, i} \vdash(\text { Write True a }(D, f) A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ } \\
& \frac{\neg \operatorname{cond}(\vartheta(t \mapsto m \mathrm{a})) \quad \mathrm{a} \in \operatorname{dom} \mathcal{S} \cup \mathcal{O} \quad \forall j<|\mathcal{O} s| . i \neq j \longrightarrow \mathcal{R}_{[j]} \text { a } \neq\lfloor\text { False }\rfloor}{\mathcal{O} s, \mathcal{R} s, i \vdash(\text { RMW a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \operatorname{cond}(\vartheta(t \mapsto m a)) \quad a \in \operatorname{dom} \mathcal{S} \cup \mathcal{O} \quad \forall j<\left|\mathcal{O}_{s}\right| . i \neq j \longrightarrow a \notin \mathcal{O}_{s_{[j]}} \cup \operatorname{dom} \mathcal{R}_{[j]} \\
& \text { a } \notin \text { read-only } \mathcal{S} \quad \forall j<\left|\mathcal{O}_{s}\right| . i \neq j \longrightarrow A \cap\left(\mathcal{O}_{S_{[j]}} \cup \operatorname{dom} \mathcal{R}_{s_{[j]}}\right)=\emptyset \\
& A \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} \quad L \subseteq A \quad R \subseteq \mathcal{O} \quad A \cap R=\emptyset \\
& \mathcal{O} s, \mathcal{R}_{s, i} \vdash(\mathrm{RMW} \text { a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ } \\
& \overline{\mathcal{O}_{s}, \mathcal{R}_{s, i} \vdash(\mathrm{Fence} \# \text { is, } \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& A \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} \\
& \frac{L \subseteq A \subseteq \mathcal{O} \quad A \cap R=\emptyset \quad \forall j<\left|\mathcal{O}_{s}\right| . i \neq j \longrightarrow A \cap\left(\mathcal{O}_{[j]} \cup \operatorname{dom} \mathcal{R}_{[j]}\right)=\emptyset}{\mathcal{O} s, \mathcal{R}_{s, i} \vdash(\operatorname{Ghost} A L R W \# \text { is, } 1, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }} \\
& \mathcal{O}_{s}, \mathcal{R}_{s}, i \vdash([], \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }
\end{aligned}
$$

Fig. 9: Safe configurations of a virtual machine (delayed-releases)
$\left(m, W_{\text {RITE }}^{\text {sb }}\right.$ False a sop v $\left.A L R W \# s b, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{sbh}}(m(\mathrm{a}:=\mathrm{v}), \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
$\frac{\mathcal{O}^{\prime}=\mathcal{O} \cup A-R \quad \mathcal{S}^{\prime}=\mathcal{S} \oplus_{W} R \ominus_{A} L}{\left(m, W_{R I T E} \text { sb }\right.}$ True a $\operatorname{sop}$ v $\left.A L R W \# s b, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\text {sbh }}\left(m(\mathrm{a}:=\mathrm{v}), \mathrm{sb}, \mathcal{O}^{\prime}, \lambda x . \perp, \mathcal{S}^{\prime}\right)$
$\overline{\left(m, R E A D_{s b} \text { volatile a } t v \# s b, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{sbh}}(m, s b, \mathcal{O}, \mathcal{R}, \mathcal{S})}$
$\overline{\left(m, \text { Prog }_{\text {sb }} p p^{\prime} \text { is } \# s b, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\text {sbh }}(m, s b, \mathcal{O}, \mathcal{R}, \mathcal{S})}$
$\frac{\mathcal{O}^{\prime}=\mathcal{O} \cup A-R \quad \mathcal{R}^{\prime}=\operatorname{aug}(\operatorname{dom} \mathcal{S}) R \mathcal{R} \quad \mathcal{S}^{\prime}=\mathcal{S} \oplus_{W} R \ominus_{A} L}{\left(m, \operatorname{GHOST}_{\mathrm{sb}} A L R W \# \operatorname{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{sbh}}\left(m, \mathrm{sb}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)}$

Fig. 10: Store buffer transitions with history
information is analogous to the corresponding operations in the virtual machine. The memory transitions defined in Figure 11 are straightforward extensions of the store buffer transitions of Figure 11 augmented with ghost state and recording history information in the store buffer. Note how we deal with the ghost state. Only the dirty flag is updated when the instruction enters the store buffer, the ownership transfer takes effect when the instruction leaves the store buffer. The global transitions $\left(t s_{\text {sbh }}, m, \mathcal{S}\right) \stackrel{\text { sbh }}{\Rightarrow}\left(t s_{\text {sbh }}{ }^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)$

```
    \(\frac{v=\left(\text { case buffered-val } s b \text { a of } \perp \Rightarrow m \text { a } \mid\left\lfloor v^{\prime}\right\rfloor \Rightarrow v^{\prime}\right) \quad s b^{\prime}=s b @\left[R E A D_{s b} \text { volatile a } t v\right]}{(\text { READ volatile a } t \# \text { is, } \vartheta, s b, m, g h s t) \xrightarrow{\text { sbh }}\left(\text { is, } \vartheta(t \mapsto v), s b^{\prime}, m, g h s t\right)}\)
    \(\frac{s b^{\prime}=s b @\left[W_{R i t e}^{s b} \text { False a }(D, f)(f \vartheta) A L R W\right]}{(\text { Write False a }(D, f) A L R W \# \text { is, } \vartheta, s b, m, g h s t) \xrightarrow{\text { sbh }_{\mathrm{m}}}\left(\text { is, } \vartheta, \mathrm{sb}^{\prime}, \mathrm{m}, \mathrm{ghst}\right)}\)
            \(s b^{\prime}=s b @\left[W_{R I T E}^{s b}\right.\) True a \(\left.(D, f)(f \vartheta) A L R W\right]\)
            ghst \(=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad\) ghst \({ }^{\prime}=(\) True, \(\mathcal{O}, \mathcal{R}, \mathcal{S})\)
    \(\overline{(\text { Write True a }(D, f) A L R W \# \text { is, } \vartheta, s b, m, g h s t) \xrightarrow{\text { sbh }}\left(\mathrm{m}, \vartheta, s b^{\prime}, m, g h s t^{\prime}\right)}\)
\(\frac{\neg \operatorname{cond}(\vartheta(t \mapsto m \text { a })) \quad \text { ghst }=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad \text { ghst } t^{\prime}=(\text { False, } \mathcal{O}, \lambda x . \perp, \mathcal{S})}{(\mathrm{RMW} \text { a } t(D, f) \text { cond ret } A L R W \# \text { is, } \vartheta,[], m, \text { ghst }) \xrightarrow{\text { sbh }}_{\mathrm{m}}\left(\text { is, } \vartheta(t \mapsto m a),[], m, g h s t^{\prime}\right)}\)
\(\operatorname{cond}(\vartheta(t \mapsto m a)) \quad \vartheta^{\prime}=\vartheta(t \mapsto \operatorname{ret}(m a)(f(\vartheta(t \mapsto m a)))) \quad m^{\prime}=m(a:=f(\vartheta(t \mapsto m a)))\)
        ghst \(=(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad\) ghst \({ }^{\prime}=\left(\right.\) False, \(\left.\mathcal{O} \cup A-R, \lambda x . \perp, \mathcal{S} \oplus_{W} R \ominus_{A} L\right)\)
    (RMW a \(t(D, f)\) cond ret \(A L R W \#\) is, \(\vartheta,[], m, g h s t) \xrightarrow{\text { shb }_{m}}\left(i s, \vartheta^{\prime},[], m^{\prime}\right.\), ghst')
```



```
    (Ghost \(A L R W \#\) is, \(\vartheta, s b, m, G) \xrightarrow{\text { sbh }}_{\mathrm{m}}\left(i s, \vartheta, s b @\left[\operatorname{Ghost}_{\mathrm{sb}} A L R W\right], m, G\right)\)
```

Fig. 11: Memory transitions of store buffer machine with history
are analogous to the rules in Figure 2 replacing the memory transtions and store buffer transtiontions accordingly.

### 5.2 Coupling relation

After this introduction of the immediate models we can proceed to the details of the coupling relation, which relates configurations of the store buffer machine with histroy and the virtual machine with delayed releases. Remember the basic idea of the coupling relation: the state of the virtual machine is obtained from the state of the store buffer machine, by executing each store buffer until we reach the first volatile write. The remaining store buffer entries are suspended as instructions. The instructions now also include the history entries for reads, program steps and ghost operations. The suspended reads are not yet visible in the temporaries of the virtual machine. Similar the ownership effects (and program steps) of the suspended operations are not yet visible in the virtual machine. The coupling relation between the store buffer machine and the virtual machine is illustrated in Figure 12. The threads issue instructions to the store buffers from the right and the instructions emerge from the store buffers to main memory from the left. The dotted line illustrates the state of the virtual machines memory. It is obtained from the memory of the store buffer machine by executing the purely non-volatile prefixes of the store buffers. The remaining entries of the store buffer are still (suspended) instructions in the virtual machine.


Fig. 12: Illustration of coupling relation

Consider the following configuration of a thread $t s_{\mathrm{sbh}}[j]$ in the store buffer machine, where $i_{\mathrm{k}}$ are the instructions and $s_{\mathrm{k}}$ the store buffer entries. Let $s_{\mathrm{v}}$ be the first volatile write in the store buffer. Keep in mind that new store buffer entries are appended to the end of the list and entries exit the store buffer and are issued to memory from the front of the list.

$$
t s_{\mathrm{sbh}}[j]=\left(p,\left[i_{1}, \ldots, i_{\mathrm{n}}\right], \vartheta,\left[s_{1}, \ldots, s_{\mathrm{v}}, s_{\mathrm{v}+1}, \ldots, s_{\mathrm{m}}\right], \mathcal{D}, \mathcal{O}, \mathcal{R}\right)
$$

The corresponding configuration $t_{[j]}$ in the virtual machine is obtained by suspending all store buffer entries beginning at $s_{\mathrm{v}}$ to the front of the instructions. A store buffer READ $\mathrm{Rb}_{\mathrm{sb}}$ / $\mathrm{Write}_{\text {sb }}$ / Ghost $\mathrm{sb}_{\text {sb }}$ is converted to a Read / Write / Ghost instruction. We take the freedom to make this coercion implicit in the example. The store buffer entries preceding $s_{\mathrm{v}}$ have already made their way to memory, whereas the suspended read operations are not yet visible in the temporaries $\vartheta^{\prime}$. Similar, the suspended updates to the ownership sets and dirty flag are not yet recorded in $\mathcal{O}^{\prime}, \mathcal{R}^{\prime}$ and $\mathcal{D}^{\prime}$.

$$
t s_{[j]}=\left(p,\left[s_{\mathrm{v}}, s_{\mathrm{v}+1}, \ldots, s_{\mathrm{m}}, i_{1}, \ldots, i_{\mathrm{n}}\right], \vartheta^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)
$$

This example illustrates that the virtual machine falls behind the store buffer machine in our simulation, as store buffer instructions are suspended and reads (and ghost operations) are delayed and not yet visible in the temporaries (and the ghost state). This delay can also propagate to the level of the programming language, which communicates with the memory system by reading the temporaries and issuing new instructions. For example the control flow can depend on the temporaries, which store the result of branching conditions. It may happen that the store buffer machine already has evaluated the branching condition by referring to the values in the store buffer, whereas the virtual machine still has to wait. Formally this manifests in still undefined temporaries. Now consider that the program in the store buffer machine makes a step from $p$ to ( $p^{\prime}$, is'), which results in a thread configuration where the program state has switched to $p^{\prime}$, the instructions is' are appended and the program step is recorded in the store buffer:

$$
t s_{\mathrm{sbh}}{ }_{[j]}^{\prime}=\left(p^{\prime},\left[i_{1}, \ldots, i_{\mathrm{n}}\right] @ \text { is } s^{\prime}, \vartheta,\left[s_{1}, \ldots, s_{\mathrm{v}}, \ldots, s_{\mathrm{m}}, \mathrm{ProG}_{\mathrm{sb}} p p^{\prime} \text { is }\right], \mathcal{D}, \mathcal{O}, \mathcal{R}\right)
$$

The virtual machine however makes no step, since it still has to evaluate the suspended instructions before making the program step. The instructions is' are not yet issued and the program state is still $p$. We also take these program steps into account in our final coupling relation $\left(t t_{\text {sbh }}, m_{\text {sbh }}, \mathcal{S}_{\text {sbh }}\right) \sim(t s, m, \mathcal{S})$, defined in Figure 13 . We denote the already simulated store buffer entries by execs and the suspended ones by suspends. The function instrs converts them back to instructions, which are a prefix of the instructions of the virtual

```
                                    m = exec-all-until-volatile-write tssbh msbh
        S = share-all-until-volatile-write tssbh}\mp@subsup{\mathcal{S}}{\mathrm{ sbh }}{}\quad|t\mp@subsup{t}{\textrm{sbh}}{}|=|ts
\foralli<|t\mp@subsup{t}{\textrm{sbh}}{}|.
    let ( }\mp@subsup{p}{\textrm{sbh}}{},i\mp@subsup{s}{\textrm{sbh}}{},\mp@subsup{0}{\textrm{sbh}}{},sb,\mp@subsup{\mathcal{D}}{\mathrm{ sbh }}{},\mp@subsup{\mathcal{O}}{\mathrm{ sbh }}{},\mp@subsup{\mathcal{R}}{\mathrm{ sbh }}{})=t\mp@subsup{s}{\textrm{sbh}[i]}{}
        execs = takeWhile not-volatile-write sb;
        suspends = dropWhile not-volatile-write sb
    in \existsis \mathcal{D}. instrs suspends @ is sbh = is @ prog-instrs suspends ^
                \mathcal{D}
                ts[i]}
                (hd-prog p}\mp@subsup{p}{\mathrm{ sbh suspends, is, }\mp@subsup{0}{\mathrm{ sbh }}{}\mp@subsup{\Gamma}{(- read-tmps suspends)}{},\mathcal{D}\mathrm{ ,}}{\mathrm{ ,}
                acquire execs }\mp@subsup{\mathcal{O}}{\mathrm{ sbh }}{}\mathrm{ , release execs (dom }\mp@subsup{\mathcal{S}}{\mathrm{ sbh }}{})\mp@subsup{\mathcal{R}}{\mathrm{ sbh }}{}
                    (tssbbh
```

Fig. 13: Coupling relation
machine. We collect the additional instructions which were issued by program instructions but still recorded in the remainder of the store buffer with function prog-instrs. These instructions have already made their way to the instructions of the store buffer machine but not yet on the virtual machine. This situation is formalized as instrs suspends @ $i s_{\text {sbh }}=$ is @ prog-instrs suspends, where is are the instructions of the virtual machine. The program state of the virtual machine is either the same as in the store buffer machine or the first program state recorded in the suspended part of the store buffer. This state is selected by hd-prog. The temporaries of the virtual machine are obtained by removing the suspended reads from $\vartheta$. The memory is obtained by executing all store buffers until the first volatile write is hit, excluding it. Thereby only non-volatile writes are executed, which are all thread local, and hence could be executed in any order with the same result on the memory. Function exec-all-until-volatile-write executes them in order of appearance. Similarly the sharing map of the virtual machine is obtained by executing all store buffers until the first volatile write via the function share-all-until-volatile-write. For the local ownership set $\mathcal{O}_{\text {sbh }}$ the auxiliary function acquire calculates the outstanding effect of the already simulated parts of the store buffer. Analogously release calculates the effect for the released addresses $\mathcal{R}_{\text {sbh }}$.

### 5.3 Simulation

Theorem 2 is our core inductive simulation theorem. Provided that all reachable states of the virtual machine (with delayed releases) are safe, a step of the store buffer machine (with history) can be simulated by a (potentially empty) sequence of steps on the virtual machine, maintaining the coupling relation and an invariant on the configurations of the store buffer machine.

Theorem 2 (Simulation).

```
\(\left(t s_{\mathrm{sbh}}, m_{\mathrm{sbh}}, \mathcal{S}_{\mathrm{sbh}}\right) \stackrel{\mathrm{sbh}}{\Rightarrow}\left(t \mathrm{~s}_{\mathrm{sbh}}{ }^{\prime}, m_{\mathrm{sbh}}{ }^{\prime}, \mathcal{S}_{\mathrm{sbh}}{ }^{\prime}\right) \wedge\left(t s_{\mathrm{sbh}}, m_{\mathrm{sbh}}, \mathcal{S}_{\mathrm{sbh}}\right) \sim(t s, m, \mathcal{S}) \wedge\)
    safe-reach-delayed \((t s, m, \mathcal{S}) \wedge\) invariant \(t_{\text {sbh }} \mathcal{S}_{\text {sbh }} m_{\text {sbh }} \longrightarrow\)
    invariant \(t \mathrm{~s}_{\mathrm{sbh}}{ }^{\prime} \mathcal{S}_{\mathrm{sbh}}{ }^{\prime} \mathrm{m}_{\text {sbh }}{ }^{\prime} \wedge\)
    \(\left(\exists t s^{\prime} \mathcal{S}^{\prime} m^{\prime} .(t s, m, \mathcal{S}) \stackrel{\mathrm{v}_{\mathrm{d}}}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right) \wedge\left(t s_{\mathrm{sbh}}{ }^{\prime}, m_{\mathrm{sbh}}{ }^{\prime}, \mathcal{S}_{\mathrm{sbh}}{ }^{\prime}\right) \sim\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)\right)\)
```

In the following we discuss the invariant invariant $t s_{\mathrm{sbh}} S_{\mathrm{sbh}} m_{\mathrm{sbh}}$, where we commonly refer to a thread configuration $t s_{\mathbf{s b h}_{[i]}}=(p$, is, $\vartheta, s b, \mathcal{D}, \mathcal{O}, \mathcal{R})$ for $i<\left|t s_{\mathrm{sbh}}\right|$. By outstanding references we refer to read and write operations in the store buffer. The invariant is a conjunction of several sub-invariants grouped by their content:
invariant $t s_{\text {sbh }} S_{\text {sbh }} m_{\text {sbh }} \equiv$ ownership-inv $S_{\text {sbh }} t s_{\text {sbh }} \wedge$ sharing-inv $S_{\text {sbh }} t s_{\text {sbh }} \wedge$
temporaries-inv $t s_{\mathrm{sbh}} \wedge$ data-dependency-inv $t s_{\mathrm{sbh}} \wedge$ history-inv $t s_{\mathrm{sbh}} \mathrm{m}_{\mathrm{sbh}} \wedge$ flush-inv $t s_{\mathrm{sbh}} \wedge$ valid $t s_{\mathrm{sbh}}$

Ownership. (i) For every thread all outstanding non-volatile references have to be owned or refer to read-only memory. (ii) Every outstanding volatile write is not owned by any other thread. (iii) Outstanding accesses to read-only memory are not owned. (iv) The ownership sets of every two different threads are distinct.

Sharing. (i) All outstanding non volatile writes are unshared. (ii) All unowned addresses are shared. (iii) No thread owns read-only memory. (iv) The ownership annotations of outstanding ghost and write operations are consistent (e.g., released addresses are owned at the point of release). (v) There is no outstanding write to read-only memory.

Temporaries. Temporaries are modeled as an unlimited store for temporary registers. We require certain distinctness and freshness properties for each thread. (i) The temporaries referred to by read instructions are distinct. (ii) The temporaries referred to by reads in the store buffer are distinct. (iii) Read and write temporaries are distinct. (iv) Read temporaries are fresh, i.e., are not in the domain of $\vartheta$.

Data dependency. Data dependency means that store operations may only depend on previous read operations. For every thread we have: (i) Every operation $(D, f)$ in a write instruction or a store buffer write is valid according to valid-sop $(D, f)$, i.e., function $f$ only depends on domain $D$. (ii) For every suffix of the instructions of the form Write volatile a $(D, f) A L R W \#$ is the domain $D$ is distinct from the temporaries referred to by future read instructions in is. (iii) The outstanding writes in the store buffer do not depend on the read temporaries still in the instruction list.

History. The history information of program steps and read operations we record in the store buffer have to be consistent with the trace. For every thread: (i) The value stored for a non volatile read is the same as the last write to the same address in the store buffer or the value in memory, in case there is no write in the buffer. (ii) All reads have to be clean. This results from our flushing policy. Note that the value recorded for a volatile read in the initial part of the store buffer (before the first volatile write), may become stale with respect to the memory. Remember that those parts of the store buffer are already executed in the virtual machine and thus cause no trouble. (iii) For every read the recorded value coincides with the corresponding value in the temporaries. (iv) For every Write ${ }_{\text {sb }}$ volatile a $(D, f) \vee A L R W$ the recorded value $v$ coincides with $f \vartheta$, and domain $D$ is subset of dom $\vartheta$ and is distinct from the following read temporaries. Note that the consistency of the ownership annotations is already covered by the aforementioned invariants. (v) For every suffix in the store buffer of the form $\mathrm{PROG}_{\mathbf{s b}} p_{1} p_{2}$ is ${ }^{\prime} \# s b^{\prime}$, either $p_{1}=p$ in case there is no preceding program node in the buffer or it corresponds to the last program state recorded there. Moreover, the program transition $\vartheta \upharpoonright_{\left(- \text {read-tmps } s b^{\prime}\right)} \vdash p_{1} \rightarrow_{\mathrm{p}}\left(p_{2}\right.$, is $)$ is possible, i.e., it was possible to execute the program transition at that point. (vi) The program configuration $p$ coincides with the last program configuration recorded in the store buffer. (vii) As the instructions from a program step are at the one hand appended to the instruction list and on the other hand recorded in the store buffer, we have for every suffix $s b^{\prime}$ of the store buffer: $\exists i s^{\prime}$. instrs $s b^{\prime} @$ is $=i s^{\prime} @$ prog-instrs $s b^{\prime}$, i.e., the remaining instructions is correspond to a suffix of the recorded instructions prog-instrs $s b^{\prime}$.

Flushes. If the dirty flag is unset there are no outstanding volatile writes in the store buffer.

Program step. The program-transitions are still a parameter of our model. In order to make the proof work, we have to assume some of the invariants also for the program steps. We allow the program-transitions to employ further invariants on the configurations, these are modeled by the parameter valid. For example, in the instantiation later on the program keeps a counter for the temporaries, for each thread. We maintain distinctness of temporaries by restricting all temporaries occurring in the memory system to be below that counter, which is expressed by instantiating valid. Program steps, memory steps and store buffer steps have to maintain valid. Furthermore we assume the following properties of a program step: (i) The program step generates fresh, distinct read temporaries, that are neither in $\vartheta$ nor in the store buffer temporaries of the memory system. (ii) The generated memory instructions respect data dependencies, and are valid according to valid-sop.

Proof sketch. We do not go into details but rather first sketch the main arguments for simulation of a step in the store buffer machine by a potentially empty sequence of steps in the virtual machine, maintaining the coupling relation. Second we exemplarically focus on some cases to illustrate common arguments in the proof. The first case distinction in the proof is on the global transitions in Figure 2. (i) Program step: we make a case distinction whether there is an outstanding volatile write in the store buffer or not. If not the configuration of the virtual machine corresponds to the executed store buffer and we can make the same step. Otherwise the virtual machine makes no step as we have to wait until all volatile writes have exited the store buffer. (ii) Memory step: we do case distinction on the rules in Figure 11. For read, non volatile write and ghost instructions we do the same case distinction as for the program step. If there is no outstanding volatile write in the store buffer we can make the step, otherwise we have to wait. When a volatile write enters the store buffer it is suspended until it exists the store buffer. Hence we do no step in the virtual machine. The read-modify-write and the fence instruction can all be simulated immediately since the store buffer has to be empty. (iii) Store Buffer step: we do case distinction on the rules in Figure 10. When a read, a non volatile write, a ghost operation or a program history node exits the store buffer, the virtual machine does not have to do any step since these steps are already visible. When a volatile write exits the store buffer, we execute all the suspended operations (including reads, ghost operations and program steps) until the next suspended volatile write is hit. This is possible since all writes are non volatile and thus memory modifications are thread local.

In the following we exemplarically describe some cases in more detail to give an impression on the typical arguments in the proof. We start with a configuration $c_{\mathrm{sbh}}=\left(t s_{\mathrm{sbh}}\right.$, $m_{\text {sbh }}, \mathcal{S}_{\text {sbh }}$ ) of the store buffer machine, where the next instruction to be executed is a read of thread $i$ : READ sb $_{\text {sb }}$ volatile a $t$. The configuration of the virtual machine is $c f g=(t s, m$, $\mathcal{S})$. We have to simulate this step on the virtual machine and can make use of the coupling relations $\left(t_{s_{s b h}}, m_{\text {sbh }}, \mathcal{S}_{\text {sbh }}\right) \sim(t s, m, \mathcal{S})$, the invariants invariant $t s_{\text {sbh }} \mathcal{S}_{\text {sbh }} m_{\text {sbh }}$ and the safety of all reachable states of the virtual machine: safe-reach-delayed ( $t \mathrm{~s}, \mathrm{~m}, \mathcal{S}$ ). The state of the store buffer machine and the coupling with the volatile machine is depicted in Figure 14. Note that if there are some suspended instructions in thread i, we cannot directly exploit the 'safety of the read', as the virtual machine has not yet reached the state where thread $i$ is poised to do the read. But fortunately we have safety of the virtual machien of all reachable states. Hence we can just execute all suspended instructions of thread $i$ until we reach the read. We refer to this configuration of the virtual machine as $c f g^{\prime \prime}=\left(t s^{\prime \prime}, m^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)$, which is depicted in Figure 15.

For now we want to consider the case where the read goes to memory and is not forwarded from the store buffer. The value read is $v=m_{\text {sbh }}$ a. Moreover, we make a case distinction wheter there is an outstanding volatile write in the store buffer of thread $i$ or


Fig. 14: Thread i poised to read


Fig. 15: Forwarded computation of virtual machine
not. This determines if there are suspended instructions in the virtual machine or not. We start with the case where there is no such write. This means that there are no suspended instructions in thread $i$ and therefore $c f g^{\prime \prime}=c f g$. We have to show that the virtual machine reads the same value from memory: $v=m$ a. So what can go wrong? When can the the memory of the virtual machine hold a different value? The memory of the virtual machine is obtained from the memory of the store buffer machine by executing all store buffers until we hit the first volatile write. So if there is a discrepancy in the value this has to come from a non-volatile write in the executed parts of another thread, let us say thread $j$. This write is marked as x in Figure 16.


Fig. 16: Conflicting write in thread $j$ (marked $x$ )

We refer to x both for the write operation itself and to characterize the point in time in the computation of the virtual machine where the write was executed. At the point x the write was safe according to rules in Figure 9 for non-volatile writes. So it was owned by thread $j$ and unshared. This knowledge about the safety of write x is preserved in the invariants, namely (Ownership.i) and (Sharing.i). Additionally from invariant (Sharing.v) we know that address a was not read-only at point x . Now we combine this information with the safety of the read of thread $i$ in the current configuration cfg: address a either has to be owned by thread $i$, or has to be read-only or the read is volatile and a is shared. Additionally there are the constraints on the released addresses which we will exploit below. Let us address all cases step by step. First, we consider that address a is currently owned by thread $i$. As it was owned by thread $j$ at time x there has to be an release of a in the executed prefix of the store buffer of thread $j$. This release is recorded in the release set, so we know $a \in \operatorname{dom} \mathcal{R}_{[j]}$. This contradicts the safety of the read. Second, we consider that address a is currently read-only. At time x address a was owned by thread $j$, unshared and not read-only. Hence there was a release of address a in the executed prefix of the store buffer of $j$, where it made a transition unshared and owned to shared. With the monotonicity of the release sets this means a $\in \operatorname{dom} \mathcal{R} s_{[j]}$, even more precisely $\mathcal{R} s_{[j]} a=\lfloor$ False $\rfloor$. Hence there is no chance to get the read safe (neiter a volatile nor a non-volatile). Third, consider a volatile read and that address a is currently shared. This is ruled out by the same line of reasoning as in the previous case. So ultimately we have ruled out all races that could destroy the value at address a and have shown that we can simulate the step on the virtual machine. This completes the simulation of the case where there is no store buffer forwarding and no volatile write in the store buffer of thread i. The other cases are handled similar. The main arguments are obtained by arguing about safety of configuration $c f g^{\prime \prime}$ and exploiting the invariants to rule out conflicting operations
in other store buffers. When there is a volatile write in he store buffer of thread $i$ there are still pending suspended instructions in the virtual machine. Hence the virtual machine makes no step and we have to argue that the simulation relation as well as all invariants still hold.

Up to now we have focused on how to simulate the read and in particular on how to argue that the value read in the store buffer machine is the same as the value read in the virtual machine. Besided these simulation properties another major part of the proof is to show that all invariants are maintained. For example if the non-volatile read enters the store buffer we have to argue that this new entry is either owned or refers to an read-only address (Ownership.i). As for the simulation above this follows from safety of the virtual machine in configuration cfg". However, consider an ghost operation that acquires an address a. From safety of the configuration $c f g^{\prime \prime}$ we can only infer that there is no conflicting acquire in the non-volaitle prefixes of the other store buffers. In case an conflicting acquire is in the suspended part of a store buffer of thread $j$ safety of configuration $c f^{\prime \prime}$ is not enough. But as we have safety of all reachable states we can forward the computation of thread $j$ until the conflicting acquire is about to be executed and construct an unsafe state which rules out the conflict.

Last we want to comment on the case where the store buffer takes a step. The major case destinction is wheter a volatile write leaves the store buffer or not. In the former case the virtual machine has to simulate a whole bunch of instructions at once to simulate the store buffer machine up to the next volatile write in the store buffer. In the latter case the virtual machine does no step at all, since the instruction leaving the store buffer is already simulated. In both cases one key argument is commutativity of non-volatile operations with respect to global effects on the memory or the sharing map. Consider a non-volatile store buffer step of thread $i$. In the configuration of the virtual machine before the store buffer step of thread $i$, the simulation relation applies the update to the memory and the sharing map of the store buffer machine, within the operations exec-all-until-volatile-write and share-all-until-volatile-write 'somewhere in the middle' to obtain the memory and the sharing map of the virtual machine. After the store buffer step however, when the nonvolatile operations has left the store buffer, the effect is applied to the memory and the sharing map right in the beginning. The invariants and safety sideconditions for nonvolatile operations guarantee 'locality' of the operation which manifests in commutativity properties. For example, a non-volatile write is thread local. There is no conflicting write in any other store buffer and hence the write can be safely moved to the beginning.

This conludes the discussion on the proof of Theorem 2.
The simulation theorem for a single step is inductive and can therefor be extended to arbitrary long computations. Moreover, the coupling relation as well as the invariants become trivial for a initial configuration where all store buffers are empty and the ghost state is setup appropriately. To arrive at our final Theorem 1 we need the following steps:

1. simulate the computation of the store buffer machine $\left(t s_{\mathrm{sb}}, m\right) \stackrel{\text { sb }}{\Rightarrow}{ }^{*}\left(t s_{\mathrm{sb}}, m^{\prime}\right)$ by a computation of a store buffer machine with history $\left(t s_{\mathrm{sbh}}, m, \mathcal{S}\right) \stackrel{\text { sbh }}{\Rightarrow}\left(t s_{\mathrm{sbh}}{ }^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)$,
2. simulate the computation of the store buffer machine with history by a computation of the virtual machine with delayed releases $(t s, m, \mathcal{S}) \stackrel{v^{*}}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)$ by Theorem 2 (extended to the reflexive transitive closure),
3. simulate the computation of the virtual machine with delayed releases by a computation of the virtual machine with free flowing releases $(t s, m, \mathcal{S}) \stackrel{{ }^{*}}{\Rightarrow}\left(t s^{\prime}, m^{\prime}, \mathcal{S}^{\prime}\right)^{5}$.
[^2]Step 1 is trivial since the bookkeeping within the additional ghost and history state does not affect the control flow of the transition systems and can be easily removed. Similar the additional $\mathcal{R}$ ghost component can be ignored in Step 3. However, to apply Theorem 2 in Step 2 we have to convert from safe-reach $(t s, m, \mathcal{S})$ provided by the preconditions of Theorem 1 to the required safe-reach-delayed ( $t s, m, \mathcal{S}$ ). This argument is more involved and we only give a short sketch here. The other direction is trivial as every single case for delayed releases (cf. Figure 9) immediately implies the corresponding case for free flowing releases (cf. Figure 7).

First keep in mind that the predicates ensure that all reachable configurations starting from $(t s, m, \mathcal{S})$ are safe, according to the rules for free flowing releases or delayed releases respectively. We show the theorem by contraposition and start with a computation which reaches a configuration $c$ that is unsafe according to the rules for delayed releases and want to show that there has to be a (potentially other) computation (starting from the same initial state) that leads to an unsafe configuration $c^{\prime}$ accroding to free flowing releases. If $c$ is already unsafe according to free flowing releases we have $c^{\prime}=c$ and are finished. Otherwise we have to find another unsafe configuration. Via induction on the length of the global computation we can also assume that for all shorter computations both safety notions coincide. A configuration can only be unsafe with respect to delayed releases and safe with respect to free flowing releases if there is a race between two distinct Threads $i$ and $j$ on an address a that is in the release set $\mathcal{R}$ of one of the threads, lets say Thread i. For example Thread $j$ attempts to write to an address a which is in the release set of Thread $i$. If the release map would be empty there cannot be such an race (it would simulataneously be unsafe with respect to free flowing releases). Now we aim to find a configuration $c^{\prime}$ that is also reachable from the initial configuration and is unsafe with respect to free flowing releases. Intuitively this is a configuration where Thread $i$ is rewinded to the state just before the release of address a and Thread $j$ is in the same state as in configuration $c$. Before the release of a the address has to be owned by Thread $i$, which is unsafe according to free flowing releases as well as delayed releases. So we can argue that either Thread $j$ can reach the same state although Thread $i$ is rewinded or we even hit an unsafe configuration before. What kind of steps can Thread i perform between between the free flowing release point (point of the ghost instruction) and the delayed release point (point of next volatile write, interlocked operation or fence at which the release map is emptied)? How can these actions affect Thread $j$ ? Note that the delayed release point is not yet reached as this would empty the release map (which we know not to be empty). Thus Thread i does only perform reads, ghost instructions, program steps or non-volatile writes. All of these instructions of Thread $i$ either have no influence on the computation of Thread $j$ at all (e.g. a read, program step, non-volatile write or irrelevant ghost operation) or may cause a safety violation already in a shorter computation (e.g. acquiring an address that another thread holds). This is fine for our inductive argument. So either we can replay every step of Thread $j$ and reach the final configuration $c^{\prime}$ which is now also unsafe according to free flowing releases, or we hit a configuration $c^{\prime \prime}$ in a shorter computation which violates the rules of delayed as well as free flowing releases (using the induction hypothesis).

## 6 PIMP

PIMP is a parallel version of IMP [11], a canonical WHILE-language.
An expression $e$ is either (i) Const v, a constant value, (ii) Mem volatile a, a (volatile) memory lookup at address a, (iii) TmP sop, reading from the temporaries with a operation sop which is an intermediate expression occurring in the transition rules for statements,
(iv) Unop $f e$, a unary operation where $f$ is a unary function on values, and finally (v) Binop $f e_{1} e_{2}$, a binary operation where $f$ is a binary function on values.

A statement $s$ is either (i) Skip, the empty statement, (ii) Assign volatile a e A $L$ $R W$, a (volatile) assignment of expression $e$ to address expression a, (iii) CAS a $c_{\mathrm{e}} s_{\mathrm{e}}$ A $L R W$, atomic compare and swap at address expression a with compare expression $c_{\mathrm{e}}$ and swap expression $s_{\mathrm{e}}$, (iv) SEQ $s_{1} s_{2}$, sequential composition, (v) Cond e $s_{1} s_{2}$, the if-then-else statement, (vi) While e s, the loop statement with condition e, (vii) SGhost, and SFence as stubs for the corresponding memory instructions.

The key idea of the semantics is the following: expressions are evaluated by issuing instructions to the memory system, then the program waits until the memory system has made all necessary results available in the temporaries, which allows the program to make another step. Figure 17 defines expression evaluation. The function used-tmps e calculates

```
issue-expr t (Const v) = []
issue-expr t(MEM volatile a) = [READ volatile a t]
issue-expr t (Tmp (D,f)) = []
issue-expr t(Unor f e) = issue-expr t e
issue-expr t (BinOP f e e e e ) = issue-expr t e e @ issue-expr (t+used-tmps e e ) e
eval-expr t (Const v) = (\emptyset, \lambda0.v)
eval-expr t(MEM volatile a) = ({t}, \lambda0. the (0t))
eval-expr t(TMp (D,f)) = (D,f)
eval-expr t(Unop f e) = let (D, fe})=\mathrm{ eval-expr te in (D, \Q.f (fe }0)
eval-expr t(BINOP f e e e e ) = let (D, (D, f
    (D},\mp@subsup{D}{2}{})=\mathrm{ eval-expr (t+ used-tmps e}\mp@subsup{e}{1}{})\mp@subsup{e}{2}{
    in ( }\mp@subsup{D}{1}{}\cup\mp@subsup{D}{2}{},\lambda0.f(\mp@subsup{f}{1}{}0)(\mp@subsup{f}{2}{}0)
```

Fig. 17: Expression evaluation
the number of temporaries that are necessary to evaluate expression $e$, where every Mem expression accounts to one temporary. With issue-expr $t$ e we obtain the instruction list for expression $e$ starting at temporary $t$, whereas eval-expr $t$ e constructs the operation as a pair of the domain and a function on the temporaries.

The program transitions are defined in Figure 18. We instantiate the program state by a tuple ( $s, t$ ) containing the statement $s$ and the temporary counter $t$. To assign an expression $e$ to an address(-expression) a we first create the memory instructions for evaluation the address a and transforming the expression to an operation on temporaries. The temporary counter is incremented accordingly. When the value is available in the temporaries we continue by creating the memory instructions for evaluation of expression $e$ followed by the corresponding store operation. Note that the ownership annotations can depend on the temporaries and thus can take the calculated address into account.

Execution of compare and swap CAS involves evaluation of three expressions, the address a the compare value $c_{\mathrm{e}}$ and the swap value $s_{\mathrm{e}}$. It is finally mapped to the read-modify-write instruction RMW of the memory system. Recall that execution of RMW first stores the memory content at address a to the specified temporary. The condition compares this value with the result of evaluating $c_{e}$ and writes swap value $s_{a}$ if successful. In either case the temporary finally returns the old value read.

Sequential composition is straightforward. An if-then-else is computed by first issuing the memory instructions for evaluation of condition $e$ and transforming the condition to an operation on temporaries. When the result is available the transition to the first or second statement is made, depending on the result of isTrue. Execution of the loop is defined

$$
\begin{aligned}
& \frac{\left.\forall \text { sop. } \mathrm{a} \neq \text { Tmp sop } \quad \mathrm{a}^{\prime}=\text { Tmp (eval-expr } t \mathrm{a}\right) \quad t^{\prime}=t+\text { used-tmps a } \quad \text { is }=\text { issue-expr } t \mathrm{a}}{\vartheta \vdash(\text { Assign volatile a e } A L R W, t) \rightarrow_{\mathrm{p}}\left(\left(\text { AsSIGn volatile } \mathrm{a}^{\prime} \text { e } A L R W, t^{\prime}\right), \text { is }\right)} \\
& \left.D \subseteq \operatorname{dom} \vartheta \quad \text { is }=\text { issue-expr } t \text { e @ [WRITE volatile }\left(\begin{array}{ll}
\text { a } \vartheta
\end{array}\right)(\text { eval-expr } t e)(A \vartheta)\binom{L}{\hline}(R \vartheta)\left(\begin{array}{ll}
W & \vartheta
\end{array}\right)\right] \\
& \vartheta \vdash(\operatorname{Assign} \text { volatile }(\operatorname{Tmp}(D, \mathrm{a})) \text { e } A L R W, t) \rightarrow_{\mathrm{p}}((\operatorname{SKIP}, t+\text { used-tmps } e) \text {, is }) \\
& \frac{\forall \text { sop. } \mathrm{a} \neq \text { TMP sop } \quad \mathrm{a}^{\prime}=\mathrm{Tmp}(\text { eval-expr } t \mathrm{a}) \quad t^{\prime}=t+\text { used-tmps a } \quad \text { is }=\text { issue-expr } t \mathrm{a}}{\vartheta \vdash\left(\operatorname{CAS} \text { a } c_{\mathrm{e}} S_{\mathrm{e}} A L R W, t\right) \rightarrow_{\mathrm{p}}\left(\left(\mathrm{CAS} \mathrm{a}^{\prime} c_{\mathrm{e}} S_{\mathrm{e}} A L R W, t^{\prime}\right), \text { is }\right)} \\
& \frac{\left.\forall \text { sop. } c_{\mathrm{e}} \neq \text { TMP sop } \quad c_{\mathrm{e}}{ }^{\prime}=\text { TMP (eval-expr } t c_{\mathrm{e}}\right) \quad t^{\prime}=t+\text { used-tmps } c_{\mathrm{e}} \quad \text { is }=\text { issue-expr } t c_{\mathrm{e}}}{\vartheta \vdash\left(\operatorname { C A S } ( \text { TMP a) } c _ { \mathrm { e } } S _ { \mathrm { e } } A L R W , t ) \rightarrow _ { \mathrm { p } } \left(\left(\text { CAS }\left(\text { TMP a) } c_{\mathrm{e}}{ }^{\prime} s_{\mathrm{e}} A L R W, t^{\prime}\right), \text { is }\right)\right.\right.} \\
& D_{\mathrm{a}} \subseteq \operatorname{dom} \vartheta \\
& D_{c} \subseteq \operatorname{dom} \vartheta \quad \text { eval-expr } t s_{\mathrm{e}}=(D, f) \quad t^{\prime}=t+\text { used-tmps } s_{\mathrm{e}} \quad \text { cond }=\left(\lambda \theta \text {. the }\left(\theta t^{\prime}\right)=c \theta\right) \\
& \text { ret }=\left(\lambda v_{1} v_{2} . v_{1}\right) \quad \text { is }=\text { issue-expr } t s_{\mathrm{e}} @\left[R M W(\operatorname{a} \vartheta) t^{\prime}(D, f) \operatorname{cond} \operatorname{ret}(A \vartheta)(L \vartheta)(R \vartheta)\left(\begin{array}{ll}
W & (L)
\end{array}\right]\right. \\
& \vartheta \vdash\left(\operatorname{CAS}\left(\operatorname{Tmp}\left(D_{\mathrm{a}}, \mathrm{a}\right)\right)\left(\operatorname{Tmp}\left(D_{\mathrm{c}}, c\right)\right) s_{\mathrm{e}} A L R W, t\right) \rightarrow_{\mathrm{p}}\left(\left(\operatorname{SkIP}, \text { Suc } t^{\prime}\right) \text {, is }\right) \\
& \frac{\vartheta \vdash\left(s_{1}, t\right) \rightarrow_{\mathrm{p}}\left(\left(s_{1}{ }^{\prime}, t^{\prime}\right), \text { is }\right)}{\vartheta \vdash\left(\operatorname{SEQ~} s_{1} s_{2}, t\right) \rightarrow_{\mathrm{p}}\left(\left(\operatorname{SEQ} s_{1}{ }^{\prime} s_{2}, t^{\prime}\right), \text { is }\right)} \\
& \overline{\vartheta \vdash\left(\operatorname{SEQ} \operatorname{SKIP} s_{2}, t\right) \rightarrow_{\mathrm{p}}\left(\left(s_{2}, t\right),[]\right)} \\
& \begin{array}{ll}
\forall \text { sop. } e \neq \text { TMP sop } & \left.e^{\prime}=\text { TMP (eval-expr } t e\right) \quad t^{\prime}=t+\text { used-tmps } e \quad \text { is }=\text { issue-expr } t e \\
\vartheta \vdash\left(\operatorname{CoND} \text { e } s_{1} S_{2}, t\right) \rightarrow_{\mathrm{p}}\left(\left(\operatorname{CoND}^{\prime}{ }_{\left.\left.S_{1} S_{2}, t^{\prime}\right), \text { is }\right)}\right.\right.
\end{array} \\
& D \subseteq \operatorname{dom} \vartheta \quad \text { isTrue }(e \vartheta) \\
& \overline{\vartheta \vdash\left(\operatorname{Cond}(\operatorname{TMP}(D, e)) s_{1} s_{2}, t\right) \rightarrow_{\mathrm{p}}\left(\left(s_{1}, t\right),[]\right)} \\
& \frac{D \subseteq \operatorname{dom} \vartheta}{\vartheta \vdash\left(\operatorname{Cond}(\operatorname{TMP}(D, e)) s_{1} s_{2}, t\right) \rightarrow_{\mathrm{p}}\left(\left(s_{2}, t\right),[]\right)} \\
& \overline{\vartheta \vdash(\text { While } e s, t) \rightarrow_{\mathrm{p}}((\operatorname{Cond} e(\operatorname{SEQ} s(\text { While } e s)) \text { Skip, } t),[])} \\
& \overline{\vartheta \vdash(\mathrm{SGhost} A L R W, t) \rightarrow_{\mathrm{p}}((\operatorname{Skip}, t),[\operatorname{Ghost}(A \vartheta)(L \vartheta)(R \vartheta)(W \vartheta)])} \\
& \vartheta \vdash(\mathrm{SFENCE}, t) \rightarrow_{\mathrm{p}}((\mathrm{SKIP}, t),[\mathrm{FENCE}])
\end{aligned}
$$

Fig. 18: Program transitions
by stepwise unfolding. Ghost and fence statements are just propagated to the memory system.

To instantiate Theorem 2 with PIMP we define the invariant parameter valid, which has to be maintained by all transitions of PIMP, the memory system and the store buffer. Let $\vartheta$ be the valuation of temporaries in the current configuration, for every thread configuration $t s_{\mathrm{sb}_{[i]}}=((s, t)$, is, $\vartheta, s b, \mathcal{D}, \mathcal{O})$ where $i<\left|t s_{\mathrm{sb}}\right|$ we require: (i) The domain of all intermediate TMP $(D, f)$ expressions in statement $s$ is below counter $t$. (ii) All temporaries in the memory system including the store buffer are below counter $t$. (iii) All temporaries less than counter $t$ are either already defined in the temporaries $\vartheta$ or are outstanding read temporaries in the memory system.

For the PIMP transitions we prove these invariants by rule induction on the semantics. For the memory system (including the store buffer steps) the invariants are straightforward. The memory system does not alter the program state and does not create new temporaries, only the PIMP transitions create new ones in strictly ascending order.

## 7 Conclusion

We have presented a practical and flexible programming discipline for concurrent programs that ensures sequential consistency on TSO machines, such as present x64 architectures. Our approach covers a wide variety of concurrency control, covering locking, data races, single writer multiple readers, read only and thread local portions of memory. We minimize the need for store buffer flushes to optimize the usage of the hardware. Our theorem is not coupled to a specific logical framework like separation logic but is based on more fundamental arguments, namely the adherence to the programming discipline which can be discharged within any program logic using the standard sequential consistent memory model, without any of the complications of TSO.

Related work. Disclaimer. This contribution presents the state of our work from 2010 [8]. Finally, 8 years later, we made the AFP submission for Isabelle2018. This related work paragraph does not thoroughly cover publications that came up in the meantime.

A categorization of various weak memory models is presented in [2]. It is compatible with the recent revisions of the Intel manuals [10] and the revised x86 model presented in [15]. The state of the art in formal verification of concurrent programs is still based on a sequentially consistent memory model. To justify this on a weak memory model often a quite drastic approach is chosen, allowing only coarse-grained concurrency usually implemented by locking. Thereby data races are ruled out completely and there are results that data race free programs can be considered as sequentially consistent for example for the Java memory model $[3,18]$ or the x86 memory model [15]. Ridge [17] considers weak memory and data-races and verifies Peterson's mutual exclusion algorithm. He ensures sequentially consistency by flushing after every write to shared memory. Burckhardt and Musuvathi [6] describe an execution monitor that efficiently checks whether a sequentially consistent TSO execution has a single-step extension that is not sequentially consistent. Like our approach, it avoids having to consider the store buffers as an explicit part of the state. However, their condition requires maintaining in ghost state enough history information to determine causality between events, which means maintaining a vector clock (which is itself unbounded) for each memory address. Moreover, causality (being essentially graph reachability) is already not first-order, and hence unsuitable for many types of program verification. Closely related to our work is the draft of Owens [14] which also investigates on the conditions for sequential consistent reasoning within TSO. The notion of a triangular-race free trace is established to exactly characterize the traces on
a TSO machine that are still sequentially consistent. A triangular race occurs between a read and a write of two different threads to the same address, when the reader still has some outstanding writes in the store buffer. To avoid the triangular race the reader has to flush the store buffer before reading. This is essentially the same condition that our framework enforces, if we limit every address to be unowned and every access to be volatile. We regard this limitation as too strong for practical programs, where non-volatile accesses (without any flushes) to temporarily local portions of memory (e.g. lock protected data) is common practice. This is our core motivation for introducing the ownership based programming discipline. We are aware of two extensions of our work that were published in the meantime. Chen et al. [7] also take effects of the MMU into account and generalize our reduction theorem to handle programs that edit page tables. Oberhauser [13] improves on the flushing policy to also take non-triangular races into account and facilitates an alternative proof approach.

Limitations. There is a class of important programs that are not sequentially consistent but nevertheless correct.

First consider a simple spinlock implementation with a volatile lock 1 , where $1=0$ indicates that the lock is not taken. The following code acquires the lock:

```
while(!interlocked_test_and_set(l));
<critical section accessing protected objects>,
```

and with the assignment $1=0$ we can release the lock again. Within our framework address 1 can be considered unowned (and hence shared) and every access to it is volatile. We do not have to transfer ownership of the lock 1 itself but of the objects it protects. As acquiring the lock is an expensive interlocked oprations anyway there are no additional restrictions from our framework. The interesting point is the release of the lock via the volatile write $1=0$. This leaves the dirty bit set, and hence our programming discipline requires a flushing instruction before the next volatile read. If 1 is the only volatile variable this is fine, since the next operation will be a lock acquire again which is interlocked and thus flushes the store buffer. So there is no need for an additonal fence. But in general this is not the case and we would have to insert a fence after the lock release to make the dirty bit clean again and to stay sequentially consistent. However, can we live without the fence? For the correctness of the mutal-exclusion algorithm we can, but we leave the domain of sequential consistent reasoning. The intuitive reason for correctness is that the threads waiting for the lock do no harm while waiting. They only take some action if they see the lock being zero again, this is when the lock release has made its way out of the store buffer.

Another typical example is the following simplified form of barrier synchronization: each processor has a flag that it writes (with ordinarry volatile writes without any flushing) and other processors read, and each processor waits for all processors to set their flags before continuing past the barrier. This is not sequentially consistent - each processor might see his own flag set and later see all other flags clear - but it is still correct.

Common for these examples is that there is only a single writer to an address, and the values written are monotonic in a sense that allows the readers to draw the correct conlcusion when they observe a certain value. This pattern is named Publication Idiom in Owens work [14].

Future work. The first direction of future work is to try to deal with the limitations of sequential consistency described above and try to come up with a more general reduction
theorem that can also handle non sequential consistent code portions that follow some monotonicity rules.

Another direction of future work is to take compiler optimization into account. Our volatile accesses correspond roughly to volatile memory accesses within a C program. An optimizing compiler is free to convert any sequence of non-volatile accesses into a (sequentially semantically equivalent) sequence of accesses. As long as execution is sequentially consistent, equivalence of these programs (e.g., with respect to final states of executions that end with volatile operations) follows immediately by reduction. However, some compilers are a little more lenient in their optimizations, and allow operations on certain local variables to move across volatile operations. In the context of C (where pointers to stack variables can be passed by pointer), the notion of "locality" is somewhat tricky, and makes essential use of C forbidding (semantically) address arithmetic across memory objects.

## Acknowledgements

We thank Mark Hillebrand for discussions and feedback on this work and extensive comments on this report.

## A Appendix

After the explanatory text in the main body of the document we now show the plain theory files.

```
theory ReduceStoreBuffer
imports Main
begin
```


## A. 1 Memory Instructions

```
type-synonym addr = nat
type-synonym val = nat
type-synonym tmp = nat
type-synonym tmps = tmp g val option
type-synonym sop = tmp set }\times(\textrm{tmps}=>\textrm{val})-\mathrm{ domain and function
locale valid-sop =
fixes sop :: sop
assumes valid-sop: }\\textrm{D}f\vartheta\mathrm{ .
    \llbracketsop=(D,f); D\subseteqdom \vartheta\rrbracket
    C
    f \vartheta = f (\vartheta|'D)
```

type-synonym memory $=$ addr $\Rightarrow$ val
type-synonym owns $=$ addr set
type-synonym rels $=$ addr $\Rightarrow$ bool option
type-synonym shared $=$ addr $\Rightarrow$ bool option
type-synonym acq $=$ addr set
type-synonym rel = addr set

```
type-synonym lcl = addr set
type-synonym wrt \(=\) addr set
type-synonym cond \(=\mathrm{tmps} \Rightarrow\) bool
type-synonym ret \(=\mathrm{val} \Rightarrow \mathrm{val} \Rightarrow \mathrm{val}\)
datatype instr \(=\) Read bool addr tmp
| Write bool addr sop acq lcl rel wrt
| RMW addr tmp sop cond ret acq lcl rel wrt
Fence
| Ghost acq lcl rel wrt
type-synonym instrs \(=\) instr list
type-synonym ('p,'sb,'dirty,'owns,'rels) thread-config =
    ' \(\mathrm{p} \times\) instrs \(\times \mathrm{tmps} \times\) 'sb \(\times\) 'dirty \(\times\) 'owns \(\times\) 'rels
type-synonym ('p,'sb,'dirty,'owns,'rels,'shared) global-config \(=\)
    ('p,'sb,'dirty,'owns,'rels) thread-config list \(\times\) memory \(\times\) 'shared
definition owned \(\mathrm{t}=(\) let \((\mathrm{p}\), instrs \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{t}\) in \(\mathcal{O})\)
lemma owned-simp [simp]: owned (p,instrs,, ,sb, \(\mathcal{D}, \mathcal{O}, \mathcal{R})=(\mathcal{O})\)
    by (simp add: owned-def)
definition \(\mathcal{O}\)-sb \(\mathrm{t}=(\mathrm{let}(\mathrm{p}\), instrs \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{t}\) in \((\mathcal{O}, \mathrm{sb}))\)
lemma \(\mathcal{O}\)-sb-simp [simp]: \(\mathcal{O}\)-sb \((\mathrm{p}\), instrs \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=(\mathcal{O}, \mathrm{sb})\)
    by (simp add: \(\mathcal{O}\)-sb-def)
definition released \(\mathrm{t}=(\) let \((\mathrm{p}\), instrs \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{t}\) in \(\mathcal{R})\)
lemma released-simp [simp]: released (p,instrs,, ,sb, \(\mathcal{D}, \mathcal{O}, \mathcal{R})=(\mathcal{R})\)
    by (simp add: released-def)
lemma list-update-id': v = xs ! i \(\Longrightarrow \mathrm{xs}[\mathrm{i}:=\mathrm{v}]=\mathrm{xs}\)
    by simp
lemmas converse-rtranclp-induct5 \(=\)
converse-rtranclp-induct \(\left[\right.\) where \(\quad \mathrm{a}=(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \quad\) and \(\quad \mathrm{b}=\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}\right)\),
split-rule,consumes 1, case-names refl step]
```


## A. 2 Abstract Program Semantics

locale memory-system $=$
fixes
memop-step :: (instrs $\times$ tmps $\times$ 'sb $\times$ memory $\times$ 'dirty $\times$ 'owns $\times$ 'rels $\times$ 'shared $) \Rightarrow$ (instrs $\times$ tmps $\times$ 'sb $\times$ memory $\times$ 'dirty $\times$ 'owns $\times$ 'rels $\times$ 'shared) $\Rightarrow$ bool $\left(-\rightarrow_{m}-[60,60] 100\right)$ and
storebuffer-step:: (memory $\times$ 'sb $\times$ 'owns $\times$ 'rels $\times$ 'shared) $\Rightarrow$ (memory $\times$ 'sb $\times$ 'owns $\times$ 'rels $\times$ 'shared $) \Rightarrow$ bool $\left(-\rightarrow_{\text {sb }}-[60,60] 100\right)$

```
locale program \(=\)
    fixes
    program-step \(::\) tmps \(\Rightarrow{ }^{\prime} \mathrm{p} \Rightarrow\) 'p \(\times\) instrs \(\Rightarrow\) bool \(\left(-\vdash-\rightarrow_{\mathrm{p}}-[60,60,60] 100\right)\)
```

    - A program only accesses the shared memory indirectly, it can read the temporaries
    and can output a sequence of memory instructions
locale computation $=$ memory-system + program +
constrains
- The constrains are only used to name the types 'sb and ' $p$
storebuffer-step:: (memory $\times$ 'sb $\times$ 'owns $\times$ 'rels $\times$ 'shared) $\Rightarrow$ (memory $\times$ 'sb $\times$ 'owns
$\times$ 'rels $\times$ 'shared) $\Rightarrow$ bool and
memop-step ::
(instrs $\times$ tmps $\times$ 'sb $\times$ memory $\times$ 'dirty $\times$ 'owns $\times$ 'rels $\times$ 'shared) $\Rightarrow$
(instrs $\times$ tmps $\times$ 'sb $\times$ memory $\times$ 'dirty $\times$ 'owns $\times$ 'rels $\times$ 'shared) $\Rightarrow$ bool
and
program-step :: tmps $\Rightarrow$ 'p $\Rightarrow$ 'p $\times$ instrs $\Rightarrow$ bool
fixes
record $::$ ' $\Rightarrow$ 'p $\Rightarrow$ instrs $\Rightarrow$ 'sb $\Rightarrow$ 'sb

## begin

inductive concurrent-step ::
('p,'sb,'dirty,'owns,'rels,'shared) global-config $\Rightarrow$ ('p,'sb,'dirty,'owns,'rels,'shared)
global-config $\Rightarrow$ bool

$$
(-\Rightarrow-[60,60] 100)
$$

## where

Program:

$$
\begin{aligned}
& \llbracket \mathrm{i}<\text { length ts } ; \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \text { is }, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \\
& \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}, \mathrm{is}\right) \rrbracket \Longrightarrow \\
& (\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \text { is } @ \mathrm{is}^{\prime}, \vartheta, \text { record } \mathrm{p} \mathrm{p}^{\prime} \text { is' } \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right], \mathrm{m}, \mathcal{S}\right)
\end{aligned}
$$

| Memop:

$$
\begin{aligned}
& \llbracket \mathrm{i} \text { < length ts; ts!i }=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \\
& \quad(\mathrm{is}, \vartheta, \mathrm{sb}, \mathrm{~m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{m}}\left(\mathrm{is}{ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right) \rrbracket \\
& \Longrightarrow \\
& (\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right], \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)
\end{aligned}
$$

| StoreBuffer:

$$
\begin{aligned}
& \llbracket \mathrm{i}<\text { length ts; ts }!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \\
& \quad(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}\left(\mathrm{~m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right) \rrbracket \Longrightarrow \\
& (\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right], \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)
\end{aligned}
$$

definition final:: ('p,'sb,'dirty, 'owns,'rels,'shared) global-config $\Rightarrow$ bool where
final $c=\left(\neg\left(\exists c^{\prime} . c \Rightarrow c^{\prime}\right)\right)$

```
lemma store-buffer-steps:
assumes sb-step: storebuffer-step \({ }^{\wedge} * *(\mathrm{~m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S})\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows \(\bigwedge\) ts. \(\mathrm{i}<\) length ts \(\Longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \Longrightarrow\)
    concurrent-step \(* *(\mathrm{ts}, \mathrm{m}, \mathcal{S})\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right], \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
using sb-step
proof (induct rule: converse-rtranclp-induct5)
    case refl then show ?case
        by (simp add: list-update-id')
next
    case (step \(\mathrm{m} \operatorname{sb} \mathcal{O} \mathcal{R} \mathcal{S} \mathrm{m}^{\prime \prime} \mathrm{sb}^{\prime \prime} \mathcal{O}^{\prime \prime} \mathcal{R}^{\prime \prime} \mathcal{S}^{\prime \prime}\) )
    note i -bound \(=\langle\mathrm{i}<\) length ts \(\rangle\)
    note ts- \(\mathrm{i}=\langle\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{\vartheta}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle\)
    note step \(=\left\langle(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}\left(\mathrm{m}^{\prime \prime}, \mathrm{sb}^{\prime \prime}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\right\rangle\)
    let \(? \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\), is, \(\left.\left.\vartheta, \mathrm{sb}^{\prime \prime}, \mathcal{D}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}\right)\right]\)
    from StoreBuffer [OF i-bound ts-i step]
    have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow\left(? \mathrm{ts}^{\prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\).
    also
    from i-bound have i-bound': \(\mathrm{i}<\) length ?ts \({ }^{\prime}\) by simp
    from i-bound have ts \({ }^{\prime}\) - i ? \(? \mathrm{ts}{ }^{\prime}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime \prime}, \mathcal{D}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}\right)\)
        by simp
    from step.hyps (3) [OF i-bound' ts' \({ }^{\prime}\) i] i-bound
    have concurrent-step** \(\left(? \mathrm{ts}^{\prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is, \(\left.\left.\left.\vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right], \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
        by ( \(\operatorname{simp}\) )
    finally
    show ?case .
qed
lemma step-preserves-length-ts:
    assumes step: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
    shows length \(\mathrm{ts}^{\prime}=\) length ts
using step
apply (cases)
apply auto
done
end
lemmas concurrent-step-cases \(=\) computation.concurrent-step.cases
[cases set, consumes 1, case-names Program Memop StoreBuffer]
definition augment-shared:: shared \(\Rightarrow\) addr set \(\Rightarrow\) addr set \(\Rightarrow\) shared \(\left(-\oplus_{-}-[61,1000,60]\right.\)
61)
where
\(\mathcal{S} \oplus \mathrm{w} \mathrm{S} \equiv(\lambda \mathrm{a}\). if \(\mathrm{a} \in \mathrm{S}\) then Some \((\mathrm{a} \in \mathrm{W})\) else \(\mathcal{S} \mathrm{a})\)
definition restrict-shared:: shared \(\Rightarrow\) addr set \(\Rightarrow\) addr set \(\Rightarrow\) shared (- \(\ominus_{-}-[51,1000,50]\)
51)
where
\(\mathcal{S} \ominus_{\mathrm{A}} \mathrm{L} \equiv(\lambda \mathrm{a}\). if \(\mathrm{a} \in \mathrm{L}\) then None
    else (case \(\mathcal{S}\) a of None \(\Rightarrow\) None
```

$$
\mid \text { Some writeable } \Rightarrow \text { Some ( } \mathrm{a} \in \mathrm{~A} \vee \text { writeable) }) \text { ) }
$$

definition read-only :: shared $\Rightarrow$ addr set where
read-only $\mathcal{S} \equiv\{\mathrm{a} .(\mathcal{S} \mathrm{a}=$ Some False $)\}$
definition shared-le:: shared $\Rightarrow$ shared $\Rightarrow$ bool (infix $\subseteq_{s} 50$ )
where
$\mathrm{m}_{1} \subseteq_{\mathrm{s}} \mathrm{m}_{2} \equiv \mathrm{~m}_{1} \subseteq_{\mathrm{m}} \mathrm{m}_{2} \wedge$ read-only $\mathrm{m}_{1} \subseteq$ read-only $\mathrm{m}_{2}$
lemma shared-leD: $\mathrm{m}_{1} \subseteq_{\mathrm{s}} \mathrm{m}_{2} \Longrightarrow \mathrm{~m}_{1} \subseteq_{\mathrm{m}} \mathrm{m}_{2} \wedge$ read-only $\mathrm{m}_{1} \subseteq$ read-only $\mathrm{m}_{2}$ by (simp add: shared-le-def)
lemma shared-le-map-le: $\mathrm{m}_{1} \subseteq_{\mathrm{s}} \mathrm{m}_{2} \Longrightarrow \mathrm{~m}_{1} \subseteq_{\mathrm{m}} \mathrm{m}_{2}$
by (simp add: shared-le-def)
lemma shared-le-read-only-le: $\mathrm{m}_{1} \subseteq_{\mathrm{s}} \mathrm{m}_{2} \Longrightarrow$ read-only $\mathrm{m}_{1} \subseteq$ read-only $\mathrm{m}_{2}$
by (simp add: shared-le-def)
lemma dom-augment [simp]: dom $(\mathrm{m} \oplus \mathrm{w} \mathrm{S})=\operatorname{dom} \mathrm{m} \cup \mathrm{S}$
by (auto simp add: augment-shared-def)
lemma augment-empty [simp]: $\mathrm{S} \oplus_{\mathrm{x}}\{ \}=\mathrm{S}$
by (simp add: augment-shared-def)
lemma inter-neg [simp]: $\mathrm{X} \cap-\mathrm{L}=\mathrm{X}-\mathrm{L}$
by blast
lemma dom-restrict-shared [simp]: $\operatorname{dom}\left(\mathrm{m} \ominus_{\mathrm{A}} \mathrm{L}\right)=\operatorname{dom} \mathrm{m}-\mathrm{L}$
by (auto simp add: restrict-shared-def split: option.splits)
lemma restrict-shared-UNIV [simp]: $\left(\mathrm{m} \ominus_{\mathrm{A}}\right.$ UNIV) $=$ Map.empty
by (auto simp add: restrict-shared-def split: if-split-asm option.splits)
lemma restrict-shared-empty [simp]: (Map.empty $\ominus_{\mathrm{A}} \mathrm{L}$ ) $=$ Map.empty apply (rule ext)
by (auto simp add: restrict-shared-def split: if-split-asm option.splits)
lemma restrict-shared-in [simp]: $\mathrm{a} \in \mathrm{L} \Longrightarrow\left(\mathrm{m} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=$ None
by (auto simp add: restrict-shared-def split: if-split-asm option.splits)
lemma restrict-shared-out: $\mathrm{a} \notin \mathrm{L} \Longrightarrow\left(\mathrm{m} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=$ map-option ( $\lambda$ writeable. ( $\mathrm{a} \in \mathrm{A} \vee$ writeable)) ( m a)
by (auto simp add: restrict-shared-def split: if-split-asm option.splits)
lemma restrict-shared-out '[simp]:
$\mathrm{a} \notin \mathrm{L} \Longrightarrow \mathrm{m} \mathrm{a}=$ Some writeable $\Longrightarrow\left(\mathrm{m} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=$ Some $(\mathrm{a} \in \mathrm{A} \vee$ writeable $)$
by (simp add: restrict-shared-out)
lemma augment-mono-map': $\mathrm{A} \subseteq_{m} \mathrm{~B} \Longrightarrow\left(\mathrm{~A} \oplus_{x} \mathrm{C}\right) \subseteq_{m}\left(\mathrm{~B} \oplus_{x} \mathrm{C}\right)$
by (auto simp add: augment-shared-def map-le-def domIff)
lemma augment-mono-map: $\mathrm{A} \subseteq_{s} \mathrm{~B} \Longrightarrow\left(\mathrm{~A} \oplus_{\mathrm{X}} \mathrm{C}\right) \subseteq_{s}\left(\mathrm{~B} \oplus_{x} \mathrm{C}\right)$
by (auto simp add: augment-shared-def shared-le-def map-le-def read-only-def dom-def split: option.splits if-split-asm)
lemma restrict-mono-map: $\mathrm{A} \subseteq_{s} \mathrm{~B} \Longrightarrow\left(\mathrm{~A} \ominus_{x} \mathrm{C}\right) \subseteq_{s}\left(\mathrm{~B} \ominus_{x} \mathrm{C}\right)$
by (auto simp add: restrict-shared-def shared-le-def map-le-def read-only-def dom-def split: option.splits if-split-asm)
lemma augment-mono-aux: $\operatorname{dom} \mathrm{A} \subseteq \operatorname{dom} \mathrm{B} \Longrightarrow \operatorname{dom}\left(\mathrm{A} \oplus_{x} \mathrm{C}\right) \subseteq \operatorname{dom}\left(\mathrm{B} \oplus_{x} \mathrm{C}\right)$
by auto
lemma restrict-mono-aux: $\operatorname{dom} \mathrm{A} \subseteq \operatorname{dom} \mathrm{B} \Longrightarrow \operatorname{dom}\left(\mathrm{A} \ominus_{x} \mathrm{C}\right) \subseteq \operatorname{dom}\left(\mathrm{B} \ominus_{x} \mathrm{C}\right)$
by auto
lemma read-only-mono: $S \subseteq_{m} S^{\prime} \Longrightarrow a \in$ read-only $S \Longrightarrow a \in$ read-only $S^{\prime}$ by (auto simp add: map-le-def domIff read-only-def dest!: bspec)
lemma in-read-only-restrict-conv:
$\mathrm{a} \in \operatorname{read}$-only $\left(\mathcal{S} \ominus_{\mathrm{A}} \mathrm{L}\right)=(\mathrm{a} \in \operatorname{read}-$ only $\mathcal{S} \wedge \mathrm{a} \notin \mathrm{L} \wedge \mathrm{a} \notin \mathrm{A})$
by (auto simp add: read-only-def restrict-shared-def split: option.splits if-split-asm)
lemma in-read-only-augment-conv: a $\in \operatorname{read-only}(\mathcal{S} \oplus \mathrm{w} \mathrm{R})=($ if $\mathrm{a} \in \mathrm{R}$ then a $\notin \mathrm{W}$ else a $\in$ read-only $\mathcal{S}$ )
by (auto simp add: read-only-def augment-shared-def)
lemmas in-read-only-convs $=$ in-read-only-restrict-conv in-read-only-augment-conv
lemma read-only-dom: read-only $\mathcal{S} \subseteq \operatorname{dom} \mathcal{S}$
by (auto simp add: read-only-def dom-def)
lemma read-only-empty [simp]: read-only Map.empty $=\{ \}$
by (auto simp add: read-only-def)
lemma restrict-shared-fuse: $S \ominus_{A} L \ominus_{B} M=\left(S \ominus_{(A \cup B)}(L \cup M)\right)$
apply (rule ext)
apply (auto simp add: restrict-shared-def split: option.splits if-split-asm)
done
lemma restrict-shared-empty-set [simp]: $\mathrm{S} \ominus_{\{ \}}\{ \}=\mathrm{S}$
apply (rule ext)
apply (auto simp add: restrict-shared-def split: option.splits if-split-asm)
done
definition augment-rels:: addr set $\Rightarrow$ addr set $\Rightarrow$ rels $\Rightarrow$ rels
where
augment-rels S R $\mathcal{R}=(\lambda a$. if $a \in R$
then (case $\mathcal{R}$ a of
None $\Rightarrow$ Some $(a \in S)$
$\mid$ Some $\mathrm{s} \Rightarrow$ Some $(\mathrm{s} \wedge(\mathrm{a} \in \mathrm{S})))$
else $\mathcal{R}$ a)
declare domIff [iff del]

## A. 3 Memory Transitions

locale gen-direct-memop-step $=$
fixes emp::'rels and aug::owns $\Rightarrow$ rel $\Rightarrow$ 'rels $\Rightarrow$ 'rels
begin
inductive gen-direct-memop-step :: (instrs $\times$ tmps $\times$ unit $\times$ memory $\times$ bool $\times$ owns $\times$ 'rels $\times$ shared ) $\Rightarrow$

```
(instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) 'rels \(\times\) shared ) \(\Rightarrow\) bool
    \((-\rightarrow-[60,60] 100)\)
```

where
Read: (Read volatile a $\mathrm{t} \#$ is $, \vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$

$$
(\mathrm{is}, \vartheta(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a}), \mathrm{x}, \mathrm{~m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})
$$

## | WriteNonVolatile:

(Write False a (D,f) A L R W\#is, $\mathcal{\vartheta}$, x, m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$ (is, $\vartheta, \mathrm{x}, \mathrm{m}(\mathrm{a}:=\mathrm{f} \vartheta), \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$

## | WriteVolatile:

(Write True a (D,f) A L R W\# is, $\mathcal{\vartheta}$, x, m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$ (is, $\boldsymbol{\vartheta}, \mathrm{x}, \mathrm{m}(\mathrm{a}:=\mathrm{f} \boldsymbol{\vartheta})$, True, $\mathcal{O} \cup \mathrm{A}-\mathrm{R}, \mathrm{emp}, \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
| Fence:
(Fence $\#$ is, $\vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow(\mathrm{is}, \vartheta, \mathrm{x}, \mathrm{m}$, False, $\mathcal{O}, \mathrm{emp}, \mathcal{S})$

## | RMWReadOnly:

$\llbracket \neg$ cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
(RMW a t (D,f) cond ret A L R W \# is, $\vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow(\mathrm{is}, \vartheta(\mathrm{t} \mapsto \mathrm{m}$ a), $\mathrm{x}, \mathrm{m}$, False, $\mathcal{O}, \mathrm{emp}, \mathcal{S})$

## | RMWWrite:

$\llbracket \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
(RMW at (D,f) cond ret A L R W\# is, $\vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$ (is, $\vartheta(\mathrm{t} \mapsto \mathrm{ret}(\mathrm{m}$ a) $(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a $)))), \mathrm{x}, \mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a) $))$, False, $\mathcal{O} \cup \mathrm{A}-\mathrm{R}, \mathrm{emp}$, $\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
| Ghost:
(Ghost A L R W \# is, $\vartheta \mathcal{\vartheta}, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$

$$
\text { (is, } \left.\vartheta, \mathrm{x}, \mathrm{~m}, \mathcal{D}, \mathcal{O} \cup \mathrm{~A}-\mathrm{R}, \operatorname{aug}(\operatorname{dom} \mathcal{S}) \mathrm{R} \mathcal{R}, \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
$$

end
interpretation direct-memop-step: gen-direct-memop-step Map.empty augment-rels .

```
term direct-memop-step.gen-direct-memop-step
abbreviation direct-memop-step :: (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\)
rels \(\times\) shared ) \(\Rightarrow\)
                                    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) rels \(\times\) shared \() \Rightarrow\) bool
                                    \((-\rightarrow-[60,60] 100)\)
```

```
where
direct-memop-step \(\equiv\) direct-memop-step.gen-direct-memop-step
\(\operatorname{term} \mathrm{x} \rightarrow \mathrm{Y}\)
abbreviation direct-memop-steps ::
    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) rels \(\times\) shared \() \Rightarrow\)
    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) rels \(\times\) shared )
    \(\Rightarrow\) bool
    \(\left(-\rightarrow^{*}-[60,60] 100\right)\)
where
direct-memop-steps \(==(\text { direct-memop-step })^{\wedge} * *\)
\(\operatorname{term} \mathrm{x} \rightarrow^{*} \mathrm{Y}\)
interpretation virtual-memop-step: gen-direct-memop-step () ( \(\lambda \mathrm{S} R \mathcal{R} .())\).
abbreviation virtual-memop-step :: (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\)
unit \(\times\) shared ) \(\Rightarrow\)
    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) unit \(\times\) shared \() \Rightarrow\) bool
    \(\left(-\rightarrow_{v}-[60,60] 100\right)\)
where
virtual-memop-step \(\equiv\) virtual-memop-step.gen-direct-memop-step
\(\operatorname{term} \mathrm{x} \rightarrow_{\mathrm{v}} \mathrm{Y}\)
abbreviation virtual-memop-steps ::
    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) unit \(\times\) shared \() \Rightarrow\)
    (instrs \(\times\) tmps \(\times\) unit \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) unit \(\times\) shared )
    \(\Rightarrow\) bool
    \(\left(-\rightarrow_{v}{ }^{*}-[60,60] 100\right)\)
where
virtual-memop-steps \(==(\text { virtual-memop-step })^{\wedge} * *\)
\(\operatorname{term} \mathrm{x} \rightarrow{ }^{*} \mathrm{Y}\)
```

lemma virtual-memop-step-simulates-direct-memop-step:
assumes step:
(is, $\vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{x}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
shows (is, $\vartheta, \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O},(), \mathcal{S}) \rightarrow_{\mathrm{v}}\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{x}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime},(), \mathcal{S}^{\prime}\right)$
using step
apply (cases)
apply (auto intro: virtual-memop-step.gen-direct-memop-step.intros)
done

## A. 4 Safe Configurations of Virtual Machines

```
inductive safe-direct-memop-state :: owns list \(\Rightarrow\) nat \(\Rightarrow\)
    (instrs \(\times\) tmps \(\times\) memory \(\times\) bool \(\times\) owns \(\times\) shared \() \Rightarrow\) bool
        \((-,-\vdash-\sqrt{ }[60,60,60] 100)\)
```

where
Read: $\llbracket \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in \operatorname{read}$-only $\mathcal{S} \vee($ volatile $\wedge \mathrm{a} \in \operatorname{dom} \mathcal{S})$;
volatile $\longrightarrow \neg \mathcal{D} \rrbracket$

| WriteNonVolatile:
$\llbracket \mathrm{a} \in \mathcal{O} ; \mathrm{a} \notin \operatorname{dom} \mathcal{S} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathrm{i} \vdash($ Write False a $(\mathrm{D}, \mathrm{f}) \mathrm{A} \mathrm{L} \mathrm{R} \mathrm{W} \#$ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| WriteVolatile:
$\llbracket \forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \mathcal{O} \mathrm{s}!j ;$
$\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap \mathcal{O} s!j=\{ \} ;$
a $\notin$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathrm{i}+($ Write $\operatorname{True}$ a $(\mathrm{D}, \mathrm{f}) \mathrm{A} \operatorname{LR} \mathrm{W} \#$ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Fence:
$\mathcal{O} \mathrm{s}, \mathrm{i} \vdash($ Fence $\#$ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Ghost:
$\llbracket \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap \mathcal{O} s!j=\{ \} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathrm{i} \vdash($ Ghost A L R W\# is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| RMWReadOnly:
$\llbracket \neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) ; \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in \operatorname{dom} \mathcal{S} \rrbracket \Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathrm{i} \vdash(\mathrm{RMW}$ at $(\mathrm{D}, \mathrm{f})$ cond ret A L R W\# is, $\boldsymbol{\vartheta}$, m, $\mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| RMWWrite:
$\llbracket \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}))$;
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \mathcal{O} \mathrm{s}!$;
$\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap \mathcal{O} \mathrm{s}!\mathrm{j}=\{ \} ;$
a $\notin$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathrm{i} \vdash(\mathrm{RMW}$ at $(\mathrm{D}, \mathrm{f})$ cond ret A L R W\# is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Nil: $\mathcal{O} s, i \vdash([], \vartheta, m, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
inductive safe-delayed-direct-memop-state $::$ owns list $\Rightarrow$ rels list $\Rightarrow$ nat $\Rightarrow$ (instrs $\times$ tmps $\times$ memory $\times$ bool $\times$ owns $\times$ shared) $\Rightarrow$ bool $(-,-,-\vdash-\sqrt{ }[60,60,60,60] 100)$

## where

Read: $\llbracket \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in$ read-only $\mathcal{S} \vee$ (volatile $\wedge \mathrm{a} \in \operatorname{dom} \mathcal{S}$ );

```
\(\forall \mathrm{j}<\) length \(\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow(\mathcal{R s}!\mathrm{j})\) a \(\neq\) Some False; - no release of unshared address
\(\neg\) volatile \(\longrightarrow(\forall \mathrm{j}<\) length \(\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \operatorname{dom}(\mathcal{R s}!\mathrm{j})\) );
volatile \(\longrightarrow \neg \mathcal{D} \rrbracket\)
\(\Longrightarrow\)
\(\mathcal{O} \mathrm{s}, \mathcal{R s}, \mathrm{i} \vdash(\) Read volatile a \(\mathrm{t} \#\) is, \(\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\)
```

| WriteNonVolatile:
$\llbracket \mathrm{a} \in \mathcal{O} ; \mathrm{a} \notin \operatorname{dom} \mathcal{S} ; \forall \mathrm{j}<\operatorname{length} \mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \operatorname{dom}(\mathcal{R s}!\mathrm{j}) \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathcal{R s}, \mathrm{i} \vdash($ Write False a $(\mathrm{D}, \mathrm{f})$ A L R W\#is, $\mathcal{\vartheta}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| WriteVolatile:
$\llbracket \forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin(\mathcal{O} \mathrm{s}!\mathrm{j} \cup \operatorname{dom}(\mathcal{R s}!j))$;
$\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap(\mathcal{O} \mathrm{s}!\mathrm{j} \cup \operatorname{dom}(\mathcal{R s}!\mathrm{j}))=\{ \} ;$
a $\notin$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathcal{R s}, \mathrm{i} \vdash($ Write True a $(\mathrm{D}, \mathrm{f}) \mathrm{A}$ L R W\# is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Fence:
$\mathcal{O} \mathrm{s}, \mathcal{R s}, \mathrm{i} \vdash($ Fence $\#$ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Ghost:
$\llbracket \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap(\mathcal{O} s!\mathrm{j} \cup \operatorname{dom}(\mathcal{R} s!j))=\{ \} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathcal{R} \mathrm{s}, \mathrm{i} \vdash($ Ghost A L R W\# is, $\vartheta \mathcal{\vartheta}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| RMWReadOnly:
$\llbracket \neg$ cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) ; \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in \operatorname{dom} \mathcal{S} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow(\mathcal{R s}!\mathrm{j})$ a $\neq$ Some False - no release of unshared address』
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathcal{R} \mathrm{s}, \mathrm{i} \vdash(\mathrm{RMW}$ at $(\mathrm{D}, \mathrm{f})$ cond ret A L R W\# is, $\vartheta \mathcal{\vartheta}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| RMWWrite:
$\llbracket \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) ; \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in \operatorname{dom} \mathcal{S} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin(\mathcal{O} \mathrm{s}!\mathrm{j} \cup \operatorname{dom}(\mathcal{R s}!\mathrm{j})) ;$
$\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O} ; \mathrm{L} \subseteq \mathrm{A} ; \mathrm{R} \subseteq \mathcal{O} ; \mathrm{A} \cap \mathrm{R}=\{ \} ;$
$\forall \mathrm{j}<$ length $\mathcal{O} \mathrm{s} . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap(\mathcal{O} \mathrm{s}!\mathrm{j} \cup \operatorname{dom}(\mathcal{R s}!\mathrm{j}))=\{ \} ;$
a $\notin$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow$
$\mathcal{O} \mathrm{s}, \mathcal{R s}, \mathrm{i} \vdash(\mathrm{RMW}$ at $(\mathrm{D}, \mathrm{f})$ cond ret A L R W\# is, $\mathcal{\vartheta}$, m, $\mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
| Nil: $\mathcal{O} \mathrm{s}, \mathcal{R} s, i \vdash([], \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$

```
lemma memop-safe-delayed-implies-safe-free-flowing:
    assumes safe-delayed: \(\mathcal{O} \mathrm{s}, \mathcal{R} \mathrm{s}, \mathrm{i} \vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\)
    shows \(\mathcal{O} \mathrm{s}, \mathrm{i} \vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\)
using safe-delayed
proof (cases)
    case Read thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.intros)
next
    case WriteNonVolatile thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.intros)
next
    case WriteVolatile thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.intros)
next
    case Fence thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.intros)
next
    case Ghost thus ? thesis
    by (fastforce intro!: safe-direct-memop-state.Ghost)
next
    case RMWReadOnly thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.intros)
next
    case RMWWrite thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.RMWWrite)
next
    case Nil thus ?thesis
        by (fastforce intro!: safe-direct-memop-state.Nil)
qed
lemma memop-empty-rels-safe-free-flowing-implies-safe-delayed:
    assumes safe: \(\mathcal{O}\) s,i \(\vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\)
    assumes empty: \(\forall \mathcal{R} \in\) set \(\mathcal{R}\) s. \(\mathcal{R}=\) Map.empty
    assumes leq: length \(\mathcal{O}_{\mathrm{s}}=\) length \(\mathcal{R} \mathrm{s}\)
    assumes unowned-shared: \((\forall \mathrm{a} .(\forall \mathrm{i}<\) length \(\mathcal{O}\) s. a \(\notin(\mathcal{O} \mathrm{s}!\mathrm{i})) \longrightarrow \mathrm{a} \in \operatorname{dom} \mathcal{S})\)
    assumes Os-i: \(\mathcal{O}\) s!i \(=\mathcal{O}\)
    shows \(\mathcal{O}\) s \(, \mathcal{R} \mathrm{s}, \mathrm{i} \vdash(\) is, \(\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\)
using safe
proof (cases)
    case Read thus ?thesis
        using leq empty
        by (fastforce intro!: safe-delayed-direct-memop-state.Read dest: nth-mem)
next
    case WriteNonVolatile thus ?thesis
        using leq empty
        by (fastforce intro!: safe-delayed-direct-memop-state.intros dest: nth-mem)
next
    case WriteVolatile thus ?thesis
    using leq empty
        apply clarsimp
```

```
    apply (rule safe-delayed-direct-memop-state.WriteVolatile)
    apply (auto)
    apply (drule nth-mem)
    apply fastforce
    apply (drule nth-mem)
    apply fastforce
    done
next
    case Fence thus ?thesis
        by (fastforce intro!: safe-delayed-direct-memop-state.intros)
next
    case Ghost thus ?thesis
    using leq empty
        apply clarsimp
        apply (rule safe-delayed-direct-memop-state.Ghost)
        apply (auto)
        apply (drule nth-mem)
        apply fastforce
        done
next
    case RMWReadOnly thus ?thesis
    using leq empty
        by (fastforce intro!: safe-delayed-direct-memop-state.intros dest: nth-mem)
next
    case (RMWWrite cond t a A L R D f ret W) thus ?thesis
    using leq empty unowned-shared [rule-format, where a=a] Os-i
        apply clarsimp
        apply (rule safe-delayed-direct-memop-state.RMWWrite)
        apply (auto)
        apply (drule nth-mem)
        apply fastforce
        apply (drule nth-mem)
        apply fastforce
        done
next
    case Nil thus ?thesis
        by (fastforce intro!: safe-delayed-direct-memop-state.Nil)
qed
inductive id-storebuffer-step::
(memory \(\times\) unit \(\times\) owns \(\times\) rels \(\times\) shared \() \Rightarrow(\) memory \(\times\) unit \(\times\) owns \(\times\) rels \(\times\) shared \()\)
\(\Rightarrow\) bool \(\left(-\rightarrow_{\mathrm{I}}-[60,60] 100\right)\)
where
\(\mathrm{Id}:(\mathrm{m}, \mathrm{x}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow \mathrm{I}(\mathrm{m}, \mathrm{x}, \mathcal{O}, \mathcal{R}, \mathcal{S})\)
definition empty-storebuffer-step:: (memory \(\times\) 'sb \(\times\) 'owns \(\times\) 'rels \(\times\) 'shared) \(\Rightarrow\) (memory \(\times\) 'sb \(\times\) 'owns \(\times\) 'rels \(\times\) 'shared) \(\Rightarrow\) bool
where
empty-storebuffer-step c c \({ }^{\prime}=\) False
```


## context program

begin
abbreviation direct-concurrent-step ::
('p,unit,bool,owns,rels,shared) global-config $\Rightarrow$ ('p,unit,bool,owns,rels,shared)
global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{d}-[100,60] 100\right)$
where
direct-concurrent-step $\equiv$
computation.concurrent-step direct-memop-step.gen-direct-memop-step
empty-storebuffer-step program-step
( $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$ )
abbreviation direct-concurrent-steps::
('p,unit,bool,owns,rels,shared) global-config $\Rightarrow$ ('p,unit,bool,owns,rels,shared) global-config $\Rightarrow$ bool

$$
\left(-\Rightarrow_{d}^{*}-[60,60] 100\right)
$$

## where

direct-concurrent-steps $==$ direct-concurrent-step ${ }^{*} * *$
abbreviation virtual-concurrent-step ::
('p,unit,bool,owns,unit,shared) global-config $\Rightarrow$ ('p,unit,bool,owns,unit,shared)
global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{v}-[100,60] 100\right)$
where
virtual-concurrent-step $\equiv$
computation.concurrent-step virtual-memop-step.gen-direct-memop-step
empty-storebuffer-step program-step
( $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$ )
abbreviation virtual-concurrent-steps::
('p,unit,bool,owns,unit,shared) global-config $\Rightarrow$ ('p,unit,bool,owns,unit,shared)
global-config $\Rightarrow$ bool

$$
\left(-\Rightarrow_{v}{ }^{*}-[60,60] 100\right)
$$

## where

virtual-concurrent-steps $==$ virtual-concurrent-step ${ }^{\wedge} * *$

```
\(\operatorname{term} \mathrm{x} \Rightarrow_{\mathrm{v}} \mathrm{Y}\)
\(\operatorname{term} \mathrm{x} \Rightarrow_{\mathrm{d}} \mathrm{Y}\)
\(\operatorname{term} \mathrm{x} \Rightarrow_{\mathrm{d}}{ }^{*} \mathrm{Y}\)
\(\operatorname{term} \mathrm{x} \Rightarrow{ }_{\mathrm{v}}{ }^{*} \mathrm{Y}\)
end
definition
safe-reach step safe cfg \(\equiv\)
    \(\forall \operatorname{cfg}^{\prime}\). step \({ } * * \operatorname{cfg} \mathrm{cfg}^{\prime} \longrightarrow\) safe \(\mathrm{cfg}^{\prime}\)
```

lemma safe-reach-safe-refl: safe-reach step safe cfg $\Longrightarrow$ safe cfg apply (auto simp add: safe-reach-def) done
lemma safe-reach-safe-rtrancl: safe-reach step safe $\operatorname{cfg} \Longrightarrow$ step ${ }^{*} * * \operatorname{cfg}^{\operatorname{cfg}}{ }^{\prime} \Longrightarrow \operatorname{safe}^{\operatorname{cfg}}{ }^{\prime}$ by (simp only: safe-reach-def)
lemma safe-reach-steps: safe-reach step safe $\mathrm{cfg} \Longrightarrow$ step $^{\wedge} * * \mathrm{cfg}^{\mathrm{cfg}}{ }^{\prime} \Longrightarrow$ safe-reach step safe $\mathrm{cfg}^{\prime}$
apply (auto simp add: safe-reach-def intro: rtranclp-trans)
done
lemma safe-reach-step: safe-reach step safe cfg $\Longrightarrow$ step cfg cfg ${ }^{\prime} \Longrightarrow$ safe-reach step safe $\mathrm{cfg}^{\prime}$
apply (erule safe-reach-steps)
apply (erule r-into-rtranclp)
done
context program
begin

## abbreviation

safe-reach-direct $\equiv$ safe-reach direct-concurrent-step
lemma safe-reac-direct-def':
safe-reach-direct safe cfg $\equiv$
$\forall \operatorname{cfg}^{\prime} . \mathrm{cfg} \Rightarrow{ }_{\mathrm{d}}{ }^{*} \mathrm{cfg}^{\prime} \longrightarrow$ safe $\mathrm{cfg}^{\prime}$
by ( simp add: safe-reach-def)

## abbreviation

safe-reach-virtual $\equiv$ safe-reach virtual-concurrent-step
lemma safe-reac-virtual-def':
safe-reach-virtual safe cfg $\equiv$
$\forall \operatorname{cfg}^{\prime} . \operatorname{cfg} \Rightarrow{ }_{\mathrm{v}}{ }^{*} \mathrm{cfg}^{\prime} \longrightarrow$ safe $\mathrm{cfg}^{\prime}$
by ( simp add: safe-reach-def)
end

## definition

safe-free-flowing cfg $\equiv$ let ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ ) $=\mathrm{cfg}$
in $(\forall \mathrm{i}<$ length ts. let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in
map owned ts, $\mathrm{i} \vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ })$
lemma safeE: $\llbracket$ safe-free-flowing (ts,m,S);i<length ts; ts!i=(p,is, $, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow$ map owned ts,i $\vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$
by (auto simp add: safe-free-flowing-def)

## definition

safe-delayed $\mathrm{cfg} \equiv$ let $(\mathrm{ts}, \mathrm{m}, \mathcal{S})=\mathrm{cfg}$
in $(\forall \mathrm{i}<$ length ts. let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in map owned ts,map released ts,i $\vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ })$
lemma safe-delayedE: $\llbracket$ safe-delayed (ts,m,S);i<length ts; ts!i=(p,is, $, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$ $\Longrightarrow$ map owned ts,map released ts, $\mathrm{i} \vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$ by (auto simp add: safe-delayed-def)
definition remove-rels $\equiv \operatorname{map}(\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O},()))$
theorem (in program) virtual-simulates-direct-step:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
shows (remove-rels ts, $\mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{v}}$ (remove-rels $\left.\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
using step
proof -
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$.
interpret virtual-computation:
computation virtual-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$.
from step show ?thesis
proof (cases)
case (Program jp is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is')
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\operatorname{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}\right.\right.$, is $\left.\left.@ i \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$m^{\prime}: m^{\prime}=m$ and
j-bound: $\mathrm{j}<$ length ts and
ts-j: ts! $j=(p, i s, \vartheta, s b, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
prog-step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $)$
by auto
from ts-j j-bound have
vts-j: remove-rels ts! $\mathfrak{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O},())$ by (auto simp add: remove-rels-def)
from virtual-computation.Program [OF - vts-j prog-step, of m S $]$ j-bound ts ${ }^{\prime}$ show ?thesis
by (clarsimp $\operatorname{simp}$ add: $\mathcal{S}^{\prime} \mathrm{m}^{\prime}$ remove-rels-def map-update)
next
case (Memop jp is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R}$ is $\left.\vartheta^{\prime} \operatorname{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]$ and
j -bound: $\mathrm{j}<$ length ts and
$\mathrm{ts}-\mathrm{j}: \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
mem-step: (is, $\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
by auto
from ts-j j-bound have
vts-j: remove-rels ts! $\mathfrak{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O},())$ by (auto simp add: remove-rels-def)
from virtual-computation.Memop[OF - vts-j vir-tual-memop-step-simulates-direct-memop-step [OF mem-step]] j-bound ts ${ }^{\prime}$

```
    show ?thesis
    by (clarsimp simp add: remove-rels-def map-update)
next
    case (StoreBuffer - p is \vartheta sb \mathcal{O}\mathcal{O}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime})
    hence False
        by (auto simp add: empty-storebuffer-step-def)
    thus ?thesis ..
qed
qed
```

lemmas converse-rtranclp-induct-sbh-steps $=$ converse-rtranclp-induct [of - ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ ) $\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$, split-rule, consumes 1, case-names refl step]

```
theorem (in program) virtual-simulates-direct-steps:
    assumes steps: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
    shows (remove-rels ts, \(\mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{v}}{ }^{*}\left(\right.\) remove-rels \(\left.\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
using steps
proof (induct rule: converse-rtranclp-induct-sbh-steps)
    case refl thus ?case by auto
next
    case (step ts \(\mathrm{m} \mathcal{S} \mathrm{ts}^{\prime \prime} \mathrm{m}^{\prime \prime} \mathcal{S}^{\prime \prime}\) )
    then obtain
```

        first: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}^{\prime \prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\) and
        hyp: (remove-rels ts \(\left.{ }^{\prime \prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right) \Rightarrow_{\mathrm{v}^{*}}\left(\right.\) remove-rels \(\left.\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
        by blast
    note virtual-simulates-direct-step [OF first] also note hyp
    finally
    show ?case by blast
    qed
locale simple-ownership-distinct $=$
fixes ts::('p,'sb,'dirty,owns,'rels) thread-config list
assumes simple-ownership-distinct:
$\bigwedge_{\mathrm{i}} \mathrm{j} \mathrm{p}_{\mathrm{i}}$ is $\mathrm{S}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
[i $<$ length ts; $\mathrm{j}<$ length ts; $\mathrm{i} \neq \mathrm{j}$;
$\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right) ; \mathrm{ts}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
$\rrbracket \Longrightarrow \mathcal{O}_{\mathrm{i}} \cap \mathcal{O}_{\mathrm{j}}=\{ \}$
lemma (in simple-ownership-distinct)
simple-ownership-distinct-nth-update:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\vartheta \mathcal{O} \mathcal{R} \mathcal{D}$ xs sb.
$\llbracket i<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$
$\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.$ let $\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathcal{\vartheta}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)=\mathrm{ts}!\mathrm{j}$
in $\left.\left(\mathcal{O}^{\prime}\right) \cap\left(\mathcal{O}_{\mathrm{j}}\right)=\{ \}\right) \rrbracket \Longrightarrow$
simple-ownership-distinct $\left(\operatorname{tss}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$
apply (unfold-locales)
apply (clarsimp simp add: nth-list-update split: if-split-asm)
apply (force dest: simple-ownership-distinct simp add: Let-def)
apply (fastforce dest: simple-ownership-distinct simp add: Let-def) apply (fastforce dest: simple-ownership-distinct simp add: Let-def) done
locale read-only-unowned $=$
fixes $\mathcal{S}:$ :shared and ts::('p,'sb,'dirty,owns,'rels) thread-config list assumes read-only-unowned:

```
\i p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\textrm{sb}
    \llbracket \mathrm { i } < \text { length ts; ts!i} = ( \mathrm { p } , \mathrm { is } , \vartheta , \mathrm { sb } , \mathcal { D } , \mathcal { O } , \mathcal { R } ) \rrbracket
    \Longrightarrow
    \mathcal { O } \cap \text { read-only } \mathcal { S } = \{ \}
```

lemma (in read-only-unowned)
read-only-unowned-nth-update:
$\bigwedge \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D}$ acq $\vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; $\mathcal{O} \cap$ read-only $\mathcal{S}=\{ \} \rrbracket \Longrightarrow$
read-only-unowned $\mathcal{S}(\operatorname{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
apply (unfold-locales)
apply (auto dest: read-only-unowned
simp add: nth-list-update split: if-split-asm)
done
locale unowned-shared $=$
fixes $\mathcal{S}$ ::shared and ts::('p,'sb, 'dirty,owns,'rels) thread-config list
assumes unowned-shared: $-\bigcup((\lambda(-,-,-,-,-\mathcal{O},-) . \mathcal{O}) \cdot$ set ts $) \subseteq \operatorname{dom} \mathcal{S}$
lemma (in unowned-shared)
unowned-shared-nth-update:
assumes i-bound: i < length ts
assumes ith: ts $!i=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
assumes subset: $\mathcal{O} \subseteq \mathcal{O}^{\prime}$
shows unowned-shared $\mathcal{S}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$
proof -
from i-bound ith subset
have $\bigcup((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})$ ' set ts $) \subseteq$
$\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot \operatorname{set}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\right)$
apply (auto simp add: in-set-conv-nth nth-list-update split: if-split-asm)
subgoal for $\mathrm{x} \mathrm{p}^{\prime \prime}$ is ${ }^{\prime \prime} \mathrm{xs}^{\prime \prime} \mathrm{sb}^{\prime \prime} \mathcal{D}^{\prime \prime} \mathcal{O}^{\prime \prime} \mathcal{R}^{\prime \prime} \mathrm{j}$
apply (case-tac $\mathrm{j}=\mathrm{i}$ )
apply (rule-tac $\mathrm{x}=\left(\mathrm{p}^{\prime}\right.$, is $\left.^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)$ in bexI)
apply fastforce
apply (fastforce simp add: in-set-conv-nth)
apply (rule-tac $\mathrm{x}=\left(\mathrm{p}^{\prime \prime}, \mathrm{is}^{\prime \prime}, \mathrm{xs}^{\prime \prime}, \mathrm{sb}^{\prime \prime}, \mathcal{D}^{\prime \prime}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}\right)$ in bexI)
apply fastforce
apply (fastforce simp add: in-set-conv-nth)
done
done
hence $-\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}){ }^{‘}\right.$ set $\left.\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\right) \subseteq$
$-\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}){ }^{\prime}\right.$ set ts)

```
    by blast
    also note unowned-shared
    finally
    show ?thesis
    by (unfold-locales)
qed
lemma (in unowned-shared) a-unowned-by-others-owned-or-shared:
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-unowned-others:
    j}<l=\mp@code{length (map owned ts). i f j }
        (let }\mp@subsup{\mathcal{O}}{\textrm{j}}{}=(\mathrm{ map owned ts)!j in a }\not\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}
    shows a }\in\mathcal{O}\vee\textrm{a}\in\operatorname{dom}\mathcal{S
proof -
    {
```



```
        assume a-unowned: a }\not\in\mathcal{O
        assume j-bound: j < length ts
        assume jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\textrm{sb
        have a }\not\in\mp@subsup{\mathcal{O}}{\textrm{j}}{
        proof (cases i=j)
            case True with a-unowned ts-i jth
            show ?thesis
    by auto
        next
            case False
            from a-unowned-others [rule-format, of j] j-bound jth False
            show ?thesis
    by auto
        qed
    } note lem = this
    {
        assume a }\not\in\mathcal{O
        from lem [OF this]
        have a }\in-\bigcup((\lambda(-,-,-,-,-,\mathcal{O},-).\mathcal{O})'set ts
            by (fastforce simp add: in-set-conv-nth)
        with unowned-shared have a }\in\operatorname{dom}\mathcal{S
            by auto
    }
    then
    show ?thesis
        by auto
qed
lemma (in unowned-shared) unowned-shared':
    assumes notin: }\forall\textrm{i}<l= length ts. a & owned (ts!i
    shows a }\in\operatorname{dom}\mathcal{S
proof -
```

```
    from notin have a }\in-\bigcup((\lambda(-,-,-,-,-,\mathcal{O},-).\mathcal{O})'set ts
    by (force simp add: in-set-conv-nth)
    with unowned-shared show ?thesis by blast
qed
```

lemma unowned-shared-def ${ }^{\prime}$ : unowned-shared $\mathcal{S}$ ts $=(\forall$ a. $(\forall \mathrm{i}<$ length ts. a $\notin$ owned
$(\mathrm{ts}!\mathrm{i})) \longrightarrow \mathrm{a} \in \operatorname{dom} \mathcal{S})$
apply rule
apply clarsimp
apply (rule unowned-shared.unowned-shared')
apply fastforce
apply fastforce
apply (unfold unowned-shared-def)
apply clarsimp
subgoal for x
apply (drule-tac $x=x$ in spec)
apply (erule impE)
apply clarsimp
apply (case-tac (ts!i))
apply (drule nth-mem)
apply auto
done
done

## definition

initial cfg $\equiv$ let $(\mathrm{ts}, \mathrm{m}, \mathcal{S})=\mathrm{cfg}$
in unowned-shared $\mathcal{S}$ ts $\wedge$ $(\forall \mathrm{i}<$ length ts. let $(\mathrm{p}$, is $, \vartheta, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in $\mathcal{R}=$ Map.empty )
lemma initial-empty-rels: initial ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Longrightarrow \forall \mathcal{R} \in$ set (map released ts). $\mathcal{R}=$ Map.empty
by (fastforce simp add: initial-def simp add: in-set-conv-nth)
lemma initial-unowned-shared: initial $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Longrightarrow$ unowned-shared $\mathcal{S}$ ts
by (fastforce simp add: initial-def )
lemma initial-safe-free-flowing-implies-safe-delayed:
assumes init: initial c
assumes safe: safe-free-flowing c
shows safe-delayed c
proof -
obtain ts $\mathcal{S} \mathrm{m}$ where $\mathrm{c}: \mathrm{c}=(\mathrm{ts}, \mathrm{m}, \mathcal{S})$ by (cases c)
from initial-empty-rels [OF init [simplified c]]
have rels-empty: $\forall \mathcal{R} \in \operatorname{set}$ (map released ts). $\mathcal{R}=$ Map.empty.
from initial-unowned-shared [OF init [simplified c]] have unowned-shared $\mathcal{S}$ ts by auto
hence us: $(\forall$ a. $(\forall \mathrm{i}<$ length $(m a p$ owned ts). a $\notin(m a p$ owned ts $!i)) \longrightarrow \mathrm{a} \in \operatorname{dom} \mathcal{S})$
by (simp add:unowned-shared-def')
\{
fix ip is $\vartheta \times \mathcal{D} \mathcal{O} \mathcal{R}$
assume i-bound: i < length ts
assume ts-i: ts!i $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
have map owned ts,map released ts,i $\vdash($ is $, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$ proof -
from safeE [OF safe [simplified c] i-bound ts-i]
have map owned ts, $i \vdash($ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$.
from memop-empty-rels-safe-free-flowing-implies-safe-delayed [OF this rels-empty us] i-bound ts-i
show ?thesis by simp
qed
\}
then show?thesis
by (fastforce simp add: c safe-delayed-def)
qed
locale program-progress $=$ program +
assumes progress: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $\left.^{\prime}\right) \Longrightarrow \mathrm{p}^{\prime} \neq \mathrm{p} \vee$ is ${ }^{\prime} \neq[]$
The assumption 'progress' could be avoided if we introduce stuttering steps in lemma undo-local-step or make the scheduling of threads explicit, such that we can directly express that 'thread i does not make a step'.lemma (in program-progress) undo-local-step:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
assumes i-bound: i $<$ length ts
assumes unchanged: ts $!i=t s!$ i
assumes safe-delayed-undo: safe-delayed (u-ts,u-m,u-shared) - proof should also work
with weaker safe-free-flowing
assumes leq: length $u-t s=$ length ts
assumes others-same: $\forall \mathrm{j}<$ length ts. $\mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{u}$-ts! $\mathrm{j}=$ ts! j
assumes u-ts-i: u-ts!i=(u-p,u-is,u-tmps,u-x,u-dirty,u-owns,u-rels)
assumes u-m-other: $\forall \mathrm{a}$. a $\notin \mathrm{u}$-owns $\longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{ma}$
assumes u-m-shared: $\forall \mathrm{a} . \mathrm{a} \in \mathrm{u}$-owns $\longrightarrow \mathrm{a} \in$ dom $u$-shared $\longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}$ a
assumes u-shared: $\forall \mathrm{a}$. a $\notin \mathrm{u}$-owns $\longrightarrow \mathrm{a} \notin$ owned (ts!i) $\longrightarrow \mathrm{u}$-shared $\mathrm{a}=\mathcal{S} \mathrm{a}$
assumes dist: simple-ownership-distinct u-ts
assumes dist-ts: simple-ownership-distinct ts
shows $\exists u$-ts' $u$-shared ${ }^{\prime} u$-m ${ }^{\prime}$. (u-ts,u-m,u-shared $) \Rightarrow_{d}\left(u-t s^{\prime}, u-m^{\prime}, u\right.$-shared $) \wedge$

- thread $i$ is unchanged
u-ts! $!\mathbf{i}=\mathrm{u}-\mathrm{ts}!\mathrm{i} \wedge$
( $\forall \mathrm{a} \in \mathrm{u}$-owns. u -shared ${ }^{\prime} \mathrm{a}=\mathrm{u}$-shared a$) \wedge$
$\left(\forall \mathrm{a} \in\right.$ u-owns. $\left.\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in\right.$ u-owns. $\left.u-\mathrm{m}^{\prime} \mathrm{a}=\mathrm{u}-\mathrm{m} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in\right.$ u-owns. $\left.\mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}\right) \wedge$
- other threads are simulated
$(\forall \mathrm{j}<$ length ts. $\mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{u}-\mathrm{ts}!\mathrm{j}=\mathrm{ts}!\mathrm{j}) \wedge$
$\left(\forall \mathrm{a} . \mathrm{a} \notin \mathrm{u}\right.$-owns $\longrightarrow \mathrm{a} \notin$ owned $(\mathrm{ts}!\mathrm{i}) \longrightarrow \mathrm{u}$-shared $\left.{ }^{\prime} \mathrm{a}=\mathcal{S}^{\prime} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} . \mathrm{a} \notin \mathrm{u}\right.$-owns $\left.\longrightarrow \mathrm{u}-\mathrm{m}^{\prime} \mathrm{a}=\mathrm{m}^{\prime} \mathrm{a}\right)$
proof -
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$. from dist interpret simple-ownership-distinct $u$-ts .
from step
show ?thesis
proof (cases)
case (Program j p is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is')
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\operatorname{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}\right.\right.$, is $\left.\left.@ \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
j-bound: $\mathrm{j}<$ length ts and
ts-j: ts! $\mathfrak{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
prog-step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is')
by auto
from progress [OF prog-step] i-bound unchanged ts-j ts ${ }^{\prime}$
have neq-j-i: $\mathrm{j} \neq \mathrm{i}$
by auto
from others-same [rule-format, OF j-bound neq-j-i] ts-j
have u-ts-j: u-ts! $\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by simp
from leq j -bound have j -bound': $\mathrm{j}<$ length u -ts by simp
from leq i-bound have i-bound': i < length u-ts by $\operatorname{simp}$
from direct-computation.Program [OF j-bound ${ }^{\prime}$ u-ts-j prog-step]
have ustep: (u-ts, u-m, u-shared) $\Rightarrow_{\mathrm{d}}\left(\mathrm{u}-\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is @ is', $\left.\left.\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$, u-m,
u-shared). show ?thesis
apply -
apply (rule exI)
apply (rule exI)
apply (rule exI)
apply (rule conjI)
apply (rule ustep)
using neq-j-i others-same u-m-other $u$-shared j-bound leq ts-j
apply (auto simp add: nth-list-update $\mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}$ )
done
next
case (Memop j p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R}$ is $\left.\vartheta^{\prime} \operatorname{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]$ and
j-bound: $\mathrm{j}<$ length ts and
$\mathrm{ts}-\mathrm{j}: \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
mem-step: (is, $\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
by auto
from mem-step i-bound unchanged ts-j

```
have neq-j-i: j\not=i
    by cases (auto simp add: ts')
from others-same [rule-format, OF j-bound neq-j-i] ts-j
have u-ts-j: u-ts!j = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    by simp
from leq j-bound have j-bound': j < length u-ts
    by simp
from leq i-bound have i-bound': i < length u-ts
    by simp
from safe-delayedE [OF safe-delayed-undo j-bound' u-ts-j]
have safe-j: map owned u-ts,map released u-ts,j }\vdash(\mathrm{ (is, }\vartheta\mathrm{ , u-m, }\mathcal{D},\mathcal{O}, u-shared) \sqrt{}{}
from simple-ownership-distinct [OF j-bound' i-bound' neq-j-i u-ts-j u-ts-i]
have owns-u-owns: }\mathcal{O}\capu\mathrm{ u-owns ={} .
from mem-step
show ?thesis
proof (cases)
    case (Read volatile a t)
    then obtain
    is: is = Read volatile a t # is' and
    \vartheta}:\mp@subsup{\vartheta}{}{\prime}=\vartheta(\textrm{t}\mapsto\textrm{m}a)\mathrm{ and
    sb': sb'=sb and
    m':m'=m and
    \mathcal{D}
    \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \text { and}
    \mathcal{R}
    \mathcal{S}
    by auto
    note eqs' = \vartheta' sb' m' D}\mp@subsup{D}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from safe-j [simplified is]
obtain
    access-cond: a }\in\mathcal{O}\vee a \in read-only u-shared V
                                    (volatile }\wedge a \in dom u-shared)
    and
    clean: volatile }\longrightarrow\neg\mathcal{D
    by cases auto
    have u-m-a-eq: u-m a = m a
    proof (cases a }\in\mathrm{ u-owns)
    case True
    with simple-ownership-distinct [OF j-bound' i-bound' neq-j-i u-ts-j u-ts-i]
    have a }\not\in\mathcal{O}\mathrm{ by auto
    with access-cond read-only-dom [of u-shared] have a }\in\mathrm{ dom u-shared
        by auto
    from u-m-shared [rule-format, OF True this]
    show ?thesis .
    next
    case False
    from u-m-other [rule-format, OF this]
```

show ?thesis .
qed
note Read $^{\prime}=$ direct-memop-step.Read [of volatile at is ${ }^{\prime} \vartheta$ sb u-m $\mathcal{D} \mathcal{O} \mathcal{R}$ u-shared]
from direct-computation.Memop [OF j-bound' $\mathbf{u}$-ts-j, simplified is, OF Read' $]$
have ustep: ( $\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared $) \Rightarrow_{\mathrm{d}}\left(\mathrm{u}-\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}\right.\right.\right.$, is $\left.\left.{ }^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}), \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$,
$\mathrm{u}-\mathrm{m}$, u-shared).
show ?thesis
apply -
apply (rule exI)
apply (rule exI)
apply (rule exI)
apply (rule conjI)
apply (rule ustep)
using neq- j -i others-same u -m-other u -shared j -bound leq ts- j
by (auto simp add: nth-list-update ts' eqs' $\mathrm{u}-\mathrm{m}-\mathrm{a}-\mathrm{eq}$ )
next
case (WriteNonVolatile a D f A L R W)
then obtain
is: is = Write False a (D, f) A L R W \# is' and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f} \vartheta)$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]
obtain
owned: $\mathrm{a} \in \mathcal{O}$ and unshared: $\mathrm{a} \notin$ dom $u$-shared
by cases auto
from simple-ownership-distinct [OF j-bound' i-bound' neq-j-i u-ts-j u-ts-i] owned
have a-unowned-i: a $\notin u$-owns
by auto
note Write ${ }^{\prime}=$ direct-memop-step.WriteNonVolatile $\left[\right.$ of a D f A L R W is ${ }^{\prime} \vartheta$ sb u-m $\mathcal{D} \mathcal{O} \mathcal{R}$ u-shared]
from direct-computation.Memop [OF j-bound' u -ts-j, simplified is, OF Write ${ }^{\prime}$ ]
have ustep: $(\mathrm{u}$-ts, u -m, u-shared $) \Rightarrow_{\mathrm{d}}\left(\mathrm{u}\right.$-ts $\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$, $\mathrm{u}-\mathrm{m}(\mathrm{a}:=\mathrm{f}$ ๆ), u-shared).
show ?thesis
apply -
apply (rule exI)
apply (rule exI)
apply (rule exI)
apply (rule conjI)
apply (rule ustep)
using neq-j-i others-same u-m-other u-shared j-bound leq ts-j a-unowned-i

```
    apply (auto simp add: nth-list-update ts' eqs')
    done
next
    case (WriteVolatile a D f A L R W)
    then obtain
        is: is = Write True a (D, f) A L R W # is' and
        \vartheta':}\mp@subsup{\vartheta}{}{\prime}=\vartheta\mathrm{ and
        sb': sb'=sb and
        m': m'=m(a:=f \vartheta) and
        \mathcal{D}
        \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \cup \mathrm { A } - \mathrm { R } \text { and}
        \mathcal{R}
        \mathcal{S}
        by auto
    note eqs'=}\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{m}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from safe-j [simplified is]
    obtain
```

        a-unowned-others: \(\forall \mathrm{k}<\) length u -ts. \(\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin\) (map owned u -ts! \(\mathrm{k} \cup\) dom (map
    released $u-t s!k)$ ) and
$\mathrm{A}: \mathrm{A} \subseteq$ dom u-shared $\cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}$
$=\{ \}$ and
A-unowned-others: $\forall \mathrm{k}<$ length $u$-ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned u-ts!k $\cup$ dom (map
released $\mathrm{u}-\mathrm{ts}!\mathrm{k}))=\{ \}$ and
a-not-ro: a $\notin$ read-only u-shared
by cases auto
note Write ${ }^{\prime}=$ direct-memop-step.WriteVolatile $[$ of a D f A L R W is' $\vartheta$ sb u-m $\mathcal{D} \mathcal{O}$
$\mathcal{R}$ u-shared]
from direct-computation.Memop [OF j-bound' u -ts-j, simplified is, OF Write' ${ }^{\prime}$ ]
have ustep: ( $u$-ts, $u-\mathrm{m}, \mathrm{u}$-shared) $\Rightarrow_{\mathrm{d}}$
(u-ts[j $:=\left(p\right.$, is ${ }^{\prime}, \vartheta$, sb, True, $\mathcal{O} \cup A-R$, Map.empty)], u-m (a $\left.:=\mathrm{f} \vartheta\right)$,
u-shared $\left.\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
from A-unowned-others [rule-format, OF i-bound' neq-j-i] u-ts-i i-bound'
have A -u-owns: $\mathrm{A} \cap$ u-owns $=\{ \}$ by auto
\{
fix a
assume a-u-owns: $\mathrm{a} \in \mathrm{u}$-owns
have (u-shared $\oplus_{W} R \ominus_{A} L$ ) $a=u$-shared $a$
using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
\}
note $u$-owned-shared $=$ this
from a-unowned-others [rule-format, OF i-bound' neq-j-i] u-ts-i i-bound' have
a-u-owns: a $\notin$ u-owns by auto
\{
fix a
assume a-u-owns: a $\notin u$-owns
assume a-u-owns-orig: a $\notin$ owned (ts!i)

```
    from u-shared [rule-format, OF a-u-owns a-u-owns-orig]
    have (u-shared \(\left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}\)
    using R-owns A-R L-A A-u-owns owns-u-owns
        by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
    \}
    note \(u\)-unowned-shared \(=\) this
    \{
        fix a
        assume a-u-owns: \(a \in u-o w n s\)
        have \(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\mathcal{S} \mathrm{a}\)
        using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
        by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
    \}
note \(\mathcal{S}^{\prime}\)-shared \(=\) this
show ?thesis
    apply -
    apply (rule exI)
    apply (rule exI)
    apply (rule exI)
    apply (rule conjI)
    apply (rule ustep)
            using neq-j-i others-same u-m-other u-shared j-bound leq ts-j u-owned-shared
a-u-owns u-unowned-shared \(\mathcal{S}^{\prime}\)-shared
    apply (auto simp add: nth-list-update ts' eqs')
        done
    next
    case Fence
    then obtain
            is: is = Fence \# is \({ }^{\prime}\) and
            \(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
            \(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
            \(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
            \(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
            \(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
            \(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
            \(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\)
            by auto
            note eqs \({ }^{\prime}=\mathcal{\vartheta}^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
            note Fence \(^{\prime}=\) direct-memop-step.Fence \(\left[\right.\) of is \({ }^{\prime} \vartheta\) sb u-m \(\mathcal{D} \mathcal{O} \mathcal{R}\) u-shared]
            from direct-computation.Memop [OF j-bound' \(u\)-ts-j, simplified is, OF Fence \({ }^{\prime}\) ]
                            have ustep: (u-ts, u-m, u-shared) \(\Rightarrow_{\mathrm{d}}\left(\mathrm{u}-\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta\right.\right.\right.\), sb, False, \(\mathcal{O}\), Map.empty)],
u-m, u-shared).
                            show ?thesis
        apply -
        apply (rule exI)
        apply (rule exI)
        apply (rule exI)
        apply (rule conjI)
```

```
apply (rule ustep)
using neq-j-i others-same \(u\)-m-other \(u\)-shared j-bound leq ts-j
by (auto simp add: nth-list-update ts \({ }^{\prime}\) eqs \({ }^{\prime}\) )
next
case (RMWReadOnly cond t a D f ret A L R W)
then obtain
is: is \(=\) RMW a t \((\mathrm{D}, \mathrm{f})\) cond ret \(\mathrm{ALRW} \#\) is' and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
\(m^{\prime}: m^{\prime}=m\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
cond: \(\neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a \())\)
by auto
note eqs \({ }^{\prime}=\mathcal{\vartheta}^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
from safe-j [simplified is] owns-u-owns u-ts-i i-bound' neq-j-i
obtain
access-cond: a \(\notin \mathrm{u}\)-owns \(\vee(\mathrm{a} \in \operatorname{dom} u\)-shared \(\wedge \mathrm{a} \in \mathrm{u}\)-owns \()\)
by cases auto
from u-m-other u-m-shared access-cond
have \(u-m\) - \(a\)-eq: \(u-m a=m a\)
by auto
from cond \(\mathrm{u}-\mathrm{m}-\mathrm{a}-\mathrm{eq}\) have cond \({ }^{\prime}: \neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}))\)
by auto
note RMWReadOnly \({ }^{\prime}=\) direct-memop-step.RMWReadOnly [of cond \(\vartheta\) t u-m a D f
ret \(\mathrm{A} L \mathrm{R} \mathrm{W}\) is' sb \(\mathcal{D} \mathcal{O} \mathcal{R}\) u-shared,
    OF cond \({ }^{\prime}\)
                                    from direct-computation.Memop \(\left[\mathrm{OF}\right.\) j-bound \({ }^{\prime} \mathrm{u}\)-ts-j, simplified is, OF
RMWReadOnly \({ }^{\prime}\) ]
    have ustep: (u-ts, u-m, u-shared) \(\Rightarrow_{\mathrm{d}}\left(\mathrm{u}-\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}{ }^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}), \mathrm{sb}\right.\right.\right.\), False, \(\mathcal{O}\),
Map.empty)], u-m, u-shared).
    show ?thesis
        apply -
    apply (rule exI)
    apply (rule exI)
    apply (rule exI)
    apply (rule conjI)
    apply (rule ustep)
    using neq-j-i others-same \(u\)-m-other \(u\)-shared j-bound leq ts-j
    by (auto simp add: nth-list-update ts' eqs' \({ }^{\prime}\) u-m-a-eq)
next
    case (RMWWrite cond t a D f ret A L R W)
    then obtain
    is: is \(=\) RMW a t ( \(\mathrm{D}, \mathrm{f}\) ) cond ret A L R W \# is' and
    \(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \mathrm{a})(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} a)))\) ) and
    \(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
    \(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} a)))\) and
```

$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=$ False and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ Map.empty and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{W} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ and
cond: cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a) $)$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is] owns-u-owns u-ts-i i-bound ${ }^{\prime}$ neq-j-i
obtain
access-cond: $\mathrm{a} \notin \mathrm{u}$-owns $\vee$ ( $\mathrm{a} \in \operatorname{dom} \mathrm{u}$-shared $\wedge \mathrm{a} \in \mathrm{u}$-owns)
by cases auto
from u-m-other u-m-shared access-cond
have u-m-a-eq: u-m a $=\mathrm{m}$ a
by auto
from cond u-m-a-eq have cond': cond $(\vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}))$
by auto
from safe-j [simplified is] cond ${ }^{\prime}$
obtain
a-unowned-others: $\forall \mathrm{k}<$ length u -ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin$ (map owned u -ts! $\mathrm{k} \cup$ dom (map released u-ts!k)) and

A: $\mathrm{A} \subseteq \operatorname{dom}$ u-shared $\cup \mathcal{O}$ and L-A: $\mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}$ $=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length u -ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned u-ts! $\mathrm{k} \cup$ dom (map released u-ts! k$)$ ) $=\{ \}$ and
a-not-ro: a $\notin$ read-only u-shared
by cases auto
note $\mathrm{Write}^{\prime}=$ direct-memop-step.RMWWrite [of cond $\vartheta \mathrm{t} u-\mathrm{m}$ a D fret A L R W is ${ }^{\prime}$ $\operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R} u$-shared,

OF cond]
from direct-computation.Memop [OF j-bound ${ }^{\prime}$ u-ts-j, simplified is, OF Write ${ }^{\prime}$ ]
have ustep: ( u -ts, u -m, u-shared) $\Rightarrow_{\mathrm{d}}$

$$
\left(\mathrm{u}-\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{u}-\mathrm{m} \mathrm{a})(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a})))) \text {, sb, False, } \mathcal{O} \cup \mathrm{A}-\right.\right.\right.
$$

R, Map.empty $)]$, $u-m(a:=f(\vartheta(t \mapsto u-m a)))$,
u-shared $\left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
from A-unowned-others [rule-format, OF i-bound' neq-j-i] u-ts-i i-bound ${ }^{\prime}$
have A -u-owns: $\mathrm{A} \cap \mathrm{u}$-owns $=\{ \}$ by auto
\{
fix a
assume a-u-owns: $\mathrm{a} \in \mathrm{u}$-owns
have ( $u$-shared $\oplus_{W} R \ominus_{A} L$ ) $a=u$-shared $a$
using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
\}
note $u$-owned-shared $=$ this
from a-unowned-others [rule-format, OF i-bound' neq-j-i] u-ts-i i-bound' have a-u-owns: a $\notin$ u-owns by auto

```
    {
        fix a
        assume a-u-owns: a & u-owns
        assume a-u-owns-orig: a & owned (ts!i)
        from u-shared [rule-format, OF a-u-owns this]
        have (u-shared }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a
        using R-owns A-R L-A A-u-owns owns-u-owns
        by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
    }
note u-unowned-shared = this
    {
        fix a
        assume a-u-owns: a }\inu\mathrm{ u-owns
        have (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{a}
        using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
        by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
    }
    note S'-shared = this
show ?thesis
    apply -
    apply (rule exI)
    apply (rule exI)
    apply (rule exI)
    apply (rule conjI)
    apply (rule ustep)
            using neq-j-i others-same u-m-other u-shared j-bound leq ts-j u-owned-shared
a-u-owns u-unowned-shared S'}\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ -shared
    apply (auto simp add: nth-list-update ts' eqs')
    done
    next
    case (Ghost A L R W)
    then obtain
    is: is = Ghost A L R W # is' and
    \vartheta ^ { \prime } : \vartheta ^ { \prime } = \vartheta \text { and}
    sb': sb'=sb and
    m': m'=m and
    \mathcal{D}
    \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \cup \mathrm { A } - \mathrm { R } \text { and}
    \mathcal{R}
    \mathcal{S}
    by auto
    note eqs' = \vartheta'sb' m' }\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from safe-j [simplified is]
    obtain
    A: A \subseteq dom u-shared \cup\mathcal{O}\mathrm{ and L-A: L }\subseteqA\mathrm{ and R-owns: R }\subseteq\mathcal{O}\mathrm{ and A-R: A }\cap\textrm{R}
= {} and
    A-unowned-others: }\forall\textrm{k}< length u-ts. \textrm{j}\not=\textrm{k}\longrightarrow\textrm{A}\cap\mathrm{ (map owned u-ts!k }\cup\mathrm{ dom (map
released u-ts!k)) = {}
```

by cases auto
note Ghost $^{\prime}=$ direct-memop-step.Ghost [of A L R W is' $\vartheta$ sb u-m $\mathcal{D} \mathcal{O} \mathcal{R}$ u-shared] from direct-computation.Memop [OF j-bound' $u$-ts-j, simplified is, OF Ghost ${ }^{\prime}$ ]
have ustep: (u-ts, u-m, u-shared) $\Rightarrow_{\mathrm{d}}$
(u-ts $\left[\mathrm{j}:=\left(\mathrm{p}\right.\right.$, is $^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O} \cup \mathrm{A}-\mathrm{R}$, augment-rels (dom u-shared) $\mathrm{R} \mathcal{R}$
)], u-m,

$$
\text { u-shared } \left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right) .
$$

from A-unowned-others [rule-format, OF i-bound' neq-j-i] u-ts-i i-bound ${ }^{\prime}$
have A -u-owns: $\mathrm{A} \cap \mathrm{u}$-owns $=\{ \}$ by auto
\{
fix a
assume a-u-owns: a $\in$ u-owns
have (u-shared $\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ) $\mathrm{a}=\mathrm{u}$-shared a
using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
\}
note $u$-owned-shared $=$ this
\{
fix a
assume a-u-owns: a $\notin u$-owns
assume a $\notin$ owned (ts!i)
from u-shared [rule-format, OF a-u-owns this]
have (u-shared $\left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$
using R-owns A-R L-A A-u-owns owns-u-owns
by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
\}
note $u$-unowned-shared $=$ this
\{
fix a
assume a-u-owns: $a \in u$-owns
have $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\mathcal{S} \mathrm{a}$
using R-owns A-R L-A A-u-owns owns-u-owns a-u-owns
by (auto simp add: restrict-shared-def augment-shared-def split: option.splits)
\}
note $\mathcal{S}^{\prime}$-shared $=$ this
from dist-ts
interpret dist-ts-inter: simple-ownership-distinct ts .
from dist-ts-inter.simple-ownership-distinct [OF j-bound i-bound neq-j-i ts-j]
have $\mathcal{O} \cap$ owned (ts!i) $=\{ \}$
apply (cases ts!i)
apply fastforce+
done
with simple-ownership-distinct [OF j-bound' i-bound' neq-j-i u-ts-j u-ts-i] R-owns u-shared
have augment-eq: augment-rels (dom u-shared) $\mathrm{R} \mathcal{R}=\operatorname{augment}-\mathrm{rels}(\operatorname{dom} \mathcal{S}) \mathrm{R} \mathcal{R}$
apply -
apply (rule ext)
apply (fastforce simp add: augment-rels-def split: option.splits simp add: domIff)
done
show ?thesis
apply -
apply (rule exI)
apply (rule exI)
apply (rule exI)
apply (rule conjI)
apply (rule ustep)
using neq-j-i others-same $u$-m-other $u$-shared $j$-bound leq ts-j $u$-owned-shared u-unowned-shared $\mathcal{S}^{\prime}$-shared
apply (auto simp add: nth-list-update ts' eqs' augment-eq)
done
qed
next
case (StoreBuffer - p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathrm{sb}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}$ )
hence False
by (auto simp add: empty-storebuffer-step-def)
thus ?thesis ..
qed
qed
theorem (in program) safe-step-preserves-simple-ownership-distinct:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{tss}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
assumes safe: safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
assumes dist: simple-ownership-distinct ts
shows simple-ownership-distinct ts'
proof -
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda$ p $\mathrm{p}^{\prime}$ is sb . sb .
from dist interpret simple-ownership-distinct ts .
from step
show ?thesis
proof (cases)
case (Program jp is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is')
then obtain
ts': ts ${ }^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}\right.\right.$, is $\left.\left.@ i s^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
j -bound: $\mathrm{j}<$ length ts and
ts-j: ts! $j=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
prog-step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $)$
by auto

```
from simple-ownership-distinct [OF j-bound - - ts-j]
show simple-ownership-distinct ts'
    apply (simp only: ts')
    apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
    apply force
    done
next
case (Memop j p is \vartheta sb \mathcal{D O R is'}\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime})
then obtain
    ts': ts' = ts[j:=(p,is',\mp@subsup{\vartheta}{}{\prime},\mp@subsup{\textrm{sb}}{}{\prime},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mathcal{R}})]\mathrm{ ] and
    j-bound: j < length ts and
    ts-j: ts!j = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R}) and
    mem-step: (is, \vartheta, sb, m, \mathcal{D, \mathcal{O},\mathcal{R},\mathcal{S})->(is', \vartheta', sb',m', D}
    by auto
from safe-delayedE [OF safe j-bound ts-j]
have safe-j: map owned ts,map released ts,j}\vdash(\mathrm{ is, }\vartheta, m,\mathcal{D},\mathcal{O},\mathcal{S})\sqrt{}{
from mem-step
show ?thesis
proof (cases)
    case (Read volatile a t)
    then obtain
        is: is = Read volatile a t # is' and
        \vartheta':}\mp@subsup{\vartheta}{}{\prime}=\vartheta(\textrm{t}\mapsto\textrm{m}\mathrm{ a) and
        sb': sb'=sb and
        m': m'=m and
        \mathcal{D}
        \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \text { and}
        R}\mp@subsup{\mathcal{R}}{}{\prime}:\mp@subsup{\mathcal{R}}{}{\prime}=\mathcal{R}\mathrm{ and
        \mathcal{S}
        by auto
    note eqs' = 䀠 sb'm
    from simple-ownership-distinct [OF j-bound - - ts-j]
    show simple-ownership-distinct ts'
        apply (simp only: ts' eqs')
        apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
        apply force
        done
```

next
case (WriteNonVolatile a D f A L R W)
then obtain
is: is = Write False a (D, f) A L R W \# is' and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f} \vartheta)$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and

```
    \mathcal{O}
    \mathcal{R}}\mp@subsup{}{}{\prime}:\mp@subsup{\mathcal{R}}{}{\prime}=\mathcal{R}\mathrm{ and
    \mp@subsup{\mathcal{S}}{}{\prime}:}\mp@subsup{\mathcal{S}}{}{\prime}=\mathcal{S
    by auto
note eqs'= 诠sb' m
from simple-ownership-distinct [OF j-bound - - ts-j]
show simple-ownership-distinct ts'
    apply (simp only: ts' eqs')
    apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
    apply force
    done
```

next
case (WriteVolatile a D f A L R W)
then obtain
is: is = Write True a (D, f) A L R W \# is' and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f} \vartheta)$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=$ True and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ Map.empty and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{w}^{\mathrm{R}} \ominus_{\mathrm{A}} \mathrm{L}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]
obtain
a-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin$ (map owned ts!k $\cup$ dom (map
released ts!k)) and
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$
and
A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts! $\mathrm{k} \cup$ dom (map
released ts!k)) $=\{ \}$ and
a-not-ro: a $\notin$ read-only $\mathcal{S}$
by cases auto
from simple-ownership-distinct [OF j-bound - - ts-j] R-owns A-R A-unowned-others
show simple-ownership-distinct ts ${ }^{\prime}$
apply (simp only: ts' eqs')
apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
apply force
done
next
case Fence
then obtain
is: is = Fence \# is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=$ False and

```
        O}\mp@subsup{\mathcal{O}}{}{\prime}:\mp@subsup{\mathcal{O}}{}{\prime}=\mathcal{O}\mathrm{ and
        \mathcal{R}
        \mathcal{S}
        by auto
    note eqs' = \vartheta '
    from simple-ownership-distinct [OF j-bound -- ts-j]
    show simple-ownership-distinct ts'
    apply (simp only: ts' eqs')
    apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
    apply force
    done
next
    case (RMWReadOnly cond t a D f ret A L R W)
    then obtain
    is: is = RMW a t (D, f) cond ret A L R W # is' and
    \vartheta':}\mp@subsup{\vartheta}{}{\prime}=\vartheta(\textrm{t}\mapsto\textrm{m}\mathrm{ a) and
    sb': sb'=sb and
    m': m'=m and
    \mathcal{D}}\mp@subsup{}{}{\prime}:\mp@subsup{\mathcal{D}}{}{\prime}=\mathrm{ False and
    \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \text { and}
    \mathcal{R}
    \mathcal{S}}:\mp@subsup{\mathcal{S}}{}{\prime}=\mathcal{S}\mathrm{ and
    cond: \neg cond (\vartheta(t \mapsto m a))
    by auto
    note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from simple-ownership-distinct [OF j-bound - - ts-j]
    show simple-ownership-distinct ts'
        apply (simp only: ts' eqs')
        apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
        apply force
        done
next
    case (RMWWrite cond t a D f ret A L R W)
    then obtain
        is: is = RMW a t (D,f) cond ret A L R W # is' and
        \vartheta':}\mp@subsup{\vartheta}{}{\prime}=\vartheta(\textrm{t}\mapsto\textrm{ret}(\textrm{m}a)(\textrm{f}(\vartheta(\textrm{t}\mapsto\textrm{m}a))))\mathrm{ and
        sb': sb'=sb and
        m': m'=m(a := f(\vartheta(t\mapstoma))) and
        \mathcal{D}}\mp@subsup{\mathcal{'}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}=\mathrm{ False and
        \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \cup \mathrm { A } - \mathrm { R } \text { and}
        R}\mp@subsup{\mathcal{R}}{}{\prime}:\mp@subsup{\mathcal{R}}{}{\prime}=\mathrm{ Map.empty and
        \mathcal{S}
        cond: cond (\vartheta(t \mapsto m a))
        by auto
    note eqs'= \vartheta'sb' m
    from safe-j [simplified is] cond
    obtain
```

    a-unowned-others: \(\forall \mathrm{k}<\) length ts. \(\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin\) (map owned ts! \(\mathrm{k} \cup\) dom (map
    released ts!k)) and
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts! $\mathrm{k} \cup$ dom (map released $\mathrm{ts}!\mathrm{k}))=\{ \}$ and
a-not-ro: a $\notin$ read-only $\mathcal{S}$
by cases auto
from simple-ownership-distinct [OF j-bound - - ts-j] R-owns A-R A-unowned-others
show simple-ownership-distinct ts ${ }^{\prime}$
apply (simp only: ts ${ }^{\prime}$ eqs ${ }^{\prime}$ )
apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
apply force
done
next
case (Ghost A L R W)
then obtain
is: is = Ghost A L R W \# is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ augment-rels (dom $\mathcal{S}$ ) $\mathrm{R} \mathcal{R}$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{W}^{\mathrm{R}} \ominus_{\mathrm{A}} \mathrm{L}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]
obtain
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts! $\mathrm{k} \cup$ dom (map released ts!k)) $=\{ \}$
by cases auto
from simple-ownership-distinct [OF j-bound - - ts-j] R-owns A-R A-unowned-others
show simple-ownership-distinct ts ${ }^{\prime}$
apply (simp only: ts ${ }^{\prime}$ eqs')
apply (rule simple-ownership-distinct-nth-update [OF j-bound ts-j])
apply force
done
qed
next
case (StoreBuffer - p is $\left.\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R} \operatorname{sb}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
hence False
by (auto simp add: empty-storebuffer-step-def)
thus ?thesis ..
qed
qed
theorem (in program) safe-step-preserves-read-only-unowned:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
assumes safe: safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
assumes dist: simple-ownership-distinct ts
assumes ro-unowned: read-only-unowned $\mathcal{S}$ ts
shows read-only-unowned $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$
proof -
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb . sb .
from dist interpret simple-ownership-distinct ts .
from ro-unowned interpret read-only-unowned $\mathcal{S}$ ts .
from step
show ?thesis
proof (cases)
case (Program jp is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is')
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\mathrm{is}} \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$m^{\prime}: m^{\prime}=m$ and
j -bound: $\mathrm{j}<$ length ts and
ts-j: ts! $\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
prog-step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is')
by auto
from read-only-unowned [OF j-bound ts-j]
show read-only-unowned $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$
apply ( $\operatorname{simp}$ only: $\mathrm{ts}^{\prime} \mathcal{S}^{\prime}$ )
apply (rule read-only-unowned-nth-update [OF j-bound])
apply force
done
next
case (Memop j p is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R}$ is $\left.\vartheta^{\prime} \operatorname{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]$ and
j -bound: $\mathrm{j}<$ length ts and
$\mathrm{ts}-\mathrm{j}: \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
mem-step: (is, $\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow\left(\right.$ is $\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
by auto
from safe-delayedE [OF safe j-bound ts-j]
have safe-j: map owned ts, map released ts, $\mathfrak{j} \vdash($ is, $\vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$.
from mem-step
show ?thesis
proof (cases)
case (Read volatile a t)
then obtain
is: is $=$ Read volatile a $t \#$ is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a) and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and

```
    \mathcal{D}
    \mathcal{O}
    \mathcal{R}}\mp@subsup{}{}{\prime}:\mp@subsup{\mathcal{R}}{}{\prime}=\mathcal{R}\mathrm{ and
    \mathcal{S}
    by auto
note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
from read-only-unowned [OF j-bound ts-j]
show read-only-unowned }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
apply (simp only: ts' eqs')
apply (rule read-only-unowned-nth-update [OF j-bound])
apply force
done
next
    case (WriteNonVolatile a D f A L R W)
    then obtain
        is: is = Write False a (D, f) A L R W # is' and
        \vartheta ^ { \prime } : \vartheta ^ { \prime } = \vartheta \text { and}
        sb': sb'=sb and
        m': m'=m(a:=f \vartheta) and
        \mathcal{D}
        \mathcal { O } ^ { \prime } : \mathcal { O } ^ { \prime } = \mathcal { O } \text { and}
        \mathcal{R}
        \mp@subsup{\mathcal{S}}{}{\prime}:\mp@subsup{\mathcal{S}}{}{\prime}=\mathcal{S}
        by auto
    note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from read-only-unowned [OF j-bound ts-j]
    show read-only-unowned }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
    apply (simp only: ts' eqs')
    apply (rule read-only-unowned-nth-update [OF j-bound])
    apply force
    done
next
    case (WriteVolatile a D f A L R W)
    then obtain
        is: is = Write True a (D, f) A L R W # is' and
        \vartheta ^ { \prime } : \vartheta ^ { \prime } = \vartheta \text { and}
        sb': sb'=sb and
        m': m'=m(a:=f \vartheta) and
        \mathcal{D}}\mp@subsup{}{\prime}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}=\mathrm{ True and
        \mathcal{O}
        \mathcal{R}
        \mathcal{S}
        by auto
    note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    from safe-j [simplified is]
    obtain
```

a-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin$ (map owned ts! $\mathrm{k} \cup$ dom (map released ts!k)) and
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts!k $\cup$ dom (map released ts! k$)$ ) $=\{ \}$ and
a-not-ro: a $\notin$ read-only $\mathcal{S}$
by cases auto
show read-only-unowned $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$
proof (unfold-locales)
fix i $\mathrm{p}_{\mathrm{i}} \mathrm{is}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \boldsymbol{\vartheta}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$
assume i-bound: $\mathrm{i}<$ length ts ${ }^{\prime}$
assume ts' i : $\mathrm{ts}!$ ! $=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
show $\mathcal{O}_{\mathrm{i}} \cap$ read-only $\mathcal{S}^{\prime}=\{ \}$
proof (cases $\mathrm{i}=\mathrm{j}$ )
case True
with read-only-unowned [OF j-bound ts-j] ts'-i A L-A R-owns A-R j-bound
show ?thesis
by (auto simp add: eqs' ts' read-only-def augment-shared-def restrict-shared-def domIff split: option.splits)
next
case False
from simple-ownership-distinct [OF j-bound - False [symmetric] ts-j] ts'-i i-bound j-bound False
have $\mathcal{O} \cap \mathcal{O}_{\mathrm{i}}=\{ \}$ by (fastforce simp add: ts ${ }^{\prime}$ )
with A L-A R-owns A-R j-bound A-unowned-others [rule-format, of i]
read-only-unowned [of i pi is $\vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}}$ ]
False i-bound ts'-i False
show ?thesis
by (force simp add: eqs ${ }^{\prime}$ ts ${ }^{\prime}$ read-only-def augment-shared-def restrict-shared-def domIff split: option.splits)
qed
qed
next
case Fence
then obtain
is: is = Fence \# is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=$ False and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ Map.empty and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from read-only-unowned [OF j-bound ts-j]
show read-only-unowned $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$

```
    apply ( \(\operatorname{simp}\) only: ts' eqs')
    apply (rule read-only-unowned-nth-update [OF j-bound])
    apply force
    done
next
case (RMWReadOnly cond t a D f ret A L R W)
then obtain
is: is \(=\) RMW at \((D, f)\) cond ret A L R W \# is \({ }^{\prime}\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
\(m^{\prime}: m^{\prime}=m\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
cond: \(\neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a \())\)
by auto
note eqs \({ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
from read-only-unowned [OF j-bound ts-j]
show read-only-unowned \(\mathcal{S}^{\prime}\) ts \({ }^{\prime}\)
apply (simp only: ts' eqs')
apply (rule read-only-unowned-nth-update [OF j-bound])
apply force
done
next
case (RMWWrite cond t a D f ret A L R W)
then obtain
is: is \(=\) RMW a t ( \(\mathrm{D}, \mathrm{f}\) ) cond ret A L R W \# is' and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} a)(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} a))))\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
\(m^{\prime}: m^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} a)))\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{W} R \ominus_{\mathrm{A}} \mathrm{L}\) and
cond: cond \((\vartheta(\mathrm{t} \mapsto \mathrm{m} a))\)
by auto
note eqs \({ }^{\prime}=\mathcal{\vartheta}^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
from safe-j [simplified is] cond
obtain
a-unowned-others: \(\forall \mathrm{k}<\) length ts. \(\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin\) (map owned ts! \(\mathrm{k} \cup\) dom (map
released ts!k)) and
\(\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}\) and \(\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}\) and R -owns: \(\mathrm{R} \subseteq \mathcal{O}\) and \(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\)
A-unowned-others: \(\forall \mathrm{k}<\) length ts. \(\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap\) (map owned ts!k \(\cup\) dom (map
released ts!k)) \(=\{ \}\) and
a-not-ro: a \(\notin\) read-only \(\mathcal{S}\)
by cases auto
show read-only-unowned \(\mathcal{S}^{\prime}\) ts \({ }^{\prime}\)
```

and

```
proof (unfold-locales)
```

fix i $p_{i}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$
assume i-bound: i $<$ length ts ${ }^{\prime}$
assume ts ${ }^{\prime}$ - $:$ ts ${ }^{\prime}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
show $\mathcal{O}_{\mathrm{i}} \cap$ read-only $\mathcal{S}^{\prime}=\{ \}$
proof (cases $\mathrm{i}=\mathrm{j}$ )
case True
with read-only-unowned [OF j-bound ts-j] ts ${ }^{\prime}$-i A L-A R-owns A-R j-bound show ?thesis
by (auto simp add: eqs' ts $^{\prime}$ read-only-def augment-shared-def restrict-shared-def
domIff split: option.splits)
next
case False
from simple-ownership-distinct [OF j-bound - False [symmetric] ts-j] ts'-i i-bound j-bound False
have $\mathcal{O} \cap \mathcal{O}_{\mathrm{i}}=\{ \}$
by (fastforce simp add: ts')
with A L-A R-owns A-R j-bound A-unowned-others [rule-format, of i ]
read-only-unowned [of i $\mathrm{p}_{\mathrm{i}}$ is $\vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}}$ ]
False i-bound ts'-i False
show ?thesis
by (force simp add: eqs ${ }^{\prime}$ ts ${ }^{\prime}$ read-only-def augment-shared-def restrict-shared-def
domIff split: option.splits)
qed
qed
next
case (Ghost A L R W)
then obtain
is: is $=$ Ghost A L R W \# is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ augment-rels (dom $\left.\mathcal{S}\right) \mathrm{R} \mathcal{R}$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{W} R \ominus_{\mathrm{A}} \mathrm{L}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]
obtain
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts!k $\cup$ dom (map released ts!k)) $=\{ \}$
by cases auto
show read-only-unowned $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$
proof (unfold-locales)
fix i $\mathrm{p}_{\mathrm{i}} \mathrm{is} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$
assume i-bound: i < length ts ${ }^{\prime}$
assume ts' ${ }^{\prime}$ : ts ${ }^{\prime}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
show $\mathcal{O}_{\mathrm{i}} \cap$ read-only $\mathcal{S}^{\prime}=\{ \}$
proof (cases $\mathrm{i}=\mathrm{j}$ )
case True
with read-only-unowned [OF j-bound ts-j] ts'-i A L-A R-owns A-R j-bound show ?thesis
by (auto simp add: eqs' $\mathrm{ts}^{\prime}$ read-only-def augment-shared-def restrict-shared-def
domIff split: option.splits)
next
case False
from simple-ownership-distinct [OF j-bound - False [symmetric] ts-j] ts'-i i-bound j-bound False
have $\mathcal{O} \cap \mathcal{O}_{\mathrm{i}}=\{ \}$
by (fastforce simp add: ts')
with A L-A R-owns A-R j-bound A-unowned-others [rule-format, of i]
read-only-unowned [of i $\mathrm{p}_{\mathrm{i}}$ is $\mathrm{\vartheta}_{\mathrm{i}} \operatorname{sb}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}}$ ]
False i-bound ts'-i False
show ?thesis
by (force simp add: eqs ${ }^{\prime}$ ts ${ }^{\prime}$ read-only-def augment-shared-def restrict-shared-def domIff split: option.splits)
qed
qed
qed
next
case (StoreBuffer - p is $\vartheta$ sb $\left.\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{sb}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
hence False
by (auto simp add: empty-storebuffer-step-def)
thus ?thesis ..
qed
qed
theorem (in program) safe-step-preserves-unowned-shared:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
assumes safe: safe-delayed (ts,m,S)
assumes dist: simple-ownership-distinct ts
assumes unowned-shared: unowned-shared $\mathcal{S}$ ts
shows unowned-shared $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$
proof -
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb . sb .
from dist interpret simple-ownership-distinct ts .
from unowned-shared interpret unowned-shared $\mathcal{S}$ ts .
from step
show ?thesis
proof (cases)
case (Program jp is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is')
then obtain
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}\right.\right.$, is $\left.\left.@ i s^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and

```
    S': S'=S and
    m': m'=m and
    j-bound: j < length ts and
    ts-j: ts!j = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\mathrm{ and}
    prog-step: \vartheta\vdash p }\mp@subsup{->}{\textrm{p}}{}(\mp@subsup{\textrm{p}}{}{\prime},\mathrm{ is')
    by auto
show unowned-shared }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
    apply (simp only: ts'}\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ )
    apply (rule unowned-shared-nth-update [OF j-bound ts-j])
    apply force
    done
next
    case (Memop j p is \vartheta sb \mathcal{D O}\mathcal{R}\mathrm{ is' }\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime})
then obtain
    ts': ts '}=\textrm{ts}[\textrm{j}:=(\textrm{p},\mp@subsup{\textrm{is}}{}{\prime},\mp@subsup{\vartheta}{}{\prime},\mp@subsup{\textrm{sb}}{}{\prime},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime})]\mathrm{ and
    j-bound: j < length ts and
    ts-j: ts!j = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R}) and
    mem-step: (is, \vartheta, sb, m,\mathcal{D},\mathcal{O},\mathcal{R},\mathcal{S})->(is',}\mp@subsup{\vartheta}{}{\prime},\mp@subsup{\textrm{sb}}{}{\prime},\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime}
    by auto
from safe-delayedE [OF safe j-bound ts-j]
have safe-j: map owned ts,map released ts,j}\vdash(is,\vartheta, m,\mathcal{D},\mathcal{O},\mathcal{S})\sqrt{}{
from mem-step
show ?thesis
proof (cases)
    case (Read volatile a t)
    then obtain
    is: is = Read volatile a t # is' and
    \vartheta}:\mp@subsup{\vartheta}{}{\prime}=\vartheta(\textrm{t}\mapsto\textrm{m a})\mathrm{ and
    sb': sb'=sb and
    m': m'=m and
    \mathcal{D}
    \mathcal{O}
    \mathcal{R}}\mp@subsup{}{}{\prime}:\mp@subsup{\mathcal{R}}{}{\prime}=\mathcal{R}\mathrm{ and
    \mathcal{S}
    by auto
    note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
    show unowned-shared }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
    apply (simp only: ts' eqs')
    apply (rule unowned-shared-nth-update [OF j-bound ts-j])
    apply force
    done
next
    case (WriteNonVolatile a D f A L R W)
    then obtain
    is: is = Write False a (D,f) A L R W # is' and
    \vartheta ^ { \prime } : \vartheta ^ { \prime } = \vartheta \text { and}
```

```
    sb': sb
    m': m'=m(a:=f \vartheta) and
    \mathcal{D}
    \mathcal{O}
    \mathcal{R}
    \mp@subsup{\mathcal{S}}{}{\prime}:\mp@subsup{\mathcal{S}}{}{\prime}=\mathcal{S}
    by auto
note eqs'}=\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime
show unowned-shared }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
    apply (simp only: ts' eqs')
    apply (rule unowned-shared-nth-update [OF j-bound ts-j])
    apply force
    done
```

next
case (WriteVolatile a D f A L R W)
then obtain
is: is $=$ Write True a $(\mathrm{D}, \mathrm{f}) \mathrm{ALR} \mathrm{W} \#$ is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$m^{\prime}: m^{\prime}=m(a:=f \vartheta)$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=$ True and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ Map.empty and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]
obtain
a-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin$ (map owned ts!k $\cup$ dom (map
released ts!k)) and
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$
and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts! $\mathrm{k} \cup$ dom (map released ts! k$))=\{ \}$ and
a-not-ro: a $\notin$ read-only $\mathcal{S}$
by cases auto
show unowned-shared $\mathcal{S}^{\prime}$ ts $^{\prime}$
apply (clarsimp simp add: unowned-shared-def')
using A R-owns L-A A-R A-unowned-others ts-j j-bound
apply (auto simp add: $\mathcal{S}^{\prime}$ ts ${ }^{\prime} \mathcal{O}^{\prime}$ )
apply (rule unowned-shared')
apply clarsimp
apply (drule-tac $x=i$ in spec)
apply (case-tac $\mathrm{i}=\mathrm{j}$ )
apply clarsimp

```
apply clarsimp
apply (drule-tac \(x=j\) in spec)
apply auto
done
next
    case Fence
    then obtain
    is: is = Fence \# is \({ }^{\prime}\) and
    \(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
    \(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
    \(m^{\prime}: m^{\prime}=m\) and
    \(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
    \(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
    \(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
    \(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\)
    by auto
    note eqs \({ }^{\prime}=\mathcal{\vartheta}^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
    show unowned-shared \(\mathcal{S}^{\prime}\) ts \({ }^{\prime}\)
    apply (simp only: ts \({ }^{\prime}\) eqs \({ }^{\prime}\) )
    apply (rule unowned-shared-nth-update [OF j-bound ts-j])
    apply force
    done
next
    case (RMWReadOnly cond t a D f ret A L R W)
    then obtain
        is: is \(=\) RMW a t \((\mathrm{D}, \mathrm{f})\) cond ret A L R W \# is' and
        \(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a) and
        \(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
        \(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
        \(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
        \(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
        \(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\) Map.empty and
        \(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
        cond: \(\neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a \())\)
        by auto
    note eqs \({ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\)
    show unowned-shared \(\mathcal{S}^{\prime}\) ts \({ }^{\prime}\)
    apply (simp only: ts' eqs')
    apply (rule unowned-shared-nth-update [OF j-bound ts-j])
    apply force
    done
next
    case (RMWWrite cond t a D f ret A L R W)
    then obtain
    is: is \(=\) RMW at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W \# is \({ }^{\prime}\) and
    \(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{ma})(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{ma}))))\) and
    \(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
    \(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})))\) and
    \(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\) False and
```

$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ Map.empty and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ and
cond: cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a $))$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is] cond
obtain
a-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{a} \notin$ (map owned ts!k $\cup$ dom (map released ts!k)) and
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts!k $\cup$ dom (map released ts!k) $=\{ \}$ and
a-not-ro: a $\notin$ read-only $\mathcal{S}$
by cases auto
show unowned-shared $\mathcal{S}^{\prime}$ ts $^{\prime}$
apply (clarsimp simp add: unowned-shared-def')
using A R-owns L-A A-R A-unowned-others ts-j j-bound
apply (auto simp add: $\mathcal{S}^{\prime} \operatorname{ts}^{\prime} \mathcal{O}^{\prime}$ )
apply (rule unowned-shared')
apply clarsimp
apply (drule-tac $x=i$ in spec)
apply (case-tac $\mathrm{i}=\mathrm{j}$ )
apply clarsimp
apply clarsimp
apply (drule-tac $x=j$ in spec)
apply auto
done
next
case (Ghost A L R W)
then obtain
is: is = Ghost A L R W \# is ${ }^{\prime}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=$ augment-rels (dom $\mathcal{S}$ ) $\mathrm{R} \mathcal{R}$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S} \oplus \mathrm{w}^{\mathrm{R}} \ominus_{\mathrm{A}} \mathrm{L}$
by auto
note eqs ${ }^{\prime}=\vartheta^{\prime} \mathrm{sb}^{\prime} \mathrm{m}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
from safe-j [simplified is]

## obtain

$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and

A-unowned-others: $\forall \mathrm{k}<$ length ts. $\mathrm{j} \neq \mathrm{k} \longrightarrow \mathrm{A} \cap$ (map owned ts! $\mathrm{k} \cup$ dom (map released ts!k) $=\{ \}$

```
            by cases auto
            show unowned-shared }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'
            apply (clarsimp simp add: unowned-shared-def')
            using A R-owns L-A A-R A-unowned-others ts-j j-bound
            apply (auto simp add: }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\mathrm{ ts}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mathrm{ )
            apply (rule unowned-shared')
            apply clarsimp
            apply (drule-tac x=i in spec)
            apply (case-tac i=j)
            apply clarsimp
            apply clarsimp
            apply (drule-tac }x=j\mathrm{ in spec)
            apply auto
            done
        qed
next
    case (StoreBuffer - p is \vartheta sb \mathcal{D O}\mathcal{R sb' }\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime})
    hence False
                by (auto simp add: empty-storebuffer-step-def)
    thus ?thesis ..
qed
qed
locale program-trace = program +
```



```
fixes n::nat - starting index
fixes k::nat - steps
assumes step: }\1.\textrm{l}<\textrm{k}\Longrightarrow\textrm{c}(\textrm{n}+\textrm{l})\not\mp@subsup{=>}{\textrm{d}}{\textrm{c}}(\textrm{n}+(\mathrm{ Suc l )}
abbreviation (in program)
trace \equiv program-trace program-step
lemma (in program) trace-0 [simp]: trace c n 0
apply (unfold-locales)
apply auto
done
lemma split-less-Suc: (\forallx<Suc k. P x ) = (P k ^ (\forallx<k. P x ) )
    apply rule
    apply clarsimp
    apply clarsimp
    apply (case-tac x = k)
    apply auto
    done
lemma split-le-Suc: (\forallx\leqSuc k. P x ) = (P (Suc k) ^( }\forall\textrm{x}\leq\textrm{k}.\textrm{P}x)
    apply rule
    apply clarsimp
    apply clarsimp
```

```
    apply (case-tac \(\mathrm{x}=\) Suc k )
    apply auto
    done
lemma (in program) steps-to-trace:
assumes steps: \(\mathrm{x} \Rightarrow_{\mathrm{d}}{ }^{*} \mathrm{y}\)
shows \(\exists \mathrm{ck}\). trace c \(0 \mathrm{k} \wedge \mathrm{c} 0=\mathrm{x} \wedge \mathrm{ck}=\mathrm{y}\)
using steps
proof (induct)
    case base
    thus ?case
    apply (rule-tac \(x=\lambda k . x\) in exI)
    apply (rule-tac \(x=0\) in exI)
    by (auto simp add: program-trace-def)
next
    case (step y z)
    have first: \(\mathrm{x} \Rightarrow_{\mathrm{d}}{ }^{*} \mathrm{y}\) by fact
    have last: \(\mathrm{y} \Rightarrow_{\mathrm{d}} \mathrm{z}\) by fact
    from step.hyps obtain ck where
        trace: trace c 0 k and c-0: c \(0=\mathrm{x}\) and c-k: c \(\mathrm{k}=\mathrm{y}\)
        by auto
    define \(\mathrm{c}^{\prime}\) where \(\mathrm{c}^{\prime}==\lambda\) i. (if \(\mathrm{i} \leq \mathrm{k}\) then c i else z )
    from trace last \(\mathrm{c}-\mathrm{k}\) have trace \(\mathrm{c}^{\prime} 0(\mathrm{k}+1)\)
        apply (clarsimp simp add: c'-def program-trace-def)
        apply (subgoal-tac l=k)
        apply (simp)
        apply (simp)
        done
    with c-0
    show ?case
        apply -
        apply (rule-tac \(x=c^{\prime}\) in exI)
        apply (rule-tac \(x=k+1\) in exI)
        apply (auto simp add: c'-def)
        done
qed
```

lemma (in program) trace-preserves-length-ts:
$\bigwedge \mathrm{l} \mathrm{x}$. trace $\mathrm{c} \mathrm{nk} \Longrightarrow \mathrm{l} \leq \mathrm{k} \Longrightarrow \mathrm{x} \leq \mathrm{k} \Longrightarrow$ length $(\mathrm{fst}(\mathrm{c}(\mathrm{n}+\mathrm{l}))$ ) $=$ length $($ fst $(\mathrm{c}(\mathrm{n}+$
$\mathrm{x})$ )
proof (induct k)
case 0
thus ?case by auto
next
case (Suc k)
then obtain trace-suc: trace c n (Suc k) and
l-suc: $\mathrm{l} \leq$ Suc k and
x -suc: $\mathrm{x} \leq$ Suc k
by simp
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$.

## from trace-suc obtain

trace-k: trace c n k and
last-step: $\mathrm{c}(\mathrm{n}+\mathrm{k}) \Rightarrow_{\mathrm{d}} \mathrm{c}(\mathrm{n}+($ Suc k$))$
by (clarsimp simp add: program-trace-def)
obtain ts $\mathcal{S} \mathrm{m}$ where c-k: c $(\mathrm{n}+\mathrm{k})=(\mathrm{ts}, \mathrm{m}, \mathcal{S})$ by (cases c $(\mathrm{n}+\mathrm{k}))$
obtain $\mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime}$ where c-suc-k: c $(\mathrm{n}+($ Suc k$))=\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$ by (cases c $(\mathrm{n}+$ (Suc k))
from direct-computation.step-preserves-length-ts [OF last-step [simplified c-k c-suc-k]]
c-k c-suc-k
have leq: length $($ fst $(\mathrm{c}(\mathrm{n}+\operatorname{Suc} \mathrm{k})))=\operatorname{length}(\operatorname{fst}(\mathrm{c}(\mathrm{n}+\mathrm{k})))$
by simp
show ?case
proof (cases l=Suck)
case True
note l-suc $=$ this
show ? thesis
proof (cases $x=$ Suc k)
case True with l-suc show ?thesis by simp
next
case False
with $x$-suc have $\mathrm{x} \leq \mathrm{k}$ by simp
from Suc.hyps [OF trace-k this, of $k$ ]
have length $($ fst $(c(n+x)))=$ length $(f s t(c(n+k)))$
by simp
with leq show ?thesis using l-suc by simp
qed
next
case False
with l-suc have l-k: $\mathrm{l} \leq \mathrm{k}$
by auto
show ?thesis
proof (cases $x=$ Suc k)
case True
from Suc.hyps [OF trace-k l-k, of k]
have length $($ fst $(c(n+1)))=$ length $(f s t(c(n+k)))$ by simp with leq True show ?thesis by simp
next
case False
with x -suc have $\mathrm{x} \leq \mathrm{k}$ by $\operatorname{simp}$
from Suc.hyps [OF trace-k l-k this]
show ?thesis by simp
qed
qed
qed
lemma (in program) trace-preserves-simple-ownership-distinct:

```
    assumes dist: simple-ownership-distinct (fst (c n))
    shows \l. trace c n k \Longrightarrow (\forallx < k. safe-delayed (c (n + x ) )) \Longrightarrow
        l }\leq\textrm{k}\Longrightarrow\mathrm{ simple-ownership-distinct (fst (c (n + l)))
proof (induct k)
    case 0 thus ?case using dist by auto
next
    case (Suc k)
    then obtain
    trace-suc: trace c n (Suc k) and
    safe-suc: }\forall\textrm{x}<\mathrm{ Suc k. safe-delayed (c (n + x)) and
    l-suc: l \leq Suc k
    by simp
```

    from trace-suc obtain
    trace-k: trace c \(\mathrm{n} k\) and
    last-step: \(\mathrm{c}(\mathrm{n}+\mathrm{k}) \Rightarrow_{\mathrm{d}} \mathrm{c}(\mathrm{n}+(\) Suc k\())\)
    by (clarsimp simp add: program-trace-def)
    obtain ts \(\mathcal{S} \mathrm{m}\) where c-k: c \((\mathrm{n}+\mathrm{k})=(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by (cases c \((\mathrm{n}+\mathrm{k}))\)
    obtain \(\mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime}\) where c-suc-k: c \((\mathrm{n}+(\) Suc k\())=\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\) by (cases c \((\mathrm{n}+\) (Suc
    k))
from safe-suc c-suc-k c-k
obtain
safe-up-k: $\forall \mathrm{x}<\mathrm{k}$. safe-delayed $(\mathrm{c}(\mathrm{n}+\mathrm{x})$ ) and
safe-k: safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
by (auto simp add: split-le-Suc)
from Suc.hyps [OF trace-k safe-up-k]
have hyp: $\forall \mathrm{l} \leq \mathrm{k}$. simple-ownership-distinct $($ fst $(\mathrm{c}(\mathrm{n}+\mathrm{l})))$
by simp
from Suc.hyps [OF trace-k safe-up-k, of k] c-k
have simple-ownership-distinct ts
by simp
from safe-step-preserves-simple-ownership-distinct [OF last-step[simplified c-k c-suc-k]
safe-k this]
have simple-ownership-distinct ts ${ }^{\prime}$.
then show ?case
using c-suc-k hyp l-suc
apply (cases l=Suc k)
apply (auto simp add: split-less-Suc)
done
qed
lemma (in program) trace-preserves-read-only-unowned:
assumes dist: simple-ownership-distinct (fst (c n))
assumes ro: read-only-unowned (snd (snd (c n))) (fst (c n))
shows $\bigwedge$ l. trace c n $\mathrm{k} \Longrightarrow(\forall \mathrm{x}<\mathrm{k}$. safe-delayed $(\mathrm{c}(\mathrm{n}+\mathrm{x}))) \Longrightarrow$
$\mathrm{l} \leq \mathrm{k} \Longrightarrow$ read-only-unowned $($ snd $(\operatorname{snd}(\mathrm{c}(\mathrm{n}+\mathrm{l}))))(\mathrm{fst}(\mathrm{c}(\mathrm{n}+\mathrm{l})))$

```
proof (induct k)
    case 0 thus ?case using ro by auto
next
    case (Suc k)
    then obtain
        trace-suc: trace c n (Suc k) and
        safe-suc: }\forall\textrm{x}<\mathrm{ Suc k. safe-delayed (c (n + x)) and
        l-suc: l }\leq\mathrm{ Suc k
        by simp
    from trace-suc obtain
        trace-k: trace c n k and
        last-step: c ( }\textrm{n}+\textrm{k})\mp@subsup{=>}{\textrm{d}}{\textrm{c}}(\textrm{n}+(\mathrm{ Suc k))
        by (clarsimp simp add: program-trace-def)
    obtain ts }\mathcal{S}\textrm{m}\mathrm{ where c-k: c (n + k) = (ts, m, S})\mathrm{ by (cases c (n + k))
    obtain ts' }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\textrm{m}}{}{\prime}\mathrm{ where c-suc-k: c (n + (Suc k)) = (ts', m
k)))
from safe-suc c-suc-k c-k
obtain
safe-up-k: }\forall\textrm{x}<\textrm{k}.\mathrm{ safe-delayed (c (n + x)) and
safe-k: safe-delayed (ts,m,S)
by (auto simp add: split-le-Suc)
from Suc.hyps [OF trace-k safe-up-k]
have hyp: }\forall\textrm{l}\leq\textrm{k}.\mathrm{ read-only-unowned (snd (snd (c (n + l)))) (fst (c (n + l)))
by simp
    from Suc.hyps [OF trace-k safe-up-k, of k] c-k
have ro': read-only-unowned }\mathcal{S}\mathrm{ ts
    by simp
    from trace-preserves-simple-ownership-distinct [where c=c and n=n, OF dist trace-k
safe-up-k, of k] c-k
    have dist': simple-ownership-distinct ts by simp
    from safe-step-preserves-read-only-unowned [OF last-step[simplified c-k c-suc-k] safe-k
dist' ro']
    have read-only-unowned }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts'.
    then show ?case
    using c-suc-k hyp l-suc
        apply (cases l=Suc k)
        apply (auto simp add: split-less-Suc)
        done
qed
lemma (in program) trace-preserves-unowned-shared:
    assumes dist: simple-ownership-distinct (fst (c n))
    assumes ro: unowned-shared (snd (snd (c n))) (fst (c n))
    shows \l. trace c n k \Longrightarrow (\forallx < k. safe-delayed (c (n + x ))) \Longrightarrow
```

```
    l sk\Longrightarrow unowned-shared (snd (snd (c (n + l)))) (fst (c (n + l)))
proof (induct k)
    case 0 thus ?case using ro by auto
next
    case (Suc k)
    then obtain
        trace-suc: trace c n (Suc k) and
        safe-suc: }\forall\textrm{x}<\mathrm{ Suc k. safe-delayed (c (n + x)) and
        l-suc: l }\leq\mathrm{ Suc k
        by simp
```

    from trace-suc obtain
        trace-k: trace c n k and
        last-step: \(\mathrm{c}(\mathrm{n}+\mathrm{k}) \Rightarrow_{\mathrm{d}} \mathrm{c}(\mathrm{n}+(\) Suc k\())\)
        by (clarsimp simp add: program-trace-def)
    obtain ts \(\mathcal{S} \mathrm{m}\) where c-k: c \((\mathrm{n}+\mathrm{k})=(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by (cases c \((\mathrm{n}+\mathrm{k}))\)
    obtain \(\mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime}\) where c-suc-k: c \((\mathrm{n}+(\) Suc k\())=\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}\right)\) by (cases c \((\mathrm{n}+(\) Suc
    k))
from safe-suc c-suc-k c-k
obtain
safe-up-k: $\forall \mathrm{x}<\mathrm{k}$. safe-delayed $(\mathrm{c}(\mathrm{n}+\mathrm{x}))$ and
safe-k: safe-delayed (ts,m,S)
by (auto simp add: split-le-Suc)
from Suc.hyps [OF trace-k safe-up-k]
have hyp: $\forall \mathrm{l} \leq \mathrm{k}$. unowned-shared $(\operatorname{snd}(\operatorname{snd}(\mathrm{c}(\mathrm{n}+\mathrm{l}))))(\mathrm{fst}(\mathrm{c}(\mathrm{n}+\mathrm{l})))$
by simp
from Suc.hyps [OF trace-k safe-up-k, of k] c-k
have ro': unowned-shared $\mathcal{S}$ ts
by $\operatorname{simp}$
from trace-preserves-simple-ownership-distinct [where $\mathrm{c}=\mathrm{c}$ and $\mathrm{n}=\mathrm{n}$, OF dist trace- k
safe-up-k, of k] c-k
have dist': simple-ownership-distinct ts by simp
from safe-step-preserves-unowned-shared [OF last-step[simplified c-k c-suc-k] safe-k dist ${ }^{\prime}$
ro]
have unowned-shared $\mathcal{S}^{\prime}$ ts ${ }^{\prime}$.
then show ?case
using c-suc-k hyp l-suc
apply (cases l=Suc k)
apply (auto simp add: split-less-Suc)
done
qed
theorem (in program-progress) undo-local-steps:
assumes steps: trace c n k
assumes c-n: c $\mathrm{n}=(\mathrm{ts}, \mathrm{m}, \mathcal{S})$

assumes safe: safe-delayed (u-ts, u-m, u-shared)
assumes leq: length u-ts $=$ length ts
assumes i-bound: i < length ts
assumes others-same: $\forall \mathrm{j}<$ length ts. $\mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{u}-\mathrm{ts}!\mathrm{j}=\mathrm{ts}!\mathrm{j}$
assumes u-ts-i: u-ts! $i=(u-p, u-i s, u-t m p s, u-s b, u-d i r t y, u-o w n s, u-r e l s)$
assumes u-m-other: $\forall \mathrm{a} . \mathrm{a} \notin \mathrm{u}$-owns $\longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}$ a
assumes u-m-shared: $\forall \mathrm{a}$. $\mathrm{a} \in \mathrm{u}$-owns $\longrightarrow \mathrm{a} \in$ dom u-shared $\longrightarrow \mathrm{u}-\mathrm{ma}=\mathrm{m} \mathrm{a}$
assumes u-shared: $\forall \mathrm{a}$. a $\notin \mathrm{u}$-owns $\longrightarrow \mathrm{a} \notin$ owned $(\mathrm{ts}!\mathrm{i}) \longrightarrow$ u-shared $\mathrm{a}=\mathcal{S}$ a
assumes dist: simple-ownership-distinct u-ts
assumes dist-ts: simple-ownership-distinct ts
assumes safe-orig: $\forall \mathrm{x} . \mathrm{x}<\mathrm{k} \longrightarrow$ safe-delayed $(\mathrm{c}(\mathrm{n}+\mathrm{x})$ )
shows $\exists \mathrm{c}^{\prime} \mathrm{l} . \mathrm{l} \leq \mathrm{k} \wedge$ trace $\mathrm{c}^{\prime} \mathrm{n} l \wedge$

$$
\begin{aligned}
& \mathrm{c}^{\prime} \mathrm{n}=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u} \text {-shared }) \wedge \\
& \left(\forall \mathrm{x} \leq \mathrm{l} . \text { length }\left(\text { fst }\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})\right)\right)=\text { length }(\text { fst }(\mathrm{c}(\mathrm{n}+\mathrm{x})))\right) \wedge \\
& \left(\forall \mathrm{x}<\text { l. safe-delayed }\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})\right)\right) \wedge \\
& \left(\mathrm{l}<\mathrm{k} \longrightarrow \neg \text { safe-delayed }\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{l})\right)\right) \wedge
\end{aligned}
$$

$\left(\forall \mathrm{x} \leq 1 . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}} \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime} \cdot \mathrm{c}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow \mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})=\right.$ $\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{i}=\mathrm{u}-\mathrm{ts}!\mathrm{i} \wedge$
$\left(\forall \mathrm{a} \in\right.$ u-owns. $\mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{u}$-shared a$) \wedge$
$\left(\forall \mathrm{a} \in\right.$ u-owns. $\left.\mathcal{S}_{\times} \mathrm{a}=\mathcal{S} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in \mathrm{u}\right.$-owns. $\left.\mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{u}-\mathrm{m} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in \mathrm{u}\right.$-owns. $\left.\left.\mathrm{m}_{\mathrm{x}} \mathrm{a}=\mathrm{m} \mathrm{a}\right)\right) \wedge$
$\left(\forall \mathrm{x} \leq 1 . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}} \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime} \cdot \mathrm{c}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow \mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})=\right.$ $\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\left(\forall \mathrm{j}<\right.$ length $\left.\mathrm{ts}_{\mathrm{x}} \cdot \mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathrm{j}=\mathrm{ts}_{\mathrm{x}}!\mathrm{j}\right) \wedge$
$\left(\forall\right.$ a. a $\notin$ u-owns $\longrightarrow \mathrm{a} \notin$ owned $\left.(\mathrm{ts}!\mathrm{i}) \longrightarrow \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{x}} \mathrm{a}\right) \wedge$
$\left(\forall\right.$ a. a $\notin \mathrm{u}$-owns $\left.\left.\longrightarrow \mathrm{m}_{\mathrm{x}}^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{x}} \mathrm{a}\right)\right)$
using steps unchanged safe-orig
proof (induct k)
case 0
show ?case
apply (rule-tac $\mathrm{x}=\lambda \mathrm{l} .(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared) in exI)
apply (rule-tac $x=0$ in exI)
thm c-n
apply (simp add: c-n)
apply (clarsimp simp add: 0 leq others-same u-m-other u-shared) done
next
case (Suc k)
then obtain
trace-suc: trace c n (Suc k) and
unchanged-suc: $\forall \mathrm{l} \leq$ Suc $\mathrm{k} . \forall \mathrm{ts}_{\mathrm{l}} \mathcal{S}_{\mathrm{l}} \mathrm{m}_{\mathrm{l}} . \mathrm{c}(\mathrm{n}+\mathrm{l})=\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right) \longrightarrow \mathrm{ts}_{\mathrm{l}}!\mathrm{i}=\mathrm{ts}!\mathrm{i}$ and
safe-orig: $\forall \mathrm{x}<\mathrm{k}$. safe-delayed $(\mathrm{c}(\mathrm{n}+\mathrm{x}))$
by $\operatorname{simp}$
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb}$.
from trace-suc obtain
trace-k: trace c n k and
last-step: $\mathrm{c}(\mathrm{n}+\mathrm{k}) \Rightarrow_{\mathrm{d}} \mathrm{c}(\mathrm{n}+($ Suc k$))$
by (clarsimp simp add: program-trace-def)
from unchanged-suc obtain
unchanged-k: $\forall \mathrm{l} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{l}} \mathcal{S}_{\mathrm{l}} \mathrm{m}_{\mathrm{l}} . \mathrm{c}(\mathrm{n}+\mathrm{l})=\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right) \longrightarrow \mathrm{ts}_{\mathrm{l}}!\mathrm{i}=\mathrm{ts}!\mathrm{i}$ and
unchanged-suc-k: $\forall \mathrm{ts}_{\mathrm{l}} \mathcal{S}_{\mathrm{l}} \mathrm{m}_{\mathrm{l}} . \mathrm{c}(\mathrm{n}+($ Suc k$))=\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right) \longrightarrow \mathrm{ts}_{\mathrm{l}}!\mathrm{i}=\mathrm{ts}!\mathrm{i}$
apply -
apply (rule that)
apply auto
apply (drule-tac $x=1$ in spec)
apply simp
done
from Suc.hyps [OF trace-k unchanged-k safe-orig] obtain c'l where
$\mathrm{l}-\mathrm{k}: \mathrm{l} \leq \mathrm{k}$ and
trace-c ${ }^{\prime}$-l: trace $c^{\prime} \mathrm{n} l$ and
safe-l: $\left(\forall \mathrm{x}<\mathrm{l}\right.$. safe-delayed $\left.\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})\right)\right)$ and
unsafe-l: $\left(\mathrm{l}<\mathrm{k} \longrightarrow \neg\right.$ safe-delayed $\left.\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{l})\right)\right)$ and
$\mathrm{c}^{\prime}-\mathrm{n}: \mathrm{c}^{\prime} \mathrm{n}=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared) and
leq-l: $\left(\forall \mathrm{x} \leq\right.$ l. length $\left(\right.$ fst $\left.\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})\right)\right)=$ length $\left.(\mathrm{fst}(\mathrm{c}(\mathrm{n}+\mathrm{x})))\right)$ and
unchanged-i: $\left(\forall \mathrm{x} \leq \mathrm{l} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}} \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.
$\mathrm{c}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{i}=\mathrm{u}-\mathrm{ts}!\mathrm{i} \wedge$
( $\forall \mathrm{a} \in$ u-owns. $\mathcal{S}_{\times}{ }^{\prime} \mathrm{a}=$ u-shared a$) \wedge$
( $\forall \mathrm{a} \in$ u-owns. $\left.\mathcal{S}_{\mathrm{x}} \mathrm{a}=\mathcal{S} \mathrm{a}\right) \wedge$
( $\forall \mathrm{a} \in \mathrm{u}$-owns. $\left.\mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{u}-\mathrm{m} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in \mathrm{u}\right.$-owns. $\left.\left.\mathrm{m}_{\mathrm{x}} \mathrm{a}=\mathrm{m} \mathrm{a}\right)\right)$ and
$\operatorname{sim}:\left(\forall \mathrm{x} \leq \mathrm{l} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}} \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.

$$
\begin{aligned}
& \mathrm{c}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{~m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow \\
& \mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})=\left(\mathrm{ts}_{\mathrm{x}}^{\prime}, \mathrm{m}_{\mathrm{x}}^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow \\
& \left(\forall \mathrm{j}<\text { length ts }_{\mathrm{x}} \cdot \mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{j}=\mathrm{ts}_{\mathrm{x}}!\mathrm{j}\right) \wedge \\
& \left(\forall \text { a. a } \notin \mathrm{u}-\mathrm{owns} \longrightarrow \mathrm{a} \notin \text { owned }(\mathrm{ts}!\mathrm{i}) \longrightarrow \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{x}} \mathrm{a}\right) \wedge \\
& \left.\left(\forall \text { a. a } \notin \mathrm{u}-\text { owns } \longrightarrow \mathrm{m}_{\mathrm{x}}^{\prime}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{x}} \mathrm{a}\right)\right)
\end{aligned}
$$

by auto
show ?case
proof (cases l<k)
case True
with True trace-c'-l safe-l unsafe-l unchanged-i sim leq-l $c^{\prime}-n$
show ?thesis
apply -
apply (rule-tac $x=c^{\prime}$ in exI)

```
    apply (rule-tac \(\mathrm{x}=\mathrm{l}\) in exI)
    apply auto
    done
next
case False
with l-k have l-k: \(\mathrm{l}=\mathrm{k}\) by auto
show ?thesis
proof (cases safe-delayed ( \(\left.\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{k})\right)\) )
case False
with False l-k trace-c'-l safe-l unsafe-l unchanged-i sim leq-l c'-n
show ?thesis
    apply -
    apply (rule-tac \(x=c^{\prime}\) in exI)
    apply (rule-tac \(x=k\) in exI)
    apply auto
    done
next
case True
note safe- \(\mathrm{k}=\) this
obtain \(\mathrm{ts}_{\mathrm{k}} \mathcal{S}_{\mathrm{k}} \mathrm{m}_{\mathrm{k}}\) where c-k: \(\mathrm{c}(\mathrm{n}+\mathrm{k})=\left(\mathrm{ts}_{\mathrm{k}}, \mathrm{m}_{\mathrm{k}}, \mathcal{S}_{\mathrm{k}}\right)\)
by \((\operatorname{cases} \mathrm{c}(\mathrm{n}+\mathrm{k}))\)
obtain \(\mathrm{ts}_{\mathrm{k}}{ }^{\prime} \mathcal{S}_{\mathrm{k}}{ }^{\prime} \mathrm{m}_{\mathrm{k}}{ }^{\prime}\) where c-suc-k: c \((\mathrm{n}+(\) Suc k\())=\left(\mathrm{ts}_{\mathrm{k}}{ }^{\prime}, \mathrm{m}_{\mathrm{k}}{ }^{\prime}, \mathcal{S}_{\mathrm{k}}{ }^{\prime}\right)\)
    by ( cases c \((\mathrm{n}+(\) Suc k\()))\)
obtain \(u-t_{s} u-\) shared \(_{k} u-m_{k}\) where \(c^{\prime}-\mathrm{k}\) : \(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{k})=\left(\mathrm{u}-\mathrm{ts}_{\mathrm{k}}, \mathrm{u}-\mathrm{m}_{\mathrm{k}}, \mathrm{u}-\right.\) shared \(\left._{\mathrm{k}}\right)\)
by \(\left(\operatorname{cases} \mathrm{c}^{\prime}(\mathrm{n}+\mathrm{k})\right)\)
from trace-preserves-length-ts [OF trace-k, of k 0] c-n c-k i-bound
have i-bound-k: \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{k}}\)
    by simp
    from leq-1 [rule-format, simplified l-k, of k ] \(\mathrm{c}-\mathrm{k} \mathrm{c}^{\prime}-\mathrm{k}\)
    have leq: length \(u-\mathrm{ts}_{\mathrm{k}}=\) length \(\mathrm{ts}_{\mathrm{k}}\)
        by \(\operatorname{simp}\)
    note last-step \(=\) last-step \([\) simplified c-k c-suc-k]
    from unchanged-suc-k c-suc-k
    have \(\mathrm{ts}_{\mathrm{k}}!\) ! \(\mathrm{i}=\mathrm{ts}!\mathrm{i}\)
        by auto
    moreover from unchanged-k [rule-format, of k ] c-k
    have unch- \(\mathrm{k}-\mathrm{i}: \mathrm{ts}_{\mathrm{k}}!\mathrm{i}=\mathrm{ts}!\mathrm{i}\)
        by auto
    ultimately have ts -eq: \(\mathrm{ts}_{\mathrm{k}}!\mathrm{i}=\mathrm{ts}_{\mathrm{k}}\) ! i
        by simp
    from unchanged-i [simplified l-k, rule-format, OF - c-k c'-k]
obtain
u - \(\mathrm{ts}-\mathrm{eq}: \mathrm{u}-\mathrm{ts}_{\mathrm{k}}!\mathrm{i}=\mathrm{u}-\mathrm{ts}!\mathrm{i}\) and
```

unchanged-shared: $\forall \mathrm{a} \in \mathrm{u}$-owns. u -shared $\mathrm{k}_{\mathrm{k}} \mathrm{a}=\mathrm{u}$-shared a and
unchanged-shared-orig: $\forall \mathrm{a} \in$ u-owns. $\mathcal{S}_{\mathrm{k}} \mathrm{a}=\mathcal{S}$ a and
unchanged-owns: $\forall \mathrm{a} \in \mathrm{u}$-owns. $\mathrm{u}-\mathrm{m}_{\mathrm{k}} \mathrm{a}=\mathrm{u}-\mathrm{m}$ a and
unchanged-owns-orig: $\forall \mathrm{a} \in \mathrm{u}$-owns. $\mathrm{m}_{\mathrm{k}} \mathrm{a}=\mathrm{m} \mathrm{a}$
by fastforce
from u-ts-eq u-ts-i
have $u-\mathrm{ts}_{\mathrm{k}}-\mathrm{i}: \mathrm{u}-\mathrm{ts}_{\mathrm{k}}!\mathrm{i}=(\mathrm{u}-\mathrm{p}, \mathrm{u}$-is,u-tmps,u-sb,u-dirty,u-owns,u-rels)
by auto
from sim [simplified l-k, rule-format, of k , $\mathrm{OF}-\mathrm{c}-\mathrm{k} \mathrm{c}^{\prime}-\mathrm{k}$ ]
obtain
ts-sim: $\left(\forall \mathrm{j}<\right.$ length $\left.\mathrm{ts}_{\mathrm{k}} \cdot \mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{u}-\mathrm{ts}_{\mathrm{k}}!\mathrm{j}=\mathrm{ts}_{\mathrm{k}}!\mathrm{j}\right)$ and
shared-sim: $\left(\forall \mathrm{a} . \mathrm{a} \notin \mathrm{u}\right.$-owns $\longrightarrow \mathrm{a} \notin$ owned $\left(\mathrm{ts}_{\mathrm{k}}!\mathrm{i}\right) \longrightarrow \mathrm{u}$-shared $\left.\mathrm{c}_{\mathrm{k}} \mathrm{a}=\mathcal{S}_{\mathrm{k}} \mathrm{a}\right)$ and mem-sim: $\left(\forall\right.$ a. a $\notin \mathrm{u}$-owns $\left.\longrightarrow \mathrm{u}-\mathrm{m}_{\mathrm{k}} \mathrm{a}=\mathrm{m}_{\mathrm{k}} \mathrm{a}\right)$
by (auto simp add: unch-k-i)
from unchanged-owns-orig unchanged-owns u-m-shared unchanged-shared
have unchanged-owns-shared: $\forall \mathrm{a} . \mathrm{a} \in \mathrm{u}$-owns $\longrightarrow \mathrm{a} \in \operatorname{dom} \mathrm{u}$-shared $\mathrm{s}_{\mathrm{k}} \longrightarrow \mathrm{u}-\mathrm{m}_{\mathrm{k}} \mathrm{a}$ $=m_{\mathrm{k}} \mathrm{a}$
by (auto simp add: simp add: domIff)
from safe-l l-k safe-k
have safe-up- k : $\forall \mathrm{x}<\mathrm{k}$. safe-delayed $\left(\mathrm{c}^{\prime}(\mathrm{n}+\mathrm{x})\right)$
apply clarsimp
done
from trace-preserves-simple-ownership-distinct [OF - trace-c'-l [simplified l-k] safe-up-k,
simplified $\mathrm{c}^{\prime}-\mathrm{n}$, simplified, OF dist, of k$] \mathrm{c}^{\prime}-\mathrm{k}$
have dist ${ }^{\prime}$ : simple-ownership-distinct $\mathrm{u}-\mathrm{ts}_{\mathrm{k}}$ by $\operatorname{simp}$
from trace-preserves-simple-ownership-distinct [OF - trace-k, simplified c-n, simplified, OF dist-ts safe-orig, of $k$ ]
c-k
have dist-orig': simple-ownership-distinct $\mathrm{ts}_{\mathrm{k}}$ by $\operatorname{simp}$
from undo-local-step [OF last-step i-bound-k ts-eq safe-k [simplified c'-k] leq ts-sim u-ts ${ }_{k}$-i mem-sim
unchanged-owns-shared shared-sim dist' dist-orig']
obtain $u$-ts ${ }^{\prime} u$-shared ${ }^{\prime} u-m^{\prime}$ where
step $^{\prime}:\left(\mathrm{u}-\mathrm{ts}_{\mathrm{k}}, \mathrm{u}-\mathrm{m}_{\mathrm{k}}, \mathrm{u}-\right.$ shared $\left._{\mathrm{k}}\right) \Rightarrow_{\mathrm{d}}\left(\mathrm{u}-\mathrm{ts}^{\prime}, \mathrm{u}-\mathrm{m}^{\prime}, \mathrm{u}-\right.$ shared $\left.^{\prime}\right)$ and
ts-eq': u-ts ${ }^{\prime}!\mathrm{i}=\mathrm{u}-\mathrm{ts}_{\mathrm{k}}$ ! i and
unchanged-shared ${ }^{\prime}:\left(\forall \mathrm{a} \in \mathrm{u}\right.$-owns. u-shared ${ }^{\prime} \mathrm{a}=\mathrm{u}$-shared $\left.{ }_{\mathrm{k}} \mathrm{a}\right)$ and
unchanged-shared-orig': ( $\forall \mathrm{a} \in \mathrm{u}$-owns. $\left.\mathcal{S}_{\mathrm{k}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{k}} \mathrm{a}\right)$ and
unchanged-owns': ( $\forall \mathrm{a} \in \mathrm{u}-\mathrm{owns} . \mathrm{u}-\mathrm{m}^{\prime} \mathrm{a}=\mathrm{u}-\mathrm{m}_{\mathrm{k}} \mathrm{a}$ ) and
unchanged-owns-orig': ( $\forall \mathrm{a} \in \mathrm{u}$-owns. $\left.\mathrm{m}_{\mathrm{k}}^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{k}} \mathrm{a}\right)$ and
sim-ts ${ }^{\prime}:\left(\forall \mathrm{j}<\right.$ length $\left.\mathrm{ts}_{\mathrm{k}} \cdot \mathrm{j} \neq \mathrm{i} \longrightarrow \mathrm{u}-\mathrm{ts}^{\prime}!\mathrm{j}=\mathrm{ts}_{\mathrm{k}}{ }^{\prime}!\mathrm{j}\right)$ and
sim-shared ${ }^{\prime}:\left(\forall\right.$ a. a $\notin \mathrm{u}$-owns $\longrightarrow \mathrm{a} \notin$ owned $\left.\left(\mathrm{ts}_{\mathrm{k}}!\mathrm{i}\right) \longrightarrow \mathrm{u}^{\text {-shared }}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{k}}{ }^{\prime} \mathrm{a}\right)$ and sim-m': ( $\forall \mathrm{a}$. $\mathrm{a} \notin \mathrm{u}$-owns $\left.\longrightarrow \mathrm{u}-\mathrm{m}^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{k}}{ }^{\prime} \mathrm{a}\right)$
by auto
define $\mathrm{c}^{\prime \prime}$ where $\mathrm{c}^{\prime \prime}==\lambda l$. if $\mathrm{l} \leq \mathrm{n}+\mathrm{k}$ then $\mathrm{c}^{\prime}$ l else ( $u$-ts', $\mathrm{u}-\mathrm{m}^{\prime}$, u -shared )
have [simp]: $\forall \mathrm{x} \leq \mathrm{n}+\mathrm{k} . \mathrm{c}^{\prime \prime} \mathrm{x}=\mathrm{c}^{\prime} \mathrm{x}$
by (auto simp add: $c^{\prime \prime}$-def)
have $[\operatorname{simp}]: \mathrm{c}^{\prime \prime}(\operatorname{Suc}(\mathrm{n}+\mathrm{k}))=\left(\mathrm{u}-\mathrm{ts}^{\prime}, \mathrm{u}-\mathrm{m}^{\prime}\right.$, u -shared $\left.{ }^{\prime}\right)$
by (auto simp add: $c^{\prime \prime}$-def)
from trace- $\mathrm{c}^{\prime}-\mathrm{l}$ l-k step ${ }^{\prime} \mathrm{c}^{\prime}-\mathrm{k}$ have trace': trace $\mathrm{c}^{\prime \prime} \mathrm{n}$ (Suc k)
apply (simp add: program-trace-def)
apply (clarsimp simp add: split-less-Suc)
done
from direct-computation.step-preserves-length-ts [OF last-step]
have leq- $\mathrm{ts}_{\mathrm{k}}{ }^{\prime}$ : length $\mathrm{ts}_{\mathrm{k}}{ }^{\prime}=$ length $\mathrm{ts}_{\mathrm{k}}$.
with direct-computation.step-preserves-length-ts [OF step ] leq
have leq': length u-ts ${ }^{\prime}=$ length $\mathrm{ts}_{\mathrm{k}}$
by simp
show ?thesis
apply (rule-tac $x=c^{\prime \prime}$ in exI)
apply (rule-tac $\mathrm{x}=$ Suc k in exI)
using safe-l l-k unchanged-i sim c-suc-k leq-l c'-n leq'
apply (clarsimp simp add: split-less-Suc split-le-Suc safe-k trace ${ }^{\prime}$ leq- $\mathrm{ts}_{\mathrm{k}}{ }^{\prime}$ sim-ts ${ }^{\prime}$ sim-shared' sim-m' unch-k-i
ts-eq' $\mathbf{u - t s - e q}$
unchanged-shared' unchanged-shared unchanged-shared-orig un-changed-shared-orig'
unchanged-owns' unchanged-owns unchanged-owns-orig' unchanged-owns-orig )
done
qed
qed
qed
locale program-safe-reach-upto $=$ program +
fixes $n$ fixes safe fixes $c_{0}$
assumes safe-config: $\llbracket \mathrm{k} \leq \mathrm{n}$; trace c $0 \mathrm{k} ; \mathrm{c} 0=\mathrm{c}_{0} ; \mathrm{l} \leq \mathrm{k} \rrbracket \Longrightarrow$ safe (c l)
abbreviation (in program)
safe-reach-upto $\equiv$ program-safe-reach-upto program-step
lemma (in program) safe-reach-upto-le:
assumes safe: safe-reach-upto $n$ safe $c_{0}$
assumes m-n: $\mathrm{m} \leq \mathrm{n}$

```
    shows safe-reach-upto m safe co
using safe m-n
apply (clarsimp simp add: program-safe-reach-upto-def)
    subgoal for k c
    apply (subgoal-tac k}\leqn
        apply blast
    apply simp
    done
done
```

lemma (in program) last-action-of-thread:
assumes trace: trace c 0 k
shows
- thread i never executes
$(\forall \mathrm{l} \leq \mathrm{k}$. fst $(\mathrm{c} \mathrm{l})!\mathrm{i}=\mathrm{fst}(\mathrm{ck})!\mathrm{i}) \vee$
- thread i has a last step in the trace
( $\exists$ last $<\mathrm{k}$.
fst (c last) $!i \neq$ fst ( $\mathbf{c}($ Suc last $))!i \wedge$
$(\forall \mathrm{l}$. last $<\mathrm{l} \longrightarrow \mathrm{l} \leq \mathrm{k} \longrightarrow \mathrm{fst}(\mathrm{c} \mathrm{l})!\mathrm{i}=\mathrm{fst}(\mathrm{ck})!\mathrm{i}))$
using trace
proof (induct k)
case 0 thus ?case
by auto
next
case (Suc k)
hence trace c 0 (Suc k) by simp
then
obtain
trace-k: trace c 0 k and
last-step: c $\mathrm{k} \Rightarrow_{\mathrm{d}} \mathrm{c}$ (Suc k)
by (clarsimp simp add: program-trace-def)
show ?case
proof (cases fst (c k)!i=fst (c (Suc k))!i)
case False
then show? thesis
apply -
apply (rule disjI2)
apply (rule-tac $\mathrm{x}=\mathrm{k}$ in exI)
apply clarsimp
apply (subgoal-tac l=Suc k)
apply auto
done
next
case True
note idle-i $=$ this
\{

```
    assume same: ( }\forall\textrm{l}\leq\textrm{k}. fst (c l)!i = fst (c k)!i)
    have ?thesis
        apply -
        apply (rule disjI1)
        apply clarsimp
        apply (case-tac l=Suc k)
        apply (simp add: idle-i)
        apply (rule same [simplified idle-i, rule-format])
        apply simp
        done
    }
    moreover
    {
        fix last
        assume last-k: last < k
        assume last-step: fst (c last) ! i f fst (c (Suc last)) ! i
        assume idle: ( }\forall\textrm{l}>\mathrm{ last. l }\leq\textrm{k}\longrightarrow\textrm{fst}(\textrm{c l ) ! i = fst (c k) ! i)
        have ?thesis
            apply -
            apply (rule disjI2)
            apply (rule-tac x=last in exI)
            using last-k
            apply (simp add: last-step)
            using idle [simplified idle-i]
            apply clarsimp
            apply (case-tac l=Suc k)
            apply clarsimp
            apply clarsimp
            done
    }
    moreover note Suc.hyps [OF trace-k]
    ultimately
    show ?thesis
        by blast
    qed
qed
lemma (in program) sequence-traces:
assumes trace1: trace c}\mp@subsup{c}{1}{}0\textrm{k
assumes trace2: trace ce m l
assumes seq: c}\mp@subsup{c}{2}{}m=\mp@subsup{c}{1}{}
assumes c-def: c = ( \lambdax. if x }\leq\textrm{k}\mathrm{ then c}\mp@subsup{\textrm{c}}{1}{}\textrm{x}\mathrm{ else (c
shows trace c 0 (k + l)
proof -
    from trace1
    interpret trace1: program-trace program-step c}\mp@subsup{c}{1}{}0\textrm{k}
    from trace2
    interpret trace2: program-trace program-step c}\mp@subsup{c}{2}{}\textrm{ml}
    {
        fix x
```

```
    assume x-bound: x < (k + l)
    have c x }\mp@subsup{=>}{\textrm{d}}{c
    proof (cases x < k)
        case True
        from trace1.step [OF True] True
        show ?thesis
        by (simp add: c-def)
    next
        case False
        hence k-x: k \leq x
        by auto
        with x-bound have bound: x - k < l
        by auto
        from k-x have eq: (Suc (m + x) - k) = Suc (m + x - k)
            by simp
        from trace2.step [OF bound] k-x seq
        show ?thesis
        by (auto simp add: c-def eq)
    qed
}
thus ?thesis
    by (auto simp add: program-trace-def)
qed
theorem (in program-progress) safe-free-flowing-implies-safe-delayed:
assumes init: initial co
assumes dist: simple-ownership-distinct (fst \(\mathrm{c}_{0}\) )
assumes read-only-unowned: read-only-unowned (snd (snd \(\mathrm{c}_{0}\) )) (fst \(\mathrm{c}_{0}\) )
assumes unowned-shared: unowned-shared ( \(\mathrm{snd}\left(\mathrm{snd}_{\mathrm{c}} \mathbf{0}\right)\) ) (fst \(\mathrm{c}_{0}\) )
assumes safe-reach-ff: safe-reach-upto \(n\) safe-free-flowing \(c_{0}\)
shows safe-reach-upto \(n\) safe-delayed \(c_{0}\)
using safe-reach-ff
proof (induct n )
case 0
hence safe-reach-upto 0 safe-free-flowing \(c_{0}\) by simp
hence safe-free-flowing \(c_{0}\)
by (auto simp add: program-safe-reach-upto-def)
from initial-safe-free-flowing-implies-safe-delayed [OF init this]
have safe-delayed \(\mathrm{c}_{0}\).
then show ?case
by (simp add: program-safe-reach-upto-def)
next
case (Suc n)
hence safe-reach-suc: safe-reach-upto (Suc n) safe-free-flowing co by simp
then interpret safe-reach-suc-inter: program-safe-reach-upto program-step (Suc n)
safe-free-flowing \(\mathrm{c}_{0}\).
from safe-reach-upto-le [OF safe-reach-suc ]
have safe-reach-n: safe-reach-upto \(n\) safe-free-flowing \(c_{0}\) by simp
from Suc.hyps [OF this]
have safe-delayed-reach-n: safe-reach-upto \(n\) safe-delayed \(c_{0}\).
```

then interpret safe-delayed-reach-inter: program-safe-reach-upto program-step $n$ safe-delayed $c_{0}$.
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb . sb .
show ?case
proof (cases safe-reach-upto (Suc n) safe-delayed $\mathrm{c}_{0}$ )
case True thus ?thesis .
next
case False
from safe-delayed-reach-n False
obtain c where
trace: trace c 0 (Suc n) and
$\mathrm{c}-0$ : c $0=\mathrm{c}_{0}$ and
safe-delayed-upto-n: $\forall \mathrm{k} \leq \mathrm{n}$. safe-delayed (c k) and
violation-delayed-suc: $\neg$ safe-delayed (c (Suc n))
proof -
from False
obtain c k l where
k -suc: $\mathrm{k} \leq$ Suc n and
trace-k: trace c 0 k and
$\mathrm{l}-\mathrm{k}: \mathrm{l} \leq \mathrm{k}$ and
violation: $\neg$ safe-delayed (c l) and
start: c $0=c_{0}$
by (clarsimp simp add: program-safe-reach-upto-def)
show ?thesis
proof (cases $\mathrm{k}=$ Suc n )
case False
with k -suc have $\mathrm{k} \leq \mathrm{n}$
by auto
from safe-delayed-reach-inter.safe-config [where $\mathrm{c}=\mathrm{c}$, OF this trace-k start l-k]
have safe-delayed (cl).
with violation have False by simp
thus ?thesis ..
next
case True
note k -suc- $\mathrm{n}=$ this
from trace-k True have trace-n: trace c 0 n
by (auto simp add: program-trace-def)
show ?thesis
proof (cases l=Suc n)
case False
with k-suc-n l-k have $1 \leq n$ by simp
from safe-delayed-reach-inter.safe-config [where $c=c$, OF - trace-n start this ]
have safe-delayed (c l) by simp
with violation have False by simp
thus ?thesis ..
next
case True
from safe-delayed-reach-inter.safe-config [where c=c, OF - trace-n start]

```
    have }\forall\textrm{k}\leqn. safe-delayed (c k) by sim
    with True k-suc-n trace-k start violation
    show ?thesis
        apply -
        apply (rule that)
        apply auto
        done
    qed
    qed
qed
from trace
interpret trace-inter: program-trace program-step c 0 Suc n .
from safe-reach-suc-inter.safe-config [where c=c, OF - trace c-0]
have safe-suc: safe-free-flowing (c (Suc n))
    by auto
obtain ts S m where c-suc: c (Suc n) = (ts,m,\mathcal{S}) by (cases c (Suc n))
from violation-delayed-suc c-suc
obtain i p is \vartheta sb \mathcal{D O}\mathcal{R}\mathrm{ where}
    i-bound: i < length ts and
    ts-i: ts ! i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\mathrm{ and}
    violation-i: \neg map owned ts,map released ts,i }\vdash(\textrm{is},\vartheta,\textrm{m},\mathcal{D},\mathcal{O},\mathcal{S})\sqrt{}{
    by (fastforce simp add: safe-free-flowing-def safe-delayed-def)
from trace-preserves-unowned-shared [where c=c and n=0 and l=Suc n,
        simplified c-0, OF dist unowned-shared trace] safe-delayed-upto-n c-suc
have unowned-shared S ts by auto
then interpret unowned-shared S ts.
```

from violation-i obtain ins is ${ }^{\prime}$ where is: is $=$ ins\#is ${ }^{\prime}$
by (cases is) (auto simp add: safe-delayed-direct-memop-state.Nil)
from safeE [OF safe-suc [simplified c-suc] i-bound ts-i]
have safe-i: map owned ts, $\mathrm{i} \vdash(\mathrm{is}, \vartheta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }$.
define races where races $==\lambda \mathcal{R}$. (case ins of
Read volatile $\mathrm{at} \Rightarrow(\mathcal{R} \mathrm{a}=$ Some False $) \vee(\neg$ volatile $\wedge \mathrm{a} \in \operatorname{dom} \mathcal{R})$
| Write volatile a sop A L R W $\Rightarrow(\mathrm{a} \in \operatorname{dom} \mathcal{R} \vee($ volatile $\wedge \mathrm{A} \cap \operatorname{dom} \mathcal{R} \neq\{ \}))$
| Ghost A L R W $\Rightarrow(\mathrm{A} \cap \operatorname{dom} \mathcal{R} \neq\{ \})$
$\mid \mathrm{RMW}$ a $\mathrm{t}(\mathrm{D}, \mathrm{f})$ cond ret $\mathrm{A} \mathrm{L} \mathrm{R} \mathrm{W} \Rightarrow($ if cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}))$
then $\mathrm{a} \in \operatorname{dom} \mathcal{R} \vee \mathrm{A} \cap \operatorname{dom} \mathcal{R} \neq\{ \}$
else $\mathcal{R} \mathrm{a}=$ Some False)
|- $\Rightarrow$ False)
\{
assume no-race:

```
    j. j < length ts }\longrightarrow\textrm{j}\not=\textrm{i}\longrightarrow\neg\mathrm{ races (released (ts!j))
from safe-i
have map owned ts,map released ts,i }\vdash(\mathrm{ is, , ,m, D},\mathcal{O},\mathcal{S})\sqrt{}{
proof cases
    case Read
    thus ?thesis
        using is no-race
        by (auto simp add: races-def intro: safe-delayed-direct-memop-state.intros)
next
    case WriteNonVolatile
    thus ?thesis
        using is no-race
        by (auto simp add: races-def intro: safe-delayed-direct-memop-state.intros)
next
    case WriteVolatile
    thus ?thesis
        using is no-race
        apply (clarsimp simp add: races-def)
        apply (rule safe-delayed-direct-memop-state.intros)
        apply auto
        done
next
    case Fence
    thus ?thesis
        using is no-race
        by (auto simp add: races-def intro: safe-delayed-direct-memop-state.intros)
next
    case Ghost
    thus ?thesis
        using is no-race
        apply (clarsimp simp add: races-def)
        apply (rule safe-delayed-direct-memop-state.intros)
        apply auto
        done
next
    case RMWReadOnly
    thus ?thesis
        using is no-race
        by (auto simp add: races-def intro: safe-delayed-direct-memop-state.intros)
next
    case (RMWWrite cond t a - - A - O)
    thus ?thesis
        using is no-race unowned-shared' [rule-format, of a] ts-i
        apply (clarsimp simp add: races-def)
        apply (rule safe-delayed-direct-memop-state.RMWWrite)
        apply auto
        apply force
        done
    next
        case Nil with is show ?thesis by auto
```

```
    qed
}
with violation-i
obtain j where
    j-bound: j < length ts and
    neq-j-i: j\not=i and
    race: races (released (ts!j))
    by auto
obtain p}\mp@subsup{\textrm{p}}{\textrm{j}}{}\mp@subsup{\textrm{is}}{\textrm{j}}{}\mp@subsup{\vartheta}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ where
    ts-j: ts!j = (p
    apply (cases ts!j)
    apply force
    done
from race
have }\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ -non-empty: }\mp@subsup{\mathcal{R}}{\textrm{j}}{}\not=\mathrm{ Map.empty
    by (auto simp add: ts-j races-def split: instr.splits if-split-asm)
{
    assume idle-j: \foralll\leqSuc n. fst (c l) ! j = fst (c (Suc n))!j
    have ?thesis
    proof -
        from idle-j [rule-format, of 0] c-suc c-0 ts-j
        have co-j: fst co ! j = ts!j
        by clarsimp
    from trace-preserves-length-ts [OF trace, of 0 Suc n] c-0 c-suc
    have length (fst co) = length ts
        by clarsimp
    with j-bound have j < length (fst co)
        by simp
    with nth-mem [OF this] init co-j ts-j
    have }\mp@subsup{\mathcal{R}}{\textrm{j}}{}=\mathrm{ Map.empty
        by (auto simp add: initial-def)
    with }\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ -non-empty have False
        by simp
        thus ?thesis ..
    qed
}
moreover
{
    fix last
    assume last-bound: last<Suc n
    assume last-step-changed-j: fst (c last) ! j = fst (c (Suc last)) ! j
    assume idle-rest: \foralll>last. l \leq Suc n \longrightarrow fst (c l) ! j = fst (c (Suc n)) ! j
    have ?thesis
    proof -
        obtain ts|}\mp@subsup{\mathcal{S}}{|}{}\mp@subsup{\textrm{m}}{|}{}\mathrm{ where
            c-last: c last = (ts_,m, ,\mathcal{S}
            by (cases c last)
```

```
obtain \(\mathrm{ts}_{\mathrm{l}}{ }^{\prime} \mathcal{S}_{\mid}{ }^{\prime} \mathrm{m}_{\mathrm{l}}{ }^{\prime}\) where
    c-last': c (Suc last) \(=\left(\mathrm{ts}^{\prime}{ }^{\prime}, \mathrm{m}^{\prime}{ }^{\prime}, \mathcal{S}^{\prime}{ }^{\prime}\right)\)
    by (cases c (Suc last))
from idle-rest [rule-format, of Suc last ] c-suc c-last \({ }^{\prime}\) last-bound
have \(\mathrm{ts}_{\mathrm{l}}{ }^{\prime}-\mathrm{j}: \mathrm{ts}^{\prime}!\mathrm{l} j=\mathrm{ts}!\mathrm{j}\)
    by auto
from last-step-changed-j c-last c-last \({ }^{\prime}\)
have \(j\)-changed: \(\mathrm{ts}_{!}!j \neq \mathrm{ts}_{\mathrm{l}}!\) !
        by auto
from trace-inter.step [OF last-bound] c-last c-last'
have last-step: \(\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}_{\mathrm{l}}{ }^{\prime}, \mathrm{m}^{\prime}{ }^{\prime}, \mathcal{S}_{\mathrm{l}}\right)\)
    by \(\operatorname{simp}\)
obtain \(\mathrm{p}_{\mathrm{l}}\) is \(\vartheta_{\mid} \mathrm{sb}_{\mid} \mathcal{D}_{\mid} \mathcal{O}_{\mid} \mathcal{R}_{\mid}\)where
    \(\mathrm{ts}_{\mathrm{l}} \mathrm{j}: \mathrm{ts}_{\mathrm{l}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{l}}, \mathrm{is}, \vartheta_{\mid}, \mathrm{sb}_{\mathrm{l}}, \mathcal{D}_{\mathrm{l}}, \mathcal{O}_{\mathrm{l}}, \mathcal{R}_{\mathrm{l}}\right)\)
    apply (cases \(\mathrm{ts}_{\mathrm{s}}!\mathrm{j}\) )
    apply force
    done
```

from trace-preserves-length-ts [OF trace, of last Suc n] c-last c-suc last-bound
have leqı: length $\mathrm{ts}_{\mathrm{I}}=$ length ts
by simp
with j-bound have j-bound ${ }_{\mathrm{I}}$ : $\mathrm{j}<$ length $_{\text {ts }}$
by simp
from trace have trace-n: trace c 0 n
by (auto simp add: program-trace-def)
from safe-delayed-reach-inter.safe-config [where $\mathrm{k}=\mathrm{n}$ and $\mathrm{c}=\mathrm{c}$ and $\mathrm{l}=$ last, OF -
trace-n c-0] last-bound c-last
have safe-delayed-last: safe-delayed $\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right)$
by auto
from safe-delayed-reach-inter.safe-config [where c=c, OF - trace-n c-0]
have safe-delayed-upto-n: $\forall \mathrm{x}<\mathrm{n}$. safe-delayed $(\mathrm{c}(0+\mathrm{x}))$
by auto
from trace-preserves-simple-ownership-distinct [where $\mathrm{c}=\mathrm{c}$ and $\mathrm{n}=0$ and $\mathrm{l}=$ last,
simplified c-0, OF dist trace-n safe-delayed-upto-n]
last-bound c-last
have dist-last: simple-ownership-distinct ts|
by auto
from trace-preserves-read-only-unowned [where $\mathrm{c}=\mathrm{c}$ and $\mathrm{n}=0$ and $\mathrm{l}=$ last
simplified c-0, OF dist read-only-unowned trace-n safe-delayed-upto-n]
last-bound c-last
have ro-last-last: read-only-unowned $\mathcal{S}_{\mathrm{I}} \mathrm{ts}$ |
by auto
from safe-delayed-reach-inter.safe-config [where $\mathrm{c}=\mathrm{c}$, OF - trace-n c -0]
have safe-delayed-upto-suc-n: $\forall \mathrm{x}<$ Suc n . safe-delayed ( $\mathrm{c}(0+\mathrm{x})$ )
by auto
from trace-preserves-simple-ownership-distinct [where $c=c$ and $n=0$ and $l=S u c$ last, simplified c-0, OF dist trace safe-delayed-upto-suc-n] last-bound c-last ${ }^{\prime}$
have dist-last': simple-ownership-distinct $\mathrm{ts}^{\prime}{ }^{\prime}$ by auto
from trace last-bound have trace-last: trace c 0 last by (auto simp add: program-trace-def)
from trace last-bound have trace-rest: trace c (Suc last) ( n - last) by (auto simp add: program-trace-def)
from idle-rest last-bound
have idle-rest': $\forall \mathrm{l} \leq \mathrm{n}$ - last.
$\forall \mathrm{ts}_{\mathrm{l}} \mathcal{S}_{\mathrm{l}} \mathrm{m}_{\mathrm{l}} . \mathrm{c}($ Suc last +l$)=\left(\mathrm{ts} \mid, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{I}}\right) \longrightarrow \mathrm{ts}_{\mathrm{l}}!\mathrm{j}=\mathrm{ts}^{\prime}!\mathrm{j}$
apply clarsimp
apply (drule-tac $x=$ Suc (last +1 ) in spec)
apply (auto simp add: c-last ${ }^{\prime} \mathrm{c}$-suc $\mathrm{ts}^{\prime}-\mathrm{j}$ )
done
from safe-delayed-upto-suc-n [rule-format, of last] last-bound
have safe-delayed-last: safe-delayed ( $\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}$ )
by (auto simp add: c-last)
from safe-delayedE [OF this j-bound ${ }_{1} \mathrm{ts}_{1}-\mathrm{j}$ ]
have safe l : map owned $\mathrm{ts}_{\mathrm{l},}$, map released $\mathrm{ts}_{\mathrm{l}}, \mathrm{j} \vdash\left(\mathrm{is}_{\mathrm{l}}, \vartheta_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{D}_{\mathrm{l}}, \mathcal{O}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right) \sqrt{ }$.
from safe-delayed-reach-inter.safe-config [where $\mathrm{c}=\mathrm{c}$, OF - trace-n c-0]
have safe-delayed-upto-last: $\forall \mathrm{x}<\mathrm{n}-$ last. safe-delayed (c (Suc (last +x )))
by auto
from last-step
show ?thesis
proof (cases)
case (Program i' - - - - - $\mathrm{p}^{\prime}$ is ${ }^{\prime}$ )
with j-changed j-bound, $\mathrm{ts}_{\mathrm{l}} \mathrm{j} \mathrm{j}$
obtain
$\mathrm{ts}_{l}: \mathrm{ts}^{\prime}{ }^{\prime}=\mathrm{ts}_{1}\left[\mathrm{j}:=\left(\mathrm{p}^{\prime}, \mathrm{is}_{\mid} @ \mathrm{is}^{\prime}, \boldsymbol{\vartheta}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\right]$ and $\mathcal{S}_{1}^{\prime}: \mathcal{S}_{1}^{\prime}=\mathcal{S}_{1}$ and $m_{1}^{\prime}: m_{1}^{\prime}=m_{1}$ and prog-step: $\vartheta_{\mathrm{l}} \vdash \mathrm{p}_{\mathrm{l}} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $)$ by (cases $\mathrm{i}^{\prime}=\mathrm{j}$ ) auto
from ts, ${ }^{\prime}-\mathrm{j}$ ts ${ }^{\prime}{ }^{\prime}$ ts-j j-bound ${ }_{1}$
obtain eqs: $\mathrm{p}^{\prime}=\mathrm{p}_{\mathrm{j}}$ is $@_{\mathrm{is}}{ }^{\prime}=\mathrm{is} \mathrm{s}_{\mathrm{j}} \vartheta_{\mathrm{l}}=\vartheta_{\mathrm{j}} \mathcal{D}_{\mathrm{l}}=\mathcal{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{l}}=\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{l}}=\mathcal{R}_{\mathrm{j}}$
by auto
from undo-local-steps [where $\mathrm{c}=\mathrm{c}$, OF trace-rest c-last' idle-rest' safe-delayed-last, simplified $\mathrm{ts}_{\mathrm{I}}{ }^{\prime}$,
simplified,
OF j-bound ${ }_{\boldsymbol{I}} \mathrm{ts}_{\mid}-\mathrm{j}$ [simplified], simplified $\mathrm{m}_{\mid}{ }^{\prime} \mathcal{S}_{\mid}{ }^{\prime}$, simplified, OF dist-last dist-last ${ }^{\prime}$ [simplified $\mathrm{ts}^{\prime}{ }^{\prime}$,simplified] safe-delayed-upto-last]
obtain $c^{\prime} k$ where
k -bound: $\mathrm{k} \leq \mathrm{n}$ - last and
trace-c': trace c ${ }^{\prime}$ (Suc last) k and
$\mathrm{c}^{\prime}$-first: $\mathrm{c}^{\prime}$ (Suc last) $=\left(\mathrm{ts}_{\mathrm{l}}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right)$ and
$\mathrm{c}^{\prime}$-leq: $\left(\forall \mathrm{x} \leq \mathrm{k}\right.$. length $\left(\right.$ fst $\left(\mathrm{c}^{\prime}(\operatorname{Suc}(\right.$ last +x$\left.\left.))\right)\right)=\operatorname{length}($ fst $(\mathrm{c}(\operatorname{Suc}($ last +x$\left.))))\right)$
and
$c^{\prime}$-safe: $\left(\forall x<k\right.$. safe-delayed $\left(c^{\prime}(\right.$ Suc $($ last $\left.+x))\right)$ ) and
$c^{\prime}$-unsafe: $\left(\mathrm{k}<\mathrm{n}-\right.$ last $\longrightarrow \neg$ safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +k$\left.))\right)$ ) and
$c^{\prime}$-unch:
$\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.$.
$\mathrm{c}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.
$c^{\prime}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{j}=\mathrm{ts}_{\mathrm{l}}!\mathrm{j} \wedge$
$\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{l}} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in \mathcal{O}_{I} . \mathcal{S}_{\mathrm{x}} \mathrm{a}=\mathcal{S}_{I} \mathrm{a}\right) \wedge$
$\left.\left.\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{l}} \cdot \mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\right) \wedge\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{l}} \cdot \mathrm{m}_{\mathrm{x}} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\right)\right)\right)$ and
$c^{\prime}-\operatorname{sim}$ :
$\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.$.
$\mathrm{c}(\operatorname{Suc}($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.
$c^{\prime}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\left(\forall \mathrm{ja}<\right.$ length $\left.\mathrm{ts}_{\mathrm{x}} . \mathrm{ja} \neq \mathrm{j} \longrightarrow \mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{ja}=\mathrm{ts}_{\mathrm{x}}!\mathrm{ja}\right) \wedge$
$\left(\forall\right.$ a. a $\left.\notin \mathcal{O}_{I} \longrightarrow \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{x}} \mathrm{a}\right) \wedge$ $\left.\left.\left(\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{x}} \mathrm{a}\right)\right)\right)$
by auto
obtain c-undo where c-undo: c -undo $=\left(\lambda \mathrm{x}\right.$. if $\mathrm{x} \leq$ last then c x else $\mathrm{c}^{\prime}$ (Suc last $+\mathrm{x}-$ last))
by blast
have c-undo- 0 : c-undo $0=c_{0}$
by (auto simp add: c-undo c-0)
from sequence-traces [OF trace-last trace- $\mathrm{c}^{\prime}$, simplified c-last, OF c'-first c-undo]
have trace-undo: trace c-undo 0 (last +k ).
obtain u-ts u-shared u-m where
c-undo-n: c-undo $\mathrm{n}=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared $)$
by (cases c-undo n)
with last-bound $c^{\prime}$-first c-last
have $c^{\prime}$-suc: $c^{\prime}($ Suc $n)=(u-t s, u-m, u-s h a r e d)$
apply (auto simp add: c-undo split: if-split-asm)
apply (subgoal-tac $\mathrm{n}=$ last)
apply auto
done
show ?thesis
proof (cases $\mathrm{k}<\mathrm{n}-$ last)
case True
with c'-unsafe have unsafe: $\neg$ safe-delayed (c-undo (last +k ))
by (auto simp add: c-undo c-last c'-first)
from True have last $+\mathrm{k} \leq \mathrm{n}$ by auto
from safe-delayed-reach-inter.safe-config [OF this trace-undo, of last +k ]
have safe-delayed (c-undo (last +k )) by (auto simp add: c-undo c-0)
with unsafe have False by simp
thus ?thesis ..
next
case False
with k -bound have $\mathrm{k}: \mathrm{k}=\mathrm{n}$ - last by auto
have eq': Suc (last $+(\mathrm{n}-$ last $))=$ Suc n using last-bound by simp
from $\mathrm{c}^{\prime}$-unch [rule-format, of k , simplified $\mathrm{keq}{ }^{\prime}$, OF - c-suc $\mathrm{c}^{\prime}$-suc]
obtain $u-\mathrm{ts}-\mathrm{j}: \mathrm{u}-\mathrm{ts}!\mathrm{j}=\mathrm{ts}!$ ! and shared-unch: $\forall \mathrm{a} \in \mathcal{O}_{I}$. u-shared $\mathrm{a}=\mathcal{S}_{I}$ a and shared-orig-unch: $\forall \mathrm{a} \in \mathcal{O}_{I} . \mathcal{S} \mathrm{a}=\mathcal{S}_{I}$ a and mem-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}_{\mathrm{I}}$ a and mem-unch-orig: $\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathrm{ma}=\mathrm{m}_{\mathrm{l}} \mathrm{a}$ by auto
from $c^{\prime}$-sim [rule-format, of k , simplified k eq', OF - c -suc $\mathrm{c}^{\prime}$-suc] i -bound neq-j-i
obtain u-ts-i: u-ts!i $=$ ts! $!$ and
shared-sim: $\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\boldsymbol{l}} \longrightarrow$ u-shared $\mathrm{a}=\mathcal{S}$ a and
mem-sim: $\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m} \mathrm{a}$
by auto
from $c^{\prime}$-leq [rule-format, of $k$ ] $c^{\prime}$-suc c-suc
have leq-u-ts: length $u$-ts $=$ length ts
by (auto simp add: eq ${ }^{\prime} \mathrm{k}$ )
from j-bound leq-u-ts
have j-bound-u: $\mathrm{j}<$ length u -ts by simp
from i-bound leq-u-ts
have i-bound-u: $\mathrm{i}<$ length u -ts by simp
from k last-bound have l-k-eq: last $+\mathrm{k}=\mathrm{n}$ by auto
from safe-delayed-reach-inter.safe-config [OF - trace-undo, simplified l-k-eq] k c-0 last-bound

```
have safe-delayed-c-undo': \(\forall \mathrm{x} \leq \mathrm{n}\). safe-delayed (c-undo x )
    by (auto simp add: c-undo split: if-split-asm)
hence safe-delayed-c-undo: \(\forall \mathrm{x}<\mathrm{n}\). safe-delayed (c-undo x )
    by (auto)
from trace-preserves-simple-ownership-distinct [OF - trace-undo,
    simplified l-k-eq c-undo-0, simplified, OF dist this, of n] dist c-undo-n
have dist-u-ts: simple-ownership-distinct u-ts
    by auto
then interpret dist-u-ts-inter: simple-ownership-distinct u-ts .
\{
    fix a
    have \(\mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m} \mathrm{a}\)
    proof (cases a \(\in \mathcal{O}_{1}\) )
        case True with mem-unch
        have \(\mathrm{u}-\mathrm{ma}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\)
            by auto
        moreover
        from True mem-unch-orig
        have \(\mathrm{ma}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\)
            by auto
        ultimately show?thesis by simp
    next
        case False
        with mem-sim
        show ?thesis
            by auto
    qed
\} hence \(u-m-e q: u-m=m\) by - (rule ext, auto)
\{
    fix a
    have \(u\)-shared \(\mathrm{a}=\mathcal{S}\) a
    proof (cases a \(\in \mathcal{O}_{1}\) )
        case True with shared-unch
        have u-shared \(\mathrm{a}=\mathcal{S}_{\mathrm{I}} \mathrm{a}\)
            by auto
        moreover
        from True shared-orig-unch
        have \(\mathcal{S} \mathrm{a}=\mathcal{S}_{1} \mathrm{a}\)
            by auto
        ultimately show ?thesis by simp
    next
        case False
        with shared-sim
        show ?thesis
            by auto
    qed
\} hence u-shared-eq: u-shared \(=\mathcal{S}\) by - (rule ext, auto)
```

```
{
assume safe: map owned u-ts,map released u-ts,i}\vdash(\textrm{is},\vartheta,\textrm{u}-\textrm{m},\mathcal{D},\mathcal{O},\textrm{u}\mathrm{ -shared)}\sqrt{}{
then have False
proof cases
    case Read
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case WriteNonVolatile
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case WriteVolatile
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
next
    case Fence
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case Ghost
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
next
    case (RMWReadOnly cond t a D f ret A L R W)
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
next
    case RMWWrite
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
        apply fastforce+
        done
next
    case Nil
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
qed
```

```
    }
    hence }\neg\mathrm{ safe-delayed (u-ts, u-m, u-shared)
    apply (clarsimp simp add: safe-delayed-def)
    apply (rule-tac x=i in exI)
    using u-ts-i ts-i i-bound-u
    apply auto
    done
    moreover
    from safe-delayed-c-undo' [rule-format, of n] c-undo-n
    have safe-delayed (u-ts, u-m, u-shared)
        by simp
    ultimately have False
        by simp
        thus ?thesis
        by simp
    qed
next
    case (Memop i' ----- - is, ' }\mp@subsup{}{|}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{|}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{D}}{|}{\prime}\mp@subsup{\mathcal{O}}{|}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{|}{\prime}
    with j-changed j-bound, tsl-j
    obtain
```



```
        mem-step: (is,, \vartheta
```



```
        by (cases \mp@subsup{i}{}{\prime}=\textrm{j})\mathrm{ auto}
    from mem-step
    show ?thesis
    proof (cases)
        case (Read volatile a t)
        then obtain
            is,: is, = Read volatile a t # is, |}\mathrm{ 'and
            \mp@subsup{v}{1}{\prime}:\mp@subsup{\vartheta}{|}{\prime}=\mp@subsup{v}{1}{\prime}(\textrm{t}\mapsto\mp@subsup{\textrm{m}}{1}{}\textrm{a})\mathrm{ and}
            s\mp@subsup{b}{1}{\prime}}:\mp@subsup{\textrm{sb}}{1}{\prime}=\mp@subsup{\textrm{sb}}{1}{}\mathrm{ and
            \mathcal { D } _ { 1 } ^ { \prime } : \mathcal { D } _ { 1 } ^ { \prime } = \mathcal { D } _ { 1 } \text { and}
            \mp@subsup{\mathcal{O}}{1}{\prime}}:\mp@subsup{\mathcal{O}}{1}{\prime}=\mp@subsup{\mathcal{O}}{l}{}\mathrm{ and
            \mp@subsup{\mathcal{R}}{1}{\prime}:\mp@subsup{\mathcal{R}}{1}{\prime}=\mp@subsup{\mathcal{R}}{1}{}\mathrm{ and}
            \mp@subsup{\mathcal{S}}{1}{\prime}:\mp@subsup{\mathcal{S}}{1}{\prime}=\mp@subsup{\mathcal{S}}{1}{}\mathrm{ and}
            m}\mp@subsup{|}{1}{\prime}:\mp@subsup{m}{1}{\prime}=\mp@subsup{m}{1}{\prime
            by auto
```



```
    from ts, '-j ts, ' ts-j j-bound, eqs'
```



```
        by auto
```

from undo-local-steps [where c=c, OF trace-rest c-last' idle-rest' safe-delayed-last, simplified $\mathrm{ts}^{\prime}{ }^{\prime}$,
simplified,
OF j-bound, $\mathrm{ts}_{1-\mathrm{j}}$ [simplified], simplified $\mathrm{m}_{1}{ }^{\prime} \mathcal{S}_{1}{ }^{\prime}$, simplified, OF dist-last dist-last' [simplified ts| ${ }^{\prime}$,simplified] safe-delayed-upto-last]

## obtain $\mathrm{c}^{\prime} \mathrm{k}$ where

k-bound: $\mathrm{k} \leq \mathrm{n}$ - last and
trace-c': trace c ${ }^{\prime}$ (Suc last) k and
$c^{\prime}$-first: $\mathrm{c}^{\prime}($ Suc last $)=\left(\mathrm{ts}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{I}}\right)$ and
$\mathrm{c}^{\prime}$-leq: $\left(\forall \mathrm{x} \leq \mathrm{k}\right.$. length $\left(\right.$ fst $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.\left.))\right)\right)=$ length $($ fst $(\mathrm{c}($ Suc $($ last +
$\mathrm{x})$ ))) and
$c^{\prime}$-safe: $\left(\forall \mathrm{x}<\mathrm{k}\right.$. safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.))\right)$ ) and
$c^{\prime}$-unsafe: $\left(\mathrm{k}<\mathrm{n}-\right.$ last $\longrightarrow \neg$ safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +k$))$ ) ) and
$c^{\prime}$-unch:
$\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.$.
$\mathrm{c}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.
$\mathrm{c}^{\prime}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\mathrm{ts}_{\mathrm{x}}$ ! $\mathrm{j}=\mathrm{ts}_{\mathrm{s}}!\mathrm{j} \wedge$
$\left(\forall \mathrm{a} \in \mathcal{O}_{\mid} \cdot \mathcal{S}_{x}{ }^{\prime} \mathrm{a}=\mathcal{S}_{1} \mathrm{a}\right) \wedge$
$\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathcal{S}_{\mathrm{x}} \mathrm{a}=\mathcal{S}_{\mathrm{I}} \mathrm{a}\right) \wedge$
$\left.\left.\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} \cdot \mathrm{m}_{\mathrm{x}}^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\right) \wedge\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} \cdot \mathrm{m}_{\mathrm{x}} \mathrm{a}=\mathrm{m}_{\mathrm{I}} \mathrm{a}\right)\right)\right)$ and
c'sim:
$\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.$.
$\mathrm{c}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow$
$\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.$.
$\mathrm{c}^{\prime}($ Suc $($ last +x$))=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow$
$\left(\forall \mathrm{ja}<\right.$ length $\left.\mathrm{ts}_{\mathrm{x}} . \mathrm{ja} \neq \mathrm{j} \longrightarrow \mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{ja}=\mathrm{ts}_{\mathrm{x}}!\mathrm{ja}\right) \wedge$
$\left(\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{I}} \longrightarrow \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{x}} \mathrm{a}\right) \wedge$
$\left.\left(\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{x}} \mathrm{a}\right)\right)$ )
by (clarsimp simp add: $\mathcal{O}_{1}$ )
obtain c -undo where c -undo: c -undo $=\left(\lambda \mathrm{x}\right.$. if $\mathrm{x} \leq$ last then c x else $\mathrm{c}^{\prime}$ (Suc last $+\mathrm{x}-$ last))
by blast
have c-undo- 0 : c -undo $0=\mathrm{c}_{0}$
by (auto simp add: c-undo c-0)
from sequence-traces [OF trace-last trace-c', simplified c-last, OF c'-first c-undo]
have trace-undo: trace c -undo 0 (last +k ).
obtain $u$-ts $u$-shared $u$-m where
c -undo-n: c-undo $\mathrm{n}=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared $)$
by (cases c-undo n)
with last-bound $\mathrm{c}^{\prime}$-first c-last
have c'suc: $\mathrm{c}^{\prime}($ Suc n$)=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared $)$
apply (auto simp add: c-undo split: if-split-asm)
apply (subgoal-tac $\mathrm{n}=$ last)
apply auto
done
show ?thesis
proof (cases k < n - last)
case True
with $\mathrm{c}^{\prime}$-unsafe have unsafe: $\neg$ safe-delayed (c-undo (last +k ))
by (auto simp add: c-undo c-last c'-first)
from True have last $+\mathrm{k} \leq \mathrm{n}$
by auto
from safe-delayed-reach-inter.safe-config [OF this trace-undo, of last +k ]
have safe-delayed ( c -undo (last +k ))
by (auto simp add: c-undo c-0)
with unsafe have False by simp
thus ?thesis ..
next
case False
with k -bound have k : $\mathrm{k}=\mathrm{n}$ - last
by auto
have eq': Suc (last $+(\mathrm{n}-$ last $))=$ Suc n
using last-bound
by simp
from c'-unch [rule-format, of k , simplified k eq', OF - c -suc $\mathrm{c}^{\prime}$-suc]
obtain u-ts-j: u-ts! $\mathrm{j}=\mathrm{ts}!\mathrm{j}$ and
shared-unch: $\forall \mathrm{a} \in \mathcal{O}_{1}$. u-shared $\mathrm{a}=\mathcal{S}_{1}$ a and
shared-orig-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mid} . \mathcal{S} \mathrm{a}=\mathcal{S}_{1}$ a and
mem-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}$ and
mem-unch-orig: $\forall \mathrm{a} \in \mathcal{O}_{1} . \mathrm{ma}=\mathrm{m}_{\mathrm{l}} \mathrm{a}$
by auto
from $\mathrm{c}^{\prime}$-sim [rule-format, of k , simplified k eq', OF - c -suc $\mathrm{c}^{\prime}$-suc] i-bound neq-j-i obtain u-ts-i: u-ts!i = ts!i and
shared-sim: $\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{u}$-shared $\mathrm{a}=\mathcal{S} \mathrm{a}$ and
mem-sim: $\forall \mathrm{a}$. $\mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{u}-\mathrm{ma}=\mathrm{m} \mathrm{a}$
by auto
from c'-leq [rule-format, of k ] c'suc c-suc
have leq-u-ts: length $u$-ts $=$ length $t s$
by (auto simp add: eq' $k$ )
from j -bound leq-u-ts
have j-bound-u: j < length u-ts
by simp
from i-bound leq-u-ts
have i-bound-u: i < length u-ts by simp
from k last-bound have $1-\mathrm{k}$-eq: last $+\mathrm{k}=\mathrm{n}$ by auto
from safe-delayed-reach-inter.safe-config [OF - trace-undo, simplified l-k-eq] k c-0 last-bound
have safe-delayed-c-undo $: ~ \forall \mathrm{x} \leq \mathrm{n}$. safe-delayed (c-undo x )
by (auto simp add: c-undo split: if-split-asm)
hence safe-delayed-c-undo: $\forall \mathrm{x}<\mathrm{n}$. safe-delayed (c-undo x ) by (auto)
from trace-preserves-simple-ownership-distinct [OF - trace-undo, simplified l-k-eq c-undo-0, simplified, OF dist this, of n] dist c-undo-n
have dist-u-ts: simple-ownership-distinct u-ts
by auto
then interpret dist-u-ts-inter: simple-ownership-distinct u-ts .

```
{
    fix a
    have u-m a = m a
    proof (cases a }\in\mp@subsup{\mathcal{O}}{1}{}\mathrm{ )
        case True with mem-unch
        have u-m a = mla
            by auto
        moreover
        from True mem-unch-orig
        have ma= mla
            by auto
        ultimately show ?thesis by simp
    next
        case False
        with mem-sim
        show ?thesis
            by auto
    qed
} hence u-m-eq: u-m = m by - (rule ext, auto)
{
fix a
    have u-shared a =S S a
    proof (cases a }\in\mp@subsup{\mathcal{O}}{1}{}\mathrm{ )
        case True with shared-unch
        have u-shared a = \mathcal{S}}\textrm{a
            by auto
        moreover
        from True shared-orig-unch
        have \mathcal{S a}=\mp@subsup{\mathcal{S}}{1}{}\textrm{a}
            by auto
        ultimately show ?thesis by simp
    next
        case False
        with shared-sim
        show ?thesis
            by auto
        qed
} hence u-shared-eq: u-shared =\mathcal{S}}\mathrm{ by - (rule ext, auto)
{
assume safe: map owned u-ts,map released u-ts,i}\vdash(is,\vartheta,u-m,\mathcal{D},\mathcal{O},u-shared) \sqrt{}{
    then have False
    proof cases
        case Read
        then show ?thesis
        using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
            by (auto simp add:eqs races-def split: if-split-asm)
```

```
    next
    case WriteNonVolatile
    then show ?thesis
    using ts-i tsı-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
    next
    case WriteVolatile
    then show ?thesis
    using ts-i ts,-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
    next
    case Fence
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
    next
    case Ghost
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
    next
    case (RMWReadOnly cond t a D f ret A L R W)
    then show ?thesis
    using ts-i tsı-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
    next
    case RMWWrite
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
        apply fastforce+
        done
    next
    case Nil
    then show ?thesis
    using ts-i tsı-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
    qed
}
hence }\neg\mathrm{ safe-delayed (u-ts, u-m, u-shared)
    apply (clarsimp simp add: safe-delayed-def)
    apply (rule-tac x=i in exI)
    using u-ts-i ts-i i-bound-u
    apply auto
    done
moreover
```

```
    from safe-delayed-c-undo' [rule-format, of n] c-undo-n
    have safe-delayed (u-ts, u-m, u-shared)
            by simp
    ultimately have False
            by simp
    thus?thesis
            by simp
    qed
next
    case (WriteNonVolatile a D f A L R W)
    then obtain
    is,: is 
    v,
    s\mp@subsup{b}{1}{\prime}}:\mp@subsup{\textrm{sb}}{1}{\prime}=\mp@subsup{\textrm{sb}}{1}{}\mathrm{ and
    \mp@subsup{\mathcal{D}}{1}{\prime}:\mp@subsup{\mathcal{D}}{1}{\prime}=\mp@subsup{\mathcal{D}}{1}{}\mathrm{ and}
    \mp@subsup{\mathcal{O}}{1}{\prime}:\mp@subsup{\mathcal{O}}{1}{\prime}}=\mp@subsup{\mathcal{O}}{1}{\prime}\mathrm{ and
    \mp@subsup{\mathcal{R}}{1}{\prime}:{\mp@subsup{\mathcal{R}}{1}{\prime}=\mp@subsup{\mathcal{R}}{1}{\prime}}\mathrm{ and
    \mathcal{S}
    m
    by auto
```



```
    from tsl}\mp@subsup{}{\prime}{\prime}-\textrm{j ts}\mp@subsup{}{|}{\prime}\mp@subsup{}{}{\prime}\mathrm{ ts-j j-bound, eqs'
```



```
        \mathcal{R}}=\mp@subsup{\mathcal{R}}{\textrm{j}}{
        by auto
```

    from safe [simplified is,]
    obtain a-owned: \(\mathrm{a} \in \mathcal{O}_{\text {I }}\) and a-unshared: \(\mathrm{a} \notin \operatorname{dom} \mathcal{S}_{\text {I }}\)
        by cases auto
    have \(\mathrm{m}_{\mathrm{l}}\)-unch-unowned: \(\forall \mathrm{a}^{\prime} . \mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{m}_{\mathrm{l}} \mathrm{a}^{\prime}=\left(\mathrm{m}_{\mathrm{l}}\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{l}}\right)\right) \mathrm{a}^{\prime}\)
    using a-owned by auto
    have \(\mathrm{m}_{\mathrm{I}}\)-unch-unshared: \(\forall \mathrm{a}^{\prime} . \mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{I}} \longrightarrow \mathrm{a}^{\prime} \in \operatorname{dom} \mathcal{S}_{\mathrm{I}} \longrightarrow \mathrm{m}_{\mathrm{I}} \mathrm{a}^{\prime}=\left(\mathrm{m}_{\mathrm{l}}(\mathrm{a}:=\mathrm{f}\right.\)
    $\left.\vartheta_{1}\right) \mathrm{a}^{\prime}$
using a-unshared by auto
from undo-local-steps [where $\mathrm{c}=\mathrm{c}, \mathrm{OF}$ trace-rest c -last' idle-rest' safe-delayed-last, simplified $\mathrm{ts}^{\prime}{ }^{\prime}$,
simplified,
OF j-bound ${ }_{\mathrm{l}} \mathrm{ts}_{\mathrm{l}}$-j [simplified], simplified $\mathrm{m}_{\mathrm{l}}{ }^{\prime} \mathcal{S}^{\prime}{ }^{\prime}, \mathrm{OF} \quad \mathrm{m}_{\mathrm{l}}$-unch-unowned $\mathrm{m}_{\mathrm{l}}$-unch-unshared, simplified,

OF dist-last dist-last' [simplified $\mathrm{ts}^{\prime}$, ,simplified] safe-delayed-upto-last]
obtain $c^{\prime} k$ where
k -bound: $\mathrm{k} \leq \mathrm{n}$ - last and
trace-c': trace c' (Suc last) k and
$c^{\prime}$-first: $\mathrm{c}^{\prime}($ Suc last $)=\left(\mathrm{ts}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{I}}\right)$ and
$\mathrm{c}^{\prime}$-leq: $\left(\forall \mathrm{x} \leq \mathrm{k}\right.$. length $\left(\right.$ fst $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.\left.))\right)\right)=$ length (fst (c (Suc (last + x))))) and
$c^{\prime}$-safe: $\left(\forall \mathrm{x}<\mathrm{k}\right.$. safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.\left.))\right)\right)$ and
$c^{\prime}$-unsafe: $\left(\mathrm{k}<\mathrm{n}-\right.$ last $\longrightarrow \neg$ safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +k$\left.\left.))\right)\right)$ and $c^{\prime}$-unch:

```
\(\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.\).
    \(\mathrm{c}(\operatorname{Suc}(\) last +x\())=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow\)
    \(\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.\).
    \(c^{\prime}(\) Suc \((\) last +x\())=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow\)
    \(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{j}=\mathrm{ts}_{\mathrm{l}}!\mathrm{j} \wedge\)
    \(\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{I}} . \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{I}} \mathrm{a}\right) \wedge\)
    \(\left(\forall \mathrm{a} \in \mathcal{O}_{\mid} . \mathcal{S}_{\mathrm{x}} \mathrm{a}=\mathcal{S}_{\mid} \mathrm{a}\right) \wedge\)
    \(\left.\left.\left(\forall \mathrm{a} \in \mathcal{O}_{\mathrm{l}} \cdot \mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}\right) \wedge\left(\forall \mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{l}} \cdot \mathrm{m}_{\mathrm{x}} \mathrm{a}^{\prime}=\left(\mathrm{m}_{\mathrm{l}}\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{l}}\right)\right) \mathrm{a}^{\prime}\right)\right)\right)\) and
\(c^{\prime}\)-sim:
    \(\left(\forall \mathrm{x} \leq \mathrm{k} . \forall \mathrm{ts}_{\mathrm{x}} \mathcal{S}_{\mathrm{x}} \mathrm{m}_{\mathrm{x}}\right.\).
            \(\mathrm{c}(\operatorname{Suc}(\) last +x\())=\left(\mathrm{ts}_{\mathrm{x}}, \mathrm{m}_{\mathrm{x}}, \mathcal{S}_{\mathrm{x}}\right) \longrightarrow\)
            \(\left(\forall \mathrm{ts}_{\mathrm{x}}{ }^{\prime} \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{m}_{\mathrm{x}}{ }^{\prime}\right.\).
            \(\mathrm{c}^{\prime}(\) Suc \((\) last +x\())=\left(\mathrm{ts}_{\mathrm{x}}{ }^{\prime}, \mathrm{m}_{\mathrm{x}}{ }^{\prime}, \mathcal{S}_{\mathrm{x}}{ }^{\prime}\right) \longrightarrow\)
            \(\left(\forall \mathrm{ja}<\right.\) length \(\left.\mathrm{ts}_{\mathrm{x}} . \mathrm{ja} \neq \mathrm{j} \longrightarrow \mathrm{ts}_{\mathrm{x}}{ }^{\prime}!\mathrm{ja}=\mathrm{ts}_{\mathrm{x}}!\mathrm{ja}\right) \wedge\)
            \(\left(\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{I}} \longrightarrow \mathcal{S}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathcal{S}_{\mathrm{x}} \mathrm{a}\right) \wedge\)
            \(\left.\left.\left(\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{m}_{\mathrm{x}}{ }^{\prime} \mathrm{a}=\mathrm{m}_{\mathrm{x}} \mathrm{a}\right)\right)\right)\)
```

by (clarsimp simp add: $\mathcal{O}^{\prime}$ )
obtain c-undo where c-undo: c-undo $=\left(\lambda \mathrm{x}\right.$. if $\mathrm{x} \leq$ last then c x else $\mathrm{c}^{\prime}$ (Suc last $+\mathrm{x}-$ last))
by blast
have c-undo- 0 : c-undo $0=\mathrm{c}_{0}$
by (auto simp add: c-undo c-0)
from sequence-traces [OF trace-last trace-c', simplified c-last, OF c'-first c-undo]
have trace-undo: trace c-undo 0 (last +k ).
obtain $u$-ts $u$-shared $u$-m where
c-undo-n: c-undo $\mathrm{n}=(\mathrm{u}-\mathrm{ts}, \mathrm{u}-\mathrm{m}, \mathrm{u}$-shared $)$
by (cases c-undo n )
with last-bound $c^{\prime}$-first c-last
have $c^{\prime}$-suc: $c^{\prime}($ Suc $n)=(u-t s, u-m, u-s h a r e d)$
apply (auto simp add: c-undo split: if-split-asm)
apply (subgoal-tac $\mathrm{n}=$ last)
apply auto
done
show ?thesis
proof (cases $\mathrm{k}<\mathrm{n}$ - last)
case True
with c'-unsafe have unsafe: $\neg$ safe-delayed (c-undo (last +k ))
by (auto simp add: c-undo c-last c'-first)
from True have last $+\mathrm{k} \leq \mathrm{n}$
by auto
from safe-delayed-reach-inter.safe-config [OF this trace-undo, of last +k ]
have safe-delayed ( c -undo (last +k ))
by (auto simp add: c-undo c-0)
with unsafe have False by simp
thus ?thesis ..

## next

case False
with k -bound have $\mathrm{k}: \mathrm{k}=\mathrm{n}$ - last by auto
have $\mathrm{eq}^{\prime}$ : Suc (last $+(\mathrm{n}-$ last $\left.)\right)=$ Suc n using last-bound by simp
from $\mathrm{c}^{\prime}$-unch [rule-format, of k , simplified $\mathrm{k} \mathrm{eq}{ }^{\prime}$, OF - c-suc $\mathrm{c}^{\prime}$-suc]
obtain u-ts-j: u-ts!j $=\mathrm{ts}_{\mathrm{s}}!\mathrm{j}$ and
shared-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mid}$. u-shared $\mathrm{a}=\mathcal{S}_{\mathrm{I}}$ a and shared-orig-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mid} . \mathcal{S} \mathrm{a}=\mathcal{S}_{\mid}$a and mem-unch: $\forall \mathrm{a} \in \mathcal{O}_{I} . \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}_{1}$ a and mem-unch-orig: $\forall \mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{I}} . \mathrm{m} \mathrm{a}^{\prime}=\left(\mathrm{m}_{\mathrm{l}}\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{l}}\right)\right) \mathrm{a}^{\prime}$ by auto
from $c^{\prime}$-sim [rule-format, of k , simplified k eq', OF - c-suc $\mathrm{c}^{\prime}$-suc] i-bound neq-j-i
obtain u-ts-i: u-ts!i = ts!i and
shared-sim: $\forall$ a. a $\notin \mathcal{O}_{1} \longrightarrow \mathrm{u}$-shared $\mathrm{a}=\mathcal{S}$ a and
mem-sim: $\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{I}} \longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m} \mathrm{a}$
by auto
from $c^{\prime}$-leq [rule-format, of $k$ c $c^{\prime}$-suc c-suc
have leq-u-ts: length $u$-ts $=$ length ts
by (auto simp add: eq ${ }^{\prime} \mathrm{k}$ )
from j-bound leq-u-ts
have j-bound-u: j < length u-ts
by simp
from i-bound leq-u-ts
have i-bound-u: i < length u-ts
by simp
from k last-bound have l-k-eq: last $+\mathrm{k}=\mathrm{n}$
by auto
from safe-delayed-reach-inter.safe-config [OF - trace-undo, simplified l-k-eq] k c-0 last-bound
have safe-delayed-c-undo': $\forall \mathrm{x} \leq \mathrm{n}$. safe-delayed (c-undo x ) by (auto simp add: c-undo split: if-split-asm)
hence safe-delayed-c-undo: $\forall \mathrm{x}<\mathrm{n}$. safe-delayed (c-undo x ) by auto
from trace-preserves-simple-ownership-distinct [OF - trace-undo, simplified l-k-eq c-undo-0, simplified, OF dist this, of n] dist c-undo-n
have dist-u-ts: simple-ownership-distinct $u$-ts by auto
then interpret dist-u-ts-inter: simple-ownership-distinct u-ts .

## \{

fix a
have u-shared $\mathrm{a}=\mathcal{S} \mathrm{a}$
proof (cases a $\in \mathcal{O}_{\text {I }}$ )
case True with shared-unch


```
        by auto
    moreover
    from True shared-orig-unch
    have }\mathcal{S}\textrm{a}=\mp@subsup{\mathcal{S}}{1}{}\textrm{a
        by auto
    ultimately show ?thesis by simp
next
    case False
    with shared-sim
    show ?thesis
        by auto
    qed
} hence u-shared-eq: u-shared =S Sy - (rule ext, auto)
{
assume safe: map owned u-ts,map released u-ts,i }\vdash(\textrm{is},\vartheta,\textrm{u}-\textrm{m},\mathcal{D},\mathcal{O},u-shared) \sqrt{}{
then have False
proof cases
    case Read
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case WriteNonVolatile
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case WriteVolatile
    then show ?thesis
    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
next
    case Fence
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        by (auto simp add:eqs races-def split: if-split-asm)
next
    case Ghost
    then show ?thesis
    using ts-i tsl-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
        apply (auto simp add:eqs races-def split: if-split-asm)
        apply fastforce
        done
next
    case (RMWReadOnly cond t a' D f ret A L R W)
    with ts-i is obtain
```

ins: ins $=$ RMW $a^{\prime} t(D, f)$ cond ret A L R W and
owned-or-shared: $\mathrm{a}^{\prime} \in \mathcal{O} \vee \mathrm{a}^{\prime} \in$ dom $u$-shared and
cond: $\neg \operatorname{cond}\left(\vartheta\left(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}^{\prime}\right)\right)$ and
rels-race: $\forall \mathrm{j}<$ length (map owned $\mathrm{u}-\mathrm{ts}) . \mathrm{i} \neq \mathrm{j} \longrightarrow(($ map released $\mathrm{u}-\mathrm{ts})!\mathrm{j})$
$a^{\prime} \neq$ Some False
by auto
from dist-u-ts-inter.simple-ownership-distinct [OF j-bound-u i-bound-u neq-j-i u-ts-j [simplified tsı-j]
u-ts-i [simplified ts-i]]
have dist: $\mathcal{O}_{1} \cap \mathcal{O}=\{ \}$
by auto
from owned-or-shared dist a-owned a-unshared shared-orig-unch
have $a^{\prime}-a$ : $a^{\prime} \neq a$
by (auto simp add: u-shared-eq domIff)
have $u-m$-eq: $u-m \mathrm{a}^{\prime}=\mathrm{m} \mathrm{a}^{\prime}$
proof (cases a' $\in \mathcal{O}_{\mathrm{l}}$ )
case True with mem-unch
have $u-m \mathrm{a}^{\prime}=\mathrm{m}_{\mathrm{I}} \mathrm{a}^{\prime}$
by auto
moreover
from True mem-unch-orig $\mathrm{a}^{\prime}-\mathrm{a}$
have $m \mathrm{a}^{\prime}=\mathrm{m}_{\mathrm{l}} \mathrm{a}^{\prime}$
by auto
ultimately show ?thesis by simp
next
case False
with mem-sim
show ?thesis
by auto
qed
with ins cond rels-race show ?thesis
using ts-i $\mathrm{ts}_{\|}-\mathrm{j}$ race is j-bound i-bound $u-\mathrm{ts}-\mathrm{i} u-\mathrm{ts}-\mathrm{j}$ leq-u-ts neq-j-i ts-j
by (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
next
case (RMWWrite cond t a' A L R D f ret W)
with ts-i is obtain
ins: ins = RMW a't (D, f) cond ret A L R W and
cond: cond $\left(\vartheta\left(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}^{\prime}\right)\right)$ and
$\mathrm{a}^{\prime}: \forall \mathrm{j}<$ length (map owned $\left.\mathrm{u}-\mathrm{ts}\right) . \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a}^{\prime} \notin($ map owned $\mathrm{u}-\mathrm{ts})!\mathrm{j} \cup$ dom ((map released u-ts)! j) and
safety:
$\mathrm{A} \subseteq$ dom u-shared $\cup \mathcal{O} \mathrm{L} \subseteq \mathrm{A} \mathrm{R} \subseteq \mathcal{O} \mathrm{A} \cap \mathrm{R}=\{ \}$
$\forall \mathrm{j}<$ length (map owned u-ts). $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A} \cap(($ map owned $\mathrm{u}-\mathrm{ts})!\mathrm{j} \cup$ dom $(($ map released u-ts $)!\mathrm{j}))=\{ \}$ $\mathrm{a}^{\prime} \notin$ read-only u-shared
by auto
from a'[rule-format, of j] j-bound-u u-ts-j tsı-j neq-j-i
have $a^{\prime} \notin \mathcal{O}_{\text {l }}$
by auto
from mem-sim [rule-format, OF this]

```
            have u-m-eq: u-m a'= m a'
                    by auto
            with ins cond safety a' show ?thesis
            using ts-i ts|-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
                    apply (auto simp add:eqs races-def u-shared-eq u-m-eq split: if-split-asm)
                    apply fastforce
                    done
        next
            case Nil
            then show ?thesis
            using ts-i tsı-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
            by (auto simp add:eqs races-def split: if-split-asm)
        qed
    }
    hence \neg safe-delayed (u-ts, u-m, u-shared)
        apply (clarsimp simp add: safe-delayed-def)
        apply (rule-tac x=i in exI)
        using u-ts-i ts-i i-bound-u
        apply auto
        done
    moreover
    from safe-delayed-c-undo' [rule-format, of n] c-undo-n
    have safe-delayed (u-ts, u-m, u-shared)
        by simp
    ultimately have False
        by simp
    thus ?thesis
        by simp
    qed
next
    case WriteVolatile
    with ts,}\mp@subsup{}{\prime}{\prime}-j t\mp@subsup{t}{|}{\prime}\mp@subsup{}{}{\prime}\mathrm{ ts-j j-bound, have }\mp@subsup{\mathcal{R}}{\textrm{j}}{}=\mathrm{ Map.empty
        by auto
    with }\mp@subsup{\mathcal{R}}{\textrm{j}}{\textrm{j}}\mathrm{ -non-empty have False by auto
    thus ?thesis ..
next
    case Fence
    with ts|}\mp@subsup{}{|}{\prime}\textrm{j} t\mp@subsup{t}{|}{\prime}\mp@subsup{}{}{\prime}\mathrm{ ts-j j-bound, have }\mp@subsup{\mathcal{R}}{\textrm{j}}{}=\mathrm{ Map.empty
        by auto
    with }\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ -non-empty have False by auto
    thus ?thesis ..
next
    case RMWReadOnly
    with ts|}\mp@subsup{}{}{\prime}-\textrm{j} t\mp@subsup{\textrm{t}}{|}{\prime}\mp@subsup{}{}{\prime}\mathrm{ ts-j j-bound,}\mathrm{ have }\mp@subsup{\mathcal{R}}{\textrm{j}}{}=\mathrm{ Map.empty
        by auto
    with }\mp@subsup{\mathcal{R}}{\textrm{j}}{\textrm{j}}\mathrm{ -non-empty have False by auto
    thus ?thesis ..
next
    case RMWWrite
```

```
with ts \({ }^{\prime}-\mathrm{j}\) ts \({ }^{\prime}{ }^{\prime}\) ts-j j-bound, have \(\mathcal{R}_{\mathrm{j}}=\) Map.empty
by auto
with \(\mathcal{R}_{\mathrm{j}}\)-non-empty have False by auto
thus ?thesis ..
next
    case (Ghost A L R W)
    then obtain
        is \(\left.\right|_{\text {: }}\) is \(\mid=\) Ghost A L R W \# is \({ }^{\prime}\) 'and
        \(\vartheta_{1}^{\prime}: \vartheta_{1}^{\prime}=\vartheta_{1}\) and
        \(\mathrm{sb}_{1}{ }^{\prime}: \mathrm{sb}_{1}{ }^{\prime}=\mathrm{sb}_{1}\) and
        \(\mathcal{D}_{1}: \mathcal{D}_{1}{ }^{\prime}=\mathcal{D}_{1}\) and
        \(\mathcal{O}_{1}^{\prime}: \mathcal{O}_{1}^{\prime}=\mathcal{O}_{1} \cup \mathrm{~A}-\mathrm{R}\) and
        \(\mathcal{R}_{1}: \mathcal{R}_{1}{ }^{\prime}=\) augment-rels \(\left(\operatorname{dom} \mathcal{S}_{1}\right) \mathrm{R} \mathcal{R}_{1}\) and
        \(\mathcal{S}_{1}: \mathcal{S}_{1}{ }^{\prime}=\mathcal{S}_{1} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) and
        \(m_{1}: m_{1}{ }^{\prime}=m_{1}\)
        by auto
    note eqs \({ }^{\prime}=\vartheta_{\mid}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathcal{D}^{\prime}{ }^{\prime} \mathcal{O}_{\mid}{ }^{\prime} \mathcal{R}_{1}{ }^{\prime} \mathcal{S}^{\prime}{ }^{\prime} \mathrm{m}_{\mid}{ }^{\prime}\)
    from ts \({ }^{\prime}-\mathrm{j}\) ts \({ }^{\prime}{ }^{\prime}\) ts-j j-bound, eqs \({ }^{\prime}\)
    obtain eqs: \(\mathrm{p}_{\mathrm{l}}=\mathrm{p}_{\mathrm{j}}\) is \({ }^{\prime}{ }^{\prime}=\mathrm{is} \mathrm{s}_{\mathrm{j}} \vartheta_{\mathrm{l}}=\vartheta_{\mathrm{j}} \mathcal{D}_{\mathrm{l}}=\mathcal{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{l}} \cup \mathrm{A}-\mathrm{R}=\mathcal{O}_{\mathrm{j}}\)
        augment-rels (dom \(\left.\mathcal{S}_{\mathrm{I}}\right) \mathrm{R} \mathcal{R}_{\mathrm{I}}=\mathcal{R}_{\mathrm{j}}\)
        by auto
```

    from safe, [simplified isı]
    obtain
    A-shared-owned: \(\mathrm{A} \subseteq \operatorname{dom} \mathcal{S}_{\mid} \cup \mathcal{O}_{\mid}\)and L - \(\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}\) and R -owns: \(\mathrm{R} \subseteq \mathcal{O}_{\text {I }}\) and
    $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
$\forall \mathrm{j}^{\prime}<$ length (map owned $\left.\mathrm{ts}_{\mathrm{I}}\right) . \mathrm{j} \neq \mathrm{j}^{\prime} \longrightarrow \mathrm{A} \cap\left(\left(\right.\right.$ map owned $\left.\mathrm{ts}_{\mathrm{I}}\right)!\mathrm{j}^{\prime} \cup$ dom ((map
released $\left.\left.\left.\mathrm{ts}_{\mathrm{I}}\right)!\mathrm{j}^{\prime}\right)\right)=\{ \}$
by cases auto
from A-shared-owned L-A R-owns A-R
have shared-eq: $\forall \mathrm{a}$. $\mathrm{a} \notin \mathcal{O}_{\mathrm{I}} \longrightarrow \mathrm{a} \notin \mathcal{O}_{\mathrm{I}}{ }^{\prime} \longrightarrow \mathcal{S}_{\mathrm{I}} \mathrm{a}=\left(\mathcal{S}_{\mathrm{I}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$
by (auto simp add: restrict-shared-def augment-shared-def $\mathcal{O}^{\prime}$ ' split: option.splits)
from undo-local-steps [where $\mathrm{c}=\mathrm{c}$, OF trace-rest c-last' idle-rest' safe-delayed-last, simplified $\mathrm{ts}_{1}{ }^{\prime}$,
simplified,
OF j-bound ${ }_{\mathrm{t}} \mathrm{ts}_{\mathrm{l}} \mathrm{j}$ [simplified], simplified $\mathrm{m}^{\prime}{ }^{\prime} \mathcal{S}^{\prime}{ }^{\prime}$, simplified,
OF shared-eq dist-last dist-last' [simplified ts ${ }^{\prime}$,simplified] safe-delayed-upto-last]
obtain c' $k$ where
k-bound: $\mathrm{k} \leq \mathrm{n}$ - last and
trace-c': trace c ${ }^{\prime}$ (Suc last) k and
$c^{\prime}$-first: $\mathrm{c}^{\prime}($ Suc last $)=\left(\mathrm{ts}, \mathrm{m}_{\mathrm{l}}, \mathcal{S}_{\mathrm{l}}\right)$ and
$c^{\prime}$-leq: $\left(\forall \mathrm{x} \leq \mathrm{k}\right.$. length $\left(\right.$ fst $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.\left.))\right)\right)=$ length (fst $(\mathrm{c}($ Suc $($ last + $\mathrm{x})$ ))) ) and
$c^{\prime}$-safe: $\left(\forall \mathrm{x}<\mathrm{k}\right.$. safe-delayed $\left(\mathrm{c}^{\prime}(\right.$ Suc $($ last +x$\left.\left.))\right)\right)$ and
$c^{\prime}$-unsafe: $\left(\mathrm{k}<\mathrm{n}-\right.$ last $\longrightarrow \neg$ safe-delayed ( $\mathrm{c}^{\prime}($ Suc $($ last +k$)$ )) ) and
$c^{\prime}$-unch:

```
( }\forall\textrm{x}\leq\textrm{k}.\forall\mp@subsup{\textrm{ts}}{\textrm{x}}{}\mp@subsup{\mathcal{S}}{\textrm{x}}{}\mp@subsup{\textrm{m}}{\textrm{x}}{}
    c (Suc (last + x)) = (tsx},\mp@subsup{m}{\textrm{x}}{},\mp@subsup{\mathcal{S}}{\textrm{x}}{})
    (}\forall\mp@subsup{\textrm{ts}}{\textrm{x}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{x}}{\prime}\mp@subsup{\textrm{m}}{\textrm{x}}{\prime
            c
            tsx
            (\forall\textrm{a}\in\mp@subsup{\mathcal{O}}{|}{}.\mp@subsup{\mathcal{S}}{x}{\prime}\mp@subsup{}{}{\prime}\textrm{a}=\mp@subsup{\mathcal{S}}{I}{}\textrm{a})\wedge
```




```
    c'-sim:
    (}\forall\textrm{x}\leq\textrm{k}.\forall\mp@subsup{\textrm{ts}}{\textrm{x}}{}\mp@subsup{\mathcal{S}}{\textrm{x}}{}\mp@subsup{\textrm{m}}{\textrm{x}}{}
            c (Suc (last + x)) = (ts. ( m
            ( }\forall\mp@subsup{\textrm{ts}}{\textrm{x}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{x}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{m}{x}{\prime}\mp@subsup{}{}{\prime}
                c}\mp@subsup{}{}{\prime}(\mathrm{ Suc (last + x ) ) = (tsx}\mp@subsup{}{}{\prime},\mp@subsup{\textrm{m}}{\textrm{x}}{\prime},\mp@subsup{\mathcal{S}}{\textrm{x}}{\prime})
                ( }\forall\textrm{ja}<l=length tsx. . ja # j \longrightarrow tsxx ! ja = tsx ! ja) ^
                (\forall\textrm{a}.\textrm{a}\not\in\mp@subsup{\mathcal{O}}{I}{}\longrightarrow\textrm{a}\not\in\mp@subsup{\mathcal{O}}{1}{\prime}\longrightarrow
                (\foralla.a }\not\in\mp@subsup{\mathcal{O}}{I}{}\longrightarrow\mp@subsup{m}{x}{\prime}\mp@subsup{}{}{\prime}\textrm{a}=\mp@subsup{\textrm{m}}{\textrm{x}}{}\textrm{a}))
    by (clarsimp)
    obtain c-undo where c-undo: c-undo = ( }\lambda\textrm{x}\mathrm{ . if }\textrm{x}\leql\mathrm{ last then c x else c' (Suc
last + x - last))
    by blast
    have c-undo-0: c-undo 0 = co
    by (auto simp add: c-undo c-0)
from sequence-traces [OF trace-last trace-c', simplified c-last, OF c'-first c-undo]
    have trace-undo: trace c-undo 0 (last + k).
    obtain u-ts u-shared u-m where
        c-undo-n: c-undo n = (u-ts,u-m, u-shared)
        by (cases c-undo n)
    with last-bound c'-first c-last
    have c'-suc: c' (Suc n) = (u-ts,u-m, u-shared)
    apply (auto simp add: c-undo split: if-split-asm)
    apply (subgoal-tac n=last)
    apply auto
    done
    show ?thesis
    proof (cases k < n - last)
    case True
    with c'-unsafe have unsafe: }\neg\mathrm{ safe-delayed (c-undo (last + k))
        by (auto simp add: c-undo c-last c'-first)
    from True have last + k \leqn
        by auto
    from safe-delayed-reach-inter.safe-config [OF this trace-undo, of last + k]
    have safe-delayed (c-undo (last + k))
        by (auto simp add: c-undo c-0)
    with unsafe have False by simp
    thus ?thesis ..
next
    case False
```

with k -bound have $\mathrm{k}: \mathrm{k}=\mathrm{n}$ - last by auto
have eq': Suc (last $+(\mathrm{n}-$ last $))=$ Suc $n$
using last-bound
by simp
from $\mathrm{c}^{\prime}$-unch [rule-format, of k , simplified $\mathrm{keq}{ }^{\prime}$, OF - c-suc $\mathrm{c}^{\prime}$-suc]
obtain u-ts-j: u-ts! $\mathrm{j}=\mathrm{ts}!!\mathrm{j}$ and
shared-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mid}$. u-shared $\mathrm{a}=\mathcal{S}_{\mathrm{I}}$ a and shared-orig-unch: $\forall \mathrm{a} \in \mathcal{O}_{\mid} . \mathcal{S} \mathrm{a}=\left(\mathcal{S}_{\mathrm{I}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ a and mem-unch: $\forall \mathrm{a} \in \mathcal{O}_{I} . \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}_{\mathrm{I}}$ a and mem-unch-orig: $\forall \mathrm{a}^{\prime} \in \mathcal{O}_{\mid} . \mathrm{m} \mathrm{a}^{\prime}=\mathrm{m}_{\mathrm{l}} \mathrm{a}^{\prime}$
by auto
from $\mathrm{c}^{\prime}$-sim [rule-format, of k , simplified k eq ${ }^{\prime}$, OF - c-suc $\mathrm{c}^{\prime}$-suc] i -bound neq-j-i
obtain u-ts-i: u-ts! $=$ ts!i and

$$
\text { shared-sim: } \forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{a} \notin \mathcal{O}_{\mathrm{l}}^{\prime} \longrightarrow \mathrm{u} \text {-shared } \mathrm{a}=\mathcal{S} \mathrm{a} \text { and }
$$ mem-sim: $\forall \mathrm{a} . \mathrm{a} \notin \mathcal{O}_{\mathrm{l}} \longrightarrow \mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{ma}$

by auto
from $c^{\prime}$-leq [rule-format, of $k$ ] $c^{\prime}$-suc c-suc
have leq-u-ts: length $u$-ts $=$ length ts
by (auto simp add: $\mathrm{eq}^{\prime} \mathrm{k}$ )
from j-bound leq-u-ts
have j-bound-u: $\mathrm{j}<$ length $u-$ ts
by simp
from i-bound leq-u-ts
have i-bound-u: i < length u-ts
by simp
from k last-bound have l-k-eq: last $+\mathrm{k}=\mathrm{n}$ by auto
from safe-delayed-reach-inter.safe-config [OF - trace-undo, simplified l-k-eq] k c-0 last-bound
have safe-delayed-c-undo $: ~ \forall \mathrm{x} \leq \mathrm{n}$. safe-delayed (c-undo x )
by (auto simp add: c-undo split: if-split-asm)
hence safe-delayed-c-undo: $\forall \mathrm{x}<\mathrm{n}$. safe-delayed (c-undo x ) by auto
from trace-preserves-simple-ownership-distinct [OF - trace-undo, simplified l-k-eq c-undo-0, simplified, OF dist this, of $n$ ] dist c-undo-n
have dist-u-ts: simple-ownership-distinct $u$-ts by auto
then interpret dist-u-ts-inter: simple-ownership-distinct u-ts .
\{
fix a
have $\mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m} \mathrm{a}$
proof (cases a $\in \mathcal{O}_{\mathrm{l}}$ )
case True with mem-unch
have $\mathrm{u}-\mathrm{m} \mathrm{a}=\mathrm{m}_{\mathrm{l}} \mathrm{a}$
by auto
moreover

```
            from True mem-unch-orig
            have ma= mla
                    by auto
            ultimately show ?thesis by simp
        next
            case False
            with mem-sim
            show ?thesis
            by auto
    qed
} hence u-m-eq: u-m = m by - (rule ext, auto)
{
assume safe: map owned u-ts,map released u-ts,i}\vdash(is,\vartheta,u-m,\mathcal{D},\mathcal{O},u-shared) \sqrt{}{
    then have False
    proof cases
        case (Read a volatile t)
        with ts-i is obtain
            ins: ins = Read volatile a t and
            access-cond: a }\in\mathcal{O}\vee\textrm{a}\in\mathrm{ read-only u-shared }\vee\mathrm{ volatile }\wedge\textrm{a}\in\mathrm{ dom u-shared
                    rels-cond: }\forall\textrm{j}<\mathrm{ length u-ts. i }\not=\textrm{j}\longrightarrow((\mathrm{ map released u-ts) ! j) a }\not=\mathrm{ Some
                    rels-non-volatile-cond: }\neg\mathrm{ volatile }\longrightarrow(\forall\textrm{j}<\mathrm{ length u-ts. i }\not=\textrm{j}\longrightarrow\textrm{a}\not\in\mathrm{ dom
((map released u-ts)! j)) and
            clean: volatile }\longrightarrow\neg\mathcal{D
            by auto
        from race ts-j
```



```
                        ( }\neg\mathrm{ volatile }\wedge\textrm{a}\in\operatorname{dom}\mathrm{ (augment-rels (dom }\mp@subsup{\mathcal{S}}{\textrm{I}}{\prime})\textrm{R}\mp@subsup{\mathcal{R}}{\textrm{l}}{\prime})
                            by (auto simp add: races-def ins eqs)
                            from rels-cond [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]]
    have }\mp@subsup{\mathcal{R}}{1}{}-\textrm{a}:\mp@subsup{\mathcal{R}}{1}{}\textrm{a}\not=\mathrm{ Some False
            by auto
                    from dist-u-ts-inter.simple-ownership-distinct [OF j-bound-u i-bound-u
neq-j-i u-ts-j [simplified tsl-j]
    u-ts-i [simplified ts-i]]
        have dist: }\mp@subsup{\mathcal{O}}{|}{}\cap\mathcal{O}={
            by auto
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
            show ?thesis
            proof (cases a }\inR\mathrm{ )
            case False
            with rc }\mp@subsup{\mathcal{R}}{1}{}-\textrm{a}\mathrm{ show False
                    by (auto simp add: augment-rels-def volatile)
```

    and
    False and
u-ts-j ts_-j j-bound-u

```
    next
    case True
    with R-owns
    have a-owns|: a }\in\mathcal{O
        by auto
    from shared-unch [rule-format, OF a-owns!]
    have u-shared-eq: u-shared a}=\mp@subsup{\mathcal{S}}{|}{}\mathrm{ a
        by auto
    from a-owns, dist have a }\not\in\mathcal{O
        by auto
    moreover
    {
        assume a }\in\mathrm{ read-only u-shared
        with u-shared-eq have }\mp@subsup{\mathcal{S}}{|}{}\textrm{a}=\mathrm{ Some False
            by (auto simp add: read-only-def)
        with rc True }\mp@subsup{\mathcal{R}}{1}{}-\textrm{a}\mathrm{ have False
            by (auto simp add: augment-rels-def split: option.splits simp add:
domIff volatile)
    }
    moreover
    {
        assume a }\in\mathrm{ dom u-shared
        with u-shared-eq rc True }\mp@subsup{\mathcal{R}}{1}{}-\textrm{a}\mathrm{ have False
            by (auto simp add: augment-rels-def split: option.splits simp add:
domIff volatile)
    }
    ultimately show False
    using access-cond
                by auto
        qed
    next
        case False
        note non-volatile = this
            from rels-non-volatile-cond [rule-format, OF False j-bound-u neq-j-i
[symmetric]] u-ts-j tsl-j j-bound-u
    have }\mp@subsup{\mathcal{R}}{\}{}-\textrm{a}:\mp@subsup{\mathcal{R}}{\}{}\textrm{a}=\mathrm{ None
        by (auto simp add: domIff)
    show ?thesis
    proof (cases a }\inR\mathrm{ )
        case False
        with rc }\mp@subsup{\mathcal{R}}{\dagger}{}-\mathrm{ -a show False
            by (auto simp add: augment-rels-def non-volatile domIff)
    next
        case True
        with R-owns
        have a-owns!: a }\in\mathcal{O
            by auto
        from shared-unch [rule-format, OF a-owns,]
        have u-shared-eq: u-shared a}=\mp@subsup{\mathcal{S}}{|}{}\textrm{a
            by auto
```

from a-owns। dist have a-unowned: a $\notin \mathcal{O}$
by auto
moreover
from ro-last-last interpret
read-only-unowned $\mathcal{S}_{\mathbf{I}} \mathrm{ts}_{\mid}$.
from read-only-unowned [ OF j -bound, $\mathrm{ts}_{\mid-\mathrm{j}}$ ] a-owns| have a-unsh: a $\notin$ read-only $\mathcal{S}_{\text {I }}$ by auto
\{
assume a $\in$ read-only u-shared
with u-shared-eq have sh: $\mathcal{S}_{1} \mathrm{a}=$ Some False
by (auto simp add: read-only-def)
with rc True $\mathcal{R}_{1}$-a access-cond $u$-shared-eq a-unowned sh a-owns| a-unsh have False
by (auto simp add: augment-rels-def split: option.splits simp add: domIff non-volatile read-only-def)
\}
moreover
\{ assume $\mathrm{a} \in$ dom u-shared
with u-shared-eq re True $\mathcal{R}_{1}$-a a-owns| a-unsh access-cond dist have
False
by (auto simp add: augment-rels-def split: option.splits simp add: domIff non-volatile read-only-def)
\}
ultimately show False
using access-cond by (auto)
qed
qed
next
case (WriteNonVolatile a D f A ${ }^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ )
with ts-i is obtain
ins: ins $=$ Write False a $(D, f) A^{\prime} L^{\prime} R^{\prime} W^{\prime}$ and
a-owned: $\mathrm{a} \in \mathcal{O}$ and a-unshared: $\mathrm{a} \notin \operatorname{dom} u$-shared and
a-unreleased: $\forall \mathrm{j}<$ length u -ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \operatorname{dom}(($ map released $\mathrm{u}-\mathrm{ts})!\mathrm{j})$
by auto
from dist-u-ts-inter.simple-ownership-distinct [OF j-bound-u i-bound-u neq-j-i u-ts-j [simplified ts|-j]
u-ts-i [simplified ts-i]]
have dist: $\mathcal{O}_{1} \cap \mathcal{O}=\{ \}$
by auto
from race ts-j
have rc: a $\in \operatorname{dom}$ (augment-rels ( $\operatorname{dom} \mathcal{S}_{1}$ ) R $\mathcal{R}_{1}$ )
by (auto simp add: races-def ins eqs)
from a-unreleased [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]] u-ts-j ts|-j j-bound-u
have $\mathcal{R}_{1}$-a: a $\notin \operatorname{dom} \mathcal{R}_{1}$
by auto
show False

```
proof (cases a \(\in R\) )
    case False
    with rc \(\mathcal{R}_{1-}\)-a show False
                by (auto simp add: augment-rels-def domIff)
    next
    case True
    with R-owns
    have a-owns|: \(\mathrm{a} \in \mathcal{O}_{\text {। }}\)
                by auto
    with a-owned dist show False
                by auto
    qed
next
case (WriteVolatile a \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{D}\) f \(\mathrm{W}^{\prime}\) )
with ts-i is obtain
ins: ins \(=\) Write True a \((\mathrm{D}, \mathrm{f}) \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) and
    a-un-owned-released: \(\forall \mathrm{j}<\) length u -ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\)
            \(\mathrm{a} \notin((\) map owned \(u-t s)!j) \wedge a \notin \operatorname{dom}((\) map released u-ts) ! j) and
    \(\mathrm{A}^{\prime}\)-owns-shared: \(\mathrm{A}^{\prime} \subseteq\) dom u-shared \(\cup \mathcal{O}\) and
    \(\mathrm{L}^{\prime}-\mathrm{A}^{\prime}: \mathrm{L}^{\prime} \subseteq \mathrm{A}^{\prime}\) and
    \(\mathrm{R}^{\prime}\)-owned: \(\mathrm{R}^{\prime} \subseteq \mathcal{O}\) and
    \(A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}\) and
    acq-ok: \(\forall \mathrm{j}<\) length u -ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A}^{\prime} \cap((\) map owned \(\mathrm{u}-\mathrm{ts})!\mathrm{j} \cup \operatorname{dom}((\) map
```

released $u-t s)!j))=\{ \}$ and
writeable: a $\notin$ read-only u-shared
by auto
from a-un-owned-released [rule-format, simplified, OF j-bound-u neq-j-i
[symmetric]] u-ts-j ts|-j j-bound-u
obtain $\mathcal{O}_{\mid}$-a: a $\notin \mathcal{O}_{\mathrm{l}}$ and $\mathcal{R}_{1}$-a: a $\notin \operatorname{dom}\left(\mathcal{R}_{1}\right)$
by auto
from acq-ok [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]] u-ts-j
tsl-j j-bound-u
obtain $\mathcal{O}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{I}}=\{ \}$ and $\mathcal{R}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \operatorname{dom}\left(\mathcal{R}_{\mathrm{I}}\right)=\{ \}$
by auto
\{
assume rc: a $\in \operatorname{dom}$ (augment-rels $\left.\left(\operatorname{dom} \mathcal{S}_{1}\right) \mathrm{R} \mathcal{R}_{1}\right)$
have False
proof (cases a $\in R$ )
case False
with rc $\mathcal{R}_{1-\mathrm{a}}$ show False
by (auto simp add: augment-rels-def domIff)
next
case True
with R-owns
have a-owns|: $\mathrm{a} \in \mathcal{O}_{\text {I }}$
by auto
with $\mathcal{O}_{1}$-a show False
by auto
qed

## \}

moreover
\{
assume rc: $\mathrm{A}^{\prime} \cap$ dom (augment-rels $\left.\left(\operatorname{dom} \mathcal{S}_{1}\right) \mathrm{R} \mathcal{R}_{1}\right) \neq\{ \}$
then obtain $a^{\prime}$ where $a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}$ and $a^{\prime}-$ aug: $a^{\prime} \in \operatorname{dom}$ (augment-rels
$\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{I}}\right) \mathrm{R} \mathcal{R}_{\mathrm{I}}\right)$
by auto
have False
proof (cases $\mathrm{a}^{\prime} \in \mathrm{R}$ )
case False
with $\mathrm{a}^{\prime}-\mathrm{aug} \mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathcal{R}_{1}-\mathrm{A}^{\prime}$ show False
by (auto simp add: augment-rels-def domIff)
next
case True
with R -owns have $\mathrm{a}^{\prime}$-owns!: $\mathrm{a}^{\prime} \in \mathcal{O}_{\text {। }}$
by auto
with $\mathcal{O}_{1}-\mathrm{A}^{\prime} \mathrm{a}^{\prime}-\mathrm{A}^{\prime}$ show False
by auto
qed
\}
ultimately show False
using race ts-j
by (auto simp add: races-def ins eqs)

## next

case Fence
then show ?thesis
using ts-i ts ${ }_{1}$-j race is j-bound i -bound u -ts-i $u$-ts-j leq-u-ts neq-j-i ts-j
by (auto simp add:eqs races-def split: if-split-asm)
next
case (Ghost A' L' R' W')
with ts-i is obtain
ins: ins $=$ Ghost $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ and
$\mathrm{A}^{\prime}$-owns-shared: $\mathrm{A}^{\prime} \subseteq$ dom u-shared $\cup \mathcal{O}$ and
$\mathrm{L}^{\prime}-\mathrm{A}^{\prime}: \mathrm{L}^{\prime} \subseteq \mathrm{A}^{\prime}$ and
$\mathrm{R}^{\prime}$-owned: $\mathrm{R}^{\prime} \subseteq \mathcal{O}$ and
$A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}$ and
acq-ok: $\forall \mathrm{j}<$ length u -ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A}^{\prime} \cap(($ map owned $\mathrm{u}-\mathrm{ts})!\mathrm{j} \cup \operatorname{dom}$ ((map
released $u$-ts) $!\mathrm{j})$ ) $=\{ \}$
by auto
from acq-ok [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]] u-ts-j
ts 1 -j j-bound-u
obtain $\mathcal{O}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{I}}=\{ \}$ and $\mathcal{R}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \operatorname{dom}\left(\mathcal{R}_{1}\right)=\{ \}$
by auto
from race ts-j
obtain $a^{\prime}$ where $a^{\prime}-\mathrm{A}^{\prime}: \mathrm{a}^{\prime} \in \mathrm{A}^{\prime}$ and
$\mathrm{a}^{\prime}$-aug: $\mathrm{a}^{\prime} \in \operatorname{dom}$ (augment-rels (dom $\mathcal{S}_{1}$ ) R $\mathcal{R}_{1}$ )
by (auto simp add: races-def ins eqs)
show False
proof (cases $a^{\prime} \in R$ )

```
            case False
            with a'-aug a'-A' }\mp@subsup{\mathcal{R}}{1}{}-\mp@subsup{A}{}{\prime}\mathrm{ 'show False
                by (auto simp add: augment-rels-def domIff)
    next
            case True
    with R-owns have a'-owns|: a' 
                by auto
    with }\mp@subsup{\mathcal{O}}{I}{}-\mp@subsup{A}{}{\prime}\mp@subsup{a}{}{\prime}-\mp@subsup{A}{}{\prime}\mathrm{ show False
                by auto
    qed
next
    case (RMWReadOnly cond t a D f ret A' L' R' W')
    with ts-i is obtain
        ins: ins = RMW a t (D,f) cond ret A' L' R' W' and
        owned-or-shared: a }\in\mathcal{O}\veea\in\operatorname{dom}u\mathrm{ u-shared and
        cond: \neg cond (\vartheta(t \mapsto u-m a)) and
    rels-race: }\forall\textrm{j}<length (map owned u-ts). i f j \longrightarrow ((map released u-ts)!j)
a}\not=\mathrm{ Some False
    by auto
                            from dist-u-ts-inter.simple-ownership-distinct [OF j-bound-u i-bound-u
neq-j-i u-ts-j [simplified ts|-j]
    u-ts-i [simplified ts-i]]
    have dist: }\mathcal{O},\cap\mathcal{O}={
    by auto
    from race ts-j cond
    have rc: augment-rels (dom }\mp@subsup{\mathcal{S}}{|}{})\mathrm{ R }\mp@subsup{\mathcal{R}}{\}{}\mathrm{ a = Some False
    by (auto simp add: races-def ins eqs u-m-eq)
    from rels-race [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]]
    u-ts-j ts_-j j-bound-u
    have }\mp@subsup{\mathcal{R}}{1}{-}\textrm{a}:\mp@subsup{\mathcal{R}}{|}{}\mathrm{ a }\not=\mathrm{ Some False
    by auto
    show ?thesis
    proof (cases a }\inR\mathrm{ )
    case False
    with rc }\mp@subsup{\mathcal{R}}{\}{\prime}\mathrm{ -a show False
        by (auto simp add: augment-rels-def)
    next
    case True
    with R-owns
    have a-owns|: a }\in\mathcal{O
        by auto
    from shared-unch [rule-format, OF a-owns।]
    have u-shared-eq: u-shared a = \mathcal{S}
        by auto
    from a-owns, dist have a }\not\in\mathcal{O
            by auto
    with u-shared-eq rc True }\mp@subsup{\mathcal{R}}{\}{\prime}\mathrm{ -a owned-or-shared show False
        by (auto simp add: augment-rels-def split: option.splits simp add: domIff)
```

qed
next
case (RMWWrite cond t a $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{D}$ f ret $\mathrm{W}^{\prime}$ )
with ts-i is obtain
ins: ins $=$ RMW at $(\mathrm{D}, \mathrm{f})$ cond ret $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ and
cond: cond $(\vartheta(\mathrm{t} \mapsto \mathrm{u}-\mathrm{m} \mathrm{a}))$ and
a-un-owned-released: $\forall \mathrm{j}<$ length (map owned u -ts). $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin$ (map
owned $u-t s)!\mathrm{j} \cup$ dom ((map released $u-\mathrm{ts})!\mathrm{j}$ ) and
A'-owns-shared $: A^{\prime} \subseteq$ dom u-shared $\cup \mathcal{O}$ and
$\mathrm{L}^{\prime}-\mathrm{A}^{\prime}: \mathrm{L}^{\prime} \subseteq \mathrm{A}^{\prime}$ and
$\mathrm{R}^{\prime}$-owned: $\mathrm{R}^{\prime} \subseteq \mathcal{O}$ and
$A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}$ and
acq-ok: $\forall \mathrm{j}<$ length (map owned $u$-ts). $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{A}^{\prime} \cap(($ map owned $\mathrm{u}-\mathrm{ts})!\mathrm{j}$
$\cup \operatorname{dom}(($ map released $u$-ts) $!\mathrm{j}))=\{ \}$ and
writeable: a $\notin$ read-only u-shared
by auto
from a-un-owned-released [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]] u-ts-j tsl-j j-bound-u
obtain $\mathcal{O}_{1}$-a: a $\notin \mathcal{O}_{1}$ and $\mathcal{R}_{1}$-a: a $\notin \operatorname{dom}\left(\mathcal{R}_{1}\right)$
by auto
from acq-ok [rule-format, simplified, OF j-bound-u neq-j-i [symmetric]] u-ts-j ts $_{1}-\mathrm{j}$ j-bound-u
obtain $\mathcal{O}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{I}}=\{ \}$ and $\mathcal{R}_{1}-\mathrm{A}^{\prime}: \mathrm{A}^{\prime} \cap \operatorname{dom}\left(\mathcal{R}_{1}\right)=\{ \}$
by auto
\{
assume rc: $\mathrm{a} \in \operatorname{dom}$ (augment-rels $\left.\left(\operatorname{dom} \mathcal{S}_{1}\right) \mathrm{R} \mathcal{R}_{\mathbf{1}}\right)$
have False
proof (cases a $\in R$ )
case False
with rc $\mathcal{R}_{1}$-a show False
by (auto simp add: augment-rels-def domIff)
next
case True
with R-owns
have a-owns|: $a \in \mathcal{O}_{\text {I }}$
by auto
with $\mathcal{O}_{1-a}$ show False
by auto
qed
\}
moreover
\{
assume rc: $\mathrm{A}^{\prime} \cap \operatorname{dom}\left(\right.$ augment-rels $\left.\left(\operatorname{dom} \mathcal{S}_{1}\right) \mathrm{R} \mathcal{R}_{\mathrm{I}}\right) \neq\{ \}$
then obtain $a^{\prime}$ where $a^{\prime}-\mathrm{A}^{\prime}: \mathrm{a}^{\prime} \in \mathrm{A}^{\prime}$ and $\mathrm{a}^{\prime}-\mathrm{aug}: \mathrm{a}^{\prime} \in \operatorname{dom}$ (augment-rels $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{I}}\right) \mathrm{R} \mathcal{R}_{\mathrm{I}}\right)$
by auto
have False
proof (cases $\mathrm{a}^{\prime} \in \mathrm{R}$ )

```
                    case False
                        with a'-aug a'-A' }\mp@subsup{\mathcal{R}}{1}{}-\mp@subsup{\textrm{A}}{}{\prime}\mathrm{ show False
                        by (auto simp add: augment-rels-def domIff)
                next
                        case True
                        with R-owns have a'-owns!: a' }\in\mathcal{O
                        by auto
                    with}\mp@subsup{\mathcal{O}}{1}{}-\mp@subsup{\textrm{A}}{}{\prime}\mp@subsup{\textrm{a}}{}{\prime}-\mp@subsup{\textrm{A}}{}{\prime}\mathrm{ show False
                        by auto
                qed
            }
            ultimately show False
            using race ts-j cond
                    by (auto simp add: races-def ins eqs u-m-eq)
            next
            next
                case Nil
                        then show ?thesis
                    using ts-i ts_-j race is j-bound i-bound u-ts-i u-ts-j leq-u-ts neq-j-i ts-j
                    by (auto simp add:eqs races-def split: if-split-asm)
                    qed
                }
            hence }\neg\mathrm{ safe-delayed (u-ts, u-m, u-shared)
                    apply (clarsimp simp add: safe-delayed-def)
                    apply (rule-tac x=i in exI)
                    using u-ts-i ts-i i-bound-u
                    apply auto
                    done
                    moreover
            from safe-delayed-c-undo' [rule-format, of n] c-undo-n
            have safe-delayed (u-ts, u-m, u-shared)
                    by simp
                    ultimately have False
                    by simp
                    thus ?thesis
                    by simp
            qed
        qed
    next
        case (StoreBuffer - p is \vartheta sb \mathcal{D O R sb' O}
        hence False
            by (auto simp add: empty-storebuffer-step-def)
        thus ?thesis ..
    qed
    qed
    }
    ultimately show ?thesis
    using last-action-of-thread [where i=j, OF trace]
        by blast
qed
```


## qed

datatype 'p memref =
Write ${ }_{\text {sb }}$ bool addr sop val acq lcl rel wrt
| $\operatorname{Read}_{\text {sb }}$ bool addr tmp val
| Prog $_{\mathrm{sb}}$ 'p 'p instrs
| Ghostsb acq lcl rel wrt
type-synonym 'p store-buffer $=$ ' p memref list
inductive flush-step:: memory $\times$ 'p store-buffer $\times$ owns $\times$ rels $\times$ shared $\Rightarrow$ memory $\times$ ' p
store-buffer $\times$ owns $\times$ rels $\times$ shared $\Rightarrow$ bool
$\left(-\rightarrow_{f}-[60,60] 100\right)$
where
Write $_{\text {sb }}: \llbracket \mathcal{O}^{\prime}=($ if volatile then $\mathcal{O} \cup \mathrm{A}-\mathrm{R}$ else $\mathcal{O})$;
$\mathcal{S}^{\prime}=\left(\right.$ if volatile then $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ else $\mathcal{S}$ );
$\mathcal{R}^{\prime}=($ if volatile then Map.empty else $\mathcal{R}) \rrbracket$
$\Longrightarrow$
( m, Write $_{\mathrm{sb}}$ volatile a sop v A L R W\# rs, $\left.\mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}\left(\mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathrm{rs}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
$\mid \operatorname{Read}_{\mathrm{sb}}:\left(\mathrm{m}, \operatorname{Read}_{\mathrm{sb}}\right.$ volatile at $\left.\mathrm{v} \# \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}(\mathrm{m}, \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
$\operatorname{Prog}_{\mathrm{sb}}:\left(\mathrm{m}, \operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $\left.\# \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}(\mathrm{m}, \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
| Ghost: $\left(\mathrm{m}\right.$, Ghost $\left._{\mathrm{sb}} \mathrm{ALR} \mathrm{W} \# \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}(\mathrm{m}, \mathrm{rs}, \mathcal{O} \cup \mathrm{A}-\mathrm{R}$, augment-rels (dom $\mathcal{S})$
$\left.\mathrm{R} \mathcal{R}, \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
abbreviation flush-steps::memory $\times$ 'p store-buffer $\times$ owns $\times$ rels $\times$ shared $\Rightarrow$ memory $\times$ 'p store-buffer $\times$ owns $\times$ rels $\times$ shared $\Rightarrow$ bool
$\left(-\rightarrow \mathrm{f}^{*}-[60,60] 100\right)$

## where

flush-steps $==$ flush-step ${ }^{* *}$
term $x \rightarrow f^{*} Y$
lemmas flush-step-induct =
flush-step.induct [split-format (complete),
consumes 1, case-names Write $_{\text {sb }} \operatorname{Read}_{\text {sb }}$ Prog $_{\text {sb }}$ Ghost]
inductive store-buffer-step:: memory $\times$ 'p store-buffer $\times$ 'owns $\times$ 'rels $\times$ 'shared $\Rightarrow$ memory $\times$ 'p memref list $\times$ 'owns $\times$ 'rels $\times$ 'shared $\Rightarrow$ bool
$\left(-\rightarrow_{w}-[60,60] 100\right)$
where
SBWrite $_{\text {sb }}$ :
$\left(\mathrm{m}\right.$, Write $_{\mathrm{sb}}$ volatile a sop v A L R W $\left.\# \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{w}}(\mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathrm{rs}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
abbreviation store-buffer-steps::memory $\times$ 'p store-buffer $\times$ 'owns $\times$ 'rels $\times$ 'shared $\Rightarrow$ memory $\times$ 'p store-buffer $\times$ 'owns $\times$ 'rels $\times$ 'shared $\Rightarrow$ bool

$$
\left(-\rightarrow \mathrm{w}^{*}-[60,60] 100\right)
$$

## where

store-buffer-steps $==$ store-buffer-step ${ }^{\wedge} * *$
term $\mathrm{x} \rightarrow \mathrm{w}^{*} \mathrm{Y}$

```
fun buffered-val :: 'p memref list \(\Rightarrow\) addr \(\Rightarrow\) val option
where
    buffered-val [] a \(=\) None
| buffered-val (r \# rs) \(\mathrm{a}^{\prime}=\)
(case rof
Write \(_{\text {sb }}\) volatile \(\mathrm{a}-\mathrm{v}-\mathrm{-}-\Rightarrow\) (case buffered-val rs a' of
                                    None \(\Rightarrow\) (if \(\mathrm{a}^{\prime}=\mathrm{a}\) then Some v else None)
                                    | Some \(\mathrm{v}^{\prime} \Rightarrow\) Some v')
|- \(\Rightarrow\) buffered-val rs a')
definition address-of :: 'p memref \(\Rightarrow\) addr set
where
address-of \(\mathrm{r}=\left(\right.\) case r of Write \(_{\text {sb }}\) volatile \(\mathrm{a}-\mathrm{v}---\mathrm{A} \Rightarrow\{\mathrm{a}\} \mid \operatorname{Read}_{\mathrm{sb}}\) volatile at \(\mathrm{v} \Rightarrow\{\mathrm{a}\} \mid\)
                        \(-\Rightarrow\{ \})\)
lemma address-of-simps [simp]:
address-of \(\left(\right.\) Write \(_{\text {sb }}\) volatile a sop v A L R W \()=\{\mathrm{a}\}\)
address-of \(\left(\operatorname{Read}_{\mathrm{sb}}\right.\) volatile a \(\left.\mathrm{t} v\right)=\{\mathrm{a}\}\)
address-of \(\left(\right.\) Prog \(_{\text {sb }}\) p p \(^{\prime}\) is \()=\{ \}\)
address-of \(\left(\right.\) Ghost \(\left._{\text {sb }} A \operatorname{LRW}\right)=\{ \}\)
    by (auto simp add: address-of-def)
definition is-volatile :: 'p memref \(\Rightarrow\) bool
where
is-volatile \(\mathrm{r}=\left(\right.\) case r of Write \(_{\text {sb }}\) volatile \(\mathrm{a}-\mathrm{v}---\Rightarrow\) volatile \(\mid \operatorname{Read}_{\mathrm{sb}}\) volatile at \(\mathrm{v} \Rightarrow\)
volatile
- \(-\Rightarrow\) False)
lemma is-volatile-simps [simp]:
is-volatile \(\left(\right.\) Write \(_{\text {sb }}\) volatile a sop v A L R W) \(=\) volatile
is-volatile \(\left(\operatorname{Read}_{\mathrm{sb}}\right.\) volatile at v\()=\) volatile
is-volatile \(\left(\operatorname{Prog}_{s b}\right.\) p p \({ }^{\prime}\) is \()=\) False
is-volatile \(\left(\right.\) Ghost \(_{\text {sb }}\) A L R W) \(=\) False
    by (auto simp add: is-volatile-def)
definition \(^{\text {is-Write }}{ }_{\text {sb }}:\) : p memref \(\Rightarrow\) bool
where
is-Write \(_{\text {sb }} \mathrm{r}=\left(\right.\) case r of Write \(_{\text {sb }}\) volatile \(\mathrm{a}-\mathrm{v}---\Rightarrow\) True \(\mid-\Rightarrow\) False \()\)
definition is-Read \({ }_{\text {sb }}\) : 'p memref \(\Rightarrow\) bool
where
is- \(\operatorname{Read}_{\mathbf{s b}} \mathrm{r}=\left(\right.\) case r of \(\operatorname{Read}_{\mathrm{sb}}\) volatile a \(\mathrm{t} \mathrm{v} \Rightarrow\) True \(\mid-\Rightarrow\) False \()\)
definition is-Prog \({ }_{s b}:\) : p memref \(\Rightarrow\) bool
where
is- Prog \(_{\text {sb }} \mathrm{r}=\left(\right.\) case r of Prog \(_{\text {sb }}--\Rightarrow\) True \(\mid-\Rightarrow\) False \()\)
definition is-Ghost \({ }_{\text {sb }}:\) : p memref \(\Rightarrow\) bool
where
is-Ghost \(_{\text {sb }} \mathrm{r}=\left(\right.\) case r of Ghost \({ }_{\text {sb }}---\Rightarrow\) True \(\mid-\Rightarrow\) False \()\)
```

lemma is-Write ${ }_{\text {sb }}$-simps [simp]:
is-Write ${ }_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $=$ True
is-Write ${ }_{s b}\left(\operatorname{Read}_{s b}\right.$ volatile a $\left.t \mathrm{v}\right)=$ False
is-Write ${ }_{\text {sb }}\left(\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $)=$ False
is-Write ${ }_{s b}\left(\right.$ Ghost $_{s b}$ A L R W) $=$ False
by (auto simp add: is-Write sb $_{\mathbf{s} \text {-def) }}$
lemma is-Read ${ }_{s b}$-simps [simp]:
is- $\operatorname{Read}_{\mathrm{sb}}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v$)=$ True
is-Read ${ }_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $)=$ False
is- $\operatorname{Read}_{s b}\left(\operatorname{Prog}_{s b}\right.$ p p ${ }^{\prime}$ is $)=$ False
is-Read $_{\text {sb }}\left(\right.$ Ghost $_{\text {sb }}$ A L R W) $=$ False
by (auto simp add: is-Read ${ }_{\text {sb }}$-def)
lemma is- Prog $_{s b-s i m p s}[\operatorname{simp}]$ :
is- $\operatorname{Prog}_{s b}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\mathrm{t} v\right)=$ False
is- Prog $_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $=$ False
is $-\operatorname{Prog}_{s b}\left(\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $)=$ True
is- $\operatorname{Prog}_{s b}\left(\right.$ Ghost $_{s b}$ A L R W) = False
by (auto simp add: is- $\operatorname{Prog}_{s b}$-def)
lemma is-Ghost ${ }_{\text {sb }}$-simps [simp]:
is-Ghost ${ }_{\text {sb }}\left(\operatorname{Read}_{s b}\right.$ volatile a $\left.t \mathrm{v}\right)=$ False
is- Ghost ${ }_{s b}\left(\right.$ Write $_{s b}$ volatile a sop v A L R W) $=$ False
is- Ghost $_{\text {sb }}\left(\operatorname{Prog}_{s b}\right.$ p p ${ }^{\prime}$ is $)=$ False
is-Ghost $_{\text {sb }}$ (Ghost $_{\text {sb }}$ A L R W) $=$ True
by (auto simp add: is-Ghost ${ }_{\text {sb }}$-def)
definition is-volatile-Write ${ }_{\text {sb }}:$ : p memref $\Rightarrow$ bool

## where

is-volatile-Write ${ }_{\text {sb }} \mathrm{r}=\left(\right.$ case r of $\mathrm{Write}_{\mathrm{sb}}$ volatile $\mathrm{a}-\mathrm{v}---\Rightarrow$ volatile $\mid-\Rightarrow$ False $)$
lemma is-volatile-Write ${ }_{\text {sb }}$-simps [simp]:
is-volatile-Write ${ }_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) = volatile
is-volatile-Write ${ }_{s b}\left(\operatorname{Read}_{s b}\right.$ volatile a $\left.t v\right)=$ False
is-volatile-Write ${ }_{s b}\left(\operatorname{Prog}_{s b}\right.$ p p $^{\prime}$ is $)=$ False
is-volatile-Write ${ }_{\text {sb }}$ (Ghost $_{\text {sb }}$ A L R W) $=$ False
by (auto simp add: is-volatile-Write sb $^{\mathbf{b}}$-def)
lemma is-volatile-Write sb-address-of [simp]: is-volatile-Write ${ }_{\text {sb }} \mathrm{x} \Longrightarrow$ address-of $\mathrm{x} \neq\{ \}$ by (cases x) auto
definition is-volatile-Read ${ }_{s b}:$ : 'p memref $\Rightarrow$ bool

## where

is-volatile-Read ${ }_{s b} r=\left(\right.$ case $r$ of $\operatorname{Read}_{s b}$ volatile a $t v \Rightarrow$ volatile $\mid-\Rightarrow$ False $)$
lemma is-volatile-Read ${ }_{\text {sb }}$-simps [simp]:
is-volatile- $\operatorname{Read}_{\mathrm{sb}}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\mathrm{t} v\right)=$ volatile
is-volatile-Read ${ }_{\text {sb }}\left(\right.$ Write $_{s b}$ volatile a sop v A L R W) $=$ False
is-volatile- $\operatorname{Read}_{\text {sb }}\left(\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $)=$ False
is-volatile-Read ${ }_{\text {sb }}\left(\right.$ Ghost $_{\text {sb }}$ A L R W) $=$ False
by (auto simp add: is-volatile-Read ${ }_{\text {sb }}$-def)
definition is-non-volatile-Write ${ }_{\text {sb }}:$ : 'p memref $\Rightarrow$ bool where
is-non-volatile-Write ${ }_{\text {sb }} r=\left(\right.$ case r of $\mathrm{Write}_{\text {sb }}$ volatile $\mathrm{a}-\mathrm{v}----\Rightarrow \neg$ volatile $\mid-\Rightarrow$ False $)$
lemma is-non-volatile-Write ${ }_{\text {sb }}$-simps [simp]:
is-non-volatile-Write ${ }_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W $)=(\neg$ volatile $)$
is-non-volatile-Write ${ }_{\text {sb }}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\mathrm{t} v\right)=$ False
is-non-volatile-Write ${ }_{\text {sb }}\left(\operatorname{Prog}_{\text {sb }} p p^{\prime}\right.$ is $)=$ False
is-non-volatile-Write ${ }_{\text {sb }}\left(\right.$ Ghost $_{\text {sb }}$ A L R W) $=$ False
by (auto simp add: is-non-volatile-Write ${ }_{\text {sb-def }}$ )
definition is-non-volatile-Read ${ }_{\mathrm{sb}}$ :: 'p memref $\Rightarrow$ bool where
is-non-volatile-Read ${ }_{\mathbf{s b}} \mathrm{r}=\left(\right.$ case r of $\operatorname{Read}_{\mathbf{s b}}$ volatile a $\mathrm{t} \mathrm{v} \Rightarrow \neg$ volatile $\mid-\Rightarrow$ False $)$
lemma is-non-volatile-Read ${ }_{s b}$-simps [simp]:
is-non-volatile-Read ${ }_{\text {sb }}\left(\operatorname{Read}_{\text {sb }}\right.$ volatile at v$)=(\neg$ volatile $)$
is-non-volatile-Read ${ }_{\text {sb }}\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $=$ False
is-non-volatile-Read ${ }_{\text {sb }}\left(\operatorname{Prog}_{\text {sb }}\right.$ p p ${ }^{\prime}$ is $)=$ False
is-non-volatile-Read ${ }_{\text {sb }}\left(\right.$ Ghostsb $_{\text {sb }}$ A R W) $)=$ False
by (auto simp add: is-non-volatile-Read ${ }_{\text {sb }}$-def)
lemma is-volatile-split: is-volatile $r=$
(is-volatile-Read $_{\text {sb }} r \vee$ is-volatile-Write ${ }_{\text {sb }} r$ )
by (cases r) auto
lemma is-non-volatile-split:
$\neg$ is-volatile $\mathrm{r}=$ (is-non-volatile-Read ${ }_{\text {sb }} \mathrm{r} \vee$ is-non-volatile-Write ${ }_{\text {sb }} \mathrm{r} \vee$ is- $_{\text {Prog }}^{\text {sb }}$ r $\vee$ is-Ghost ${ }_{\text {sb }}$ r)
by (cases r) auto
fun outstanding-refs:: ('p memref $\Rightarrow$ bool) $\Rightarrow$ ' p memref list $\Rightarrow$ addr set where
outstanding-refs P []$=\{ \}$
| outstanding-refs $\mathrm{P}(\mathrm{r} \# \mathrm{rs})=($ if Pr then (address-of r$) \cup$ (outstanding-refs P rs) else outstanding-refs P rs)
lemma outstanding-refs-conv: outstanding-refs P sb $=\bigcup$ (address-of ' $\{\mathrm{r} . \mathrm{r} \in$ set $\mathrm{sb} \wedge \mathrm{P}$ r\})
by (induct sb) auto
lemma outstanding-refs-append:
$\bigwedge$ ys. outstanding-refs vol (xs@ys) = outstanding-refs vol xs $\cup$ outstanding-refs vol ys by (auto simp add: outstanding-refs-conv)
lemma outstanding-refs-empty-negate: (outstanding-refs P sb $=\{ \}) \Longrightarrow$ (outstanding-refs (Not $\circ \mathrm{P}$ ) sb $=\bigcup$ (address-of ' set sb))
by (auto simp add: outstanding-refs-conv)
lemma outstanding-refs-mono-pred:
$\bigwedge \mathrm{sb} \mathrm{sb}^{\prime}$.
$\forall \mathrm{r} . \mathrm{P} \mathrm{r} \longrightarrow \mathrm{P}^{\prime} \mathrm{r} \Longrightarrow$ outstanding-refs P sb $\subseteq$ outstanding-refs $\mathrm{P}^{\prime}$ sb by (auto simp add: outstanding-refs-conv)
lemma outstanding-refs-mono-set:
$\bigwedge \mathrm{sb} \mathrm{sb}^{\prime}$. set sb $\subseteq$ set $\mathrm{sb}^{\prime} \Longrightarrow$ outstanding-refs P sb $\subseteq$ outstanding-refs P sb'
by (auto simp add: outstanding-refs-conv)
lemma outstanding-refs-takeWhile:
outstanding-refs $\mathrm{P}\left(\right.$ takeWhile $\mathrm{P}^{\prime}$ sb $) \subseteq$ outstanding-refs P sb
apply (rule outstanding-refs-mono-set)
apply (auto dest: set-takeWhileD)
done
lemma outstanding-refs-subsets:
outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb} \subseteq$ outstanding-refs is-Write $_{\text {sb }} \mathrm{sb}$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \subseteq$ outstanding-refs is-Write $_{\text {sb }}$ sb
outstanding-refs is-volatile-Read ${ }_{s b} s b \subseteq$ outstanding-refs is-Read $_{\text {sb }}$ sb outstanding-refs is-non-volatile-Read $\operatorname{Reb}^{s b} \subseteq$ outstanding-refs is-Read $_{\text {sb }}$ sb
outstanding-refs is-non-volatile-Write sb $^{s b} \subseteq$ outstanding-refs (Not $\circ$ is-volatile) sb outstanding-refs is-non-volatile-Read $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb} \subseteq$ outstanding-refs (Not $\circ$ is-volatile) sb
outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ sb $\subseteq$ outstanding-refs (is-volatile) sb
outstanding-refs is-volatile-Read ${ }_{\text {sb }} \mathrm{sb} \subseteq$ outstanding-refs (is-volatile) sb
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \subseteq$ outstanding-refs (Not $\circ$ is-volatile-Write $_{\text {sb }}$ ) sb
outstanding-refs is-non-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{sb} \subseteq$ outstanding-refs (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb
outstanding-refs is-volatile-Read ${ }_{s b} \mathrm{sb} \subseteq$ outstanding-refs (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb outstanding-refs is-Read $_{\text {sb }} \mathrm{sb} \subseteq$ outstanding-refs (Not $\circ$ is-volatile-Write $_{\text {sb }}$ ) sb
by (auto intro!:outstanding-refs-mono-pred simp add: is-volatile-Write sb -def is-non-volatile-Write ${ }_{s b}$-def
is-volatile-Read ${ }_{s b}$-def is-non-volatile-Read ${ }_{s b}$-def is-Read ${ }_{\text {sb }}$-def split: memref.splits)
lemma outstanding-non-volatile-refs-conv:
outstanding-refs (Not $\circ$ is-volatile) $\mathrm{sb}=$
outstanding-refs is-non-volatile-Write sb $^{s b} \cup$ outstanding-refs is-non-volatile-Read ${ }_{s b}$ sb
apply (induct sb)
apply simp

```
    subgoal for a sb
    by (case-tac a, auto)
done
```

lemma outstanding-volatile-refs-conv:
outstanding-refs is-volatile $\mathrm{sb}=$
outstanding-refs is-volatile-Write ${ }_{s b} s b \cup$ outstanding-refs is-volatile-Read ${ }_{\mathbf{s b}}$ sb
apply (induct sb)
apply simp
subgoal for a sb
by (case-tac a, auto)
done
lemma outstanding-is-Write ${ }_{\text {sb}}$-refs-conv:
outstanding-refs $^{\text {is-Write }}$ sb $\mathrm{sb}=$
outstanding-refs is-non-volatile-Write sb $^{s b} \cup$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ sb
apply (induct sb)
apply simp
subgoal for a sb
by (case-tac a, auto)
done
lemma outstanding-is-Read ${ }_{\text {sb }}$-refs-conv:
outstanding-refs is-Read sb $_{\text {sb }}$ sb
outstanding-refs is-non-volatile- $\operatorname{Read}_{s b}$ sb $\cup$ outstanding-refs is-volatile-Read $\operatorname{la}_{\text {sb }}$ sb
apply (induct sb)
apply simp
subgoal for a sb
by (case-tac a, auto)
done

is-volatile- $\operatorname{Read}_{\text {sb }}$ ) $\mathrm{sb}=$
outstanding-refs is-Write ${ }_{\text {sb }}$ sb $\cup$ outstanding-refs is-non-volatile-Read ${ }_{s b}$ sb
apply (induct sb)
apply (clarsimp)
subgoal for a sb
by (case-tac a, auto)
done
lemmas misc-outstanding-refs-convs $=$ outstanding-non-volatile-refs-conv outstand-ing-volatile-refs-conv
outstanding-is-Write $_{\mathbf{s b}}$-refs-conv outstanding-is-Read sb $_{\text {-refs-conv }}$ outstand-ing-not-volatile-Read ${ }_{\text {sb }}$-refs-conv
lemma no-outstanding-vol-write-takeWhile-append: outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ $\mathrm{sb}=\{ \} \Longrightarrow$
 xs)
apply (induct sb)
apply (auto split: if-split-asm)

## done

lemma outstanding-vol-write-takeWhile-append: outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \neq$ $\} \Longrightarrow$
takeWhile (Not $\circ$ is-volatile-Write $\left.e_{s b}\right)($ sb@xs $)=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb)
apply (induct sb)
apply (auto split: if-split-asm)
done
lemma no-outstanding-vol-write-dropWhile-append: outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ sb $=\{ \} \Longrightarrow$
dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{\mathbf{s b}}\right)($ sb@xs $)=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ xs)
apply (induct sb)
apply (auto split: if-split-asm)
done
lemma outstanding-vol-write-dropWhile-append: outstanding-refs is-volatile-Write ${ }_{s b} \mathrm{sb}$ $\neq\{ \} \Longrightarrow$
dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)\left(\mathrm{sb}_{\mathrm{sxs}}\right)=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb)@xs
apply (induct sb)
apply (auto split: if-split-asm)

## done

lemmas outstanding-vol-write-take-drop-appends $=$
no-outstanding-vol-write-takeWhile-append
outstanding-vol-write-takeWhile-append
no-outstanding-vol-write-dropWhile-append
outstanding-vol-write-dropWhile-append
lemma outstanding-refs-is-non-volatile-Write ${ }_{\text {sb }}$-takeWhile-conv:
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)=$ outstanding-refs $^{\text {is-Write }}$ sb $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb)
apply (induct sb)
apply clarsimp
subgoal for a sb
by (case-tac a, auto)
done
lemma dropWhile-not-vol-write-empty:
outstanding-refs is-volatile-Write $\mathrm{e}_{\mathrm{sb}} \mathrm{sb}=\{ \} \Longrightarrow$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb})=[]$

```
apply (induct sb)
apply (auto split: if-split-asm)
done
```

lemma takeWhile－not－vol－write－outstanding－refs：
outstanding－refs is－volatile－Write $_{\text {sb }}\left(\right.$ takeWhile $^{(N o t} \circ$ is－volatile－Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)=\{ \}$
apply（induct sb）
apply（auto split：if－split－asm）
done
lemma no－volatile－Write ${ }_{\text {sbs }}$－conv：（outstanding－refs is－volatile－Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ ）$=$ $\left(\forall \mathrm{r} \in \operatorname{set} \mathrm{sb} .\left(\forall \mathrm{v}^{\prime} \operatorname{sop}^{\prime} \mathrm{a}^{\prime}\right.\right.$ A L R W． $\mathrm{r} \neq$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime}$ A L R W））
by（force simp add：outstanding－refs－conv is－volatile－Write ${ }_{\mathbf{s b}}$－def split：memref．splits）
lemma no－volatile－Read ${ }_{\text {sb }}$ S－conv：（outstanding－refs is－volatile－Read ${ }_{s b} \mathrm{sb}=\{ \}$ ）$=$ $\left(\forall \mathrm{r} \in \operatorname{set} \mathrm{sb} .\left(\forall \mathrm{v}^{\prime} \mathrm{t}^{\prime} \mathrm{a}^{\prime} . \mathrm{r} \neq \operatorname{Read}_{\mathrm{sb}} \operatorname{True} \mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v}^{\prime}\right)\right)$
by（force simp add：outstanding－refs－conv is－volatile－Read sb －def split：memref．splits）
inductive sb－memop－step ：：（instrs $\times$ tmps $\times$＇p store－buffer $\times$ memory $\times$＇dirty $\times$＇owns $\times$＇rels $\times$＇shared $) \Rightarrow$
（instrs $\times$ tmps $\times$＇p store－buffer $\times$ memory $\times$＇dirty $\times$＇owns $\times$＇rels $\times$＇shared
）$\Rightarrow \mathrm{bool}$

$$
\left(-\rightarrow_{\text {sb }}-[60,60] 100\right)
$$

## where

SBReadBuffered：
【buffered－val sb a $=$ Some v】
$\Longrightarrow$
（Read volatile a $\mathrm{t} \#$ is $, \vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
｜SBReadUnbuffered：
【buffered－val sb a＝None】
$\Longrightarrow$
$(\operatorname{Read}$ volatile a $\mathrm{t} \#$ is，$\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}), \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$

## ｜SBWriteNonVolatile：

（Write False a（D，f）A L R W\＃is，$\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta$ ，sb＠$\left[W_{r i t e}^{s b}\right.$ False a $\left.\left.(\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta) \mathrm{A} \mathrm{L} \mathrm{R} \mathrm{W}\right], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
｜SBWriteVolatile：
（Write True a $(\mathrm{D}, \mathrm{f}) \mathrm{ALR} \mathrm{W} \#$ is， $\boldsymbol{\vartheta}$ ， $\mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta$ ，sb＠［Write ${ }_{\text {sb }}$ True a（D，f）（f $\vartheta$ ）A L R W］，m， $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}$ ）
｜SBFence：
$($ Fence $\#$ is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}($ is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$

## ｜SBRMWReadOnly：

$\llbracket \neg$ cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
（RMW at（D，f）cond ret A L R W\＃is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}($ is，$\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a），［］，m， $\mathcal{D}$ ， $\mathcal{O}, \mathcal{R}, \mathcal{S})$
｜SBRMWWrite：
$\llbracket \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
（RMW at（D，f）cond ret A L R W\＃is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta(\mathrm{t} \mapsto \mathrm{ret}(\mathrm{m}$ a）$(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})))),[], \mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}))), \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
｜SBGhost：
（Ghost A L R W\＃is，$\vartheta$ ， $\mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}$ （is，$\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S})$
inductive sbh－memop－step ：：
（instrs $\times$ tmps $\times$＇p store－buffer $\times$ memory $\times$ bool $\times$ owns $\times$ rels $\times$ shared
）$\Rightarrow$
（instrs $\times$ tmps $\times$＇p store－buffer $\times$ memory $\times$ bool $\times$ owns $\times$ rels $\times$ shared
）$\Rightarrow$ bool

$$
\left(-\rightarrow_{\text {sbh }}-[60,60] 100\right)
$$

## where

SBHReadBuffered：
【buffered－val sb a＝Some v】
$\Longrightarrow$
（Read volatile a $\mathrm{\#} \#$ is，$\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}$ （is，$\vartheta(\mathrm{t} \mapsto \mathrm{v})$ ，sb＠$\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v$\left.], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
｜SBHReadUnbuffered：
【buffered－val sb a＝None】
$\Longrightarrow$
（Read volatile a $\mathrm{t} \#$ is，$\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}$ （is，$\vartheta\left(\mathrm{t} \mapsto \mathrm{m}\right.$ a）， $\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\left.\mathrm{t}(\mathrm{m} \mathrm{a})\right], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
｜SBHWriteNonVolatile：
（Write False a（D，f）A L R W\＃is，$\vartheta$ ，sb，m， $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}$ （is，$\vartheta$ ，sb＠［Write ${ }_{\text {sb }}$ False a（D，f）（f $\vartheta$ ）A L R W］，m， $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}$ ）

## ｜SBHWriteVolatile：

（Write True a（D，f）A L R W\＃is，$\vartheta$ ，sb，m， $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}$ ）$\rightarrow_{\text {sbh }}$ （is，$\vartheta$ ，sb＠［Write ${ }_{\text {sb }} \operatorname{True}$ a（D，f）（f $\vartheta$ ）A L R W］，m，True， $\mathcal{O}, \mathcal{R}, \mathcal{S}$ ）
｜SBHFence：
（Fence \＃is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}($ is，$\vartheta,[], \mathrm{m}$, False， $\mathcal{O}$ ，Map．empty， $\mathcal{S}$ ）

## ｜SBHRMWReadOnly：

$\llbracket \neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
（RMW at（D，f）cond ret A L R W\＃is，$\vartheta,[], \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}($ is，$\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a），［］，m， False， $\mathcal{O}$ ，Map．empty， $\mathcal{S}$ ）

## SBHRMWWrite:

$\llbracket$ cond $(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})) \rrbracket \Longrightarrow$
(RMW at (D,f) cond ret A L R W\# is, $\vartheta$, [], m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}$
(is, $\vartheta(\mathrm{t} \mapsto \mathrm{ret}(\mathrm{m}$ a) $(\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})))),[], \mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a) $))$, False, $\mathcal{O} \cup \mathrm{A}-$ R,Map.empty, $\left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$

## | SBHGhost:

(Ghost A L R W\# is, $\vartheta$, sb, m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}$ (is, $\vartheta$, sb@[Ghost ${ }_{\mathrm{sb}}$ A L R W], m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}$ )
interpretation direct: memory-system direct-memop-step id-storebuffer-step .
interpretation sb: memory-system sb-memop-step store-buffer-step .
interpretation sbh: memory-system sbh-memop-step flush-step .
primrec non-volatile-owned-or-read-only:: bool $\Rightarrow$ shared $\Rightarrow$ owns $\Rightarrow$ 'a memref list $\Rightarrow$ bool
where
non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}[]=$ True
| non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}(\mathrm{x} \# \mathrm{xs})=$ (case x of Read $_{\text {sb }}$ volatile at $\mathrm{v} \Rightarrow$ $(\neg$ volatile $\longrightarrow$ pending-write $\longrightarrow(\mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in$ read-only $\mathcal{S})) \wedge$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ xs
| Write $_{\text {sb }}$ volatile a sop v A L R W $\Rightarrow$
(if volatile then non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ xs
else a $\in \mathcal{O} \wedge$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ xs)
Ghost ${ }_{\text {sb }} \mathrm{A}$ L $\mathrm{R} \mathrm{W} \Rightarrow$ non-volatile-owned-or-read-only pending-write $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{xs}$
|- $\Rightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ xs)
primrec acquired :: bool $\Rightarrow$ 'a memref list $\Rightarrow$ addr set $\Rightarrow$ addr set where
acquired pending-write [] $\mathrm{A}=$ (if pending-write then A else $\}$ )
| acquired pending-write ( $\mathrm{x} \# \mathrm{xs}$ ) $\mathrm{A}=$
(case x of
Write $_{\text {sb }}$ volatile - - A' L R W $\Rightarrow$
(if volatile then acquired True xs (if pending-write then $\left(A \cup A^{\prime}-R\right)$ else ( $\mathrm{A}^{\prime}-$
R)) else acquired pending-write xs A)
$\mid$ Ghost $_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L} R \mathrm{~W} \Rightarrow$ acquired pending-write xs (if pending-write then $\left(\mathrm{A} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)$ else A)
|- $\Rightarrow$ acquired pending-write xs A)
primrec share :: 'a memref list $\Rightarrow$ shared $\Rightarrow$ shared
where
share [] $\mathrm{S}=\mathrm{S}$
| share ( $\mathrm{x} \# \mathrm{xs}$ ) $\mathrm{S}=$ (case x of

Write $_{\text {sb }}$ volatile -- A L R W $\Rightarrow$ (if volatile then (share xs $\left(S \oplus_{W} R \ominus_{A} L\right)$ ) else share xs S)
| Ghost ${ }_{\text {sb }}$ A L R W $\Rightarrow$ share xs $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
|- $\Rightarrow$ share xs $S$ )
primrec acquired-reads :: bool $\Rightarrow$ 'a memref list $\Rightarrow$ addr set $\Rightarrow$ addr set where
acquired-reads pending-write [] $\mathrm{A}=\{ \}$
| acquired-reads pending-write ( $\mathrm{x} \# \mathrm{xs}$ ) $\mathrm{A}=$
(case x of
$\operatorname{Read}_{\mathrm{sb}}$ volatile at $\mathrm{v} \Rightarrow$ (if pending-write $\wedge \neg$ volatile $\wedge \mathrm{a} \in \mathrm{A}$
then insert a (acquired-reads pending-write xs A)
else acquired-reads pending-write xs A)
| Write $_{\text {sb }}$ volatile --- A'L R W $\Rightarrow$
(if volatile then acquired-reads True xs (if pending-write then $\left(A \cup A^{\prime}-R\right)$ else ( $\left.\mathrm{A}^{\prime}-\mathrm{R}\right)$ ) else acquired-reads pending-write xs A)
$\mid$ Ghost $_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L} \mathrm{R} \mathrm{W} \Rightarrow$ acquired-reads pending-write $\mathrm{xs}\left(\mathrm{A} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)$
$\mid-\Rightarrow$ acquired-reads pending-write xs A)
lemma union-mono-aux: $\mathrm{A} \subseteq \mathrm{B} \Longrightarrow \mathrm{A} \cup \mathrm{C} \subseteq \mathrm{B} \cup \mathrm{C}$
by blast
lemma set-minus-mono-aux: $\mathrm{A} \subseteq \mathrm{B} \Longrightarrow \mathrm{A}-\mathrm{C} \subseteq \mathrm{B}-\mathrm{C}$
by blast
lemma acquired-mono: $\wedge$ A B pending-write. $\mathrm{A} \subseteq \mathrm{B} \Longrightarrow$ acquired pending-write xs $\mathrm{A} \subseteq$ acquired pending-write xs B
apply (induct xs)
apply simp
subgoal for a xs A B pending-write
apply (case-tac a )
apply clarsimp
subgoal for volatile a1 D fv A' L R W x
apply (drule-tac $\mathrm{C}=\mathrm{A}^{\prime}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
apply clarsimp
apply clarsimp
apply clarsimp
subgoal for $\mathrm{A}^{\prime} \mathrm{L} R \mathrm{~W} \mathrm{x}$
apply (drule-tac $\mathrm{C}=\mathrm{A}^{\prime}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
done
done

```
lemma acquired-mono-in:
    assumes \(x\)-in: \(x \in\) acquired pending-write xs \(A\)
    assumes sub: \(\mathrm{A} \subseteq \mathrm{B}\)
    shows \(\mathrm{x} \in\) acquired pending-write xs B
using acquired-mono [OF sub, of pending-write xs] x -in
by blast
```

lemma acquired-no-pending-write: $\bigwedge \mathrm{A} B$. acquired False xs $\mathrm{A}=$ acquired False xs B by (induct xs) (auto split: memref.splits)
lemma acquired-no-pending-write-in:
$x \in$ acquired False $x s \mathrm{~A} \Longrightarrow \mathrm{x} \in$ acquired False xs $B$
apply (subst acquired-no-pending-write)
apply auto
done
lemma acquired-pending-write-mono-in: $\bigwedge \mathrm{AB} . \mathrm{x} \in$ acquired False $\mathrm{xs} \mathrm{A} \Longrightarrow \mathrm{x} \in$ acquired
True xs B
apply (induct xs)
apply (auto split: memref.splits if-split-asm intro: acquired-mono-in)
done
lemma acquired-pending-write-mono: acquired False xs A $\subseteq$ acquired True xs B by (auto intro: acquired-pending-write-mono-in)
lemma acquired-append: $\bigwedge \mathrm{A}$ pending-write. acquired pending-write (xs@ys) $\mathrm{A}=$ acquired (pending-write $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\neq\{ \}$ ) ys (acquired pend-ing-write xs A)
apply (induct xs)
apply (auto split: memref.splits intro: acquired-no-pending-write-in)
done
lemma acquired-take-drop:
acquired (pending-write $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ (takeWhile P xs) $\neq\{ \}$ )
(dropWhile P xs) (acquired pending-write (takeWhile P xs) A$)=$
acquired pending-write xs A

## proof -

have acquired pending-write xs $\mathrm{A}=$ acquired pending-write ((takeWhile P xs)@(dropWhile P xs)) A
by $\operatorname{simp}$
also
from acquired-append [where $\mathrm{xs}=($ takeWhile P xs$)$ and $\mathrm{ys}=($ dropWhile $\mathrm{P} x s)$ ]
have $\ldots=$ acquired (pending-write $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ (takeWhile P
xs) $\neq\{ \}$ )
(dropWhile P xs) (acquired pending-write (takeWhile P xs) A)
by simp
finally show ?thesis
by simp
qed

```
lemma share-mono: \(\wedge \mathrm{AB}\). dom \(\mathrm{A} \subseteq \operatorname{dom} \mathrm{B} \Longrightarrow \operatorname{dom}(\) share xs A\() \subseteq \operatorname{dom}(\) share xs B\()\)
apply (induct xs)
apply simp
subgoal for a xs A B
apply (case-tac a)
apply (clarsimp iff del: domIff)
subgoal for volatile a1 D fv A'L R W x
apply (drule-tac \(\mathrm{C}=\mathrm{R}\) and \(\mathrm{x}=\mathrm{W}\) in augment-mono-aux)
apply (drule-tac \(\mathrm{C}=\mathrm{L}\) in restrict-mono-aux)
apply blast
done
apply clarsimp
apply clarsimp
apply (clarsimp iff del: domIff)
subgoal for \(\mathrm{A}^{\prime} \mathrm{L} R \mathrm{R} \mathrm{x}\)
apply (drule-tac \(\mathrm{C}=\mathrm{R}\) and \(\mathrm{x}=\mathrm{W}\) in augment-mono-aux)
apply (drule-tac \(\mathrm{C}=\mathrm{L}\) in restrict-mono-aux)
apply blast
done
done
done
lemma share-mono-in:
    assumes x -in: \(\mathrm{x} \in \operatorname{dom}\) (share xs A)
    assumes sub: dom \(\mathrm{A} \subseteq \operatorname{dom} \mathrm{B}\)
    shows \(\mathrm{x} \in \operatorname{dom}\) (share xs B)
using share-mono [OF sub, of xs] x-in
by blast
lemma acquired-reads-mono:
    \(\bigwedge \mathrm{A}\) B pending-write. \(\mathrm{A} \subseteq \mathrm{B} \Longrightarrow\) acquired-reads pending-write xs \(\mathrm{A} \subseteq\) acquired-reads
pending-write xs B
apply (induct xs)
apply simp
subgoal for a xs A B pending-write
apply (case-tac a)
apply clarsimp
    subgoal for volatile a1 D fv A'L R W x
    apply (drule-tac \(\mathrm{C}=\mathrm{A}^{\prime}\) in union-mono-aux)
    apply (drule-tac \(\mathrm{C}=\mathrm{R}\) in set-minus-mono-aux)
    apply blast
    done
apply clarsimp
apply blast
apply clarsimp
apply clarsimp
subgoal for \(\mathrm{A}^{\prime} \mathrm{L} R \mathrm{~W} \mathrm{x}\)
apply (drule-tac \(\mathrm{C}=\mathrm{A}^{\prime}\) in union-mono-aux)
apply (drule-tac \(\mathrm{C}=\mathrm{R}\) in set-minus-mono-aux)
apply blast
```

```
done
done
done
lemma acquired-reads-mono-in:
    assumes x-in: x }\in\mathrm{ acquired-reads pending-write xs A
    assumes sub: A\subseteqB
    shows x}\in\mathrm{ acquired-reads pending-write xs B
using acquired-reads-mono [OF sub, of pending-write xs] x-in
by blast
lemma acquired-reads-no-pending-write: \(\bigwedge \mathrm{A} B\). acquired-reads False xs \(\mathrm{A}=\mathrm{ac}\) -quired-reads False xs B
by (induct xs) (auto split: memref.splits)
lemma acquired-reads-no-pending-write-in:
\(\mathrm{x} \in\) acquired-reads False \(\mathrm{xs} \mathrm{A} \Longrightarrow \mathrm{x} \in\) acquired-reads False xs B
apply (subst acquired-reads-no-pending-write)
apply blast
done
lemma acquired-reads-pending-write-mono:
\(\bigwedge\) A. acquired-reads False xs \(\mathrm{A} \subseteq\) acquired-reads True xs A
by (induct xs) (auto split: memref.splits intro: acquired-reads-mono-in )
lemma acquired-reads-pending-write-mono-in:
assumes x-in: \(\mathrm{x} \in\) acquired-reads False xs A
shows \(\mathrm{x} \in\) acquired-reads True xs A
using acquired-reads-pending-write-mono [of xs A] x-in
by blast
lemma acquired-reads-append: ^pending-write A. acquired-reads pending-write (xs@ys)
A =
acquired-reads pending-write xs \(\mathrm{A} \cup\)
acquired-reads (pending-write \(\vee\) (outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) xs \(\neq\{ \}\) )) ys (acquired pending-write xs A)
proof (induct xs)
case Nil thus ?case by (auto dest: acquired-reads-no-pending-write-in)
next
case (Cons x xs)
show ?case
proof (cases x)
case \(\left(\right.\) Write \(_{\text {sb }}\) volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case False
show ?thesis
using Cons.hyps
by (auto simp add: Write \({ }_{\text {sb }}\) False) next
```

```
        case True
        show ?thesis
using Cons.hyps
by (auto simp add: Writesb True)
    qed
next
    case (Read sb volatile a t v)
    show ?thesis
    proof (cases volatile)
        case False
        show ?thesis
using Cons.hyps
by (auto simp add: Readsb False)
    next
        case True
        show ?thesis
using Cons.hyps
by (auto simp add: Readsb True)
        qed
    next
        case Progsb
        with Cons.hyps show ?thesis by auto
next
        case (Ghost sb A'L R W)
        have (acquired False xs (A\cup A' -R ))=(acquired False xs A)
            by (simp add: acquired-no-pending-write)
        with Cons.hyps show ?thesis by (auto simp add: Ghostsb)
    qed
qed
lemma in-acquired-reads-no-pending-write-outstanding-write:
\A.a }\in\mathrm{ acquired-reads False xs A }\Longrightarrow\mathrm{ outstanding-refs (is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ ) xs }\not={
apply (induct xs)
apply simp
apply (auto split: memref.splits)
apply auto
done
lemma augment-read-only-mono: read-only \(\mathcal{S} \subseteq\) read-only \(\mathcal{S}^{\prime} \Longrightarrow\)
read-only (\mathcal{S}\oplus\textrm{W}R)\subseteq read-only ( }\mp@subsup{\mathcal{S}}{}{\prime}\oplus\textrm{W}R\textrm{R}
by (auto simp add: augment-shared-def read-only-def)
lemma restrict-read-only-mono: read-only S}\subseteq\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime}
    read-only (S Ө
        apply (clarsimp simp add: restrict-shared-def read-only-def split: option.splits
if-split-asm)
    apply (rule conjI)
    apply blast
    apply fastforce
    done
```

```
lemma share-read-only-mono: }\\mathcal{S}\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ . read-only }\mathcal{S}\subseteq\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime}
    read-only (share sb S) \subseteq read-only (share sb \mathcal{S}
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case False
            with Cons Write sb show ?thesis by auto
    next
            case True
            note «read-only S \subseteq read-only }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ `
            from augment-read-only-mono [OF this]
            have read-only (\mathcal{S}\oplus\textrm{w}}\textrm{R})\subseteq\mathrm{ read-only ( }\mp@subsup{\mathcal{S}}{}{\prime}\oplus\textrm{w}R)
            from restrict-read-only-mono [OF this, of A L]
            have read-only (S }\mp@subsup{\mathcal{W}}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\subseteq\mathrm{ read-only ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
            from Cons.hyps [OF this]
            show ?thesis
by (clarsimp simp add: Write sb True)
    qed
    next
        case Read sb with Cons show ?thesis
            by auto
    next
        case Prog
            by auto
    next
        case (Ghost sb A L R W)
        note «read-only S \subseteq read-only S'`
        from augment-read-only-mono [OF this]
        have read-only (\mathcal{S}\oplus\textrm{w}R)\subseteq read-only ( }\mp@subsup{\mathcal{S}}{}{\prime}\oplus\textrm{w}R)
        from restrict-read-only-mono [OF this, of A L]
        have read-only (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\subseteq\mathrm{ read-only ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}).
    from Cons.hyps [OF this]
    show ?thesis
        by (clarsimp simp add: Ghostsb)
    qed
qed
```

lemma non-volatile-owned-or-read-only-append:
$\wedge \mathcal{O} \mathcal{S}$ pending-write. non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ (xs@ys)
$=($ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ xs $\wedge$
non-volatile-owned-or-read-only (pending-write $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\neq\{ \}$ )
(share xs $\mathcal{S}$ ) (acquired True xs $\mathcal{O})$ ys)
apply (induct xs)
apply (auto split: memref.splits)
done
lemma non-volatile-owned-or-read-only-mono:
$\bigwedge \mathcal{O} \mathcal{O}^{\prime} \mathcal{S}$ pending-write. $\mathcal{O} \subseteq \mathcal{O}^{\prime} \Longrightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$
xs
$\Longrightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}^{\prime}$ xs
apply (induct xs)
apply simp
subgoal for a xs $\mathcal{O} \mathcal{O}^{\prime} \mathcal{S}$ pending-write
apply (case-tac a)
apply (clarsimp split: if-split-asm)
subgoal for volatile a1 D f v A L R W
apply (drule-tac $\mathrm{C}=\mathrm{A}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
apply fastforce
apply fastforce
apply fastforce
apply clarsimp
subgoal for $A L R W$
apply (drule-tac $\mathrm{C}=\mathrm{A}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
done
done
lemma non-volatile-owned-or-read-only-shared-mono:
$\bigwedge \mathcal{S} \mathcal{S}^{\prime} \mathcal{O}$ pending-write. $\mathcal{S} \subseteq_{s} \mathcal{S}^{\prime} \Longrightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ xs
$\Longrightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S}^{\prime} \mathcal{O}$ xs
apply (induct xs)
apply simp
subgoal for a xs $\mathcal{S} \mathcal{S}^{\prime} \mathcal{O}$ pending-write
apply (case-tac a)
apply (clarsimp split: if-split-asm)
subgoal for volatile a1 D f v A L R W
apply (frule-tac $\mathrm{C}=\mathrm{R}$ and $\mathrm{x}=\mathrm{W}$ in augment-mono-map)
apply (drule-tac $\mathrm{A}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R}$ and $\mathrm{C}=\mathrm{L}$ in restrict-mono-map)
apply (fastforce dest: read-only-mono)
done
apply (fastforce dest: read-only-mono shared-leD)
apply fastforce
subgoal for $A L R W$

```
apply (frule-tac C=R and }\textrm{x}=\textrm{W}=\textrm{W}\mathrm{ in augment-mono-map)
apply (drule-tac A=S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mathrm{ and }\textrm{C}=\textrm{L}\mathrm{ in restrict-mono-map)
apply (fastforce dest: read-only-mono)
done
done
done
```

lemma non-volatile-owned-or-read-only-pending-write-antimono:
$\bigwedge \mathcal{O} \mathcal{S}$. non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ xs
$\Longrightarrow$ non-volatile-owned-or-read-only False $\mathcal{S} \mathcal{O}$ xs by (induct xs) (auto split: memref.splits)
primrec all-acquired $::$ 'a memref list $\Rightarrow$ addr set

## where

all-acquired [] $=\{ \}$
| all-acquired (i\#is) = (case i of
Write $_{\text {sb }}$ volatile -- A L R W $\Rightarrow$ (if volatile then $\mathrm{A} \cup$ all-acquired is else all-acquired
is)

> Ghost $_{\text {sb }} \mathrm{A} L \mathrm{R} \mathrm{W} \Rightarrow \mathrm{A} \cup$ all-acquired is $\mid-\Rightarrow$ all-acquired is)
lemma all-acquired-append: all-acquired ( $x s @ y s)=$ all-acquired $x s \cup$ all-acquired ys apply (induct xs)
apply (auto split: memref.splits)
done
lemma acquired-reads-all-acquired: $\wedge \mathcal{O}$ pending-write.
acquired-reads pending-write sb $\mathcal{O} \subseteq \mathcal{O} \cup$ all-acquired sb
apply (induct sb)
apply clarsimp
apply (auto split: memref.splits)
done
lemma acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ :
$\bigwedge A$. (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) A$) \subseteq$
$\mathrm{A} \cup$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (induct sb)
apply clarsimp
subgoal for a sb A
apply (case-tac a)
apply auto
done
done
lemma acquired-False-takeWhile-non-volatile-Write ${ }_{s b}$ :
acquired False (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathrm{A}=\{ \}$
apply (induct sb)
apply simp

```
subgoal for a sb
    by (case-tac a) auto
done
```

lemma outstanding-refs-takeWhile-opposite: outstanding-refs P (takeWhile (Not $\circ \mathrm{P}$ ) xs) $=\{ \}$
apply (induct xs)
apply auto
done
lemma no-outstanding-volatile-Write sb $_{\text {-acquired: }}$
outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \} \Longrightarrow$ acquired False sb $\mathrm{A}=\{ \}$
apply (induct sb)
apply simp
subgoal for a sb
by (case-tac a) auto
done
lemma acquired-all-acquired:^pending-write A . acquired pending-write xs $\mathrm{A} \subseteq \mathrm{A} \cup$ all-acquired xs
apply (induct xs)
apply (auto split: memref.splits)
done
lemma acquired-all-acquired-in: $x \in$ acquired pending-write $x s \mathrm{~A} \Longrightarrow \mathrm{x} \in \mathrm{A} \cup$ all-acquired xS
using acquired-all-acquired by blast

```
primrec sharing-consistent:: shared }=>\mathrm{ owns }=>\mathrm{ 'a memref list }=>\mathrm{ bool
where
sharing-consistent S\mathcal{O}[]=\mathrm{ True}
| sharing-consistent \mathcal{S O}}(\textrm{r}#\textrm{rs})
        (case r of
            Write
                (if volatile then A}\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\wedge\textrm{L}\subseteq\textrm{A}\wedge\textrm{A}\cap\textrm{R}={}\wedge\textrm{R}\subseteq\mathcal{O}
                    sharing-consistent (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R}) rs
            else sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ rs)
        |Ghost sb A L R W m A\subseteqdom S \cup\mathcal{O}\wedgeL\subseteqA A ^ A \cap R={}^R\subseteq\mathcal{O}^
            sharing-consistent (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ rs
        |- m sharing-consistent \mathcal{S O rs)}
    lemma sharing-consistent-all-acquired:
    \\mathcal{O}}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\textrm{sb}\Longrightarrow\mathrm{ all-acquired sb }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O
proof (induct sb)
    case Nil thus ?case by simp
next
```

```
case (Cons x sb)
show ?case
proof (cases x)
    case (Write sb volatile a sop v A L R W)
    show ?thesis
    proof (cases volatile)
            case False
            with Cons Writesb show ?thesis by auto
    next
        case True
            from Cons.hyps [where }\mathcal{S}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]\mathrm{ Cons.prems
            show ?thesis
by (auto simp add: Writesb True)
    qed
next
    case Read
next
    case Prog
next
    case (Ghostsb A L R W)
    with Cons.hyps [where \mathcal{S}=(\mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]\mathrm{ Cons.prems show
?thesis by auto
    qed
qed
lemma sharing-consistent-append:
\\mathcal{O}}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ (xs@ys)=
    (sharing-consistent \mathcal{S O}
    sharing-consistent (share xs \mathcal{S}}\mathrm{ ) (acquired True xs }\mathcal{O}) ys
apply (induct xs)
apply (auto split: memref.splits)
done
primrec read-only-reads :: owns }=>\mathrm{ ''a memref list }=>\mathrm{ addr set
where
read-only-reads }\mathcal{O}[]={
|read-only-reads \mathcal{O}(x#xs)=
    (case x of
        Read
                                    then insert a (read-only-reads \mathcal{O xs)}
                                    else read-only-reads }\mathcal{O}\mathrm{ xs)
        | Writesb volatile - - A L R W =
            (if volatile then read-only-reads (\mathcal{O}\cup\textrm{A}-\textrm{R}) xs
            else read-only-reads }\mathcal{O}\mathrm{ xs )
        Ghost sb A L R W m read-only-reads (\mathcal{O}\cup\textrm{A}-\textrm{R})\textrm{xs}
        |- = read-only-reads \mathcal{O xs)}
```

lemma read-only-reads-append:
$\wedge \mathcal{O}$. read-only-reads $\mathcal{O}$ (xs@ys) $=$ read-only-reads $\mathcal{O}$ xs $\cup$ read-only-reads (acquired True xs $\mathcal{O}$ ) ys

```
apply (induct xs)
    apply simp
subgoal for a xs }\mathcal{O
    by (case-tac a) auto
done
lemma read-only-reads-antimono:
\mathcal{O}\mp@subsup{\mathcal{O}}{}{\prime}
\mathcal{O}\subseteq\mp@subsup{\mathcal{O}}{}{\prime}\Longrightarrow\mathrm{ read-only-reads }\mp@subsup{\mathcal{O}}{}{\prime}\textrm{sb}\subseteq\mathrm{ read-only-reads }\mathcal{O}\mathrm{ sb}
apply (induct sb)
apply simp
subgoal for a sb O}\mp@subsup{\mathcal{O}}{}{\prime
apply (case-tac a)
apply (clarsimp split: if-split-asm)
subgoal for volatile a1 D fv A L R W
apply (drule-tac C=A in union-mono-aux)
apply (drule-tac C=R in set-minus-mono-aux)
apply blast
done
apply auto
subgoal for A L R W x
apply (drule-tac C=A in union-mono-aux)
apply (drule-tac C=R in set-minus-mono-aux)
apply blast
done
done
done
```

primrec non-volatile-writes-unshared:: shared $\Rightarrow$ 'a memref list $\Rightarrow$ bool
where
non-volatile-writes-unshared $\mathcal{S}[]=$ True
| non-volatile-writes-unshared $\mathcal{S}(\mathrm{x} \# \mathrm{xs})=$
(case x of
Write $_{\text {sb }}$ volatile a sop v A L R W $\Rightarrow$ (if volatile then non-volatile-writes-unshared ( $\mathcal{S}$
$\oplus_{w} R \ominus_{A} L$ ) xs
else a $\notin \operatorname{dom} \mathcal{S} \wedge$ non-volatile-writes-unshared $\mathcal{S}$ xs)
$\mid$ Ghosts $_{\text {sb }} \mathrm{A} \mathrm{R} \mathrm{W} \Rightarrow$ non-volatile-writes-unshared $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ xs
|- $\Rightarrow$ non-volatile-writes-unshared $\mathcal{S}$ xs)
lemma non-volatile-writes-unshared-append:
$\wedge \mathcal{S}$. non-volatile-writes-unshared $\mathcal{S}$ (xs@ys)
$=($ non-volatile-writes-unshared $\mathcal{S} \times s \wedge$ non-volatile-writes-unshared (share xs $\mathcal{S})$
ys)
apply (induct xs)
apply (auto split: memref.splits)
done
lemma non-volatile-writes-unshared-antimono:
$\wedge \mathcal{S} \mathcal{S}^{\prime}$. dom $\mathcal{S} \subseteq \operatorname{dom} \mathcal{S}^{\prime} \Longrightarrow$ non-volatile-writes-unshared $\mathcal{S}^{\prime}$ xs

```
non-volatile-writes-unshared S xs
apply (induct xs)
apply simp
subgoal for a xs S S S'
apply (case-tac a)
apply (clarsimp split: if-split-asm)
            subgoal for volatile a1 D f v A L R W
            apply (drule-tac C=R in augment-mono-aux)
            apply (drule-tac C=L in restrict-mono-aux)
            apply blast
            done
apply fastforce
apply fastforce
apply fastforce
apply (clarsimp split: if-split-asm)
subgoal for A L R W
apply (drule-tac C=R in augment-mono-aux)
apply (drule-tac C=L in restrict-mono-aux)
apply blast
done
done
done
primrec no-write-to-read-only-memory:: shared }=>\mathrm{ ''a memref list }=>\mathrm{ bool
where
no-write-to-read-only-memory \mathcal{S [] = True}
| no-write-to-read-only-memory \mathcal{S}}(\textrm{x}#\textrm{xs})
    (case x of
        Write}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ volatile a sop v A L R W ma a read-only S S ^
                                    (if volatile then no-write-to-read-only-memory (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{
L) xs
                                    else no-write-to-read-only-memory S xs)
    Ghost sb A L R W m no-write-to-read-only-memory (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}) xs
    |- # no-write-to-read-only-memory S xs)
lemma no-write-to-read-only-memory-append:
\(\bigwedge \mathcal{S}\). no-write-to-read-only-memory \(\mathcal{S}\) (xs@ys)
\(=\) (no-write-to-read-only-memory \(\mathcal{S}\) xs \(\wedge\) no-write-to-read-only-memory (share xs
\(\mathcal{S}\) ) ys)
apply (induct xs)
apply simp
subgoal for a xs \(\mathcal{S}\)
by (case-tac a) auto
done
lemma no-write-to-read-only-memory-antimono:
\(\bigwedge \mathcal{S} \mathcal{S}^{\prime} . \mathcal{S} \subseteq_{s} \mathcal{S}^{\prime} \Longrightarrow\) no-write-to-read-only-memory \(\mathcal{S}^{\prime}\) xs
\(\Longrightarrow\) no-write-to-read-only-memory \(\mathcal{S}\) xs
apply (induct xs)
apply simp
```

```
subgoal for a xs S S S'
apply (case-tac a)
apply (clarsimp split: if-split-asm)
    subgoal for volatile a1 D f v A L R W
    apply (frule-tac C=R and x=W in augment-mono-map)
    apply (drule-tac A=S }\oplus\textrm{W}R\textrm{R}\mathrm{ and C=L and x=A in restrict-mono-map)
    apply (fastforce dest: read-only-mono shared-leD)
    done
apply (fastforce dest: read-only-mono shared-leD)
apply fastforce
apply fastforce
apply (clarsimp)
subgoal for A L R W
apply (frule-tac C=R and x=W in augment-mono-map)
apply (drule-tac A=S S }\mp@subsup{\textrm{w}}{\textrm{W}}{\textrm{R}}\mathrm{ and C=L and x=A in restrict-mono-map)
apply (fastforce dest: read-only-mono shared-leD)
done
done
done
locale outstanding-non-volatile-refs-owned-or-read-only =
fixes }\mathcal{S}:\mathrm{ :shared
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes outstanding-non-volatile-refs-owned-or-read-only:
\i is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb p
    |i < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
\Longrightarrow
non-volatile-owned-or-read-only False }\mathcal{S}\mathcal{O}\mathrm{ sb
locale outstanding-volatile-writes-unowned-by-others =
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes outstanding-volatile-writes-unowned-by-others:
\i pi is isio
|i < length ts; j < length ts; i\not=j;
```



```
】
\Longrightarrow
    (\mathcal{O}
locale read-only-reads-unowned =
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes read-only-reads-unowned:
```



```
    |i < length ts; j < length ts; i\not=j;
```



```
】
\Longrightarrow
    (\mathcal{O}
    read-only-reads (acquired True
```

locale ownership-distinct $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes ownership-distinct:
$\bigwedge_{\mathrm{i}}^{\mathrm{j} \mathrm{p}_{\mathrm{i}}} \mathrm{is}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}}$ is $\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
$\llbracket \mathrm{i}<$ length ts $; \mathrm{j}<$ length ts $; \mathrm{i} \neq \mathrm{j} ;$
$\operatorname{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right) ; \mathrm{ts}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
$\rrbracket \Longrightarrow\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
locale valid-ownership $=$
outstanding-non-volatile-refs-owned-or-read-only +
outstanding-volatile-writes-unowned-by-others +
read-only-reads-unowned +
ownership-distinct
locale outstanding-non-volatile-writes-unshared $=$
fixes $\mathcal{S}$ ::shared and ts::('p,'p store-buffer,bool,owns,rels) thread-config list assumes outstanding-non-volatile-writes-unshared:

```
\i p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
    |i< length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
    \Longrightarrow
```

    non-volatile-writes-unshared \(\mathcal{S}\) sb
    locale sharing-consis $=$
fixes $\mathcal{S}$ ::shared and ts::('p,'p store-buffer,bool,owns,rels) thread-config list assumes sharing-consis:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow$
sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb}$
locale no-outstanding-write-to-read-only-memory $=$
fixes $\mathcal{S}$ ::shared and ts::('p,'p store-buffer,bool,owns,rels) thread-config list assumes no-outstanding-write-to-read-only-memory:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow$
no-write-to-read-only-memory $\mathcal{S}$ sb
locale valid-sharing $=$
outstanding-non-volatile-writes-unshared +
sharing-consis +

```
read-only-unowned +
unowned-shared +
no-outstanding-write-to-read-only-memory
locale valid-ownership-and-sharing = valid-ownership +
outstanding-non-volatile-writes-unshared +
sharing-consis +
no-outstanding-write-to-read-only-memory
lemma (in read-only-reads-unowned)
    read-only-reads-unowned-nth-update:
\ p is \mathcal{O R \mathcal{D }}\vartheta\textrm{sb}.
    \llbracketi < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R});
        read-only-reads (acquired True (takeWhile (Not o is-volatile-Write sb) sb
            (dropWhile (Not ○ is-volatile-Write sb) sb)}\subseteq\mathrm{ read-only-reads (acquired True
(takeWhile (Not o is-volatile-Write sb) sb)\mathcal{O})
            (dropWhile (Not o is-volatile-Write 
        \mathcal { O } ^ { \prime } \cup \text { all-acquired } \mathrm { sb } ^ { \prime } \subseteq \mathcal { O } \cup \text { all-acquired sb} \rrbracket \Longrightarrow
        read-only-reads-unowned (ts[i := ( }\mp@subsup{\textrm{p}}{}{\prime},\mp@subsup{\textrm{is}}{}{\prime},\mp@subsup{\vartheta}{}{\prime},\mp@subsup{\textrm{sb}}{}{\prime},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime})]
    apply (unfold-locales)
    apply (clarsimp simp add: nth-list-update split: if-split-asm)
    apply (fastforce dest: read-only-reads-unowned)+
    done
```

lemma outstanding-non-volatile-refs-owned-or-read-only-tl: outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S} \quad(\mathrm{t} \# \mathrm{ts}) \quad \Longrightarrow \quad$ outstand-ing-non-volatile-refs-owned-or-read-only $\mathcal{S}$ ts
by (force simp add: outstanding-non-volatile-refs-owned-or-read-only-def)
lemma outstanding-volatile-writes-unowned-by-others-tl:
outstanding-volatile-writes-unowned-by-others $\quad(\mathrm{t} \# \mathrm{ts}) \quad \Longrightarrow$ outstand-
ing-volatile-writes-unowned-by-others ts
apply (clarsimp simp add: outstanding-volatile-writes-unowned-by-others-def)
apply fastforce
done
lemma read-only-reads-unowned-tl:
read-only-reads-unowned ( $\mathrm{t} \# \mathrm{ts}$ ) $\Longrightarrow$ read-only-reads-unowned (ts)
apply (clarsimp simp add: read-only-reads-unowned-def)
apply fastforce
done
lemma ownership-distinct-tl:
assumes dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ )

```
    shows ownership-distinct ts
proof -
    from dist
    interpret ownership-distinct t#ts .
    show ?thesis
    proof (rule ownership-distinct.intro)
        fix ijp is }\mathcal{O}\mathcal{R}\mathcal{D}\mathrm{ xs sb p' is' }\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\textrm{xs}}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime
        assume i-bound: i < length ts
        and j-bound: j < length ts
        and neq: i = j
        and ith: ts ! i = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        and jth: ts ! j = ( }\mp@subsup{p}{}{\prime},\mp@subsup{\textrm{is}}{}{\prime},\mp@subsup{xs}{}{\prime},\mp@subsup{sb}{}{\prime},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime}
    from i-bound j-bound neq ith jth
    show (\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{}{\prime}\cup\mathrm{ all-acquired sb})={}
        by - (rule ownership-distinct [of Suc i Suc j],auto)
    qed
qed
```

lemma valid-ownership-tl: valid-ownership $\mathcal{S}(\mathrm{t} \# \mathrm{ts}) \Longrightarrow$ valid-ownership $\mathcal{S}$ ts
by (auto simp add: valid-ownership-def
intro: outstanding-volatile-writes-unowned-by-others-tl
outstanding-non-volatile-refs-owned-or-read-only-tl ownership-distinct-tl
read-only-reads-unowned-tl)
lemma sharing-consistent-takeWhile:
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows sharing-consistent $\mathcal{S} \mathcal{O}$ (takeWhile P sb)
proof -
from consis have sharing-consistent $\mathcal{S} \mathcal{O}$ (takeWhile P sb @ dropWhile P sb)
by simp
with sharing-consistent-append [of - - takeWhile P sb dropWhile P sb]
show ?thesis
by simp
qed
lemma sharing-consis-tl: sharing-consis $\mathcal{S}(\mathrm{t} \# \mathrm{ts}) \Longrightarrow$ sharing-consis $\mathcal{S}$ ts
by (auto simp add: sharing-consis-def)
lemma sharing-consis-Cons:
【sharing-consis $\mathcal{S}$ ts; sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket$
$\Longrightarrow$ sharing-consis $\mathcal{S}((\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \# \mathrm{ts})$
apply (clarsimp simp add: sharing-consis-def)
subgoal for i pa isa $\mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{D}^{\prime} \vartheta^{\prime}$ sba
by (case-tac i) auto
done
lemma outstanding-non-volatile-writes-unshared-tl:
outstanding-non-volatile-writes-unshared $\mathcal{S}(\mathrm{t} \# \mathrm{ts}) \Longrightarrow$
outstanding-non-volatile-writes-unshared $\mathcal{S}$ ts
by (auto simp add: outstanding-non-volatile-writes-unshared-def)
lemma no-outstanding-write-to-read-only-memory-tl:
no-outstanding-write-to-read-only-memory $\mathcal{S}(\mathrm{t} \# \mathrm{ts}) \Longrightarrow$
no-outstanding-write-to-read-only-memory $\mathcal{S}$ ts
by (auto simp add: no-outstanding-write-to-read-only-memory-def)
lemma valid-ownership-and-sharing-tl:
valid-ownership-and-sharing $\mathcal{S}(\mathrm{t} \# \mathrm{ts}) \Longrightarrow$ valid-ownership-and-sharing $\mathcal{S}$ ts
apply (clarsimp simp add: valid-ownership-and-sharing-def)
apply (auto intro: valid-ownership-tl
outstanding-non-volatile-writes-unshared-tl
no-outstanding-write-to-read-only-memory-tl
sharing-consis-tl)
done
lemma non-volatile-owned-or-read-only-outstanding-non-volatile-writes:
$\bigwedge \mathcal{O} \mathcal{S}$ pending-write. 【non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket$ $\Longrightarrow$
outstanding-refs is-non-volatile-Write $\mathrm{s}_{\mathrm{sb}} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case $\left(\mathrm{Write}_{\text {sb }}\right.$ volatile a sop v A L R W)
show ?thesis proof (cases volatile)
case True
from Cons.hyps [of True $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ ] Cons.prems
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True)
next
case False with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case Read ${ }_{\text {sb }}$ with Cons show ?thesis
by auto
next
case $\operatorname{Prog}_{\text {sb }}$ with Cons show ?thesis by auto
next
case (Ghost sb $_{\text {sb }}$ L R W)
from Cons.hyps [of pending-write $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ ] Cons.prems show ?thesis
by (auto simp add: Ghost ${ }_{\text {sb }}$ )

```
    qed
```

qed
lemma (in outstanding-non-volatile-refs-owned-or-read-only) outstand-ing-non-volatile-writes-owned:
assumes i-bound: $\mathrm{i}<$ length ts
assumes ts-i: ts! $1=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
shows outstanding-refs is-non-volatile-Writesb $s b \subseteq \mathcal{O} \cup$ all-acquired sb
using non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF outstand-ing-non-volatile-refs-owned-or-read-only [OF i-bound ts-i]]
by blast
lemma non-volatile-reads-acquired-or-read-only:
$\bigwedge \mathcal{O} \mathcal{S}$. [non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb; sharing-consistent $\mathcal{S} \mathcal{O}$ sb』 $\Longrightarrow$
outstanding-refs is-non-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb $\cup$ read-only $\mathcal{S}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case $\left(\mathrm{Write}_{\text {sb }}\right.$ volatile a sop v A L R W)
show ?thesis proof (cases volatile)
case True
from Cons.prems obtain non-vol: non-volatile-owned-or-read-only True $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R}\right.$ $\left.\ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
A-shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Write sb $^{\text {True }}$ )
from Cons.hyps [OF non-vol consis']
have hyp: outstanding-refs is-non-volatile- $\operatorname{Read}_{s b} \mathrm{sb}$ $\subseteq \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb $\cup$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
with R-owns A-R L-A
show ?thesis
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ True )
apply (drule (1) rev-subsetD)
apply (auto simp add: in-read-only-convs split: if-split-asm)
done
next
case False with Cons show ?thesis

```
by (auto simp add: Write sb)
    qed
    next
        case Read
        by auto
    next
        case Prog
        by auto
    next
        case (Ghost sb A L R W)
        from Cons.prems obtain non-vol: non-volatile-owned-or-read-only True (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R
L) (\mathcal{O}\cupA - R) sb and
        A-shared-onws: A \subseteq dom S \cup\mathcal{O}\mathrm{ and L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:}
R\subseteq\mathcal{O}\mathrm{ and}
        consis': sharing-consistent (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R}) sb
        by (clarsimp simp add: Ghostsb )
        from Cons.hyps [OF non-vol consis]
        have hyp: outstanding-refs is-non-volatile-Read }\mp@subsup{\mp@code{sb}}{sb}{sb
            \subseteq\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb }\cup\mathrm{ read-only (S }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}).
        with R-owns A-R L-A
        show ?thesis
            apply (clarsimp simp add: Ghost sb )
        apply (drule (1) rev-subsetD)
        apply (auto simp add: in-read-only-convs split: if-split-asm)
        done
    qed
qed
```

lemma non-volatile-reads-acquired-or-read-only-reads:
$\bigwedge \mathcal{O} \mathcal{S}$ pending-write. 【non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket$ $\Longrightarrow$
outstanding-refs is-non-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb $\cup$ read-only-reads $\mathcal{O}$ sb proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case $\left(\mathrm{Write}_{s b}\right.$ volatile a sop v A L R W)
show ?thesis proof (cases volatile)
case True
from Cons.prems obtain non-vol: non-volatile-owned-or-read-only True $(\mathcal{S} \oplus \mathrm{w}$
$\left.\ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Write sb $^{\text {True }}$ )
from Cons.hyps [OF non-vol ]
have hyp: outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ sb
$\subseteq \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb $\cup$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb.
then
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True )
next
case False with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case $\operatorname{Read}_{\text {sb }}$ with Cons show ?thesis
by auto
next
case $\operatorname{Prog}_{s b}$ with Cons show ?thesis
by auto
next
case ( Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain non-vol: non-volatile-owned-or-read-only pending-write ( $\mathcal{S}$
$\left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$ by (clarsimp simp add: Ghostsb )
from Cons.hyps [OF non-vol ]
have hyp: outstanding-refs is-non-volatile-Read ${ }_{s b}$ sb
$\subseteq \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb $\cup$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb.
then
show ?thesis
by (auto simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
lemma non-volatile-owned-or-read-only-outstanding-refs:
$\wedge \mathcal{O} \mathcal{S}$ pending-write. 【non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ sb】 $\Longrightarrow$
outstanding-refs (Not $\circ$ is-volatile) $\mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb $\cup$ read-only-reads $\mathcal{O}$ sb proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.hyps [of True $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ ] Cons.prems
show ?thesis

```
by (auto simp add: Write \({ }_{\text {sb }}\) True)
    next
        case False with Cons show ?thesis
by (auto simp add: Writesb \({ }_{\text {s }}\) )
    qed
    next
        case Read \({ }_{\text {sb }}\) with Cons show ?thesis
        by auto
    next
        case \(\operatorname{Prog}_{\text {sb }}\) with Cons show ?thesis
        by auto
    next
        case (Ghost sb \(_{\text {sb }}\) L R W)
        from Cons.hyps [of pending-write \(\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) ] Cons.prems
        show ?thesis
            by (auto simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
```

lemma no-unacquired-write-to-read-only:
$\bigwedge \mathcal{S} \mathcal{O}$. $\llbracket$ no-write-to-read-only-memory $\mathcal{S}$ sb;sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} ;$
$\mathrm{a} \in \operatorname{read}-$ only $\mathcal{S} ; \mathrm{a} \notin(\mathcal{O} \cup$ all-acquired sb$) \rrbracket$
$\Longrightarrow \mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain no-wrt: no-write-to-read-only-memory $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ sb and
A-shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and a-ro: a $\in$ read-only $\mathcal{S}$ and
a-A: a $\notin \mathrm{A}$ and a-all-acq: a $\notin$ all-acquired sb and a-owns: a $\notin \mathcal{O}$ and $\mathrm{a}^{\prime}$-notin: $\mathrm{a}^{\prime} \notin$ read-only $\mathcal{S}$ by ( simp add: Write ${ }_{\text {sb }}$ True )
from $a^{\prime}$-notin a-ro have neq- $a-a^{\prime}: a \neq a^{\prime}$
by blast
from a-A a-all-acq a-owns
have a-notin': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb
by auto
from a-ro L-A a-A R-owns a-owns
have a $\in$ read-only $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by (auto simp add: in-read-only-convs split: if-split-asm)
from Cons.hyps [OF no-wrt consis' this a-notin']
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb.
with neq-a-a ${ }^{\prime}$
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
next
case False with Cons
show ?thesis
by (auto simp add: Write sb $_{\text {sb }}$ False)
qed
next
case Read ${ }_{\text {sb }}$ with Cons
show ?thesis
by (auto)
next
case $\operatorname{Prog}_{\text {sb }}$ with Cons
show ?thesis
by (auto)
next
case Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain no-wrt: no-write-to-read-only-memory $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{sb}$ and
A-shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and a-ro: a $\in$ read-only $\mathcal{S}$ and
a-A: a $\notin \mathrm{A}$ and a-all-acq: a $\notin$ all-acquired sb and a-owns: a $\notin \mathcal{O}$
by ( simp add: Ghost ${ }_{\text {sb }}$ )
from a-A a-all-acq a-owns
have a-notin': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb
by auto
from a-ro $\mathrm{L}-\mathrm{A}$ a-A R-owns a-owns
have a $\in$ read-only $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by (auto simp add: in-read-only-convs split: if-split-asm)
from Cons.hyps [OF no-wrt consis' this a-notin']
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb.
then
show ?thesis
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
lemma read-only-reads-read-only:
$\bigwedge \mathcal{S} \mathcal{O}$. 【non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb;
sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket$
$\Longrightarrow$
read-only-reads $\mathcal{O} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb $\cup$ read-only $\mathcal{S}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{s b}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain non-vol: non-volatile-owned-or-read-only True $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R}\right.$ $\left.\ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
A-shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True )
from Cons.hyps [OF non-vol consis ${ }^{\prime}$ ]
have hyp: read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
$\subseteq \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb $\cup$ read-only $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.

## \{ <br> fix $\mathrm{a}^{\prime}$

assume $\mathrm{a}^{\prime}$-in: $\mathrm{a}^{\prime} \in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
assume $\mathrm{a}^{\prime}$-unowned: $\mathrm{a}^{\prime} \notin \mathcal{O}$
assume $a^{\prime}$-unacq: $a^{\prime} \notin$ all-acquired sb
assume $a^{\prime}-A: a^{\prime} \notin A$
have $\mathrm{a}^{\prime} \in$ read-only $\mathcal{S}$
proof -
from $a^{\prime}$-in hyp $a^{\prime}$-unowned $a^{\prime}$-unacq $a^{\prime}-\mathrm{A}$
have $\mathrm{a}^{\prime} \in \operatorname{read}$-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by auto
with L-A R-owns a'-unowned
show ?thesis
by (auto simp add: in-read-only-convs split:if-split-asm)
qed
\}
then
show ?thesis
apply (clarsimp simp add: Write sb $^{\text {True simp del: o-apply) }}$

```
apply force
done
    next
        case False with Cons show ?thesis
by (auto simp add: Write sb)
    qed
next
    case Read
        by auto
next
    case Prog
        by auto
next
    case (Ghost sb A L R W)
    from Cons.prems obtain non-vol: non-volatile-owned-or-read-only True (S }\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R
L) (\mathcal{O}\cupA - R) sb and
        A-shared-onws: A \subseteq dom S \cup\mathcal{O}\mathrm{ and L-A:L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:}
R\subseteq\mathcal{O}\mathrm{ and}
            consis': sharing-consistent (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R}) sb
            by (clarsimp simp add: Ghost sb )
    from Cons.hyps [OF non-vol consis']
    have hyp: read-only-reads (\mathcal{O}\cup\textrm{A}-\textrm{R}) sb
                                    \mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb }\cup\mathrm{ read-only (S }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}).
    {
        fix a'
        assume a'-in: a' }\in\mathrm{ read-only-reads (O) 
        assume a'-unowned: a' }\not\in\mathcal{O
        assume a'-unacq: a' }\not=\mathrm{ all-acquired sb
        assume a'-A: a' }\not\in\textrm{A
        have a'\in read-only }\mathcal{S
        proof -
from a'-in hyp a'-unowned a'-unacq a'-A
have a'\in read-only (\mathcal{S}\oplus\textrm{w}}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
    by auto
with L-A R-owns a'-unowned
show ?thesis
    by (auto simp add: in-read-only-convs split:if-split-asm)
        qed
    }
    then
    show ?thesis
        apply (clarsimp simp add: Ghost sb simp del: o-apply)
        apply force
        done
```

qed
qed
lemma no-unacquired-write-to-read-only-reads:
$\bigwedge \mathcal{S} \mathcal{O} . \llbracket$ no-write-to-read-only-memory $\mathcal{S}$ sb;
non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O} \mathrm{sb}$; sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb}$;
$\mathrm{a} \in$ read-only-reads $\mathcal{O}$ sb; a $\notin(\mathcal{O} \cup$ all-acquired sb)】
$\Longrightarrow \mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain no-wrt: no-write-to-read-only-memory $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{sb}$ and
non-vol: non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and A-shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{W} R \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and a-ro: a $\in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-A: a $\notin \mathrm{A}$ and a-all-acq: a $\notin$ all-acquired sb and a-owns: a $\notin \mathcal{O}$ and
$a^{\prime}$-notin: $a^{\prime} \notin$ read-only $\mathcal{S}$
by ( simp add: Write ${ }_{\text {sb }}$ True )
from read-only-reads-read-only [OF non-vol consis ${ }^{\prime}$ ] a-ro a-owns a-all-acq a-A
have a $\in$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by auto
with $a^{\prime}$-notin R-owns a-owns have neq-a-a': $a \neq a^{\prime}$
by (auto simp add: in-read-only-convs split: if-split-asm)
from a-A a-all-acq a-owns
have a-notin': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb
by auto
from Cons.hyps [OF no-wrt non-vol consis' a-ro a-notin']
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb.
then
show ?thesis
using neq-a-a'
by (auto simp add: Write ${ }_{\text {sb }}$ True)
next
case False with Cons
show ?thesis

```
by (auto simp add: Writesb False)
        qed
    next
    case (Readsb volatile a't v)
    show ?thesis
    proof (cases volatile)
        case True
        with Cons show ?thesis
by (auto simp add: Read
    next
        case False
        note non-volatile = this
        from Cons.prems obtain no-wrt': no-write-to-read-only-memory S sb and
consis':sharing-consistent \mathcal{O}}\mathbf{\mathcal{sb}}\mathrm{ and
a-in: a }\in(\mathrm{ if a' }\not\in\mathcal{O}\mathrm{ then insert a' (read-only-reads }\mathcal{O}\mathrm{ sb)
                    else read-only-reads (\mathcal{Osb})\mathrm{ and}
a'-owns-shared: a'}\in\mathcal{O}\vee \mp@subsup{a}{}{\prime}\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
non-vol': non-volatile-owned-or-read-only True }\mathcal{S}\mathcal{O}\mathrm{ sb and
    a-owns: a }\not\in\mathcal{O}\cup\mathrm{ all-acquired sb
by (clarsimp simp add: Read}\mp@subsup{\mp@code{sb}}{}{\mathrm{ False)}
show ?thesis
proof (cases a' }\in\mathcal{O}
case True
with a-in have a }\in\mathrm{ read-only-reads }\mathcal{O}\mathrm{ sb
    by auto
from Cons.hyps [OF no-wrt' non-vol' consis' this a-owns]
show ?thesis
    by (clarsimp simp add: Read
        next
case False
note a'-unowned = this
with a-in have a-in': a \in insert a' (read-only-reads \mathcal{O sb) by auto}
from a'-owns-shared False have a'-read-only: a' }\in\mathrm{ read-only }\mathcal{S}\mathrm{ by auto
show ?thesis
proof (cases a=a')
    case False
    with a-in' have a }\in\mathrm{ (read-only-reads }\mathcal{O}\mathrm{ sb) by auto
    from Cons.hyps [OF no-wrt' non-vol' consis' this a-owns]
    show ?thesis
        by (simp add: Readsb
next
    case True
    from no-unacquired-write-to-read-only [OF no-wrt' consis' a'-read-only] a-owns True
    have a' }\not\in\mathrm{ outstanding-refs is-Write sb sb
        by auto
    then show ?thesis
        by (simp add: Readsb True)
qed
```

```
        qed
    qed
    next
        case Progsb with Cons
    show ?thesis
        by (auto)
next
    case (Ghost sb A L R W)
    from Cons.prems obtain no-wrt: no-write-to-read-only-memory (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{\prime}R\mp@subsup{|}{\textrm{A}}{\prime}}\textrm{L})\textrm{sb
and
    non-vol: non-volatile-owned-or-read-only True (\mathcal{S}\oplus\textrm{W}}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and
    A-shared-onws: A \subseteq dom \mathcal{S}\cup\mathcal{O}\mathrm{ and L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:}
R}\subseteq\mathcal{O}\mathrm{ and
        consis': sharing-consistent (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and
        a-ro: a }\in\mathrm{ read-only-reads ( }\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and
        a-A: a }\not\in\textrm{A}\mathrm{ and a-all-acq: a }\not\in\mathrm{ all-acquired sb and a-owns: a }\not\in\mathcal{O
        by ( simp add: Ghostsb )
    from read-only-reads-read-only [OF non-vol consis'] a-ro a-owns a-all-acq a-A
    have a }\in\mathrm{ read-only (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
            by auto
        from a-A a-all-acq a-owns
        have a-notin': a }\not\in\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb
            by auto
        from Cons.hyps [OF no-wrt non-vol consis' a-ro a-notin']
        have a }\not=\mathrm{ outstanding-refs is-Write esb sb.
        then
        show ?thesis
        by (auto simp add: Ghostsb
    qed
qed
```

lemma no-unacquired-write-to-read-only":
assumes no-wrt: no-write-to-read-only-memory $\mathcal{S}$ sb
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows read-only $\mathcal{S} \cap$ outstanding-refs is-Write ${ }_{\text {sb }} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb
using no-unacquired-write-to-read-only [OF no-wrt consis]
by auto
lemma no-unacquired-volatile-write-to-read-only:
assumes no-wrt: no-write-to-read-only-memory $\mathcal{S}$ sb
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows read-only $\mathcal{S} \cap$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb
proof -
have outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ sb $\subseteq$ outstanding-refs is-Write $_{\text {sb }}$ sb apply (rule outstanding-refs-mono-pred)

```
    apply (auto simp add: is-volatile-Write }\mp@subsup{\mathrm{ sb}}{\mathrm{ -def split: memref.splits)}}{\mathrm{ (den}
    done
    with no-unacquired-write-to-read-only"} [OF no-wrt consis]
    show ?thesis by blast
qed
lemma no-unacquired-non-volatile-write-to-read-only-reads:
    assumes no-wrt: no-write-to-read-only-memory }\mathcal{S}\mathrm{ sb
    assumes consis: sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb
    shows read-only \mathcal{S}\cap\mathrm{ outstanding-refs is-non-volatile-Write sb sb }\subseteq\mathcal{O}\cup\mathrm{ all-acquired sb}
proof -
    from outstanding-refs-subsets
    have outstanding-refs is-non-volatile-Write
assumption
    with no-unacquired-write-to-read-only"} [OF no-wrt consis
    show ?thesis by blast
qed
```

lemma no-unacquired-write-to-read-only-reads':
assumes no-wrt: no-write-to-read-only-memory $\mathcal{S}$ sb
assumes non-vol: non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows read-only-reads $\mathcal{O}$ sb $\cap$ outstanding-refs is-Write ${ }_{\text {sb }} \mathrm{sb} \subseteq \mathcal{O} \cup$ all-acquired sb using no-unacquired-write-to-read-only-reads [OF no-wrt non-vol consis]
by auto
lemma no-unacquired-volatile-write-to-read-only-reads:
assumes no-wrt: no-write-to-read-only-memory $\mathcal{S}$ sb
assumes non-vol: non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows read-only-reads $\mathcal{O}$ sb $\cap$ outstanding-refs is-volatile-Write sb sb $\subseteq \mathcal{O} \cup$ all-acquired
sb
proof -
have outstanding-refs is-volatile-Write sb $^{s b} \subseteq$ outstanding-refs is-Write $_{\text {sb }}$ sb
apply (rule outstanding-refs-mono-pred)
apply (auto simp add: is-volatile-Write ${ }_{\mathbf{s b}}$-def split: memref.splits) done
with no-unacquired-write-to-read-only-reads [OF no-wrt non-vol consis]
show ?thesis by blast
qed
lemma no-unacquired-non-volatile-write-to-read-only:
assumes no-wrt: no-write-to-read-only-memory $\mathcal{S}$ sb
assumes non-vol: non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows read-only-reads $\mathcal{O}$ sb $\cap$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb $\subseteq \mathcal{O} \cup$ all-acquired sb
proof -
from outstanding-refs-subsets
have outstanding-refs is-non-volatile-Write ${ }_{s b}$ sb $\subseteq$ outstanding-refs is-Write ${ }_{s b}$ sb by assumption
with no-unacquired-write-to-read-only-reads [OF no-wrt non-vol consis]
show ?thesis by blast
qed
lemma set-dropWhileD: $x \in \operatorname{set}($ dropWhile $P$ xs) $\Longrightarrow x \in$ set $x s$ by (induct xs) (auto split: if-split-asm)
lemma outstanding-refs-takeWhileD:
$\mathrm{x} \in$ outstanding-refs P (takeWhile $\left.\mathrm{P}^{\prime} \mathrm{sb}\right) \Longrightarrow \mathrm{x} \in$ outstanding-refs P sb using outstanding-refs-takeWhile
by blast
lemma outstanding-refs-dropWhileD:
$\mathrm{x} \in$ outstanding-refs P (dropWhile $\left.\mathrm{P}^{\prime} \mathrm{sb}\right) \Longrightarrow \mathrm{x} \in$ outstanding-refs P sb by (auto dest: set-dropWhileD simp add: outstanding-refs-conv)
lemma dropWhile-ConsD: dropWhile $P$ xs $=y \# y s \Longrightarrow \neg P y$ by (simp add: dropWhile-eq-Cons-conv)
lemma non-volatile-owned-or-read-only-drop: non-volatile-owned-or-read-only False $\mathcal{S} \mathcal{O}$ sb $\Longrightarrow$ non-volatile-owned-or-read-only True
(share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{S}$ )
(acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O}$ )
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
using non-volatile-owned-or-read-only-append [of False $\mathcal{S} \mathcal{O}$ (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb)
(dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb)]
apply (cases outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends
takeWhile-not-vol-write-outstanding-refs dropWhile-not-vol-write-empty)
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends
takeWhile-not-vol-write-outstanding-refs dropWhile-not-vol-write-empty )
apply (case-tac (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb))
apply (fastforce simp add: outstanding-refs-conv)
apply (frule dropWhile-ConsD)
apply (clarsimp split: memref.splits)
done
lemma read-only-share: $\bigwedge \mathcal{S} \mathcal{O}$.
sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow$
read-only $($ share $\operatorname{sb} \mathcal{S}) \subseteq$ read-only $\mathcal{S} \cup \mathcal{O} \cup$ all-acquired sb

```
proof (induct sb)
    case Nil thus ?case by auto
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            from Cons.prems obtain
    A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and L-A:L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:
R\subseteq\mathcal{O}\mathrm{ and}
    consis': sharing-consistent (\mathcal{S}\oplus\textrm{w}R
    by (clarsimp simp add: Write sb True )
        from Cons.hyps [OF consis']
        have read-only (share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}))
                \subseteq \operatorname { r e a d - o n l y ~ } ( \mathcal { S } \oplus \mathrm { W } R \mathrm { R } \ominus _ { \mathrm { A } } \mathrm { L } ) \cup ( \mathcal { O } \cup \mathrm { A } - \mathrm { R } ) \cup \text { all-acquired sb}
                by auto
            also from A-shared-owns L-A R-owns A-R
            have read-only }(\mathcal{S}\oplus\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\cup(\mathcal{O}\cup\textrm{A}-\textrm{R})\cup\mathrm{ all-acquired sb }
                read-only }\mathcal{S}\cup\mathcal{O}\cup(\textrm{A}\cup\mathrm{ all-acquired sb)
                    by (auto simp add: read-only-def augment-shared-def restrict-shared-def split:
option.splits)
            finally
            show ?thesis
                by (simp add: Write sb True)
    next
            case False with Cons show ?thesis
by (auto simp add: Writesb)
    qed
    next
        case Read
            by auto
    next
        case Prog
                by auto
    next
        case (Ghost sb A L R W)
        from Cons.prems obtain
            A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:
R}\subset\mathcal{O}\mathrm{ and
                consis': sharing-consistent (\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}L
        by (clarsimp simp add: Ghostsb )
    from Cons.hyps [OF consis']
    have read-only (share sb (\mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
                \subseteq \mp@code { r e a d - o n l y ~ } ( \mathcal { S } \oplus \mathrm { W } R \mathrm { R } \ominus _ { \mathrm { A } } \mathrm { L } ) \cup ( \mathcal { O } \cup \mathrm { A } - \mathrm { R } ) \cup \text { all-acquired sb}
        by auto
    also from A-shared-owns L-A R-owns A-R
    have read-only }(\mathcal{S}\oplus\textrm{w}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\cup(\mathcal{O}\cup\textrm{A}-\textrm{R})\cup\mathrm{ all-acquired sb }
```

read-only $\mathcal{S} \cup \mathcal{O} \cup(\mathrm{A} \cup$ all-acquired sb)
by (auto simp add: read-only-def augment-shared-def restrict-shared-def split: option.splits)
finally
show ?thesis by (simp add: Ghostsb)
qed
qed
lemma (in valid-ownership-and-sharing) outstanding-non-write-non-vol-reads-drop-disj:
assumes i-bound: i < length ts
assumes j-bound: $\mathrm{j}<$ length ts
assumes neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assumes ith: ts! $\mathrm{i}_{\mathrm{i}}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{isi}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
assumes jth: ts! $=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
shows outstanding-refs is-Write $\mathbf{e}_{\mathbf{s b}}\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathbf{i}}\right) \cap$
outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}\left(\right.$ dropWhile $\left(\right.$ Not $^{\circ} \circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb $_{\mathrm{j}}$ )
$=\{ \}$
proof -
let ? take- $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ?drop- $\mathrm{j}=\left(\right.$ drop While $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ?take- $\mathrm{i}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{i}}\right)$
let ?drop- $\mathrm{i}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{i}}\right)$
note nvo-i $=$ outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ith]
note nvo-j = outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]
note nro-i = no-outstanding-write-to-read-only-memory [OF i-bound ith]
with no-write-to-read-only-memory-append [of $\mathcal{S}$ ?take-i ?drop-i]
have nro-drop-i: no-write-to-read-only-memory (share ?take-i $\mathcal{S}$ ) ?drop-i
by simp
note nro-j $=$ no-outstanding-write-to-read-only-memory [OF j-bound jth]
with no-write-to-read-only-memory-append [of $\mathcal{S}$ ?take-j ?drop-j]
have nro-drop-j: no-write-to-read-only-memory (share ?take-j $\mathcal{S}$ ) ?drop-j
by simp
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound neq-i-j ith jth]
have dist: $\left(\mathcal{O}_{j} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{i}}=\{ \}$.
note own-dist $=$ ownership-distinct [OF i-bound j-bound neq-i-j ith jth]
from sharing-consis [OF j-bound jth]
have consis-j: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
with sharing-consistent-append [of $\mathcal{S} \mathcal{O}_{\mathrm{j}}$ ?take-j ?drop-j]
obtain
consis-take-j: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{j}}$ ?take-j and
consis-drop-j: sharing-consistent (share ?take-j $\mathcal{S}$ ) (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop-j by simp
from sharing-consis [OF i-bound ith]
have consis-i: sharing-consistent $\mathcal{S} \mathcal{O}_{i}$ sb $_{i}$.
with sharing-consistent-append [of $\mathcal{S} \mathcal{O}_{\mathrm{i}}$ ?take-i ?drop-i]
have consis-drop-i: sharing-consistent (share ?take-i $\mathcal{S}$ ) (acquired True ?take-i $\mathcal{O}_{\mathrm{i}}$ ) ?drop-i by simp

## \{

fix x
assume x -in-drop-i: $\mathrm{x} \in$ outstanding-refs is-Write $\mathrm{e}_{\mathrm{sb}}$ ?drop-i
assume x -in-drop-j: $\mathrm{x} \in$ outstanding-refs is-non-volatile-Read ${ }_{\mathrm{sb}}$ ?drop-j
have False
proof -
from x-in-drop-i have x -in-i: $\mathrm{x} \in$ outstanding-refs is-Write $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{i}}$
using outstanding-refs-append [of is-Write ${ }_{\text {sb }}$ ?take-i ?drop-i] by auto
from x -in-drop-j have x -in- $\mathrm{j}: \mathrm{x} \in$ outstanding-refs is-non-volatile-Read $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$
using outstanding-refs-append [of is-non-volatile-Read ${ }_{\text {sb }}$ ?take-j ?drop-j]
by auto
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-drop-j: non-volatile-owned-or-read-only True (share ?take-j $\mathcal{S}$ ) (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop-j.
from non-volatile-reads-acquired-or-read-only-reads [OF nvo-drop-j] x-in-drop-j
acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}\left[\right.$ of $\operatorname{sb}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ]
have x - $\mathrm{j}: \mathrm{x} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}} \cup$ read-only-reads (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ )
?drop-j
using all-acquired-append [of ?take-j ?drop-j]
by ( auto )
\{
assume x -in-vol-drop-i: $\mathrm{x} \in$ outstanding-refs is-volatile-Write $_{\text {sb }}$ ?drop-i
hence x -in-vol-i: $\mathrm{x} \in$ outstanding-refs is-volatile-Write ${ }_{s b} \mathrm{sb}_{\mathrm{i}}$
using outstanding-refs-append [of is-volatile-Write ${ }_{\text {sb }}$ ?take-i ?drop-i]
by auto
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound neq-i-j ith jth] have $\left(\mathcal{O}_{j} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile-Write $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{i}}=\{ \}$.
with x -in-vol-i x -j obtain
x -unacq- $\mathrm{j}: \mathrm{x} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$ and
x -ror-j: $\mathrm{x} \in$ read-only-reads (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop-j
by auto
from read-only-reads-unowned [OF j-bound i-bound neq-i-j [symmetric] jth ith] x-ror-j
have $\mathrm{x} \notin \mathcal{O}_{\mathrm{i}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{i}}$
by auto

```
from read-only-reads-read-only [OF nvo-drop-j consis-drop-j] x-ror-j x-unacq-j
    all-acquired-append [of ?take-j ?drop-j] acquired-takeWhile-non-volatile-Write sb [of sb }\mp@subsup{}{j}{
O
have x }\in\mathrm{ read-only (share ?take-j S
    by (auto)
            from read-only-share [OF consis-take-j] this x-unacq-j all-acquired-append [of ?take-j
?drop-j]
            have x }\in\mathrm{ read-only }\mathcal{S
        by auto
with no-unacquired-write-to-read-only" [OF nro-i consis-i] x-in-i
have x }\in\mp@subsup{\mathcal{O}}{\textrm{i}}{}\cup\mathrm{ all-acquired sbi
    by auto
ultimately have False by auto
        }
        moreover
        {
assume x-in-non-vol-drop-i: x f outstanding-refs is-non-volatile-Writesb
hence }x\in\mathrm{ outstanding-refs is-non-volatile-Write sb sb
        using outstanding-refs-append [of is-non-volatile-Write}\mp@subsup{e}{\mathrm{ sb }}{}\mathrm{ ?take-i ?drop-i]
        by auto
with non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF nvo-i]
have x }\in\mp@subsup{\mathcal{O}}{i}{}\cup\mathrm{ all-acquired sbi}\mathrm{ by auto
moreover
with x-j own-dist obtain
    x-unacq-j:x }\not\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
    x-ror-j: x }\in\mathrm{ read-only-reads (acquired True ?take-j }\mp@subsup{\mathcal{O}}{j}{}\mathrm{ ) ?drop-j
    by auto
from read-only-reads-unowned [OF j-bound i-bound neq-i-j [symmetric] jth ith] x-ror-j
have x }\not\in\mp@subsup{\mathcal{O}}{\textrm{i}}{}\cup\mathrm{ all-acquired sb
        by auto
ultimately have False
    by auto
        }
        ultimately
        show ?thesis
using x-in-drop-i x-in-drop-j
by (auto simp add: misc-outstanding-refs-convs)
        qed
}
thus ?thesis
```

by auto
qed
lemma (in valid-ownership-and-sharing) outstanding-non-volatile-write-disj:
assumes i-bound: i < length ts
assumes j-bound: $\mathrm{j}<$ length ts
assumes neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assumes ith: ts! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
assumes jth: ts! $\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
shows outstanding-refs (is-non-volatile-Write ${ }_{s b}$ ) (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\left.\mathrm{sb}_{\mathrm{i}}\right) \cap$
(outstanding-refs is-volatile-Write ${ }_{s b} \mathrm{sb}_{\mathrm{j}} \cup$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}} \cup$
outstanding-refs is-non-volatile-Read sb $\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right) \operatorname{sb}_{j}$ )
$\cup$
(outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}\left(\right.$ takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}$ )
-
read-only-reads $\mathcal{O}_{\mathrm{j}}\left(\right.$ takeWhile (Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)\right) \cup$
$\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.\mathrm{sb}_{\mathrm{s}}\right) \mathrm{sb}_{\mathrm{j}}\right)$ )
$)=\{ \}$ (is ?non-vol-writes-i $\cap$ ?not-volatile-j $=\{ \})$
proof -
note nro-i $=$ no-outstanding-write-to-read-only-memory [OF i-bound ith]
note nro-j $=$ no-outstanding-write-to-read-only-memory [OF j-bound jth]
note nvo-j $=$ outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]
note nvo-i $=$ outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ith]
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound neq-i-j ith jth] have dist: $\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{i}}=\{ \}$.
from outstanding-volatile-writes-unowned-by-others [OF j-bound i-bound neq-i-j [symmetric] jth ith]
have dist-j: $\left(\mathcal{O}_{i} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right) \cap$ outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \operatorname{sb}_{\mathrm{j}}=\{ \}$.
note own-dist $=$ ownership-distinct [OF i-bound j-bound neq-i-j ith jth]
from sharing-consis [OF j-bound jth]
have consis-j: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
from sharing-consis [OF i-bound ith]
have consis-i: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$.
let ?take-j $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)$
let ?drop- $\mathrm{j}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
\{
fix x
assume $x$-in-take-i: $x \in$ ?non-vol-writes-i
assume $x-i n-j: x \in$ ?not-volatile-j
from $x$-in-take-i have $x$-in-i: $x \in$ outstanding-refs (is-non-volatile-Write ${ }_{s b}$ ) sb $_{i}$
by (auto dest: outstanding-refs-takeWhileD)
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF nvo-i] x-in-i have x-in-owns-acq-i: $x \in \mathcal{O}_{i} \cup$ all-acquired $s_{i}$ by auto
have False
proof \{
assume $\mathrm{x}-\mathrm{in}-\mathrm{j}: \mathrm{x} \in$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$
with dist-j have x-notin: $\mathrm{x} \notin\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)$
by auto
with x-in-owns-acq-i have False
by auto
\}
moreover
\{
assume x -in-j: $\mathrm{x} \in$ outstanding-refs is-non-volatile-Write $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF nvo-j] x-in-j
have $\mathrm{x} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
by auto
with x-in-owns-acq-i own-dist
have False
by auto
\}
moreover
\{
assume $\mathrm{x}-\mathrm{in}-\mathrm{j}: \mathrm{x} \in$ outstanding-refs $^{\text {is-non-volatile-Read }}{ }_{\mathrm{sb}}$ ?drop-j
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo': non-volatile-owned-or-read-only True (share ?take-j $\mathcal{S}$ ) (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop-j.
from non-volatile-owned-or-read-only-outstanding-refs [OF nvo $]$ x-in-j
have $\mathrm{x} \in$ acquired True ?take-j $\mathcal{O}_{\mathrm{j}} \cup$ all-acquired ?drop-j $\cup$
read-only-reads (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop-j
by (auto simp add: misc-outstanding-refs-convs)

## moreover

from acquired-append [of True ?take-j ?drop-j $\mathcal{O}_{\mathrm{j}}$ ] acquired-all-acquired [of True ?take-j $\mathcal{O}_{\mathrm{j}}$ ]
all-acquired-append [of ?take-j ?drop-j]
have acquired True ?take-j $\mathcal{O}_{\mathrm{j}} \cup$ all-acquired ?drop-j $\subseteq \mathcal{O}_{\mathrm{j}} \cup$ all-acquired sb $\mathrm{j}_{\mathrm{j}}$
by auto
ultimately
have $\mathrm{x} \in$ read-only-reads (acquired True ?take-j $\mathcal{O}_{\mathrm{j}}$ ) ?drop- j
using x-in-owns-acq-i own-dist
by auto
with read-only-reads-unowned [OF j-bound i-bound neq-i-j [symmetric] jth ith] x-in-owns-acq-i
have False

```
by auto
    }
    moreover
    {
assume x-in-j: x }\in\mathrm{ outstanding-refs is-non-volatile-Read }\mp@subsup{}{\textrm{sb}}{}\mathrm{ ?take-j
assume x-notin: x }\not\in\mathrm{ read-only-reads }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ ?take-j
from non-volatile-owned-or-read-only-append [where xs=?take-j and ys=?drop-j] nvo-j
have non-volatile-owned-or-read-only False S S O
by auto
from non-volatile-owned-or-read-only-outstanding-refs [OF this] x-in-j x-notin
have }\textrm{x}\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired ?take-j
    by (auto simp add: misc-outstanding-refs-convs )
with all-acquired-append [of ?take-j ?drop-j] x-in-owns-acq-i own-dist
have False
    by auto
        }
        moreover
        {
assume x-in-j: x }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired ?take-j
moreover
from all-acquired-append [of ?take-j ?drop-j]
have all-acquired ?take-j }\subseteq\mathrm{ all-acquired sb
    by auto
ultimately have False
    using x-in-owns-acq-i own-dist
    by auto
        }
        ultimately show ?thesis
using x-in-take-i x-in-j
by (auto simp add: misc-outstanding-refs-convs)
        qed
    }
    then show ?thesis
    by auto
qed
lemma (in valid-ownership-and-sharing) outstanding-non-volatile-write-not-volatile-read-disj:
assumes i-bound: i \(<\) length ts
assumes j -bound: \(\mathrm{j}<\) length ts
assumes neq- \(-\mathrm{i} \mathrm{j}: \mathrm{i} \neq \mathrm{j}\)
assumes ith: ts! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
assumes jth: ts! \(=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{isj}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
shows outstanding-refs (is-non-volatile-Write \({ }_{\mathbf{s b}}\) ) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )
\(\left.\mathrm{sb}_{\mathrm{i}}\right) \cap\)
outstanding-refs (Not \(\circ\) is-volatile-Read \(_{\mathbf{s b}}\) ) (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) )
\(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\)
(is ?non-vol-writes-i \(\cap\) ?not-volatile-j \(=\{ \}\) )
proof -
```

have outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{\mathbf{s b}}$ ) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\left.\mathrm{sb}_{\mathrm{j}}\right) \subseteq$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \operatorname{sb}_{j} \cup$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \operatorname{sb}_{j} \cup$ outstanding-refs is-non-volatile-Read sb $^{\text {( }}$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb ${ }_{j}$ ) by (auto simp add: misc-outstanding-refs-convs dest: outstanding-refs-dropWhileD) with outstanding-non-volatile-write-disj [OF i-bound j-bound neq-i-j ith jth] show ?thesis by blast

## qed

lemma (in valid-ownership-and-sharing) outstanding-refs-is-Write ${ }_{\text {sb }}$-takeWhile-disj:
$\forall \mathrm{i}<$ length ts. $(\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$

$$
\begin{gathered}
\text { (let }\left(-,-,-, \mathrm{sb}_{\mathrm{i}},-,-,-\right)=\mathrm{ts}!\mathrm{i} ; \\
\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}
\end{gathered}
$$

in outstanding-refs is-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{i}} \cap$
outstanding-refs is-Write $_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $=$
\{\}))
proof -
\{
fix i j $p_{i}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume i-bound: i $<$ length ts
assume j-bound: j < length ts
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume ith: ts! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
assume jth: ts! $j=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
from outstanding-non-volatile-write-disj [OF j-bound i-bound neq-i-j[symmetric] jth ith]
have outstanding-refs is-Write ${ }_{s b} \mathrm{sb}_{\mathrm{i}} \cap$
outstanding-refs is-Write $_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $=\{ \}$ apply (clarsimp simp add: outstanding-refs-is-non-volatile-Write ${ }_{\mathbf{s b}}$-takeWhile-conv) apply (auto simp add: misc-outstanding-refs-convs ) done
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
fun read-tmps:: 'p store-buffer $\Rightarrow$ tmp set
where
read-tmps []$=\{ \}$
$\mid$ read-tmps (r\#rs) $=$
(case r of
Read $_{\mathrm{sb}}$ volatile a $\mathrm{t} \mathrm{v} \Rightarrow$ insert t (read-tmps rs)
|- $\Rightarrow$ read-tmps rs)
lemma in-read-tmps-conv:
$(\mathrm{t} \in \operatorname{read}-\mathrm{tmps} \mathrm{xs})=\left(\exists\right.$ volatile a $\mathrm{v} . \operatorname{Read}_{\mathrm{sb}}$ volatile att $\mathrm{v} \in$ set xs$)$
by (induct xs) (auto split: memref.splits)
lemma read-tmps-mono: $\bigwedge$ ys. set xs $\subseteq$ set ys $\Longrightarrow$ read-tmps xs $\subseteq$ read-tmps ys by (fastforce simp add: in-read-tmps-conv)

```
fun distinct-read-tmps:: 'p store-buffer \(\Rightarrow\) bool
where
    distinct-read-tmps [] = True
\(\mid\) distinct-read-tmps (r\#rs) \(=\)
    (case r of
    Read \(_{\mathrm{sb}}\) volatile a \(\mathrm{t} \mathrm{v} \Rightarrow \mathrm{t} \notin(\) read-tmps rs\() \wedge\) distinct-read-tmps rs
    | \(-\Rightarrow\) distinct-read-tmps rs)
```

lemma distinct-read-tmps-conv:
distinct-read-tmps xs $=(\forall \mathrm{i}<$ length xs. $\forall \mathrm{j}<$ length xs. $\mathrm{i} \neq \mathrm{j} \longrightarrow$
(case xs!i of
$\operatorname{Read}_{\mathrm{sb}}--\mathrm{t}_{\mathrm{i}}-\Rightarrow$ case xs!j of $\operatorname{Read}_{\mathrm{sb}}--\mathrm{t}_{\mathrm{j}}-\Rightarrow \mathrm{t}_{\mathrm{i}} \neq \mathrm{t}_{\mathrm{j}} \mid-\Rightarrow$ True
|- $\Rightarrow$ True))

- Nice lemma, ugly proof.
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a sop v)
with Cons.hyps show ?thesis
apply -
apply (rule iffI [rule-format])
apply clarsimp
subgoal for $\mathrm{i} j$
apply (case-tac i)
apply fastforce
apply (case-tac j)
apply (fastforce split: memref.splits)
apply (clarsimp cong: memref.case-cong)
done
apply clarsimp
subgoal for $\mathrm{i} j$
apply (erule-tac $x=$ Suc i in allE)
apply clarsimp
apply (erule-tac $\mathrm{x}=$ Suc j in allE)
apply (clarsimp cong: memref.case-cong)
done

```
    done
next
    case ( Readsb volatile a t v)
    with Cons.hyps show ?thesis
        apply -
        apply (rule iffI [rule-format])
        apply clarsimp
            subgoal for i j
            apply (case-tac i)
            apply clarsimp
            apply (case-tac j)
            apply clarsimp
            apply (fastforce split: memref.splits simp add: in-read-tmps-conv dest: nth-mem)
                    apply (clarsimp)
                apply (case-tac j)
            apply (fastforce split: memref.splits simp add: in-read-tmps-conv dest: nth-mem)
                        apply (clarsimp cong: memref.case-cong)
                done
    apply clarsimp
    apply (rule conjI)
    apply (clarsimp simp add: in-read-tmps-conv)
    apply (erule-tac x=0 in allE)
    apply (clarsimp simp add: in-set-conv-nth)
        subgoal for volatile' }\mp@subsup{\textrm{a}}{}{\prime}\mp@subsup{v}{}{\prime}\mathrm{ i
        apply (erule-tac x=Suc i in allE)
        apply clarsimp
        done
    apply clarsimp
    subgoal for i j
    apply (erule-tac x=Suc i in allE)
    apply clarsimp
    apply (erule-tac x=Suc j in allE)
    apply (clarsimp cong: memref.case-cong)
    done
    done
next
    case Progsb
    with Cons.hyps show ?thesis
        apply -
        apply (rule iffI [rule-format])
        apply clarsimp
            subgoal for i j
            apply (case-tac i)
            apply fastforce
            apply (case-tac j)
            apply (fastforce split: memref.splits)
            apply (clarsimp cong: memref.case-cong)
            done
        apply clarsimp
        subgoal for i j
```

```
        apply (erule-tac x=Suc i in allE)
        apply clarsimp
        apply (erule-tac x=Suc j in allE)
        apply (clarsimp cong: memref.case-cong)
        done
        done
    next
    case Ghostsb
    with Cons.hyps show ?thesis
        apply -
        apply (rule iffI [rule-format])
        apply clarsimp
            subgoal for i j
            apply (case-tac i)
            apply fastforce
            apply (case-tac j)
            apply (fastforce split: memref.splits)
            apply (clarsimp cong: memref.case-cong)
            done
        apply clarsimp
        subgoal for i j
        apply (erule-tac x=Suc i in allE)
        apply clarsimp
        apply (erule-tac x=Suc j in allE)
        apply (clarsimp cong: memref.case-cong)
        done
        done
    qed
qed
fun load-tmps:: instrs }=>\mathrm{ tmp set
where
    load-tmps [] = {}
| load-tmps (i#is) =
    (case i of
        Read volatile a t }=>\mathrm{ insert t (load-tmps is)
        | RMW - t - - - - - m insert t (load-tmps is)
        |- = load-tmps is)
lemma in-load-tmps-conv:
    (t \in load-tmps xs ) = ((\existsvolatile a. Read volatile a t \in set xs ) V
                            (\existsa sop cond ret A L R W. RMW a t sop cond ret A L R W \in set xs))
by (induct xs) (auto split: instr.splits)
lemma load-tmps-mono: \bigwedgeys. set xs \subseteq set ys }\Longrightarrow\mathrm{ load-tmps xs }\subseteq\mathrm{ load-tmps ys
    by (fastforce simp add: in-load-tmps-conv)
fun distinct-load-tmps:: instrs }=>\mathrm{ bool
where
    distinct-load-tmps [] = True
```

```
| distinct-load-tmps (r#rs) =
    (case r of
            Read volatile a t }=>\textrm{t}\not\in(\mathrm{ (load-tmps rs) ^ distinct-load-tmps rs
                |MW a t sop cond ret A L R W }=>\textrm{t}\not\in(\mathrm{ load-tmps rs) ^ distinct-load-tmps rs
                |- = distinct-load-tmps rs)
locale load-tmps-distinct =
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes load-tmps-distinct:
\ip is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
    |i < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
    C
    distinct-load-tmps is
locale read-tmps-distinct =
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes read-tmps-distinct:
\ip is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
|i < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
C
distinct-read-tmps sb
locale load-tmps-read-tmps-distinct =
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes load-tmps-read-tmps-distinct:
\ip is \mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
    | < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
    \Longrightarrow
    load-tmps is }\cap\mathrm{ read-tmps sb = {}
locale tmps-distinct =
    load-tmps-distinct +
    read-tmps-distinct +
    load-tmps-read-tmps-distinct
lemma rev-read-tmps: read-tmps (rev xs) = read-tmps xs
    by (auto simp add: in-read-tmps-conv)
lemma rev-load-tmps: load-tmps (rev xs) = load-tmps xs
    by (auto simp add: in-load-tmps-conv)
lemma distinct-read-tmps-append: \ys. distinct-read-tmps (xs @ ys)=
    (distinct-read-tmps xs ^ distinct-read-tmps ys }
    read-tmps xs \cap read-tmps ys = {})
by (induct xs) (auto split: memref.splits simp add: in-read-tmps-conv)
lemma distinct-load-tmps-append: \ys. distinct-load-tmps (xs @ ys) =
    (distinct-load-tmps xs ^ distinct-load-tmps ys }
    load-tmps xs \cap load-tmps ys ={})
```

```
apply (induct xs)
apply (auto split: instr.splits simp add: in-load-tmps-conv)
done
lemma read-tmps-append: read-tmps \((x s @ y s)=(\) read-tmps xs \(\cup\) read-tmps ys)
    by (fastforce simp add: in-read-tmps-conv)
lemma load-tmps-append: load-tmps \((x s @ y s)=(\) load-tmps xs \(\cup\) load-tmps ys)
    by (fastforce simp add: in-load-tmps-conv)
fun write-sops:: 'p store-buffer \(\Rightarrow\) sop set
where
    write-sops []\(=\{ \}\)
\(\mid\) write-sops (r\#rs) \(=\)
        (case r of
    Write \(_{\text {sb }}\) volatile a sop v--- \(\Rightarrow\) insert sop (write-sops rs)
    | - \(\Rightarrow\) write-sops rs)
lemma in-write-sops-conv:
    \((\) sop \(\in\) write-sops xs \()=\left(\exists\right.\) volatile a v A L R W. Write \({ }_{\text {sb }}\) volatile a sop v A L R W set
xs )
    apply (induct xs)
    apply simp
    apply (auto split: memref.splits)
    apply force
    apply force
    done
lemma write-sops-mono: \(\bigwedge\) ys. set xs \(\subseteq\) set ys \(\Longrightarrow\) write-sops xs \(\subseteq\) write-sops ys
    by (fastforce simp add: in-write-sops-conv)
lemma write-sops-append: write-sops (xs@ys) = write-sops xs \(\cup\) write-sops ys
    by (force simp add: in-write-sops-conv)
fun store-sops:: instrs \(\Rightarrow\) sop set
where
    store-sops []\(=\{ \}\)
| store-sops (i\#is) =
    (case i of
    Write volatile a sop \(---\Rightarrow\) insert sop (store-sops is)
        | RMW a t sop cond ret A L R W \(\Rightarrow\) insert sop (store-sops is)
        |- \(\Rightarrow\) store-sops is)
lemma in-store-sops-conv:
\((\operatorname{sop} \in\) store-sops xs \()=((\exists\) volatile a A L R W. Write volatile a sop A L R W \(\in\) set xs \()\)
V
\((\exists \mathrm{at}\) cond ret A L R W. RMW a t sop cond ret A L R W \(\in\) set xs\()\) )
by (induct xs) (auto split: instr.splits)
```

lemma store-sops-mono: $\bigwedge$ ys. set xs $\subseteq$ set ys $\Longrightarrow$ store-sops xs $\subseteq$ store-sops ys by (fastforce simp add: in-store-sops-conv)
lemma store-sops-append: store-sops $(x s @ y s)=$ store-sops $x s \cup$ store-sops ys by (force simp add: in-store-sops-conv)
locale valid-write-sops $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes valid-write-sops:
$\wedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow$
$\forall$ sop $\in$ write-sops sb. valid-sop sop
locale valid-store-sops $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes valid-store-sops:

```
\(\bigwedge_{\mathrm{i}}\) is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb.
        \(\llbracket \mathrm{i}<\) length ts; ts!i \(=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket\)
\(\Longrightarrow\)
\(\forall\) sop \(\in\) store-sops is. valid-sop sop
```

locale valid-sops $=$ valid-write-sops + valid-store-sops
The value stored in a non-volatile Read $_{\mathrm{sb}}$ in the store-buffer has to match the last value written to the same address in the store buffer or the memory content if there is no corresponding write in the store buffer. No volatile read may follow a volatile write. Volatile reads in the store buffer may refer to a stale value: e.g. imagine one writer and multiple readersfun reads-consistent:: bool $\Rightarrow$ owns $\Rightarrow$ memory $\Rightarrow$ 'p store-buffer $\Rightarrow$ bool where

```
reads-consistent pending-write }\mathcal{O}\textrm{m}[]=\mathrm{ True
| reads-consistent pending-write }\mathcal{O}\textrm{m}(\textrm{r}#\textrm{rs})
    (case r of
            Read}\mp@subsup{\textrm{sb}}{\mathrm{ volatile a t v }=>(\neg\mathrm{ volatile }\longrightarrow(\mathrm{ pending-write }\vee\textrm{a}\in\mathcal{O})\longrightarrow\textrm{v}=\textrm{m}\mathrm{ a) }\wedge}{\wedge
                                    reads-consistent pending-write }\mathcal{O m rs
            | Write sb volatile a sop v A L R W =
                (if volatile then
                    outstanding-refs is-volatile-Read sb rs = {}^
                        reads-consistent True (\mathcal{O}\cup\textrm{A}-\textrm{R})(\textrm{m}(\textrm{a}:=\textrm{v}))}\mathrm{ ) rs
                else reads-consistent pending-write \mathcal{O}(m(a := v)) rs)
                | Ghost sb A L R W }=>\mathrm{ reads-consistent pending-write (O) ( 
                |- = reads-consistent pending-write }\mathcal{O}\textrm{m}\mathrm{ rs
)
```

fun volatile-reads-consistent:: memory $\Rightarrow$ 'p store-buffer $\Rightarrow$ bool

## where

volatile-reads-consistent m [] = True
| volatile-reads-consistent $\mathrm{m}(\mathrm{r} \# \mathrm{rs})=$ (case r of
$\operatorname{Read}_{\mathrm{sb}}$ volatile at $\mathrm{v} \Rightarrow($ volatile $\longrightarrow \mathrm{v}=\mathrm{m}$ a) $\wedge$ volatile-reads-consistent m rs
| Writesb volatile a sop v A L R W $\Rightarrow$ volatile-reads-consistent ( $\mathrm{m}(\mathrm{a}:=\mathrm{v})$ ) rs

$$
{ }_{( }-\Rightarrow \text { volatile-reads-consistent m rs }
$$

```
fun flush:: 'p store-buffer \(\Rightarrow\) memory \(\Rightarrow\) memory
where
    flush [] \(\mathrm{m}=\mathrm{m}\)
| flush ( \(\mathrm{r} \# \mathrm{rs}\) ) \(\mathrm{m}=\)
    (case r of
    Write \(_{\text {sb }}\) volatile \(\mathrm{a}-\mathrm{v}-\mathrm{-}\) - \(\Rightarrow\) flush \(\mathrm{rs}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))\)
    |- \(\Rightarrow\) flush rs m)
```

lemma reads-consistent-pending-write-antimono:
$\Lambda \mathcal{O} \mathrm{m}$. reads-consistent True $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$ reads-consistent False $\mathcal{O} \mathrm{m} \mathrm{sb}$

## apply (induct sb)

apply simp
subgoal for a sb $\mathcal{O} \mathrm{m}$
by (case-tac a) auto
done
lemma reads-consistent-owns-antimono:
$\wedge \mathcal{O} \mathcal{O}^{\prime}$ pending-write m.
$\mathcal{O} \subseteq \mathcal{O}^{\prime} \Longrightarrow$ reads-consistent pending-write $\mathcal{O}^{\prime} \mathrm{m}$ sb $\Longrightarrow$ reads-consistent pending-write
$\mathcal{O} \mathrm{m} \mathrm{sb}$
apply (induct sb)
apply simp
subgoal for $\operatorname{asb} \mathcal{O} \mathcal{O}^{\prime}$ pending-write m
apply (case-tac a)
apply (clarsimp split: if-split-asm)
subgoal for volatile a D f v A L R W
apply (drule-tac $\mathrm{C}=\mathrm{A}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
apply fastforce
apply fastforce
apply clarsimp
subgoal for A L R W
apply (drule-tac $\mathrm{C}=\mathrm{A}$ in union-mono-aux)
apply (drule-tac $\mathrm{C}=\mathrm{R}$ in set-minus-mono-aux)
apply blast
done
done
done
lemma acquired-reads-mono': $\mathrm{x} \in$ acquired-reads b xs $\mathrm{A} \Longrightarrow$ acquired-reads b xs $\mathrm{B}=\{ \}$
$\Longrightarrow \mathrm{A} \subseteq \mathrm{B} \Longrightarrow$ False
apply (drule acquired-reads-mono-in [where $\mathrm{B}=\mathrm{B}]$ )
apply auto
done
lemma reads-consistent-append:
$\bigwedge \mathrm{m}$ pending-write $\mathcal{O}$. reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (xs@ys) = (reads-consistent pending-write $\mathcal{O} \mathrm{m}$ xs $\wedge$
reads-consistent (pending-write $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\neq\{ \}$ )
(acquired True xs $\mathcal{O}$ ) (flush xs m) ys $\wedge$
(outstanding-refs is-volatile-Write sb $^{\text {xs }} \neq\{ \}$
$\longrightarrow$ outstanding-refs is-volatile-Read $\operatorname{Re}_{\text {sb }}$ ys $\left.=\{ \}\right)$ )
apply (induct xs)
apply clarsimp
subgoal for a xs m pending-write $\mathcal{O}$
apply (case-tac a)
apply (auto simp add: outstanding-refs-append acquired-reads-append
dest: acquired-reads-mono-in acquired-pending-write-mono-in acquired-reads-mono ${ }^{\prime} \mathrm{ac}^{-}$ quired-mono-in)

## done

done
lemma reads-consistent-mem-eq-on-non-volatile-reads:
assumes mem-eq: $\forall \mathrm{a} \in \mathrm{A} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}$
assumes subset: outstanding-refs (is-non-volatile-Read ${ }_{\text {sb }}$ ) sb $\subseteq \mathrm{A}$

- We could be even more restrictive here, only the non volatile reads that are not buffered in sb have to be the same.
assumes consis-m: reads-consistent pending-write $\mathcal{O} \mathrm{m}$ sb
shows reads-consistent pending-write $\mathcal{O} \mathrm{m}^{\prime}$ sb
using mem-eq subset consis-m
proof (induct sb arbitrary: $\mathrm{m}^{\prime} \mathrm{m}$ pending-write $\mathcal{O}$ )
case Nil thus ?case by simp
next
case (Cons r sb)
note mem-eq $=\left\langle\forall \mathrm{a} \in \mathrm{A} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m}\right.$ a $\rangle$
note subset $=\left\langle\right.$ outstanding-refs (is-non-volatile-Read $\left.\left.{ }_{\mathrm{sb}}\right)(\mathrm{r} \# \mathrm{sb}) \subseteq \mathrm{A}\right\rangle$
note consis-m $=$ 〔reads-consistent pending-write $\mathcal{O} \mathrm{m}(\mathrm{r} \# \mathrm{sb})$ 〉
from subset have subset': outstanding-refs is-non-volatile-Read sb sb $\subseteq \mathrm{A}$
by (auto simp add: Write ${ }_{\text {sb }}$ )
show ?case
proof (cases r)
case $\left(W_{\text {rite }}^{\text {sb }}\right.$ volatile a sop v $\left.\mathrm{A}^{\prime} \mathrm{L} R \mathrm{~W}\right)$
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in \mathrm{A} .\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{a}^{\prime}=(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{a}^{\prime}$ by (auto)
show ?thesis
proof (cases volatile)
case True
from consis-m obtain
consis': reads-consistent $\operatorname{True}\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ sb and no-volatile-Read ${ }_{\mathbf{s b}}$ : outstanding-refs is-volatile-Read ${ }_{\mathrm{sb}} \mathrm{sb}=\{ \}$

```
by (simp add: Write sb True)
from Cons.hyps [OF mem-eq' subset' consis']
have reads-consistent True (\mathcal{O}\cup\mp@subsup{\textrm{A}}{}{\prime}-\textrm{R})(\mp@subsup{\textrm{m}}{}{\prime}(\textrm{a}:=\textrm{v}))}\mathrm{ ) sb.
with no-volatile-Readsb
show ?thesis
by (simp add: Write sb True)
    next
        case False
        from consis-m obtain consis': reads-consistent pending-write \mathcal{O (m(a := v)) sb}\mp@code{s}\mathrm{ )}
by (simp add: Write sb False)
    from Cons.hyps [OF mem-eq' subset' consis']
    have reads-consistent pending-write \mathcal{O}(\mp@subsup{m}{}{\prime}(\textrm{a}:=\textrm{v})) sb.
    then
    show ?thesis
by (simp add: Write sb False)
    qed
next
    case (Read sb volatile a t v)
    from mem-eq
    have mem-eq':
        \forall\mp@subsup{a}{}{\prime}\inA. m' a' = ma'
        by (auto)
    show ?thesis
    proof (cases volatile)
        case True
        from consis-m obtain
consis': reads-consistent pending-write }\mathcal{O}\textrm{m sb
by (simp add: Read sb True)
        from Cons.hyps [OF mem-eq' subset' consis']
        show ?thesis
by (simp add: Read sb True)
    next
        case False
        from consis-m obtain
consis': reads-consistent pending-write }\mathcal{O}\textrm{m}\mathrm{ sb and v: (pending-write }\vee\textrm{a}\in\mathcal{O})\longrightarrow\textrm{v}=\textrm{m
a
by (simp add: Read sb False)
        from mem-eq subset Readsb}\mathrm{ have m'a = m a
by (auto simp add: False)
        with Cons.hyps [OF mem-eq' subset' consis'] v
        show ?thesis
by (simp add: Read sb False)
        qed
    next
        case Prog
    next
        case Ghost sb with Cons show ?thesis by auto
    qed
qed
```

lemma volatile-reads-consistent-mem-eq-on-volatile-reads:
assumes mem-eq: $\forall \mathrm{a} \in \mathrm{A} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}$
assumes subset: outstanding-refs (is-volatile-Read ${ }_{\mathrm{sb}}$ ) $\mathrm{sb} \subseteq \mathrm{A}$

- We could be even more restrictive here, only the non volatile reads that are not buffered in sb have to be the same.
assumes consis-m: volatile-reads-consistent m sb
shows volatile-reads-consistent $\mathrm{m}^{\prime}$ sb
using mem-eq subset consis-m
proof (induct sb arbitrary: $\mathrm{m}^{\prime} \mathrm{m}$ )
case Nil thus ?case by simp
next
case (Cons r sb)
note mem-eq $=\left\langle\forall \mathrm{a} \in \mathrm{A} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m}\right.$ a $\rangle$
note subset $=$ <outstanding-refs (is-volatile-Read $\left.{ }_{\mathrm{sb}}\right)(\mathrm{r} \# \mathrm{sb}) \subseteq \mathrm{A}$,
note consis- $\mathrm{m}=$ \{volatile-reads-consistent $\mathrm{m}(\mathrm{r} \# \mathrm{sb})$ )
from subset have subset': outstanding-refs is-volatile-Read ${ }_{\text {sb }} \mathrm{sb} \subseteq \mathrm{A}$
by (auto simp add: Write ${ }_{\text {sb }}$ )
show ?case
proof (cases r)
case ( Write $_{\text {sb }}$ volatile a sop v A'L R W)
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in \mathrm{A} .\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{a}^{\prime}=(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{a}^{\prime}$ by (auto)
show ?thesis
proof (cases volatile)
case True
from consis-m obtain
consis': volatile-reads-consistent (m(a $:=\mathrm{v})$ ) sb
by ( simp add: Write ${ }_{\mathrm{sb}}$ True)
from Cons.hyps [OF mem-eq' subset' consis']
have volatile-reads-consistent $\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right)$ sb.
then show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ True)


## next

case False
from consis-m obtain consis': volatile-reads-consistent ( $\mathrm{m}(\mathrm{a}:=\mathrm{v}$ ) ) sb
by (simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF mem-eq' subset' consis'] have volatile-reads-consistent $\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{sb}$.
then
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ False)
qed

```
next
    case (Read sb volatile a t v)
    from mem-eq
    have mem-eq':
        * '
        by (auto)
    show ?thesis
    proof (cases volatile)
        case False
        from consis-m obtain
consis': volatile-reads-consistent m sb
by (simp add: Read
    from Cons.hyps [OF mem-eq' subset' consis']
    show ?thesis
by (simp add: Read}\mp@subsup{\mp@code{sb}}{\mathrm{ b False)}}{
    next
        case True
        from consis-m obtain
consis': volatile-reads-consistent m sb and v: v=m a
by (simp add: Readsb True)
    from mem-eq subset Read}\mp@subsup{\mp@code{sb}}{}{v}\mathrm{ have v = m'a
by (auto simp add: True)
        with Cons.hyps [OF mem-eq' subset' consis']
        show ?thesis
by (simp add: Readsb True)
    qed
    next
        case Prog
    next
        case Ghost sb with Cons show ?thesis by auto
    qed
qed
locale valid-reads =
fixes m::memory and ts::('p, 'p store-buffer,bool,owns,rels) thread-config list
assumes valid-reads: }\\textrm{i p}\mathrm{ is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\mathrm{ sb.
```



```
    reads-consistent False }\mathcal{O m sb
lemma valid-reads-Cons: valid-reads m (t#ts) =
    (let (-,-,-,sb,-,\mathcal{O},-)= t in reads-consistent False \mathcal{O m sb ^ valid-reads m ts)}
apply (auto simp add: valid-reads-def)
subgoal for p}\mp@subsup{p}{}{\prime}\mathrm{ is' }\mp@subsup{\vartheta}{}{\prime}\mp@subsup{\operatorname{sb}}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mathrm{ i p is }\vartheta\mathrm{ sb }\mathcal{D}\mathcal{O}\mathcal{R
apply (case-tac i)
apply auto
done
done
    Read}\mp@subsup{}{\textrm{sb}}{}s\mathrm{ and writes have in the store-buffer have to conform to the valuation of tem-
poraries.context program
begin
```

```
fun history-consistent:: tmps \(\Rightarrow\) 'p \(\Rightarrow\) 'p store-buffer \(\Rightarrow\) bool
where
    history-consistent \(\vartheta \mathrm{p}[]=\) True
| history-consistent \(\vartheta \mathrm{p}(\mathrm{r} \# \mathrm{rs})=\)
        (case r of
            \(\operatorname{Read}_{\text {sb }}\) vol at \(\mathrm{v} \Rightarrow\)
                    (case \(\vartheta \mathrm{t}\) of Some \(\mathrm{v}^{\prime} \Rightarrow \mathrm{v}=\mathrm{v}^{\prime} \wedge\) history-consistent \(\vartheta \mathrm{p} \mathrm{rs} \mid-\Rightarrow\) False)
        | Write \({ }_{\text {sb }}\) vol a (D,f) v \(-\cdots \Rightarrow\)
                        \(\mathrm{D} \subseteq \operatorname{dom} \vartheta \wedge \mathrm{f} \vartheta=\mathrm{v} \wedge \mathrm{D} \cap\) read-tmps rs \(=\{ \} \wedge\) history-consistent \(\vartheta \mathrm{p}\) rs
        \(\mid\) Prog \(_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2}\) is \(\Rightarrow \mathrm{p}_{1}=\mathrm{p} \wedge\)
                                    \(\left.\vartheta\right|^{\cdot}(\) dom \(\vartheta-\) read-tmps rs \() \vdash \mathrm{p}_{1} \rightarrow_{\mathrm{p}}\left(\mathrm{p}_{2}\right.\), is \() \wedge\)
                                    history-consistent \(\vartheta \mathrm{p}_{2}\) rs
        | - \(\Rightarrow\) history-consistent \(\vartheta\) p rs)
end
fun hd-prog:: 'p \(\Rightarrow\) 'p store-buffer \(\Rightarrow\) 'p
where
    hd-prog p[]\(=\mathrm{p}\)
\(\mid\) hd-prog \(\mathrm{p}(\mathrm{i} \# \mathrm{is})=\) (case i of
        \(\operatorname{Prog}_{\text {sb }} \mathrm{p}^{\prime}--\Rightarrow \mathrm{p}^{\prime}\)
    \(\mid-\Rightarrow\) hd-prog \(p\) is)
fun last-prog:: \(\mathrm{p} \Rightarrow\) 'p store-buffer \(\Rightarrow\) 'p
where
    last-prog p[]\(=\mathrm{p}\)
\(\mid\) last-prog \(\mathrm{p}(\mathrm{i} \#\) is \()=(\) case i of
        \(\operatorname{Prog}_{\mathrm{sb}}-\mathrm{p}^{\prime}-\Rightarrow \operatorname{last}-\mathrm{prog} \mathrm{p}^{\prime}\) is
    |- \(\Rightarrow\) last-prog p is)
locale valid-history \(=\) program +
constrains
    program-step :: tmps \(\Rightarrow{ }^{\prime} \mathrm{p} \Rightarrow{ }^{\prime} \mathrm{p} \times\) instrs \(\Rightarrow\) bool
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes valid-history: \(\bigwedge \mathrm{i} \mathrm{p}\) is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb.
    \(\llbracket \mathrm{i}<\) length ts; ts \(!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\)
    program.history-consistent program-step \(\vartheta\) (hd-prog p sb) sb
fun data-dependency-consistent-instrs:: addr set \(\Rightarrow\) instrs \(\Rightarrow\) bool
where
    data-dependency-consistent-instrs T [] = True
| data-dependency-consistent-instrs \(\mathrm{T}(\mathrm{i} \# \mathrm{is})=\)
        (case i of
                            Write volatile a ( \(\mathrm{D}, \mathrm{f}\) ) \(\cdots \Rightarrow \mathrm{D} \subseteq \mathrm{T} \wedge \mathrm{D} \cap\) load-tmps is \(=\{ \} \wedge\)
data-dependency-consistent-instrs T is
            | RMW a t ( \(\mathrm{D}, \mathrm{f}\) ) cond ret \(\cdots \Rightarrow \mathrm{D} \subseteq \mathrm{T} \wedge \mathrm{D} \cap\) load-tmps is \(=\{ \} \wedge\)
data-dependency-consistent-instrs (insert t T) is
    | Read - \(\mathrm{t} \Rightarrow\) data-dependency-consistent-instrs (insert t T) is
    |- \(\Rightarrow\) data-dependency-consistent-instrs T is)
lemma data-dependency-consistent-mono:
```



```
data-dependency-consistent-instrs }\mp@subsup{\textrm{T}}{}{\prime}\mathrm{ is
apply (induct is)
apply clarsimp
subgoal for a is T T'
apply (case-tac a)
apply clarsimp
    subgoal for volatile a't
    apply (drule-tac a=t in insert-mono)
    apply clarsimp
    done
apply fastforce
apply clarsimp
    subgoal for a't D f cond ret A L R W
    apply (frule-tac a=t in insert-mono)
    apply fastforce
    done
apply fastforce
apply fastforce
done
done
lemma data-dependency-consistent-instrs-append:
\ys T. data-dependency-consistent-instrs \(\mathrm{T}(\mathrm{xs} @ y s)=\) (data-dependency-consistent-instrs T xs \(\wedge\)
data-dependency-consistent-instrs ( \(\mathrm{T} \cup\) load-tmps xs) ys \(\wedge\)
load-tmps ys \(\cap \bigcup\) (fst ' store-sops xs) \(=\{ \}\) )
apply (induct xs)
apply (auto split: instr.splits simp add: load-tmps-append intro:
data-dependency-consistent-mono)
done
locale valid-data-dependency \(=\)
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list assumes data-dependency-consistent-instrs:
\(\bigwedge \mathrm{ip}\) is \(\mathcal{O} \mathcal{D} \vartheta \mathrm{sb}\).
\(\llbracket \mathrm{i}<\) length ts; ts \(!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\) data-dependency-consistent-instrs (dom \(\vartheta\) ) is
assumes load-tmps-write-tmps-distinct:
\(\bigwedge \mathrm{i}\) p is \(\mathcal{O} \mathcal{D} \vartheta \mathrm{sb}\).
\(\llbracket \mathrm{i}<\) length ts; ts \(!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\) load-tmps is \(\cap \bigcup\) (fst ' write-sops sb) \(=\{ \}\)
locale load-tmps-fresh =
fixes ts::('p, 'p store-buffer,bool,owns,rels) thread-config list assumes load-tmps-fresh:
\(\bigwedge \mathrm{i} \mathrm{p}\) is \(\mathcal{O} \mathcal{D} \vartheta \mathrm{sb}\).
\(\llbracket \mathrm{i}<\) length ts; ts \(!=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\) load-tmps is \(\cap \operatorname{dom} \vartheta=\{ \}\)
```

```
fun acquired-by-instrs :: instrs \(\Rightarrow\) addr set \(\Rightarrow\) addr set
where
    acquired-by-instrs [] A \(=\mathrm{A}\)
| acquired-by-instrs (i\#is) \(\mathrm{A}=\)
    (case i of
        Read - - \(\Rightarrow\) acquired-by-instrs is A
            | Write volatile - - \(\mathrm{A}^{\prime} \mathrm{L} R \mathrm{~W} \Rightarrow\) acquired-by-instrs is (if volatile then \(\left(\mathrm{A} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)\)
else A)
            | RMW at sop cond ret A'L R W \(\Rightarrow\) acquired-by-instrs is \(\}\)
            | Fence \(\Rightarrow\) acquired-by-instrs is \(\}\)
            | Ghost \(\mathrm{A}^{\prime} \mathrm{L} \mathrm{RW} \Rightarrow\) acquired-by-instrs is \(\left(\mathrm{A} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)\) )
fun acquired-loads :: bool \(\Rightarrow\) instrs \(\Rightarrow\) addr set \(\Rightarrow\) addr set
where
    acquired-loads pending-write [] \(\mathrm{A}=\{ \}\)
| acquired-loads pending-write (i\#is) \(\mathrm{A}=\)
    (case i of
            Read volatile a - \(\Rightarrow\) (if pending-write \(\wedge \neg\) volatile \(\wedge \mathrm{a} \in \mathrm{A}\)
                                    then insert a (acquired-loads pending-write is A)
                                    else acquired-loads pending-write is A)
                            | Write volatile - - A' L R W \(\Rightarrow\) (if volatile then acquired-loads True is (if pending-write
then \(\left(A \cup A^{\prime}-R\right)\) else \(\})\)
                                    else acquired-loads pending-write is A)
    | RMW at sop cond ret \(\mathrm{A}^{\prime} \mathrm{L}\) R \(\mathrm{W} \Rightarrow\) acquired-loads pending-write is \(\}\)
    | Fence \(\Rightarrow\) acquired-loads pending-write is \(\}\)
    | Ghost \(\mathrm{A}^{\prime} \mathrm{L} \mathrm{R} \mathrm{W} \Rightarrow\) acquired-loads pending-write is \(\left(\mathrm{A} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)\) )
lemma acquired-by-instrs-mono:
    \(\bigwedge \mathrm{AB} . \mathrm{A} \subseteq \mathrm{B} \Longrightarrow\) acquired-by-instrs is \(\mathrm{A} \subseteq\) acquired-by-instrs is B
apply (induct is)
apply simp
subgoal for a is A B
apply (case-tac a)
apply clarsimp
apply clarsimp
    subgoal for volatile a' D f A' L R W x
    apply (drule-tac \(\mathrm{C}=\mathrm{A}^{\prime}\) in union-mono-aux)
    apply (drule-tac \(\mathrm{C}=\mathrm{R}\) in set-minus-mono-aux)
    apply blast
    done
apply clarsimp
apply clarsimp
apply clarsimp
subgoal for \(\mathrm{A}^{\prime} \mathrm{L} R \mathrm{R} \mathrm{x}\)
apply (drule-tac \(\mathrm{C}=\mathrm{A}^{\prime}\) in union-mono-aux)
apply (drule-tac \(\mathrm{C}=\mathrm{R}\) in set-minus-mono-aux)
apply blast
done
done
```

done
lemma acquired-by-instrs-mono-in:
assumes x -in: $\mathrm{x} \in$ acquired-by-instrs is A
assumes sub: $\mathrm{A} \subseteq \mathrm{B}$
shows $\mathrm{x} \in$ acquired-by-instrs is B
using acquired-by-instrs-mono [OF sub, of is] x-in
by blast
locale enough-flushs $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes clean-no-outstanding-volatile-Write ${ }_{\text {sb }}$ :
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts $; \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \neg \mathcal{D} \rrbracket \Longrightarrow$
(outstanding-refs is-volatile-Write ${ }_{s b} \mathrm{sb}=\{ \}$ )
fun prog-instrs:: 'p store-buffer $\Rightarrow$ instrs
where
prog-instrs [] = []
$\mid$ prog-instrs (i\#is) $=$ (case i of
$\operatorname{Prog}_{\text {sb }}-$ is $^{\prime} \Rightarrow$ is $^{\prime} @$ prog-instrs is
$\mid-\Rightarrow$ prog-instrs is)
fun instrs:: 'p store-buffer $\Rightarrow$ instrs
where
instrs [] = []
$\mid$ instrs $(\mathrm{i} \# \mathrm{is})=$ (case i of
Write $_{\text {sb }}$ volatile a sop v A L R W $\Rightarrow$ Write volatile a sop A L R W\# instrs is
$\mid \operatorname{Read}_{\mathrm{sb}}$ volatile a t v $\Rightarrow$ Read volatile a $\mathrm{t} \#$ instrs is
| Ghost ${ }_{\text {sb }}$ A L R W $\Rightarrow$ Ghost A L R W\# instrs is |- $\Rightarrow$ instrs is)
locale causal-program-history $=$
fixes is ${ }_{\text {sb }}$ and sb
assumes causal-program-history:
$\bigwedge \mathrm{sb}_{1} \mathrm{sb}_{2} . \mathrm{sb}=\mathrm{sb}_{1} @ \mathrm{sb}_{2} \Longrightarrow \exists$ is. instrs $\mathrm{sb}_{2} @$ is $\mathrm{s}_{\mathrm{sb}}=$ is @ prog-instrs $\mathrm{sb}_{2}$
lemma causal-program-history-empty [simp]: causal-program-history is [] by (rule causal-program-history.intro) simp
lemma causal-program-history-suffix:
causal-program-history is $\mathrm{sb}_{\mathrm{sb}}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right) \Longrightarrow$ causal-program-history is $\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}^{\prime}$ by (auto simp add: causal-program-history-def)
locale valid-program-history $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes valid-program-history:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts!i $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$
causal-program-history is sb
assumes valid-last-prog:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$ last-prog $\mathrm{p} \mathrm{sb}=\mathrm{p}$
lemma (in valid-program-history) valid-program-history-nth-update: $\llbracket \mathrm{i}<$ length ts; causal-program-history is sb; last-prog $\mathrm{p} \mathrm{sb}=\mathrm{p} \rrbracket$

$$
\Longrightarrow
$$

    valid-program-history \((\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])\)
    by (rule valid-program-history.intro)
(auto dest: valid-program-history valid-last-prog
simp add: nth-list-update split: if-split-asm)
lemma (in outstanding-non-volatile-refs-owned-or-read-only)
outstanding-non-volatile-refs-owned-instructions-read-value-independent:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$
outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right)$
by (unfold-locales)
(auto dest: outstanding-non-volatile-refs-owned-or-read-only simp add: nth-list-update split: if-split-asm)
lemma (in outstanding-non-volatile-refs-owned-or-read-only) outstanding-non-volatile-refs-owned-or-read-only-nth-update:
$\wedge \mathrm{i}$ is $\mathcal{O} \mathcal{D} \mathcal{R} \vartheta$ sb.
$\llbracket i<$ length ts; non-volatile-owned-or-read-only False $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket \Longrightarrow$ outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S}(\operatorname{ts}[i:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: outstanding-non-volatile-refs-owned-or-read-only simp add: nth-list-update split: if-split-asm)
lemma (in outstanding-volatile-writes-unowned-by-others)
outstanding-volatile-writes-unowned-by-others-instructions-read-value-independent:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$
outstanding-volatile-writes-unowned-by-others $\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right)$
by (unfold-locales)
(auto dest: outstanding-volatile-writes-unowned-by-others
simp add: nth-list-update split: if-split-asm)
lemma (in read-only-reads-unowned)
read-only-unowned-instructions-read-value-independent:
$\bigwedge \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$
read-only-reads-unowned (ts $\left.\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right)$
by (unfold-locales)
(auto dest: read-only-reads-unowned
simp add: nth-list-update split: if-split-asm)
lemma Write ${ }_{\text {sb-in-outstanding-refs: }}$
Write $_{\text {sb }}$ True a sop v A L R W $\in$ set xs $\Longrightarrow a \in$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs by (induct xs) (auto split:memref.splits)

```
lemma (in outstanding-volatile-writes-unowned-by-others)
    outstanding-volatile-writes-unowned-by-others-store-buffer:
\(\wedge \mathrm{i} p\) is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}\).
    \(\llbracket i<\) length ts; ts \(!i=(p\), is \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;\)
    outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}^{\prime} \subseteq\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}\);
    all-acquired sb \({ }^{\prime} \subseteq\) all-acquired \(\mathrm{sb} \rrbracket \Longrightarrow\)
    outstanding-volatile-writes-unowned-by-others (ts[i := ( \(\left.\left.\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}\right)\right]\) )
apply (unfold-locales)
apply (fastforce dest: outstanding-volatile-writes-unowned-by-others
    simp add: nth-list-update split: if-split-asm)
done
```

```
lemma (in ownership-distinct)
    ownership-distinct-instructions-read-value-store-buffer-independent:
\(\bigwedge \mathrm{i} \mathrm{p}\) is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}\).
    \(\llbracket \mathrm{i}<\) length ts; ts \(!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;\)
        all-acquired sb' \(\subseteq\) all-acquired \(\mathrm{sb} \rrbracket \Longrightarrow\)
        ownership-distinct \(\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}\right)\right]\right)\)
    by (unfold-locales)
        (auto dest: ownership-distinct
            simp add: nth-list-update split: if-split-asm)
lemma (in ownership-distinct)
    ownership-distinct-nth-update:
\(\bigwedge \mathrm{i} p\) is \(\mathcal{O} \mathcal{R} \mathcal{D}\) xs sb.
        \(\llbracket i<\) length ts; ts \(!i=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;\)
        \(\forall \mathrm{j}<\) length ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathcal{\vartheta}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)=\mathrm{ts}!\mathrm{j}\)
            in \(\left(\mathcal{O}^{\prime} \cup\right.\) all-acquired sb' \() \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\right) \rrbracket \Longrightarrow\)
        ownership-distinct ( \(\left.\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right)\right]\right)\)
    apply (unfold-locales)
    apply (clarsimp simp add: nth-list-update split: if-split-asm)
    apply (force dest: ownership-distinct simp add: Let-def)
    apply (fastforce dest: ownership-distinct simp add: Let-def)
    apply (fastforce dest: ownership-distinct simp add: Let-def)
    done
```

lemma (in valid-write-sops) valid-write-sops-nth-update:
$\llbracket i<$ length ts; $\forall$ sop $\in$ write-sops sb. valid-sop sop $\rrbracket \Longrightarrow$
valid-write-sops $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold valid-write-sops-def)
(auto dest: valid-write-sops simp add: nth-list-update split: if-split-asm)
lemma (in valid-store-sops) valid-store-sops-nth-update:
$\llbracket i<$ length ts; $\forall$ sop $\in$ store-sops is. valid-sop sop $\rrbracket \Longrightarrow$ valid-store-sops $(\operatorname{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold valid-store-sops-def)
(auto dest: valid-store-sops simp add: nth-list-update split: if-split-asm)
lemma (in valid-sops) valid-sops-nth-update:
$\llbracket i<$ length ts; $\forall$ sop $\in$ write-sops sb. valid-sop sop;
$\forall$ sop $\in$ store-sops is. valid-sop sop $\rrbracket \Longrightarrow$
valid-sops (ts[i := (p,is,xs,sb, $\mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold valid-sops-def valid-write-sops-def valid-store-sops-def)
(auto dest: valid-write-sops valid-store-sops
simp add: nth-list-update split: if-split-asm)
lemma (in valid-data-dependency) valid-data-dependency-nth-update:
$\llbracket \mathrm{i}<$ length ts; data-dependency-consistent-instrs ( $\operatorname{dom} \vartheta$ ) is;
load-tmps is $\cap \bigcup$ (fst ' write-sops sb) $=\{ \} \rrbracket \Longrightarrow$
valid-data-dependency $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold valid-data-dependency-def)
(force dest: data-dependency-consistent-instrs load-tmps-write-tmps-distinct simp add: nth-list-update split: if-split-asm)
lemma (in enough-flushs) enough-flushs-nth-update:
$\llbracket i<$ length ts;
$\neg \mathcal{D} \longrightarrow$ (outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
$\rrbracket \Longrightarrow$
enough-flushs $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
apply (unfold-locales) apply (force simp add: nth-list-update split: if-split-asm dest: clean-no-outstanding-volatile-Write ${ }_{\text {sb }}$ )

## done

lemma (in outstanding-non-volatile-writes-unshared)
outstanding-non-volatile-writes-unshared-nth-update:
$\llbracket \mathrm{i}$ < length ts; non-volatile-writes-unshared $\mathcal{S} \mathrm{sb} \rrbracket \Longrightarrow$
outstanding-non-volatile-writes-unshared $\mathcal{S}(\operatorname{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: outstanding-non-volatile-writes-unshared simp add: nth-list-update split: if-split-asm)
lemma (in sharing-consis)
sharing-consis-nth-update:
$\llbracket i<$ length ts; sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \rrbracket \Longrightarrow$
sharing-consis $\mathcal{S}(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: sharing-consis
simp add: nth-list-update split: if-split-asm)

```
lemma (in no-outstanding-write-to-read-only-memory)
no-outstanding-write-to-read-only-memory-nth-update:
    \llbracket i < ~ l e n g t h ~ t s ; ~ n o - w r i t e - t o - r e a d - o n l y - m e m o r y ~ \mathcal { S ~ s b } \rrbracket \Longrightarrow
            no-outstanding-write-to-read-only-memory \mathcal{S}}(\textrm{ts}[\textrm{i}:=(\textrm{p},\textrm{is},\textrm{xs},\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})]
by (unfold-locales)
    (auto dest: no-outstanding-write-to-read-only-memory
        simp add: nth-list-update split: if-split-asm)
```

lemma in-Union-image-nth-conv: $\mathrm{a} \in \mathrm{U}$ (f ${ }^{\text {‘ }}$ set xs$) \Longrightarrow \exists \mathrm{i} . \mathrm{i}<$ length $\mathrm{xs} \wedge \mathrm{a} \in \mathrm{f}$ (xs!i)
by (auto simp add: in-set-conv-nth)
lemma in-Inter-image-nth-conv: $\mathrm{a} \in \bigcap$ (f ' set xs) $=(\forall \mathrm{i}<$ length xs. $\mathrm{a} \in \mathrm{f}(\mathrm{xs}!\mathrm{i}))$
by (force simp add: in-set-conv-nth)
lemma release-ownership-nth-update:
assumes R -subset: $\mathrm{R} \subseteq \mathcal{O}$
shows $\wedge \mathrm{i} . \llbracket \mathrm{i}<$ length ts; ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$;
ownership-distinct ts】
$\Longrightarrow \bigcup\left((\lambda(-,-,-,-,, \mathcal{O},-) . \mathcal{O}){ }^{\prime} \operatorname{set}\left(\operatorname{ts}\left[\mathrm{i}=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}-\mathrm{R}, \mathcal{R}^{\prime}\right)\right]\right)\right)$
$=((\bigcup((\lambda(-,-,-,-,, \mathcal{O},-) . \mathcal{O})$ 'set ts $))-\mathrm{R})$
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
note i-bound $=$ i $<$ length ( $\mathrm{t} \# \mathrm{ts}$ ) )
note ith $=\langle(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
note dist $=$ रownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) $\rangle$
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
from dist
have dist': ownership-distinct ts
by (rule ownership-distinct-tl)
show ?case
proof (cases i)
case 0
from ith 0 have $\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by simp
have $\mathrm{R} \cap(\cup((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})$ ' set ts $))=\{ \}$
proof -
\{
fix $x$
assume $x-R: x \in R$
assume $\mathrm{x}-\mathrm{ls}: \mathrm{x} \in(\bigcup((\lambda(-,-,-,-,-\mathcal{O},-) . \mathcal{O})$ ' set ts $))$
then obtain $\mathrm{j} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$ where

```
    j-bound: j < length ts and
    jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{,},\mp@subsup{\textrm{xs}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{})\mathrm{ and
    x-in: x }\in\mp@subsup{\mathcal{O}}{j}{
    by (fastforce simp add: in-set-conv-nth )
from j-bound jth 0
have }(\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{j}{}\cup\mathrm{ all-acquired sb }\mp@subsup{\textrm{s}}{\textrm{j}}{})={
    apply -
    apply (rule ownership-distinct [OF i-bound - - ith, of Suc j])
    apply clarsimp+
    apply blast
    done
with x-R R-subset x-in have False
    by auto
        }
        thus ?thesis
by blast
        qed
        then
        show ?thesis
        by (auto simp add: 0 t)
next
    case (Suc n)
```



```
        by (cases t)
    have n-bound: n < length ts
        using i-bound by (simp add: Suc)
    have nth: ts!n = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        using ith by (simp add: Suc)
    have }\textrm{R}\cap(\mp@subsup{\mathcal{O}}{1}{}\cup\mathrm{ all-acquired sb }\mp@subsup{\textrm{sb}}{\textrm{l}}{\prime})={
    proof -
        {
fix x
assume x-R: x }\in
assume x-owns|: x }\in(\mp@subsup{\mathcal{O}}{\textrm{l}}{|}\cup\mathrm{ all-acquired sb
from t
have }(\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{|}{}\cup\mathrm{ all-acquired sb
    apply -
    apply (rule ownership-distinct [OF i-bound - - ith, of 0])
    apply (auto simp add: Suc)
    done
with x-owns| x-R R-subset have False
    by auto
        }
        thus ?thesis
by blast
    qed
    with Cons.hyps [OF n-bound nth dist']
```

```
        show ?thesis
        by (auto simp add: Suc t)
    qed
qed
```

lemma acquire-ownership-nth-update:
shows $\bigwedge \mathrm{i} . \llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot \operatorname{set}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O} \cup \mathrm{A}, \mathcal{R}^{\prime}\right)\right]\right)\right)$
$=((\cup((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) '$ set ts $)) \cup \mathrm{A})$
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
note i -bound $=$ «i $<$ length ( $\mathrm{t} \# \mathrm{ts}$ ) $\rangle$
note ith $=\langle(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
show ?case
proof (cases i)
case 0
from ith 0 have $\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by simp
show ?thesis
by (auto simp add: 0 t)
next
case (Suc n)
obtain $\mathrm{p}_{\mathrm{l}}$ is $\mathcal{O}_{\mathrm{l}} \mathcal{R}_{\mid} \mathcal{D}_{\mathrm{l}} \mathrm{xs}_{\mid} \mathrm{sb}_{\mid}$where $\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{l}}, \mathrm{is}_{\mid}, \mathrm{xs}_{\mid}, \mathrm{sb}_{\mid}, \mathcal{D}_{\mathrm{l}}, \mathcal{O}_{\mathrm{l}}, \mathcal{R}_{\mid}\right)$
by (cases t)
have $n$-bound: $n<$ length ts
using i-bound by (simp add: Suc)
have nth: $\mathrm{ts}!\mathrm{n}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
using ith by (simp add: Suc)
from Cons.hyps [OF n-bound nth]
show ?thesis
by (auto simp add: Suc t)
qed
qed
lemma acquire-release-ownership-nth-update:
assumes R-subset: $\mathrm{R} \subseteq \mathcal{O}$
shows $\bigwedge \mathrm{i} . \llbracket \mathrm{i}<$ length ts; ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$; ownership-distinct ts』
$\Longrightarrow \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot \operatorname{set}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \mathrm{xs}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O} \cup \mathrm{A}-\mathrm{R}, \mathcal{R}^{\prime}\right)\right]\right)\right)$ $=\left(\left(\cup\left((\lambda(-,-,-,-,-\mathcal{O},-) . \mathcal{O})^{\prime}\right.\right.\right.$ set ts $\left.\left.)\right) \cup \mathrm{A}-\mathrm{R}\right)$
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
note i -bound $=\langle\mathrm{i}<$ length $(\mathrm{t} \# \mathrm{ts})\rangle$
note ith $=\langle(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
note dist $=$ 〈ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) $\rangle$

```
then interpret ownership-distinct t#ts.
from dist
have dist': ownership-distinct ts
    by (rule ownership-distinct-tl)
show ?case
proof (cases i)
    case 0
    from ith 0 have t: t = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by simp
    have R \cap(U((\lambda(-,-,-,-,,\mathcal{O},-).\mathcal{O})'set ts))={}
    proof -
        {
fix x
assume x-R: x }\in
assume x-ls: x \in (U ((\lambda(-,-,-,-,,\mathcal{O},-). \mathcal{O)` set ts))}
then obtain j p i is }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ where
    j-bound: j < length ts and
    jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,}\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{xSj}
    x-in: x }\in\mp@subsup{\mathcal{O}}{j}{
    by (fastforce simp add: in-set-conv-nth )
from j-bound jth 0
have }(\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb }\mp@subsup{\textrm{j}}{\textrm{j}}{})={
    apply -
    apply (rule ownership-distinct [OF i-bound - - ith, of Suc j])
    apply clarsimp+
    apply blast
    done
with x-R R-subset x-in have False
    by auto
        }
        thus ?thesis
by blast
    qed
    then
    show ?thesis
        by (auto simp add: 0 t)
next
    case (Suc n)
    obtain pl is is| }\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{1}{}\mp@subsup{\mathcal{D}}{|}{}\mp@subsup{\textrm{xs}}{|}{}\mathrm{ sb m
        by (cases t)
    have n-bound: n < length ts
        using i-bound by (simp add: Suc)
    have nth: ts!n = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        using ith by (simp add: Suc)
    have }\textrm{R}\cap(\mp@subsup{\mathcal{O}}{\}{}\cup\mathrm{ all-acquired sb 
    proof -
        {
```

```
fix \(x\)
assume \(x-R: x \in R\)
assume x -owns| \(: \mathrm{x} \in\left(\mathcal{O}_{\mathrm{l}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{l}}\right)\)
from t
have \((\mathcal{O} \cup\) all-acquired sb\() \cap\left(\mathcal{O}_{।} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{l}}\right)=\{ \}\)
    apply -
    apply (rule ownership-distinct [OF i-bound -- ith, of 0])
    apply (auto simp add: Suc)
    done
with x -owns| \(\mathrm{x}-\mathrm{R}\) R-subset have False
    by auto
        \}
        thus ?thesis
by blast
        qed
        with Cons.hyps [OF n-bound nth dist']
        show ?thesis
        by (auto simp add: Suc t)
    qed
qed
```

lemma (in valid-history) valid-history-nth-update:
$\llbracket \mathrm{i}<$ length ts; history-consistent $\vartheta$ (hd-prog p sb) sb $\rrbracket \Longrightarrow$ valid-history program-step $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: valid-history simp add: nth-list-update split: if-split-asm)
lemma (in valid-reads) valid-reads-nth-update:
$\llbracket i<$ length ts; reads-consistent False $\mathcal{O} \mathrm{m} \mathrm{sb} \rrbracket \Longrightarrow$ valid-reads m (ts[i $:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: valid-reads simp add: nth-list-update split: if-split-asm)
lemma (in load-tmps-distinct) load-tmps-distinct-nth-update:
$\llbracket \mathrm{i}<$ length ts; distinct-load-tmps is $\rrbracket \Longrightarrow$
load-tmps-distinct $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: load-tmps-distinct simp add: nth-list-update split: if-split-asm)
lemma (in read-tmps-distinct) read-tmps-distinct-nth-update:
$\llbracket i<$ length ts; distinct-read-tmps sb $\rrbracket \Longrightarrow$
read-tmps-distinct $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: read-tmps-distinct simp add: nth-list-update split: if-split-asm)
lemma (in load-tmps-read-tmps-distinct) load-tmps-read-tmps-distinct-nth-update:
$\llbracket \mathrm{i}<$ length ts; load-tmps is $\cap$ read-tmps sb $=\{ \} \rrbracket \Longrightarrow$ load-tmps-read-tmps-distinct $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(auto dest: load-tmps-read-tmps-distinct simp add: nth-list-update split: if-split-asm)
lemma (in load-tmps-fresh) load-tmps-fresh-nth-update:

$$
\llbracket \mathrm{i}<\text { length ts }
$$

load-tmps is $\cap \operatorname{dom} \vartheta=\{ \} \rrbracket \Longrightarrow$
load-tmps-fresh $(\mathrm{ts}[\mathrm{i}:=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})])$
by (unfold-locales)
(fastforce dest: load-tmps-fresh
simp add: nth-list-update split: if-split-asm)
fun flush-all-until-volatile-write::
('p,'p store-buffer, 'dirty,'owns, 'rels) thread-config list $\Rightarrow$ memory $\Rightarrow$ memory where
flush-all-until-volatile-write [] $\mathrm{m}=\mathrm{m}$
| flush-all-until-volatile-write ( $(-,-,-$, sb,-, -$) \# \mathrm{ts}) \mathrm{m}=$
flush-all-until-volatile-write ts (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m)
fun share-all-until-volatile-write::
('p,'p store-buffer, 'dirty, 'owns,'rels) thread-config list $\Rightarrow$ shared $\Rightarrow$ shared

## where

share-all-until-volatile-write [] $\mathrm{S}=\mathrm{S}$
| share-all-until-volatile-write ( $(-,-,-$, sb,-,-) \#ts) $\mathrm{S}=$
share-all-until-volatile-write ts (share (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) sb) S)
lemma takeWhile-dropWhile-real-prefix:
$\llbracket \mathrm{x} \in \operatorname{set} \mathrm{xs} ; \neg \mathrm{P} \mathrm{x} \rrbracket \Longrightarrow \exists \mathrm{y}$ ys. xs=takeWhile P xs @ y\#ys $\wedge \neg \mathrm{P} \mathrm{y} \wedge$ dropWhile P xs $=\mathrm{y} \# \mathrm{ys}$
by (induct xs) auto
lemma buffered-val-witness: buffered-val sb $\mathrm{a}=$ Some $\mathrm{v} \Longrightarrow$
$\exists$ volatile sop A L R W. Write ${ }_{\text {sb }}$ volatile a sop v A L R W $\in$ set sb
apply (induct sb)
apply $\operatorname{simp}$
apply (clarsimp split: memref.splits option.splits if-split-asm)
apply blast
apply blast
done
lemma flush-append-Read ${ }_{s b}$ :
$\bigwedge \mathrm{m}$. (flush (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) ( $\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v$\left.]\right)$ ) m) $=$ flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m
by (induct sb) (auto split: memref.splits)
lemma flush-append-write:
$\bigwedge \mathrm{m}$. (flush (sb @ [Write ${ }_{\text {sb }}$ volatile a sop v A L R W]) m) $=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$
by (induct sb) (auto split: memref.splits)
lemma flush-append- $\operatorname{Prog}_{s b}$ :
$\bigwedge \mathrm{m}$. (flush (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) ( $\left.\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\right]\right)$ ) m) $=$ (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m)
by (induct sb) (auto split: memref.splits)
lemma flush-append-Ghost ${ }_{\text {sb }}$ :
$\bigwedge \mathrm{m}$. (flush (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) (sb @ [Ghost ${ }_{\text {sb }}$ A L R W])) m) $=$ (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m)
by (induct sb) (auto split: memref.splits)
lemma share-append: $\bigwedge$ S. share (xs@ys) $S=$ share ys (share xs $S$ )
by (induct xs) (auto split: memref.splits)
lemma share-append-Read sb $_{\text {: }}$ :
$\wedge$ S. (share (takeWhile (Not o is-volatile-Write $\left.{ }_{\text {sb }}\right)\left(\mathrm{sb}\right.$ @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v$\left.\left.\left.]\right)\right) \mathrm{S}\right)$ $=$ share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) S
by (induct sb) (auto split: memref.splits)
lemma share-append-Write ${ }_{s b}$ :
^S. (share (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) (sb @ [Write ${ }_{\text {sb }}$ volatile a sop v A L R
W])) S)
$=$ share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) S
by (induct sb) (auto split: memref.splits)
lemma share-append- Prog $_{\text {sb }}$ :
^S. (share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) (sb @ $\left.\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\right]\right)$ ) S$)=$ (share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) S)
by (induct sb) (auto split: memref.splits)
lemma in-acquired-no-pending-write-outstanding-write:
$\mathrm{a} \in$ acquired False sb $\mathrm{A} \Longrightarrow$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ sb $\neq\{ \}$
apply (induct sb)
apply (auto split: memref.splits)
done
lemma flush-buffered-val-conv:
$\bigwedge \mathrm{m}$. flush sb mate (case buffered-val sba of None $\Rightarrow \mathrm{m}$ a $\mid$ Some $\mathrm{v} \Rightarrow \mathrm{v}$ )
by (induct sb) (auto split: memref.splits option.splits)
lemma reads-consistent-unbuffered-snoc:
$\bigwedge \mathrm{m}$. buffered-val $\mathrm{sb} \mathrm{a}=$ None $\Longrightarrow \mathrm{ma}=\mathrm{v} \Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}$ sb
volatile $\longrightarrow$
outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$
$\Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile atv$\left.]\right)$
by (simp add: reads-consistent-append flush-buffered-val-conv)

## lemma reads-consistent-buffered-snoc

$\bigwedge \mathrm{m}$. buffered-val sb $\mathrm{a}=$ Some $\mathrm{v} \Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$ volatile $\longrightarrow$ outstanding-refs is-volatile-Write ${ }_{s b} \mathrm{sb}=\{ \}$
$\Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile atv $\left.\left.\mathbf{v}\right]\right)$
by (simp add: reads-consistent-append flush-buffered-val-conv)
lemma reads-consistent-snoc-Write ${ }_{\text {sb }}$ :
$\bigwedge \mathrm{m}$. reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$
reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (sb @ $\left[\mathrm{Write}_{\text {sb }}\right.$ volatile a sop v A L R W])
by (simp add: reads-consistent-append)
lemma reads-consistent-snoc- Prog $_{\text {sb }}$ :
$\bigwedge \mathrm{m}$. reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (sb @ $\left.\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\right]\right)$
by (simp add: reads-consistent-append)
lemma reads-consistent-snoc-Ghost ${ }_{\text {sb }}$ :
$\bigwedge \mathrm{m}$. reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (sb @ [Ghost ${ }_{\text {sb }}$ A L R W])
by (simp add: reads-consistent-append)

```
lemma restrict-map-id [simp]:m |' dom \(\mathrm{m}=\mathrm{m}\)
    apply (rule ext)
    subgoal for x
    apply (case-tac m x)
    apply (auto simp add: restrict-map-def domIff)
    done
    done
lemma flush-all-until-volatile-write-Read-commute:
    shows \(\bigwedge \mathrm{m}\) i. \(\llbracket \mathrm{i}<\) length \(\mathrm{ls} ; \mathrm{ls}!\mathrm{i}=(\mathrm{p}, \operatorname{Read}\) volatile a \(\mathrm{t} \# \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
        】
        flush-all-until-volatile-write
            \(\left(\mathrm{ls}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta(\mathrm{t} \mapsto \mathrm{v})\right.\right.\right.\), sb @ \(\left[\operatorname{Read}_{\mathrm{sb}}\right.\) volatile a t v\(\left.\left.\left.], \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right)\right]\right) \mathrm{m}=\)
        flush-all-until-volatile-write ls m
proof (induct ls)
    case Nil thus ?case
        by simp
next
    case (Cons lls)
    note i -bound \(=\) \(\mathrm{i}<\) length (l \(\# \mathrm{l}\) s) \(\rangle\)
    note ith \(=\langle(\mathrm{l} \# \mathrm{ls})!\mathrm{i}=(\mathrm{p}\), Read volatile a \(\mathrm{t} \# \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle\)
    show ?case
```

```
proof (cases i)
    case 0
    from ith 0 have l: l=(p,Read volatile a t#is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by simp
    thus ?thesis
        by (simp add: 0 flush-append-Readsb del: fun-upd-apply )
next
    case (Suc n)
    obtain pl is il O
        by (cases l)
    from i-bound ith
    have flush-all-until-volatile-write
        (ls[n := (p,is ,\vartheta(t\mapstov), sb @ [Read sb volatile a t v ], D
        (flush (takeWhile (Not o is-volatile-Write sb) sbl})\textrm{m})
        flush-all-until-volatile-write ls (flush (takeWhile (Not o is-volatile-Write sb) sblı
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then
    show ?thesis
        by (simp add: Suc l del: fun-upd-apply)
qed
qed
lemma flush-all-until-volatile-write-append-Ghost-commute:
    \i m. \llbracketi < length ts; ts!i=(p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
        \Longrightarrow \text { flush-all-until-volatile-write (ts[i := (p} \mathrm { p } ^ { \prime } , \mathrm { is } ^ { \prime } , \vartheta ^ { \prime } , \text { sb@[Ghost sb A L R W], D}
m
        = flush-all-until-volatile-write ts m
proof (induct ts)
    case Nil thus ?case
        by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts)>
    note ith = <(l#ts)!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l=(p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
            by simp
    thus?thesis
            by (simp add: 0 flush-append-Ghost sb del: fun-upd-apply)
    next
        case (Suc n)
```



```
            by (cases l)
```

```
    from i-bound ith
    have flush-all-until-volatile-write
                    (ts[n := (p',is',\vartheta', sb@[Ghost sb A L R W], \mathcal{D}
            (flush (takeWhile (Not o is-volatile-Write sb})\mp@subsup{\textrm{sb}}{\boldsymbol{\prime}}{})\textrm{m})
        flush-all-until-volatile-write ts
            (flush (takeWhile (Not o is-volatile-Write sb
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then show ?thesis
    by (simp add: Suc l)
    qed
qed
lemma update-commute:
assumes g-unchanged: \(\forall \mathrm{am} . \mathrm{a} \notin \mathrm{G} \longrightarrow \mathrm{g} \mathrm{ma}=\mathrm{ma}\)
assumes g -independent: \(\forall \mathrm{am} . \mathrm{a} \in \mathrm{G} \longrightarrow \mathrm{g}(\mathrm{fm}) \mathrm{a}=\mathrm{g} \mathrm{m}\) a
assumes f-unchanged: \(\forall \mathrm{am} . \mathrm{a} \notin \mathrm{F} \longrightarrow \mathrm{fma}=\mathrm{m} a\)
assumes f-independent: \(\forall \mathrm{am} . \mathrm{a} \in \mathrm{F} \longrightarrow \mathrm{f}(\mathrm{g} \mathrm{m}) \mathrm{a}=\mathrm{fm} \mathrm{m}\)
assumes disj: \(\mathrm{G} \cap \mathrm{F}=\{ \}\)
shows \(\mathrm{f}(\mathrm{gm})=\mathrm{g}(\mathrm{fm})\)
proof
fix a
show \(f(g m) a=g(f m) a\)
proof (cases \(a \in G\) )
case True
with disj have a-notin-F: a \(\notin \mathrm{F}\)
by blast
from f-unchanged [rule-format, OF a-notin-F, of g m ]
have \(\mathrm{f}(\mathrm{gm}) \mathrm{a}=\mathrm{g} \mathrm{m}\) a.
also
from \(g\)-independent [rule-format, OF True]
have \(\ldots=\mathrm{g}\) ( f m ) a by simp
finally show? thesis .
next
case False
note a-notin- \(\mathrm{G}=\) this
show ?thesis
proof (cases a \(\in F\) )
case True
from f-independent [rule-format, OF True]
have \(\mathrm{f}(\mathrm{g} \mathrm{m}) \mathrm{a}=\mathrm{f} \mathrm{m}\) a by simp
also
from g -unchanged [rule-format, OF a-notin-G]
have \(\ldots=\mathrm{g}\) ( fm ) a
by simp
finally show?thesis .
```

```
    next
    case False
    from f-unchanged [rule-format, OF False]
    have f (g m)a=g m a.
    also
    from g-unchanged [rule-format, OF a-notin-G]
    have ... = m a .
    also
    from f-unchanged [rule-format, OF False]
    have .. = f m a by simp
    also
    from g-unchanged [rule-format, OF a-notin-G]
    have ... = g (f m) a
by simp
    finally show ?thesis .
    qed
qed
qed
```

lemma update-commute':
assumes g-unchanged: $\forall \mathrm{a} \mathrm{m} . \mathrm{a} \notin \mathrm{G} \longrightarrow \mathrm{g} \mathrm{ma}=\mathrm{ma}$
assumes g -independent: $\forall \mathrm{a}_{1} \mathrm{~m}_{2} . \mathrm{a} \in \mathrm{G} \longrightarrow \mathrm{g} \mathrm{m} \mathrm{m}_{1} \mathrm{a}=\mathrm{g} \mathrm{m}_{2} \mathrm{a}$
assumes f-unchanged: $\forall \mathrm{a} \mathrm{m} . \mathrm{a} \notin \mathrm{F} \longrightarrow \mathrm{fma}=\mathrm{ma}$
assumes f-independent: $\forall \mathrm{a}_{1} \mathrm{~m}_{2} . \mathrm{a} \in \mathrm{F} \longrightarrow \mathrm{f} \mathrm{m}_{1} \mathrm{a}=\mathrm{f} \mathrm{m}_{2} \mathrm{a}$
assumes disj: $\mathrm{G} \cap \mathrm{F}=\{ \}$
shows $\mathrm{f}(\mathrm{g} \mathrm{m})=\mathrm{g}(\mathrm{f} m)$
proof -
from g -independent have g -ind ${ }^{\prime}: \forall \mathrm{am} . \mathrm{a} \in \mathrm{G} \longrightarrow \mathrm{g}(\mathrm{fm}) \mathrm{a}=\mathrm{g} \mathrm{m}$ a by blast
from f-independent have f -ind': $\forall \mathrm{a} \mathrm{m} . \mathrm{a} \in \mathrm{F} \longrightarrow \mathrm{f}(\mathrm{g} \mathrm{m}) \mathrm{a}=\mathrm{f} \mathrm{m}$ a by blast
from update-commute [OF g-unchanged $g$-ind' f-unchanged f-ind' disj]
show ?thesis .
qed
lemma flush-unchanged-addresses: $\wedge \mathrm{m}$. a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }} \mathrm{sb} \Longrightarrow$ flush sb
$\mathrm{ma}=\mathrm{ma}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons r sb)
note a -notin $=\left\langle\mathrm{a} \notin\right.$ outstanding-refs is-Write $\left.{ }_{\text {sb }}(\mathrm{r} \# \mathrm{sb})\right\rangle$
show ?case
proof (cases r)
case ( Write $_{\text {sb }}$ volatile a'sop v)
from a-notin obtain neq-a-a': $\mathrm{a} \neq \mathrm{a}^{\prime}$ and $\mathrm{a}-$ notin': $\mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb by (simp add: Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF a-notin', of $m\left(\mathrm{a}^{\prime}:=\mathrm{v}\right)$ ] neq-a-a'
show ?thesis
apply (simp add: Write ${ }_{\text {sb }}$ del: fun-upd-apply)
apply simp

```
        done
    next
        case (Readsb volatile a't v)
        from a-notin obtain a-notin': a & outstanding-refs is-Write sb sb
        by (simp add: Readsb)
    from Cons.hyps [OF a-notin', of m]
    show ?thesis
        by (simp add: Readsb)
    next
    case Prog}\mp@subsup{}{sb}{}\mathrm{ with Cons show ?thesis by simp
    next
        case Ghost sb with Cons show ?thesis by simp
    qed
qed
lemma flushed-values-mem-independent:
    \m m' a. a }\in\mathrm{ outstanding-refs is-Write sb sb }\Longrightarrow\mathrm{ flush sb m' a = flush sb m a
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons r sb)
    show ?case
    proof (cases r)
        case (Write sb volatile a' sop' v}\mp@subsup{v}{}{\prime}
        have flush sb (m}(\mp@subsup{\textrm{m}}{}{\prime}:=\mp@subsup{\textrm{v}}{}{\prime}))\mp@subsup{\textrm{a}}{}{\prime}=\mathrm{ flush sb (m(a
        proof (cases a' }\mp@subsup{|}{\mathrm{ outstanding-refs is-Write }}{\mathbf{sb}
            case True
            from Cons.hyps [OF this]
            show ?thesis .
    next
                case False
                from flush-unchanged-addresses [OF False]
                show ?thesis
    by simp
        qed
        with Cons.hyps Cons.prems
        show ?thesis
            by (auto simp add: Write sb
    next
        case Read sb thus ?thesis using Cons
        by auto
    next
        case Progsb thus ?thesis using Cons
                by auto
    next
        case Ghost sb thus ?thesis using Cons
            by auto
    qed
qed
```

lemma flush-all-until-volatile-write-unchanged-addresses:
$\wedge \mathrm{m} . \mathrm{a} \notin \bigcup\left(\left(\lambda(-,-,-, \mathrm{sb},-,-,-)\right.\right.$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) ' set ls) $\Longrightarrow$ flush-all-until-volatile-write ls $\mathrm{ma}=\mathrm{m}$ a
proof (induct ls)
case Nil thus ?case by simp
next
case (Cons 1 ls)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D}$ xs sb where $\mathrm{l}: \mathrm{l}=(\mathrm{p}, \mathrm{s}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ by (cases l)
note $\varsigma \mathrm{a} \notin \bigcup\left(\left(\lambda(-,-,-\right.\right.$, sb,-,-,-, $)$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) ' set ( $\mathrm{l} \# \mathrm{ls}$ )) 〕

## then obtain

a-notin-sb: a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) and
a-notin-ls: a $\notin \bigcup\left(\left(\lambda(-,-,-, s b,-,-,-)\right.\right.$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb)) ' set ls)
by (auto simp add: l)
from Cons.hyps [OF a-notin-ls]
have flush-all-until-volatile-write ls (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) m)
a

$$
=
$$

(flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m) a.
also
from flush-unchanged-addresses [OF a-notin-sb]
have (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) m) $\mathrm{a}=\mathrm{m}$ a.
finally
show ?case
by (simp add: l)
qed
lemma notin-outstanding-non-volatile-takeWhile-lem:
a $\notin$ outstanding-refs (Not $\circ$ is-volatile) sb

$$
\Longrightarrow
$$


apply (induct sb)
apply (auto simp add: is- Write $_{\text {sb }}$-def split: if-split-asm memref.splits)
done
lemma notin-outstanding-non-volatile-takeWhile-lem':
a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb

$$
\Longrightarrow
$$

$\mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (induct sb)
apply (auto simp add: is-Write ${ }_{\text {sb }}$-def split: if-split-asm memref.splits)
done
lemma notin-outstanding-non-volatile-takeWhile-Un-lem':
$\mathrm{a} \notin \bigcup((\lambda(-,-,-$, sb,-,-,-). outstanding-refs (Not o is-volatile) sb) ' set ls)
$\Longrightarrow \mathrm{a} \notin \bigcup\left(\left(\lambda(-,-,-, \mathrm{sb},-,-,-)\right.\right.$. outstanding-refs is-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb))' set ls)
proof (induct ls)
case Nil thus ?case by simp
next
case (Cons 1 ls)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D}$ xs sb where $\mathrm{l}: \mathrm{l}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases l)

```
    from Cons.prems
    obtain
        a-notin-sb: a & outstanding-refs (Not o is-volatile) sb and
        a-notin-ls: a }\not=\bigcup((\lambda(-,-,-,sb,-,,-). outstanding-refs (Not o is-volatile) sb)' set ls
        by (force simp add: 1 simp del: o-apply)
    from notin-outstanding-non-volatile-takeWhile-lem [OF a-notin-sb]
    Cons.hyps [OF a-notin-ls]
    show ?case
        by (auto simp add: 1 simp del: o-apply)
qed
```

lemma flush-all-until-volatile-write-unchanged-addresses':
assumes notin: a $\notin \bigcup((\lambda(-,-,-$, sb,-,-,-, $)$. outstanding-refs (Not $\circ$ is-volatile) sb) ' set ls)
shows flush-all-until-volatile-write ls $\mathrm{ma}=\mathrm{m}$ a
using notin-outstanding-non-volatile-takeWhile-Un-lem' [OF notin]
by (auto intro: flush-all-until-volatile-write-unchanged-addresses)
lemma flush-all-until-volatile-wirte-mem-independent:
$\Lambda \mathrm{m} \mathrm{m}^{\prime} . \mathrm{a} \in \bigcup\left(\left(\lambda(-,-,-, \mathrm{sb},-,-,-)\right.\right.$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) ' set ls) $\Longrightarrow$
flush-all-until-volatile-write $\mathrm{ls} \mathrm{m}^{\prime} \mathrm{a}=$ flush-all-until-volatile-write ls m a
proof (induct ls)
case Nil thus ?case by simp
next
case (Cons 1 ls)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D}$ xs sb where $\mathrm{l}: \mathrm{l}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases l)
note $\mathrm{a}-\mathrm{in}=<\mathrm{a} \in \bigcup\left(\left(\lambda(-,-,-\right.\right.$, sb,-,-,-, $)$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb)) ' set ( $\mathrm{l} \# \mathrm{ls}$ ) ) )
show ?case
proof (cases a $\in \bigcup\left(\left(\lambda(-,-,-\right.\right.$, sb,-,-,,$)$. outstanding-refs is-Write ${ }_{\text {sb }}$
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)) ' set ls))
case True
from Cons.hyps [OF this]
show ?thesis
by (simp add: 1 )
next
case False

```
    with a-in
    have a }\in\mathrm{ outstanding-refs is-Write sb (takeWhile (Not ० is-volatile-Write }\mp@subsup{\mathbf{sb}}{\mathbf{sb}}{}\mathrm{ ) sb)
        by (auto simp add: l)
    from flushed-values-mem-independent [rule-format, OF this]
    have flush (takeWhile (Not o is-volatile-Write 
            flush (takeWhile (Not o is-volatile-Write sb) sb) m a.
    with flush-all-until-volatile-write-unchanged-addresses [OF False]
    show ?thesis
        by (auto simp add: l)
    qed
qed
lemma flush-all-until-volatile-write-buffered-val-conv:
    assumes no-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ : outstanding-refs is-volatile-Write 
    shows\m i. \llbracketi < length ls; ls!i = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R});
    j < length ls. i f j }
        (let (-,-,-,sbj,-,-,-) = ls!j
            in a }\not\in\mathrm{ outstanding-refs is-non-volatile-Writesb (takeWhile (Not o
is-volatile-Write sb
    flush-all-until-volatile-write ls m a =
            (case buffered-val sb a of None }=>\textrm{m}||\mathrm{ Some v }=>\textrm{v}\mathrm{ )
proof (induct ls)
    case Nil thus ?case
        by simp
next
    case (Cons l ls)
    note i-bound = <i < length (l#ls) 
    note ith = <(l#ls)!i = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    note notin = < | < length (l#ls). i = j }
                    (let (-,-,-,sb
                            in a & outstanding-refs is-non-volatile-Write}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ (takeWhile (Not o
is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{b}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{})\mathrm{ )/
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
            by simp
        from no-volatile-Write sb have take-all: takeWhile (Not o is-volatile-Write sb
            by (auto simp add: outstanding-refs-conv)
        have a }\not\in\bigcup((\lambda(-,-, -, sb, -,-,-)
                    outstanding-refs is-Write
                    (takeWhile (Not o is-volatile-Write sb
        proof
            assume a }\in\mathrm{ ?LS
            from in-Union-image-nth-conv [OF this]
            obtain j pj is j}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ where
    j-bound: j < length ls and
    jth: ls!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{})\mathrm{ and
```

a-in-j: $\mathrm{a} \in$ outstanding-refs is-Write ${ }_{\mathbf{s b}}$ (takeWhile (Not $\circ$ is-volatile-Write $\mathbf{s b}_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) by fastforce
 (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ )
apply (clarsimp simp add: outstanding-refs-conv )
subgoal for x
apply (case-tac x)
apply clarsimp
apply (frule set-takeWhileD)
apply auto
done
done
with notin [rule-format, of Suc j] j-bound jth
show False
by (force simp add: 0 outstanding-refs-conv is-non-volatile-Write ${ }_{\text {sb }}$-def
split: memref.splits)
qed
from flush-all-until-volatile-write-unchanged-addresses [OF this]
have flush-all-until-volatile-write ls (flush sb m) $\mathrm{a}=$ (flush sb m) a
by (simp add: take-all)
then
show ?thesis
by (simp add: 0 l take-all flush-buffered-val-conv)
next
case (Suc n)

by (cases l)
from i-bound ith notin
have flush-all-until-volatile-write ls
(flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $\mathrm{sb}_{\mathrm{l}}$ ) m) a
$=$ (case buffered-val sb a of None $\Rightarrow$
(flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{l}}$ ) m) a $\mid$ Some $\mathrm{v} \Rightarrow \mathrm{v}$ )
apply -
apply (rule Cons.hyps)
apply (force simp add: Suc Let-def simp del: o-apply) + done
moreover
from notin [rule-format, of 0] l
have a $\notin$ outstanding-refs is-non-volatile-Write $_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{l}}$ )
by (auto simp add: Let-def outstanding-refs-conv Suc )
then
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sblı ${ }_{\text {I }}$ )
apply (clarsimp simp add: outstanding-refs-conv is-Write ${ }_{\mathbf{s b}}$-def split: memref.splits dest: set-takeWhileD)
apply (frule set-takeWhileD)
apply force
done
from flush-unchanged-addresses [OF this]
have (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{l}}$ ) m) $\mathrm{a}=\mathrm{m} \mathrm{a} \cdot$

```
    ultimately
    show ?thesis
    by (simp add: Suc l split: option.splits)
qed
qed
```

context program
begin
abbreviation sb-concurrent-step ::
('p,'p store-buffer,'dirty,'owns,'rels,'shared) global-config $\quad \Rightarrow \quad(' p$, 'p
store-buffer, 'dirty, 'owns, 'rels, 'shared) global-config $\Rightarrow$ bool

$$
\left(-\Rightarrow_{\text {sb }}-[60,60] 100\right)
$$

where
sb-concurrent-step $\equiv$
computation.concurrent-step sb-memop-step store-buffer-step program-step ( $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb. sb)
$\operatorname{term} \mathrm{x} \Rightarrow_{\text {sb }} \mathrm{Y}$
abbreviation (in program) sb-concurrent-steps::
('p,'p store-buffer,'dirty,'owns,'rels,'shared) global-config $\quad \Rightarrow \quad(' p$, 'p
store-buffer,'dirty,'owns,'rels,'shared) global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{\mathrm{sb}}{ }^{*}-[60,60] 100\right)$
where
sb-concurrent-steps $\equiv$ sb-concurrent-step ${ }^{*} * *$
$\operatorname{term} \mathrm{x} \Rightarrow_{\mathrm{sb}}{ }^{*} \mathrm{Y}$
abbreviation sbh-concurrent-step ::
('p,'p store-buffer,bool,owns,rels,shared) global-config $\Rightarrow$ ('p,'p store-buffer,bool,owns,rels,shared) global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{\text {sbh }}-[60,60] 100\right)$
where
sbh-concurrent-step $\equiv$
computation.concurrent-step sbh-memop-step flush-step program-step $\left(\lambda \mathrm{p} \mathrm{p}^{\prime}\right.$ is $\mathrm{sb} . \mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $\left.]\right)$
term $\mathrm{x} \Rightarrow_{\text {sbh }} \mathrm{Y}$
abbreviation sbh-concurrent-steps::
('p,'p store-buffer,bool,owns,rels,shared) global-config $\Rightarrow$ ('p,'p
store-buffer,bool,owns,rels,shared) global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{\text {sbh }}{ }^{*}-[60,60] 100\right)$
where
sbh-concurrent-steps $\equiv$ sbh-concurrent-step ${ }^{* *}$
term $\mathrm{x} \Rightarrow$ sbh $^{*} \mathrm{Y}$
end
lemma instrs-append-Read ${ }_{\text {bb }}$ :
instrs $\left(\operatorname{sb} @\left[\operatorname{Read}_{\text {sb }}\right.\right.$ volatile a t v]) $=$ instrs sb @ [Read volatile a t]
by (induct sb) (auto split: memref.splits)
lemma instrs-append-Write ${ }_{\text {sb }}$ :
instrs $\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W]) $=$ instrs sb @ $[$ Write volatile a sop A L
R W]
by (induct sb) (auto split: memref.splits)
lemma instrs-append-Ghost ${ }_{\text {sb }}$ :
instrs $\left(\mathrm{sb} @\left[\right.\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A} L \mathrm{R} \mathrm{W}\right]\right)=$ instrs sb @ [Ghost A L R W]
by (induct sb) (auto split: memref.splits)
lemma prog-instrs-append-Ghost ${ }_{\text {sb }}$ :
prog-instrs (sb@[Ghostsb A L R W]) $=$ prog-instrs sb
by (induct sb) (auto split: memref.splits)
lemma prog-instrs-append-Read ${ }_{\text {sb }}$ :
$\operatorname{prog}-$ instrs $\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v$\left.]\right)=\operatorname{prog}$-instrs sb by (induct sb) (auto split: memref.splits)
lemma prog-instrs-append-Write ${ }_{\text {sb }}$ :
prog-instrs (sb@[Write ${ }_{\text {sb }}$ volatile a sop v A L R W]) $=$ prog-instrs sb
by (induct sb) (auto split: memref.splits)
lemma hd-prog-append-Read ${ }_{\text {sb }}$ :

by (induct sb) (auto split: memref.splits)
lemma hd-prog-append-Write ${ }_{\text {sb }}$ :
hd-prog $\mathrm{p}\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W] $)=$ hd-prog p sb
by (induct sb) (auto split: memref.splits)
lemma flush-update-other: $\wedge \mathrm{m}$. a $\notin$ outstanding-refs (Not $\circ$ is-volatile) sb $\Longrightarrow$
outstanding-refs (is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}=\{ \} \Longrightarrow$
flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$
by (induct sb)
(auto split: memref.splits if-split-asm simp add: fun-upd-twist)
lemma flush-update-other': $\wedge$ m. a $\notin$ outstanding-refs (is-non-volatile-Write ${ }_{\text {sb }}$ ) sb $\Longrightarrow$ outstanding-refs (is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}=\{ \} \Longrightarrow$ flush sb $(\mathrm{m}(\mathrm{a}:=\mathrm{v}))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$
by (induct sb)
(auto split: memref.splits if-split-asm simp add: fun-upd-twist)
lemma flush-update-other ${ }^{\prime \prime}: ~ \bigwedge \mathrm{~m}$. a $\notin$ outstanding-refs (is-non-volatile-Write ${ }_{\text {sb }}$ ) sb $\Longrightarrow$ $\mathrm{a} \notin$ outstanding-refs (is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb} \Longrightarrow$
flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$
by (induct sb)
(auto split: memref.splits if-split-asm simp add: fun-upd-twist)
lemma flush-all-until-volatile-write-update-other:
$\bigwedge \mathrm{m} . \forall \mathrm{j}<$ length ts.

$$
\begin{aligned}
& \text { (let }\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j} \\
& \quad \text { in a } \notin \text { outstanding-refs is-non-volatile-Write }{ }_{\text {sb }} \text { (takeWhile (Not o }
\end{aligned}
$$

is-volatile- Write $_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ )
$\Longrightarrow$
flush-all-until-volatile-write ts $(\mathrm{m}(\mathrm{a}:=\mathrm{v}))=$
(flush-all-until-volatile-write ts m$)(\mathrm{a}:=\mathrm{v})$
proof (induct ts)
case Nil thus ?case
by simp
next
case (Cons t ts)
note notin $=\langle\forall \mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts})$.
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=(\mathrm{t} \# \mathrm{ts})!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile-Write $\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}$ )) >
hence notin': $\forall \mathrm{j}<$ length ts.
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile-Write $\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{j}}$ ))
by auto
obtain $\mathrm{p}_{\mathrm{l}}$ is $\mathcal{O}_{\mathrm{l}} \mathcal{R}_{\mid} \mathcal{D}_{\mathrm{l}} \mathrm{xs} \mathrm{xb}_{\mid}$where $\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{l}}, \mathrm{is}_{\mathrm{l}}, \mathrm{xs}_{\mathrm{l}}, \mathrm{sb}_{\mid}, \mathcal{D}_{\mathrm{l}}, \mathcal{O}_{\mathrm{l}}, \mathcal{R}_{\mathrm{l}}\right)$
by (cases t )
have no-write:
outstanding-refs (is-volatile-Write $\left._{\text {sb }}\right)\left(\right.$ takeWhile (Not $\circ$ is-volatile-Write $\left._{\text {sb }}\right)$ sb $\left._{\boldsymbol{l}}\right)=\{ \}$
by (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
from notin [rule-format, of 0] t
have a-notin:
$\mathrm{a} \notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb sb $_{\boldsymbol{l}}$ ) by (auto)
from flush-update-other ${ }^{\prime}$ [OF a-notin no-write]
have (flush (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{l}}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v}))\right)=$
(flush (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) $\left.\left.\mathrm{sb}_{\mathrm{l}}\right) \mathrm{m}\right)(\mathrm{a}:=\mathrm{v})$.
with Cons.hyps [OF notin', of (flush (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sblı ${ }^{\prime}$ ) )
show ?case
by (simp add: t del: fun-upd-apply)
qed
lemma flush-all-until-volatile-write-append-non-volatile-write-commute:
assumes no-volatile-Write ${ }_{\mathbf{s b}}$ : outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb}=\{ \}$
shows $\bigwedge \mathrm{m}$ i. $\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$;
$\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)$ )】
$\Longrightarrow$ flush-all-until-volatile-write ( $\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}\right.\right.$, xs, sb @ [Write ${ }_{\text {sb }}$ False a sop v A L R $\left.\left.\left.\mathrm{W}], \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right) \mathrm{m}=$ (flush-all-until-volatile-write ts m$)(\mathrm{a}:=\mathrm{v})$
proof (induct ts)
case Nil thus ?case
by simp
next
case (Cons t ts)
note i -bound $=\langle\mathrm{i}<$ length ( $\mathrm{t} \# \mathrm{ts}$ ) $\rangle$
note ith $=\langle(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
note notin $=\langle\forall \mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) . \mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=(\mathrm{t} \# \mathrm{ts})!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o is-volatile-Write $\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}$ )) ,
show ?case
proof (cases i)
case 0
from ith 0 have $\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by simp
from no-volatile-Write ${ }_{\text {sb }}$ have take-all: takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}=\mathrm{sb}$ by (auto simp add: outstanding-refs-conv)
from no-volatile-Write ${ }_{\text {sb }}$
have take-all': takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) (sb @ [Write ${ }_{\text {sb }}$ False a sop v A L $\mathrm{R} \mathrm{W}])=$
(sb @ [Write ${ }_{\text {sb }}$ False a sop v A L R W])
by (auto simp add: outstanding-refs-conv)
from notin
have $\forall \mathrm{j}<$ length ts.
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile-Write ${ }_{s b}$ ) $\operatorname{sb}_{\mathrm{j}}$ ))
by (auto simp add: 0)

## from flush-all-until-volatile-write-update-other [OF this] show ?thesis

by (simp add: 0 t take-all' take-all flush-append-write del: fun-upd-apply)
next
case (Suc n)
obtain $\mathrm{p}_{\mathrm{l}}$ is $\mathcal{O}_{\mathrm{l}} \mathcal{R}_{\mid} \mathcal{D}_{\mathrm{l}} \mathrm{xs} \mid \mathrm{sb}_{\mid}$where $\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{l}}, \mathrm{is}_{\mathrm{l}}, \mathrm{xs}_{\mid}, \mathrm{sb}_{\mathrm{l}}, \mathcal{D}_{\mathrm{l}}, \mathcal{O}_{\mathrm{l}}, \mathcal{R}_{\mid}\right)$
by (cases t)
from i-bound ith notin
have flush-all-until-volatile-write

```
                (ts[n := (p',is',xs, sb @ [Write sb False a sop v A L R W], D
                    (flush (takeWhile (Not \circ is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{|}{})\textrm{m})
                    (flush-all-until-volatile-write ts
                    (flush (takeWhile (Not O is-volatile-Write sb) sb la m))
                    (a := v)
apply -
apply (rule Cons.hyps)
apply (auto simp add: Suc simp del: o-apply)
done
```


## then

```
show ?thesis
        by (simp add: t Suc del: fun-upd-apply)
    qed
qed
lemma flush-all-until-volatile-write-append-unflushed:
    assumes volatile-Write }\mp@subsup{}{\mathbf{sb}}{}:\neg\mp@subsup{\mathrm{ outstanding-refs is-volatile-Write }}{\mathbf{sb}}{}\textrm{sb}={
    shows \m i. \llbracketi < length ts; ts!i = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
```



```
    (flush-all-until-volatile-write ts m)
proof (induct ts)
    case Nil thus ?case
    by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts)>
    note ith = <(l#ts)!i = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    show ?case
    proof (cases i)
    case 0
    from ith 0 have l: l = (p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by simp
    from volatile-Write 
    obtain r where r-in: r \in set sb and volatile-r: is-volatile-Write sb r
        by (auto simp add: outstanding-refs-conv)
    from takeWhile-append1 [OF r-in, of (Not o is-volatile-Write sb)] volatile-r
    have (flush (takeWhile (Not ○ is-volatile-Write e
                (flush (takeWhile (Not o is-volatile-Writesb) sb ) m)
        by auto
    then
    show ?thesis
        by (simp add: 0 l)
    next
    case (Suc n)
```



```
        by (cases l)
    from Cons.hyps [of n] i-bound ith
```

```
        show ?thesis
            by (simp add: 1 Suc)
    qed
qed
```

lemma flush-all-until-volatile-nth-update-unused:
shows $\bigwedge \mathrm{m}$ i. $\llbracket \mathrm{i}<$ length ts; ts $!i=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$
$\Longrightarrow$ flush-all-until-volatile-write $\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is $\left.\left.\left.^{\prime}, \vartheta^{\prime}, \operatorname{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right) \mathrm{m}=$ (flush-all-until-volatile-write ts m)
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons l ts)
note i -bound $=$ $\mathrm{i}<$ length $(\mathrm{l} \# \mathrm{ts})$ 〉
note ith $=\langle(\mathrm{l} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
show ?case
proof (cases i)
case 0
from ith 0 have $l: l=(p, i s, \vartheta, s b, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by $\operatorname{simp}$
show ?thesis
by (simp add: 0 l)
next
case (Suc n)

by (cases l)
from Cons.hyps [of n] i-bound ith
show ?thesis
by (simp add: l Suc)
qed
qed
lemma flush-all-until-volatile-write-append-volatile-write-commute:
assumes no-volatile-Write ${ }_{\mathbf{s b}}$ : outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb}=\{ \}$
shows $\bigwedge \mathrm{m}$ i. $\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow$
flush-all-until-volatile-write
$\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is $^{\prime}, \vartheta \vartheta$, sb @ $\left[\mathrm{Write}_{\text {sb }}\right.$ True a sop v A L R W], $\left.\left.\left.\mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right) \mathrm{m}$
$=$ flush-all-until-volatile-write ts m
proof (induct ts)
case Nil thus ?case
by simp
next
case (Cons lts)
note i-bound $=$ i $<$ length ( $\mathrm{l} \# \mathrm{ts}$ ) $\rangle$
note ith $=\langle(\mathrm{l} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
show ?case
proof (cases i)
case 0

```
    from ith 0 have l: l = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by simp
    from no-volatile-Write sb
    have s1: takeWhile (Not \circ is-volatile-Write esb) sb = sb
        by (auto simp add: outstanding-refs-conv)
    from no-volatile-Write sb
    have s2: (takeWhile (Not o is-volatile-Write esb) (sb @ [Write sb True a sop v A L R W]))
= sb
        by (auto simp add: outstanding-refs-conv)
    show ?thesis
        by (simp add: 0 l s1 s2)
    next
    case (Suc n)
```



```
        by (cases l)
    from Cons.hyps [of n] i-bound ith
    show ?thesis
        by (simp add: 1 Suc)
    qed
qed
lemma reads-consistent-update:
    \pending-write }\mathcal{O}\textrm{m}.\mathrm{ reads-consistent pending-write }\mathcal{O}\textrm{m sb}
    a }\not\in\mathrm{ outstanding-refs (Not ० is-volatile) sb }
    reads-consistent pending-write }\mathcal{O}(\textrm{m}(\textrm{a}:=\textrm{v}))\textrm{sb
apply (induct sb)
apply simp
apply (clarsimp split: memref.splits if-split-asm
    simp add: fun-upd-twist)
subgoal for sb }\mathcal{O}\textrm{m}\mathrm{ x11 addr val A R pending-write
apply (case-tac a=addr)
apply simp
apply (fastforce simp add: fun-upd-twist)
done
done
lemma (in program) history-consistent-hd-prog: \p. history-consistent \vartheta p' xs
\Longrightarrow \text { history-consistent v (hd-prog p xs) xs}
apply (induct xs)
apply simp
apply (auto split: memref.splits option.splits)
done
locale valid-program = program +
    fixes valid-prog
    assumes valid-prog-inv: \llbracket\vartheta\vdashp }\mp@subsup{->}{\textrm{p}}{}(\mp@subsup{\textrm{p}}{}{\prime},\textrm{is}');\mathrm{ valid-prog p}\rrbracket\Longrightarrow valid-prog p'
```

```
lemma (in valid-program) history-consistent-appendD:
 y ys p. }\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop }
                read-tmps xs \cap read-tmps ys ={}\Longrightarrow
            history-consistent \vartheta p (xs@ys) \Longrightarrow
                        (history-consistent (\vartheta\mp@subsup{|}{}{`}(\operatorname{dom}\vartheta-\operatorname{read-tmps ys)) p xs }\wedge
            history-consistent \vartheta (last-prog p xs) ys }
            read-tmps ys \capU(fst' write-sops xs)={})
proof (induct xs)
    case Nil thus ?case
        by auto
next
case (Cons x xs)
note valid-sops = \\forallsop\inwrite-sops (x # xs). valid-sop sop`
note read-tmps-dist = rread-tmps (x#xs) \cap read-tmps ys ={}`
note consis = 〈history-consistent \vartheta p ((x#xs)@ys)〉
show ?case
proof (cases x)
    case (Write sb volatile a sop v)
    obtain D f where sop: sop=(D,f)
        by (cases sop)
    from consis obtain
            D-tmps: D\subseteqdom \vartheta and
            f-v: f \vartheta = v and
            D-read-tmps: D \cap read-tmps (xs @ ys) = {} and
            consis': history-consistent \vartheta p (xs @ ys)
            by (simp add: Write sb sop)
    from valid-sops obtain
            valid-Df: valid-sop (D,f) and
            valid-sops': }\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop
            by (auto simp add: Write sb sop)
    from valid-Df
    interpret valid-sop (D,f) .
    from read-tmps-dist have read-tmps-dist': read-tmps xs \cap read-tmps ys ={}
            by (simp add: Writesb)
        from D-read-tmps have D-ys: D \cap read-tmps ys ={}
        by (auto simp add: read-tmps-append)
    with D-tmps have D-subset: D\subseteqdom \vartheta - read-tmps ys
        by auto
    moreover
    from valid-sop [OF refl D-tmps]
    have f \vartheta = f (\vartheta '` D).
    moreover
    let ?\vartheta'=\vartheta 滴(dom \vartheta - read-tmps ys)
    from D-subset
    have ?\vartheta' |}\mp@subsup{|}{}{\prime}\textrm{D}=\vartheta\mp@subsup{|}{}{\prime}\textrm{D
        apply -
        apply (rule ext)
```

```
    by (auto simp add: restrict-map-def)
    moreover
    from D-subset
    have D-tmps': D \subseteq dom ?\vartheta'
        by auto
    ultimately
    have f-v': f ? ?' = v
        using valid-sop [OF refl D-tmps'] f-v
        by simp
    from D-read-tmps
    have D \cap read-tmps xs = {}
    by (auto simp add: read-tmps-append)
    with Cons.hyps [OF valid-sops' read-tmps-dist' consis'] D-tmps D-subset f-v' D-ys
    show ?thesis
    by (auto simp add: Write sb sop)
next
    case (Read sb volatile a t v)
    from consis obtain
    tmps-t: \vartheta t = Some v and
    consis': history-consistent v p (xs @ ys)
    by (simp add: Read sb split: option.splits)
    from read-tmps-dist
    obtain t-ys: t }\not=\mathrm{ read-tmps ys and read-tmps-dist': read-tmps xs }\cap\mathrm{ read-tmps ys ={}
    by (auto simp add: Readsb
    from valid-sops have valid-sops': }\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop
    by (auto simp add: Readsb)
    from t-ys tmps-t
    have (\vartheta | (dom \vartheta - read-tmps ys)) t = Some v
        by (auto simp add: restrict-map-def domIff)
    with Cons.hyps [OF valid-sops' read-tmps-dist' consis']
    show ?thesis
    by (auto simp add: Read sb
next
    case (Prog
    from consis obtain p}\mp@subsup{p}{1}{-p: p
    prog-step: \vartheta |
    consis': history-consistent \vartheta p2 (xs @ ys)
    by (auto simp add: Progsb)
let ?\mp@subsup{\vartheta}{}{\prime}=\vartheta |
```



```
    apply (rule ext)
    apply (auto simp add: read-tmps-append restrict-map-def domIff split: if-split-asm)
    done
```

from valid-sops have valid-sops': $\forall$ sop $\in$ write-sops xs. valid-sop sop
by (auto simp add: Prog $_{\text {sb }}$ )
from read-tmps-dist

```
    obtain read-tmps-dist': read-tmps xs \cap read-tmps ys ={}
        by (auto simp add: Progsb)
    from Cons.hyps [OF valid-sops' read-tmps-dist' consis'] p1-p prog-step eq
    show ?thesis
        by (simp add: Progsb)
    next
    case Ghostsb
    with Cons show ?thesis
        by auto
    qed
qed
lemma (in valid-program) history-consistent-appendI:
    \\vartheta ys p.}\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop }
    history-consistent (\vartheta|` (dom \vartheta - read-tmps ys)) p xs \Longrightarrow
history-consistent \vartheta (last-prog p xs) ys \Longrightarrow
read-tmps ys \capU(fst ' write-sops xs) ={} \Longrightarrow valid-prog p \Longrightarrow
                history-consistent \vartheta p (xs@ys)
proof (induct xs)
    case Nil thus ?case by simp
next
    case (Cons x xs)
    note valid-sops = \\forallsop\inwrite-sops (x # xs). valid-sop sop }
    note consis-xs = <history-consistent (\vartheta ` (dom \vartheta - read-tmps ys)) p (x # xs)>
    note consis-ys = \history-consistent \vartheta (last-prog p (x # xs)) ys`
    note dist = <read-tmps ys \cap U(fst ` write-sops (x # xs))={}`
    note valid-p = \valid-prog p `
    show ?case
    proof (cases x)
        case (Write esb volatile a sop v)
        obtain D f where sop: sop=(D,f)
        by (cases sop)
    from consis-xs obtain
            D-tmps: D \subseteq dom \vartheta - read-tmps ys and
        f-v: f (\vartheta |}(\textrm{dom}\vartheta-\textrm{read}-\textrm{tmps}ys))=v(\mathrm{ is f ? }\vartheta=v)\mathrm{ and
        D-read-tmps: D \cap read-tmps xs ={} and
        consis': history-consistent (\vartheta |' (dom \vartheta - read-tmps ys)) p xs
        by (simp add: Write sb sop)
    from D-tmps D-read-tmps
    have D \cap read-tmps (xs @ ys)={}
        by (auto simp add: read-tmps-append)
    moreover
    from D-tmps have D-tmps': D}\subseteq\operatorname{dom}
        by auto
moreover
from valid-sops obtain
    valid-Df: valid-sop (D,f) and
    valid-sops': }\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop
    by (auto simp add: Write sb sop)
```

```
from valid-Df
interpret valid-sop (D,f) .
from D-tmps
have tmps-eq: \vartheta |}((\mathrm{ dom }\vartheta-\mathrm{ read-tmps ys ) }\cap\textrm{D})=\vartheta\mp@subsup{|}{}{\prime}\textrm{D
    apply -
    apply (rule ext)
    apply (auto simp add: restrict-map-def)
    done
from D-tmps
have f ?\vartheta = f (?\vartheta |}\textrm{D}
    apply -
    apply (rule valid-sop [OF refl ])
    apply auto
    done
with valid-sop [OF refl D-tmps] f-v D-tmps
have f \vartheta=v
    by (clarsimp simp add: tmps-eq)
moreover
from consis-ys have consis-ys': history-consistent \vartheta (last-prog p xs) ys
    by (auto simp add: Write sb)
from dist have dist': read-tmps ys \cap (fst ' write-sops xs)={}
    by (auto simp add: Write sb)
moreover note Cons.hyps [OF valid-sops' consis' consis-ys' dist' valid-p]
ultimately show ?thesis
    by (simp add: Write sb sop)
next
    case (Read sb volatile a t v)
    from consis-xs obtain
        t-v: (\vartheta | (dom \vartheta - read-tmps ys)) t = Some v and
        consis-xs': history-consistent (\vartheta |}\mathrm{ '(dom }\vartheta\mathrm{ - read-tmps ys)) p xs
        by (clarsimp simp add: Readsb split: option.splits)
from t-v have \vartheta t = Some v
    by (auto simp add: restrict-map-def split: if-split-asm)
moreover
from valid-sops obtain
    valid-sops': }\forall\mathrm{ sop }\in\mathrm{ write-sops xs. valid-sop sop
    by (auto simp add: Readsb)
from consis-ys have consis-ys': history-consistent \vartheta (last-prog p xs) ys
    by (auto simp add: Readsb)
from dist have dist': read-tmps ys \cap \(fst` write-sops xs)={}
    by (auto simp add: Readsb
note Cons.hyps [OF valid-sops' consis-xs' consis-ys' dist' valid-p]
ultimately
show ?thesis
```

```
    by (simp add: Read
next
    case (Progsb p1 p
    let ?\vartheta =\vartheta |}(\mathrm{ (dom v - read-tmps ys)
    from consis-xs obtain
        p1-p: p
        prog-step: ?\vartheta |` (dom ?\vartheta - read-tmps xs)\vdash p p }\mp@subsup{->}{\textrm{p}}{(}\mathrm{ (p2, mis) and
        consis': history-consistent ?\vartheta p}\mp@subsup{\textrm{p}}{2}{}\times
        by (auto simp add: Progsb)
```

    have eq: ? \(\left.\right|^{\bullet}(\operatorname{dom} ? \vartheta-\operatorname{read}-\mathrm{tmps} x s)=\left.\vartheta\right|^{\circ}(\operatorname{dom} \vartheta-\operatorname{read}-\mathrm{tmps}(\mathrm{xs} @ y s))\)
        apply (rule ext)
        apply (auto simp add: read-tmps-append restrict-map-def domIff split: if-split-asm)
        done
    from prog-step eq
    have \(\left.\vartheta\right|^{\cdot}\left(\operatorname{dom} \vartheta-\right.\) read-tmps (xs @ ys)) \(\vdash \mathrm{p}_{1} \rightarrow_{\mathrm{p}}\left(\mathrm{p}_{2}\right.\), mis \()\) by simp
    moreover
    from valid-sops obtain
        valid-sops': \(\forall\) sop \(\in\) write-sops xs. valid-sop sop
        by (auto simp add: Progsb \(_{\text {sb }}\) )
    from consis-ys have consis-ys': history-consistent \(\vartheta\) (last-prog \(\mathrm{p}_{2} \mathrm{xs}\) ) ys
        by (auto simp add: Prog \(_{\text {sb }}\) )
    from dist have dist': read-tmps ys \(\cap \bigcup\) (fst ' write-sops xs) \(=\{ \}\)
        by (auto simp add: Progsb \(_{\text {sb }}\) )
    note Cons.hyps [OF valid-sops' \({ }^{\prime}\) consis' \({ }^{\prime}\) consis-ys \({ }^{\prime}\) dist \(^{\prime}\) valid-prog-inv [OF prog-step
    valid-p [simplified $\mathrm{p}_{1}-\mathrm{p}[$ symmetric $\left.\left.\left.]\right]\right]\right]$
ultimately
show ?thesis
by (simp add: Prog $_{\text {sb }} \mathrm{p}_{1}-\mathrm{p}$ )
next
case Ghost $_{\text {sb }}$
with Cons show ?thesis
by auto
qed
qed
lemma (in valid-program) history-consistent-append-conv:
$\Lambda \vartheta$ ys $\mathrm{p} . \forall$ sop $\in$ write-sops xs. valid-sop sop $\Longrightarrow$
read-tmps xs $\cap$ read-tmps ys $=\{ \} \Longrightarrow$ valid-prog $p \Longrightarrow$
history-consistent $\vartheta \mathrm{p}(\mathrm{xs} @ y s)=$
(history-consistent ( $\left.\vartheta\right|^{\bullet}($ dom $\vartheta$ - read-tmps ys)) p xs $\wedge$
history-consistent $\vartheta$ (last-prog p xs) ys $\wedge$ read-tmps ys $\cap \bigcup$ (fst ' write-sops xs) $=\{ \}$ )
apply rule
apply (rule history-consistent-appendD,assumption+)
apply (rule history-consistent-appendI)
apply auto
done
lemma instrs-takeWhile-dropWhile-conv:
instrs xs $=$ instrs (takeWhile P xs) @ instrs (dropWhile P xs)
by (induct xs) (auto split: memref.splits)
lemma (in program) history-consistent-hd-prog-p:
$\wedge$ p. history-consistent $\vartheta \mathrm{p}$ xs $\Longrightarrow \mathrm{p}=$ hd-prog p xs
by (induct xs) (auto split: memref.splits option.splits)
lemma instrs-append: $\bigwedge$ ys. instrs (xs@ys) = instrs xs @ instrs ys
by (induct xs) (auto split: memref.splits)
lemma prog-instrs-append: $\bigwedge$ ys. prog-instrs (xs@ys) = prog-instrs xs @ prog-instrs ys by (induct xs) (auto split: memref.splits)
lemma prog-instrs-empty: $\forall \mathrm{r} \in$ set xs. $\neg$ is- $\operatorname{Prog}_{s b} \mathrm{r} \Longrightarrow$ prog-instrs $\mathrm{xs}=[]$
by (induct xs) (auto split: memref.splits)
lemma length-dropWhile [termination-simp]: length (dropWhile P xs) $\leq$ length xs by (induct xs) auto
lemma prog-instrs-filter-is-Prog ${ }_{s b}$ : prog-instrs (filter (is-Prog ${ }_{s b}$ ) xs) $=$ prog-instrs xs by (induct xs) (auto split: memref.splits)
lemma Cons-to-snoc: $\wedge \mathrm{x}$. ヨys y. $(\mathrm{x} \# \mathrm{xs})=(\mathrm{ys} @[\mathrm{y}])$
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x1 xs)
from Cons [of x1] obtain ys y where $\mathrm{x} 1 \# \mathrm{xs}=\mathrm{ys}$ @ $[\mathrm{y}$ ]
by auto
then
show ?case by simp
qed
lemma causal-program-history-Read:
assumes causal-Read: causal-program-history (Read volatile at \# is $\mathrm{sb}_{\mathrm{sb}}$ ) sb
shows causal-program-history is $\mathrm{s}_{\mathrm{sb}}\left(\mathrm{sb}\right.$ @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v])
proof
fix $\mathrm{sb}_{1} \mathrm{sb}_{2}$
assume sb: sb @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v] $=\mathrm{sb}_{1} @ \mathrm{sb}_{2}$
from causal-Read
interpret causal-program-history Read volatile at $\#$ is s $_{\text {sb }}$ sb .
show $\exists$ is. instrs $\mathrm{sb}_{2} @$ is $\mathrm{sb}=$ is @ prog-instrs sb ${ }_{2}$
proof (cases $\mathrm{sb}_{2}$ )

```
    case Nil
    thus ?thesis
        by simp
    next
    case (Cons r sb)
    from Cons-to-snoc [of r sb ] Cons obtain ys y where sb 2-snoc: sb }\mp@subsup{\mp@code{F}}{2}{}=\textrm{ys}@[y
        by auto
    with sb obtain y: y = Readsb volatile a t v and sb: sb = sb
        by simp
    from causal-program-history [OF sb] obtain is where
        instrs ys @ Read volatile a t # is isb
        by auto
    then show ?thesis
        by (simp add: sb }\mp@subsup{\mp@code{2}}{2}{}\mathrm{ -snoc y instrs-append prog-instrs-append)
    qed
qed
lemma causal-program-history-Write:
assumes causal-Write: causal-program-history (Write volatile a sop A L R W\# is \(\mathrm{sb}_{\mathrm{sb}}\) ) sb shows causal-program-history is sb \(\left(\mathrm{sb}\right.\) @ \(\left[\mathrm{Write}_{\text {sb }}\right.\) volatile a sop v A L R W])
proof
    fix sb
    assume sb: sb @ [Write sb volatile a sop v A L R W] = sb @ @ sb 
    from causal-Write
    interpret causal-program-history Write volatile a sop A L R W# is sb sb .
    show \exists is. instrs sb }\mp@subsup{\mp@code{D}}{2}{@ is isb
    proof (cases sb 2)
        case Nil
        thus ?thesis
            by simp
    next
        case (Cons r sb')
        from Cons-to-snoc [of r sb
            by auto
        with sb obtain y: y = Writesb volatile a sop v A L R W and sb: sb = sb @@ys
            by simp
        from causal-program-history [OF sb] obtain is where
            instrs ys @ Write volatile a sop A L R W# is sb = is @ prog-instrs ys
            by auto
        then show ?thesis
            by (simp add: sb 2-snoc y instrs-append prog-instrs-append)
    qed
qed
lemma causal-program-history- \(\operatorname{Prog}_{s b}\) :
assumes causal-Write: causal-program-history is stb \(_{\text {sb }}\)
shows causal-program-history (is sb @mis) (sb @ [ \(\left.\operatorname{Prog}_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\right]\) )
```

```
proof
    fix sb
    assume sb: sb @ [Progsb p p p p mis]= sb
    from causal-Write
    interpret causal-program-history is }\mp@subsup{\textrm{s}}{\textrm{sb}}{}\textrm{sb}
```



```
    proof (cases sb }\mp@subsup{)}{2}{}\mathrm{ )
        case Nil
        thus ?thesis
        by simp
    next
        case (Cons r sb')
```



```
            by auto
        with sb obtain y: y = = Prog}\mp@subsup{\mp@code{sb}}{}{\prime}\mp@subsup{p}{1}{}\mp@subsup{p}{2}{}\mathrm{ mis and sb: sb = sb
            by simp
        from causal-program-history [OF sb] obtain is where
            instrs ys @ (is isb @ mis) = is @ prog-instrs (ys@[Prog
            by (auto simp add: prog-instrs-append)
    then show ?thesis
            by (simp add: sb }\mp@subsup{\mp@code{2}}{2}{}\mathrm{ -snoc y instrs-append prog-instrs-append)
    qed
qed
lemma causal-program-history-Ghost:
    assumes causal-Ghostsb: causal-program-history (Ghost A L R W # is crs) sb
    shows causal-program-history is sb (sb @ [Ghost sb A L R W])
proof
    fix sb
    assume sb: sb @ [Ghost sb A L R W] = sb @ @ sb 
    from causal-Ghostsb
    interpret causal-program-history Ghost A L R W # is sb sb .
    show \exists is. instrs sb }\mp@subsup{\mp@code{N}}{2}{@ is}\mp@subsup{\textrm{s}}{\mathbf{s}}{}=\mathrm{ is @ prog-instrs sb
    proof (cases sb 2)
        case Nil
        thus ?thesis
            by simp
    next
    case (Cons r sb}
    from Cons-to-snoc [of r sb ] Cons obtain ys y where sb 2-snoc: sb }\mp@subsup{\mp@code{N}}{2}{}=ys@[y
        by auto
    with sb obtain y: y = Ghostsb A L R W and sb: sb = sb @@ys
                by simp
    from causal-program-history [OF sb] obtain is where
        instrs ys @ Ghost A L R W # is isb = is @ prog-instrs ys
        by auto
    then show ?thesis
        by (simp add: sb}\mp@subsup{\mp@code{2}}{2}{}\mathrm{ -snoc y instrs-append prog-instrs-append)
```

qed
qed
lemma hd-prog-last-prog-end: $\llbracket \mathrm{p}=\mathrm{hd}$-prog p sb; last-prog $\mathrm{p} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}} \rrbracket \Longrightarrow \mathrm{p}=$ hd-prog $\mathrm{p}_{\text {sb }} \mathrm{sb}$
by (induct sb) (auto split: memref.splits)
lemma hd-prog-idem: hd-prog (hd-prog p xs ) xs $=$ hd-prog p xs by (induct xs) (auto split: memref.splits)
lemma last-prog-idem: last-prog (last-prog p sb) sb $=$ last-prog p sb by (induct sb) (auto split: memref.splits)
lemma last-prog-hd-prog-append:
last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)$ ) sb = last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}$ ) sb
apply (induct sb)
apply (auto split: memref.splits)
done
lemma last-prog-hd-prog: last-prog (hd-prog $\mathrm{p} x s) \mathrm{xs}=\operatorname{last-prog} \mathrm{p} \mathrm{xs}$
by (induct xs) (auto split: memref.splits)
lemma last-prog-append-Read ${ }_{s b}$ :
$\bigwedge \mathrm{p}$. last-prog p (sb @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v$\left.]\right)=\operatorname{last-prog} \mathrm{p} \mathrm{sb}$
by (induct sb) (auto split: memref.splits)
lemma last-prog-append-Write ${ }_{\text {sb }}$ :
$\bigwedge$ p. last-prog p (sb @ [Write ${ }_{\text {sb }}$ volatile a sop v A L R W]) = last-prog p sb
by (induct sb) (auto split: memref.splits)
lemma last-prog-append-Prog ${ }_{\text {sb }}$ :
$\Lambda \mathrm{x}$. last-prog $\mathrm{x}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime} \mathrm{mis}\right]\right)=\mathrm{p}^{\prime}$
by (induct sb) (auto split: memref.splits)
lemma hd-prog-append- $\operatorname{Prog}_{s b}:$ hd-prog $\mathrm{x}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime} \mathrm{mis}\right]\right)=$ hd-prog p sb by (induct sb) (auto split: memref.splits)
lemma hd-prog-last-prog-append- $\operatorname{Prog}_{s b}$ :
$\bigwedge \mathrm{p}^{\prime}$. hd-prog $\mathrm{p}^{\prime} \mathrm{xs}=\mathrm{p}^{\prime} \Longrightarrow$ last-prog $\mathrm{p}^{\prime} \mathrm{xs}=\mathrm{p}_{1} \Longrightarrow$ $h d-\operatorname{prog} \mathrm{p}^{\prime}\left(\mathrm{xs} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{1} \mathrm{p}_{2} \operatorname{mis}\right]\right)=\mathrm{p}^{\prime}$
apply (induct xs)
apply (auto split: memref.splits)
done
lemma hd-prog-append-Ghostsb:
hd-prog p (sb@[Ghost $\operatorname{sb}_{\text {sb }}$ A R L W]) = hd-prog p sb
by (induct sb) (auto split: memref.splits)
lemma last-prog-append-Ghost ${ }_{s b}$ :
^p. last-prog p (sb @ [Ghostsb A L R W]) = last-prog p sb
by (induct sb) (auto split: memref.splits)
lemma dropWhile-all-False-conv:
$\forall \mathrm{x} \in$ set $\mathrm{xs} . \neg \mathrm{P} \mathrm{x} \Longrightarrow$ dropWhile $\mathrm{P} \mathrm{xs}=\mathrm{xs}$
by (induct xs) auto
lemma dropWhile-append-all-False:
$\forall \mathrm{y} \in$ set ys. $\neg \mathrm{P}$ y $\Longrightarrow$ dropWhile P (xs@ys) = dropWhile P xs @ ys
apply (induct xs)
apply (auto simp add: dropWhile-all-False-conv)
done
lemma reads-consistent-append-first:
$\bigwedge \mathrm{m}$ ys. reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (xs @ ys) $\Longrightarrow$ reads-consistent pending-write $\mathcal{O} \mathrm{m}$ xs
by (clarsimp simp add: reads-consistent-append)
lemma reads-consistent-takeWhile:
assumes consis: reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb}$
shows reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (takeWhile P sb)
using reads-consistent-append [where $\mathrm{xs}=($ takeWhile P sb$)$ and ys=(dropWhile $\mathrm{P} s b)$ ]

## consis

apply (simp add: reads-consistent-append)
done
lemma flush-flush-all-until-volatile-write-Write ${ }_{\text {sb }}$-volatile-commute:
$\bigwedge \mathrm{i} \mathrm{m} . \llbracket \mathrm{i}<$ length ts; $\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \mathrm{xs}\right.$, Write $_{\text {sb }}$ True a sop v A L R W\#sb, $\left.\mathcal{D}, \mathcal{O}, \mathcal{R}\right) ;$
$\forall \mathrm{i}<$ length ts. $(\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$

$$
\begin{gathered}
\left(\operatorname{let}\left(-,-,-, \mathrm{sb}_{\mathrm{i}},-,-,-\right)=\mathrm{ts}!\mathrm{i} ;\right. \\
\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}
\end{gathered}
$$

in outstanding-refs is-Write ${ }_{s b} \mathrm{sb}_{\mathrm{i}} \cap$
outstanding-refs is-Write $_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $=$ $\})$ );
$\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=$ ts $!\mathrm{j}$ in a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile- $W_{r i t e}^{s b}$ ) $\left.\mathrm{sb}_{\mathrm{j}}\right)$ )】

$$
\Longrightarrow
$$

flush (takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right) \mathrm{sb}$ )
$(($ flush-all-until-volatile-write ts m$)(\mathrm{a}:=\mathrm{v}))=$
flush-all-until-volatile-write ( $\left.\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$
$(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$
proof (induct ts)
case Nil thus ?case

```
    by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts)`
    note ith = <(l#ts)!i = (p,is,xs,Write sb True a sop v A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    note disj = & | < length (l#ts). ( }\textrm{|j}<<l\mp@code{length (l#ts). i = j }
                (let (-,-,-,sbi,-,-,--) = (l#ts)!i;
                            (-,-,-,sbj,-,-,-)=(l#ts)!j
            in outstanding-refs is-Write sb sb;
                outstanding-refs is-Write sb (takeWhile (Not o is-volatile-Write sb
{}))>
    note a-notin = \forall j < length (l#ts). i 
                (let (-,,-,,s\mp@subsup{b}{j}{j},-,,-) = (l#ts)!j
                in a & outstanding-refs is-Write sb (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb 
show ?case
proof (cases i)
    case 0
    from ith 0 have l: l = (p,is,xs,Write sb True a sop v A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by simp
    have a-notin-ts:
            a #U((\lambda(-,-,-,sb,-,-,-). outstanding-refs is-Write sb
                            (takeWhile (Not o is-volatile-Write sb) sb))' set ts) (is a & ?U)
    proof
        assume a }\in\mathrm{ ?U
        from in-Union-image-nth-conv [OF this]
        obtain j p pis is }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ where
j-bound: j < length ts and
jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,}\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{XSj}
a-in-j: a }\in\mathrm{ outstanding-refs is-Write sb (takeWhile (Not o is-volatile-Write sb
by fastforce
    from a-notin [rule-format, of Suc j] j-bound 0 a-in-j
    show False
by (auto simp add: jth)
    qed
    from a-notin-ts
    have (flush-all-until-volatile-write ts m)(a := v) =
                flush-all-until-volatile-write ts (m(a := v))
        apply -
    apply (rule update-commute' [where F={a} and G=?U and
g=flush-all-until-volatile-write ts])
    apply (auto intro: flush-all-until-volatile-wirte-mem-independent
            flush-all-until-volatile-write-unchanged-addresses)
    done
    moreover
    let ?SB = outstanding-refs is-Write }\mp@subsup{\mathbf{sb}}{\mathbf{sb}}{(takeWhile (Not \circ is-volatile-Write sb
    have U-SB-disj: ?U \cap ?SB = {}
```

```
    proof -
    {
fix a'
assume a'-in-U: a' }\in
have a' }\not\in\mathrm{ ?SB
proof
    assume a'-in-SB: a'}\in?\mathrm{ ?SB
    hence a'-in-SB': a' 
        apply (clarsimp simp add: outstanding-refs-conv)
        apply (drule set-takeWhileD)
        subgoal for x
        apply (rule-tac x=x in exI)
        apply (auto simp add: is-Write }\mp@subsup{\mathrm{ sb}}{\mathbf{b}}{}-\mathrm{ def split:memref.splits)
        done
        done
    from in-Union-image-nth-conv [OF a'-in-U]
    obtain j p pis is }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ where
        j-bound: j < length ts and
        jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{})\mathrm{ and
        a'-in-j: a' }\in\mathrm{ outstanding-refs is-Writesb (takeWhile (Not O is-volatile-Write sb
        by fastforce
    from disj [rule-format, of 0 Suc j] 0 j-bound a'-in-SB' a'-in-j jth l
    show False
    by auto
qed
    }
    moreover
    {
fix a'
assume a'-in-SB: a' }\in\mathrm{ ?SB
hence a'-in-SB': a' }\in\mathrm{ outstanding-refs is-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ sb
    apply (clarsimp simp add: outstanding-refs-conv)
    apply (drule set-takeWhileD)
    subgoal for x
    apply (rule-tac x=x in exI)
    apply (auto simp add: is-Write sb-def split:memref.splits)
    done
    done
have a' }\not\in
proof
    assume a'}\mp@subsup{a}{}{\prime}\in?
    from in-Union-image-nth-conv [OF this]
    obtain j pj is j}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ where
    j-bound: j < length ts and
    jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{\textrm{i}},\textrm{i}\mp@subsup{\textrm{S}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{\textrm{j}},\textrm{sb
```



```
    by fastforce
```

    from disj [rule-format, of 0 Suc j] j-bound \(a^{\prime}-i n-S B^{\prime} a^{\prime}-i n-j\) jth 1
    ```
    show False
    by auto
qed
    }
    ultimately
    show ?thesis by blast
    qed
    have flush (takeWhile (Not o is-volatile-Writesb) sb)
        (flush-all-until-volatile-write ts (m(a := v))) =
        flush-all-until-volatile-write ts
            (flush (takeWhile (Not ○ is-volatile-Write 
        apply (rule update-commute' [where g = flush-all-until-volatile-write ts ,
            OF --- - U-SB-disj])
    apply (auto intro: flush-all-until-volatile-wirte-mem-independent
        flush-all-until-volatile-write-unchanged-addresses
        flush-unchanged-addresses
        flushed-values-mem-independent simp del: o-apply)
    done
ultimately
have flush (takeWhile (Not o is-volatile-Write sb) sb)
            ((flush-all-until-volatile-write ts m)(a:=v)) =
            flush-all-until-volatile-write ts
            (flush (takeWhile (Not ○ is-volatile-Write sb}) sb) (m(a :=v)))
    by simp
    then show ?thesis
    by (auto simp add: l 0 o-def simp del: fun-upd-apply)
next
    case (Suc n)
    obtain pl| is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{|}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\textrm{xs}}{|}{}\mathrm{ sbl
    by (cases l)
from i-bound ith disj a-notin
have
    flush (takeWhile (Not \circ is-volatile-Write sb) sb)
        ((flush-all-until-volatile-write ts
            (flush (takeWhile (Not ○ is-volatile-Write 
            (a := v)) =
        flush-all-until-volatile-write (ts[n := (p,is, xs, sb,\mathcal{D}}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime})]
        ((flush (takeWhile (Not ○ is-volatile-Write 
    apply -
    apply (rule Cons.hyps)
    apply (force simp add: Suc Let-def simp del: o-apply)+
    done
moreover
```

```
    let ?SB = outstanding-refs is-Write sb (takeWhile (Not \circ is-volatile-Write sb
    have a & ?SB
    proof
    assume a }\in\mathrm{ ?SB
    with a-notin [rule-format, of 0]
    show False
by (auto simp add: 1 Suc)
    qed
    then
    have ((flush (takeWhile (Not \circ is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{sb}}{})\textrm{sb
            (flush (takeWhile (Not ० is-volatile-Write e
        apply -
        apply (rule update-commute' [where m=m and F={a} and G=?SB])
        apply (auto intro:
            flush-unchanged-addresses
            flushed-values-mem-independent simp del: o-apply)
        done
    ultimately
    show ?thesis
        by (simp add: 1 Suc del: fun-upd-apply o-apply)
    qed
qed
lemma (in program)
\sb' p. history-consistent \vartheta (hd-prog p (sb@sb')) (sb@sb')\Longrightarrow
    last-prog p (sb@sb})=p
    last-prog (hd-prog p (sb@sb')) sb = hd-prog p sb'
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons r sb 1)
    have consis: history-consistent \vartheta (hd-prog p ((r # sb
        by fact
    have last-prog: last-prog p ((r# sb }\mp@subsup{)}{1}{})@ sb')= p by fact
    show ?case
    proof (cases r)
        case Write sb with Cons show ?thesis by auto
    next
        case Read sb with Cons show ?thesis by (auto split: option.splits)
    next
        case (Prog
        from last-prog have last-prog-p2: last-prog p2 (sb 1 @ sb})=
```

by ( $\operatorname{simp}$ add: Prog $_{\text {sb }}$ )
from consis obtain consis': history-consistent $\vartheta \mathrm{p}_{2}\left(\mathrm{sb}_{1} @ \mathrm{sb}^{\prime}\right)$
by ( $\operatorname{simp}$ add: Prog $_{\text {sb }}$ )
hence history-consistent $\vartheta$ (hd-prog $\left.\mathrm{p}_{2}\left(\mathrm{sb}_{1} @ \mathrm{sb}^{\prime}\right)\right)\left(\mathrm{sb}_{1} @ \mathrm{sb}^{\prime}\right)$
by (rule history-consistent-hd-prog)
from Cons.hyps [OF this ]
have last-prog $\mathrm{p}_{2} \mathrm{sb}_{1}=$ hd-prog $\mathrm{p} \mathrm{sb}{ }^{\prime}$
oops
lemma last-prog-to-last-prog-same: $\bigwedge^{\prime} \mathrm{p}^{\prime}$. last-prog $\mathrm{p}^{\prime} \mathrm{sb}=\mathrm{p} \Longrightarrow$ last-prog p sb=p
by (induct sb) (auto split: memref.splits)
lemma last-prog-hd-prog-same: $\llbracket \operatorname{last}-\mathrm{prog} \mathrm{p}^{\prime} \mathrm{sb}=\mathrm{p} ;$ hd-prog $\mathrm{p}^{\prime} \mathrm{sb}=\mathrm{p} \rrbracket \Longrightarrow$ hd-prog p $\mathrm{sb}=\mathrm{p}^{\prime}$
by (induct sb) (auto split : memref.splits)
lemma last-prog-hd-prog-last-prog:
last-prog $\mathrm{p}^{\prime}\left(\operatorname{sb} @ \mathrm{sb}^{\prime}\right)=\mathrm{p} \Longrightarrow$ hd- $\operatorname{prog} \mathrm{p}^{\prime}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)=\mathrm{p}^{\prime} \Longrightarrow$ last-prog (hd-prog p sb') sb $=$ last-prog $\mathrm{p}^{\prime} \mathrm{sb}$
apply (induct sb)
apply (simp add: last-prog-hd-prog-same)
apply (auto split : memref.splits)
done
lemma (in program) last-prog-hd-prog-append':
$\bigwedge \mathrm{sb}^{\prime} \mathrm{p}$. history-consistent $\vartheta\left(\right.$ hd-prog $\left.\mathrm{p}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right)(\mathrm{sb@sb}) \Longrightarrow$
last-prog $\mathrm{p}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)=\mathrm{p} \Longrightarrow$
last-prog (hd-prog p sb') sb $=$ hd-prog $\mathrm{p} \mathrm{sb}^{\prime}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons r sb ${ }_{1}$ )
have consis: history-consistent $\vartheta$ (hd-prog p ((r \# sb ${ }_{1}$ ) @ sb')) ((r \# sb ${ }_{1}$ ) @ sb')
by fact
have last-prog: last-prog $\mathrm{p}\left(\left(\mathrm{r} \# \mathrm{sb}_{1}\right) @ \mathrm{sb}^{\prime}\right)=\mathrm{p}$ by fact
show ?case
proof (cases r)
case Write $_{\text {sb }}$ with Cons show ?thesis by auto
next
case Read $_{\text {sb }}$ with Cons show ?thesis by (auto split: option.splits)
next
case (Prog $_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2}$ is)
from last-prog have last-prog- $\mathrm{p}_{2}$ : last-prog $\mathrm{p}_{2}\left(\mathrm{sb}_{1} @ \mathrm{sb}^{\prime}\right)=\mathrm{p}$
by (simp add: Prog $_{\text {sb }}$ )
from last-prog-to-last-prog-same [OF this]
have last-prog-p: last-prog $\left.\mathrm{p}\left(\mathrm{sb}_{1} @ \mathrm{sb}\right)^{\prime}\right)=\mathrm{p}$.
from consis obtain consis': history-consistent $\vartheta \mathrm{p}_{2}\left(\mathrm{sb}_{1} @ \mathrm{sb}^{\prime}\right)$
by (simp add: Prog $_{\text {sb }}$ )
from history-consistent-hd-prog-p [OF consis']

```
    have hd-prog-p}\mp@subsup{p}{2}{}\mathrm{ : hd-prog p}\mp@subsup{p}{2}{}(\mp@subsup{sb}{1}{}@ @sb)= p p by simp
    from consis' have history-consistent \vartheta (hd-prog p (s\mp@subsup{b}{1}{}@ sb}\mp@subsup{)}{}{\prime})(\mp@subsup{\textrm{sb}}{1}{}@ @b'
        by (rule history-consistent-hd-prog)
    from Cons.hyps [OF this last-prog-p]
    have last-prog (hd-prog p sb}) sb 1 = hd-prog p sb'.
    moreover
    from last-prog-hd-prog-last-prog [OF last-prog-p2 hd-prog-p2]
    have last-prog (hd-prog p sb}) s\mp@subsup{b}{1}{}= last-prog p p sb i .
    ultimately
    have last-prog p}\mp@subsup{\textrm{p}}{2}{}\mp@subsup{\textrm{sb}}{1}{}=\mathrm{ hd-prog p sb'
        by simp
    thus ?thesis
        by (simp add: Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ )
    next
    case Ghostsb with Cons show ?thesis by (auto split: option.splits)
qed
qed
lemma flush-all-until-volatile-write-Write }\mp@subsup{}{\mathrm{ sb}}{}-non-volatile-commute:
    \i m. \llbracketi < length ts; ts!i=(p,is,xs,Write esb False a sop v A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R});
    i < length ts. ( }\forall\textrm{j}<l\mathrm{ length ts. i }\not=\textrm{j}
                    (let (-,,-,,s\mp@subsup{b}{i}{\prime},-,,-) = tsli;
                        (-,-,,,s\mp@subsup{b}{j}{j},-,-,-)=ts!j
                    in outstanding-refs is-Write sb sb
                            outstanding-refs is-Write sb (takeWhile (Not o is-volatile-Write sb
{}));
    j < length ts. i }=\textrm{j}
                (let (-,-,,-s\mp@subsup{b}{j}{\prime},,-,,-) = ts!j in a }\not=\mathrm{ outstanding-refs is-Write 
is-volatile-Write 
    \Longrightarrow flush-all-until-volatile-write (ts[i := (p,is, xs, sb,\mathcal{D}
                        flush-all-until-volatile-write ts m
proof (induct ts)
    case Nil thus ?case
        by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts) >
    note ith = <(l#ts)!i = (p,is,xs,Write sb False a sop v A L R W#sb, \mathcal{D ,O},\mathcal{R})
    note disj = \forall \foralli < length (l#ts). ( }\forall\textrm{j}<l=length ( l#ts). i = j \longrightarrow
                    (let (-,-,-,s\mp@subsup{b}{\textrm{i}}{\textrm{i}},-,-,-)}=(\textrm{l}#\textrm{ts})!i
                            (-,-,-,-sbj,-,-,-) = (l#ts)!j
                    in outstanding-refs is-Write sb sb
                                    outstanding-refs is-Write sb (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ ) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )=
{}))>
    note a-notin = \forall j < length (l#ts). i f j }
                            (let (-,,-,,s\mp@subsup{b}{j}{j,-,,,-) = (l#ts)!j}
    in a & outstanding-refs is-Write sb (takeWhile (Not ० is-volatile-Write sb
show ?case
proof (cases i)
    case 0
```

```
    from ith 0 have l: l = (p,is,xs,Write sb False a sop v A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R})
    by simp
    thus ?thesis
        by (simp add: 0 del: fun-upd-apply)
next
    case (Suc n)
```



```
        by (cases l)
    from i-bound ith disj a-notin
    have
        flush-all-until-volatile-write (ts[n := (p,is,xs, sb, D}\mp@subsup{\mathcal{D}}{}{\prime},\mathcal{O},\mp@subsup{\mathcal{R}}{}{\prime})]
            ((flush (takeWhile (Not o is-volatile-Write sb) sb l})\textrm{m})(\textrm{a}:=\textrm{v}))
        flush-all-until-volatile-write ts
            (flush (takeWhile (Not o is-volatile-Write sb) sb s) m)
        apply -
        apply (rule Cons.hyps)
        apply (force simp add: Suc Let-def simp del: o-apply)+
        done
    moreover
```

    let ? \(\mathrm{SB}=\) outstanding-refs is-Write \({ }_{\mathbf{s b}}\left(\right.\) takeWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.{ }_{\mathbf{s b}}\right)\) sb \(\left._{\mathrm{l}}\right)\)
    have a \(\notin\) ?SB
    proof
        assume \(a \in\) ?SB
        with a-notin [rule-format, of 0]
        show False
    by (auto simp add: 1 Suc)
qed
then
have ((flush (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb} \mathrm{sb}_{\mathrm{l}}\right) \mathrm{m}\right)(\mathrm{a}:=\mathrm{v})\right)=$
(flush (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{l}}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ )
apply -
apply (rule update-commute' $[$ where $\mathrm{m}=\mathrm{m}$ and $\mathrm{F}=\{\mathrm{a}\}$ and $\mathrm{G}=? \mathrm{SB}]$ )
apply (auto intro:
flush-unchanged-addresses
flushed-values-mem-independent simp del: o-apply)
done
ultimately
show ?thesis
by (simp add: 1 Suc del: fun-upd-apply o-apply)
qed
qed
lemma (in program) history-consistent-access-last-read':
$\wedge \mathrm{p}$. history-consistent $\vartheta \mathrm{p}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v$\left.]\right) \Longrightarrow$
$\vartheta \mathrm{t}=$ Some v
apply (induct sb)

```
apply (auto split: memref.splits option.splits)
done
lemma (in program) history-consistent-access-last-read:
    history-consistent \vartheta p (rev ( (Read sb volatile at v # sb))\Longrightarrow\vartheta t=Some v
    by (simp add: history-consistent-access-last-read')
lemma flush-all-until-volatile-write-Read sb
    \i m. \llbracketi < length ts; ts!i=(p,is,\vartheta, Read sb volatile a t v #sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
        \Longrightarrow ~ f l u s h - a l l - u n t i l - v o l a t i l e - w r i t e ~ ( t s [ i ~ : = ~ ( p , i s , \vartheta , ~ s b , ~ ( \mathcal { D } , \mathcal { O } , \mathcal { R } ^ { \prime } ) ] ) \mathrm { m }
        = flush-all-until-volatile-write ts m
proof (induct ts)
    case Nil thus ?case
        by simp
next
    case (Cons l ts)
    note i-bound = {i < length (l#ts)>
    note ith = <(l#ts)!i = (p,is,\vartheta, Read sb volatile a t v vsb,\mathcal{D},\mathcal{O},\mathcal{R})`
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l=(p,is,\vartheta,\mp@subsup{\operatorname{Read}}{\textrm{sb}}{}\mathrm{ volatile a t v#sb, D, }\mathcal{O},\mathcal{R})
            by simp
        thus?thesis
            by (simp add: 0 del: fun-upd-apply)
    next
        case (Suc n)
    obtain pl is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{|}{}\mp@subsup{\mathcal{D}}{l}{}\mp@subsup{\vartheta}{|}{}\mathrm{ sbl
            by (cases l)
    from i-bound ith
    have flush-all-until-volatile-write (ts[n := (p,is,\vartheta, sb, 疎, \mathcal{O},\mp@subsup{\mathcal{R}}{}{\prime})])
                (flush (takeWhile (Not o is-volatile-Write sb) sb 
            flush-all-until-volatile-write ts
                (flush (takeWhile (Not o is-volatile-Write sb) sb l) m)
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then show ?thesis
        by (simp add: Suc l)
    qed
qed
lemma flush-all-until-volatile-write-Ghostsb-commute:
    \i m. \llbracketi < length ts; ts!i=(p,is,\vartheta,Ghost sb A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
    \Longrightarrow ~ f l u s h - a l l - u n t i l - v o l a t i l e - w r i t e ~ ( t s [ i ~ : = ~ ( p ' , i s ' , \vartheta ' , ~ s b , ~ ( \mathcal { D } , \mathcal { O } ^ { \prime } , \mathcal { R } ) ] ) ~ m
    = flush-all-until-volatile-write ts m
proof (induct ts)
```

```
    case Nil thus ?case
    by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts) >
    note ith = <(l#ts)!i = (p,is,\vartheta,Ghost sb A L R W#sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l = (p,is,\vartheta,Ghost sb A L R W#sb, D, \mathcal{O},\mathcal{R})
            by simp
    thus ?thesis
        by (simp add: 0 del: fun-upd-apply)
    next
        case (Suc n)
        obtain plo is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{|}{}\mp@subsup{\mathcal{D}}{l}{}\mp@subsup{\vartheta}{l}{}\mathrm{ sbl
            by (cases l)
    from i-bound ith
    have flush-all-until-volatile-write (ts[n := ( }\mp@subsup{\textrm{p}}{}{\prime},\mp@subsup{\mathrm{ is }}{}{\prime},\mp@subsup{\vartheta}{}{\prime},\operatorname{sb},\mp@subsup{\mathcal{D}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime})]
                (flush (takeWhile (Not ○ is-volatile-Write sb}) \mp@subsup{\textrm{sb}}{\textrm{l}}{})\textrm{m})
                flush-all-until-volatile-write ts
                    (flush (takeWhile (Not O is-volatile-Write sb
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then show ?thesis
        by (simp add: Suc l)
    qed
qed
lemma flush-all-until-volatile-write-Prog \({ }_{s b}\)-commute:
\im. \(\llbracket \mathrm{i}<\) length ts; ts! \(i=\left(\mathrm{p}, \mathrm{is}, \vartheta, \operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis} \# \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right) \rrbracket\)
\(\Longrightarrow\) flush-all-until-volatile-write \(\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}, \mathcal{R}^{\prime}\right)\right]\right) \mathrm{m}\)
\(=\) flush-all-until-volatile-write ts m
proof (induct ts)
case Nil thus ?case
by \(\operatorname{simp}\)
next
case (Cons lts)
note i -bound \(=\) i \(<\) length ( \(\mathrm{l} \# \mathrm{ts}\) ) 〉
note ith \(=\left\langle(\mathrm{l} \# \mathrm{ts})!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \vartheta, \operatorname{Prog}_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2} \operatorname{mis} \# \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right\rangle\)
show ?case
proof (cases i)
case 0
from ith 0 have \(l: l=\left(p, i s, \vartheta, \operatorname{Prog}_{s b} p_{1} p_{2} \operatorname{mis} \# s b, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\)
by simp
thus ?thesis
```

```
        by (simp add: 0 del: fun-upd-apply)
    next
        case (Suc n)
        obtain p}\mp@subsup{\textrm{p}}{|}{}\mathrm{ is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{|}{}\mp@subsup{\mathcal{D}}{|}{}\mp@subsup{\vartheta}{|}{}\textrm{sb
        by (cases l)
    from i-bound ith
    have flush-all-until-volatile-write (ts[n:=(p,is, \vartheta, sb, 疎, \mathcal{O},\mp@subsup{\mathcal{R}}{}{\prime})])
                (flush (takeWhile (Not \circ is-volatile-Write }\mp@subsup{\textrm{sb}}{}{\prime})\mp@subsup{\textrm{sb}}{\textrm{l}}{})\textrm{m})
            flush-all-until-volatile-write ts
                (flush (takeWhile (Not ○ is-volatile-Write sb
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then show ?thesis
        by (simp add: Suc l)
    qed
qed
lemma flush-all-until-volatile-write-append-Prog
\i m. \llbracketi < length ts; ts!i=(p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
```



```
O},\mp@subsup{\mathcal{R}}{}{\prime})])\textrm{m
            = flush-all-until-volatile-write ts m
proof (induct ts)
    case Nil thus ?case
        by simp
next
    case (Cons l ts)
    note i-bound = <i < length (l#ts)>
    note ith = <(l#ts)!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
            by simp
    thus ?thesis
            by (simp add: }0\mathrm{ flush-append-Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ del: fun-upd-apply)
next
    case (Suc n)
    obtain pll is|}\mp@subsup{\mathcal{O}}{l}{}\mp@subsup{\mathcal{R}}{l}{}\mp@subsup{\mathcal{D}}{l}{}\mp@subsup{\vartheta}{l}{}\mathrm{ sb
        by (cases l)
    from i-bound ith
    have flush-all-until-volatile-write
                (ts[n := ( }\mp@subsup{\textrm{p}}{2}{},\mathrm{ is@mis, }\vartheta\mathrm{ , sb@ @ Prog
                (flush (takeWhile (Not ○ is-volatile-Write 
```

```
                flush-all-until-volatile-write ts
                (flush (takeWhile (Not ○ is-volatile-Write sb) sb⿱丷⿱一⿱㇒⿴囗⿱一一夊心
    apply -
    apply (rule Cons.hyps)
    apply (auto simp add: Suc 1)
    done
    then show ?thesis
    by (simp add: Suc l)
    qed
qed
lemma (in program) history-consistent-append--Prog
    assumes step: \vartheta\vdash p -> 
    shows history-consistent \vartheta (hd-prog p xs) xs \Longrightarrow last-prog p xs = p \Longrightarrow
        history-consistent \vartheta (hd-prog p'(xs@[Prog
proof (induct xs)
    case Nil with step show ?case by simp
next
    case (Cons x xs)
    note consis = <history-consistent \vartheta (hd-prog p (x # xs)) (x # xs) 〉
    note last = <last-prog p (x#xs) = p
    show ?case
    proof (cases x)
        case Write sb with Cons show ?thesis by (auto simp add: read-tmps-append)
    next
        case Read}\mp@subsup{\mp@code{sb}}{}{\mathrm{ with Cons show ?thesis by (auto split: option.splits)}
    next
    case (Prog
    from consis obtain
```



```
        consis': history-consistent \vartheta p p xs
        by (auto simp add: Progsb read-tmps-append)
    from last have last-p}\mp@subsup{p}{2}{}\mathrm{ : last-prog p}\mp@subsup{p}{2}{}\times\textrm{xs}=\textrm{p
        by (simp add: Prog
    from last-prog-to-last-prog-same [OF this]
    have last-prog': last-prog p xs = p.
    from history-consistent-hd-prog [OF consis']
    have consis'": history-consistent \vartheta (hd-prog p xs) xs.
    from Cons.hyps [OF this last-prog']
    have history-consistent \vartheta (hd-prog p'(xs @ [Prog
        (xs @ [Progsb p p'mis]).
    from history-consistent-hd-prog [OF this]
    have history-consistent \vartheta (hd-prog p2 (xs @ [Progsb p p' mis]))
        (xs @ [Prog}\mp@subsup{\mp@code{sb p p'mis]).}}{}{\prime
    moreover
    from history-consistent-hd-prog-p [OF consis']
```

```
    have p}\mp@subsup{p}{2}{}=\mathrm{ hd-prog p}\mp@subsup{p}{2}{}\mathrm{ xs.
    from hd-prog-last-prog-append--Prog
    have hd-prog p
        by simp
    ultimately
    have history-consistent \vartheta p
        by simp
    thus ?thesis
        by (simp add: Prog
    next
    case Ghost sb with Cons show ?thesis by (auto)
    qed
qed
```

primrec release :: 'a memref list $\Rightarrow$ addr set $\Rightarrow$ rels $\Rightarrow$ rels
where
release [] $\mathrm{S} \mathcal{R}=\mathcal{R}$
| release ( $\mathrm{x} \# \mathrm{xs}$ ) $\mathrm{S} \mathcal{R}=$
(case x of
Write $_{\text {sb }}$ volatile -- A L R W $\Rightarrow$
(if volatile then release xs $(S \cup R-L)$ Map.empty
else release xs $\mathrm{S} \mathcal{R}$ )
$\mid$ Ghost $_{\text {sb }} \mathrm{ALRW} \Rightarrow$ release xs $(\mathrm{S} \cup \mathrm{R}-\mathrm{L})$ (augment-rels $\mathrm{S} R \mathcal{R}$ )
$\mid-\Rightarrow$ release xs $\mathrm{S} \mathcal{R}$ )
lemma augment-rels-shared-exchange: $\forall a \in R .\left(a \in S^{\prime}\right)=(a \in S) \Longrightarrow$ augment-rels $S R$ $\mathcal{R}=$ augment-rels $\mathrm{S}^{\prime} \mathrm{R} \mathcal{R}$
apply (rule ext)
apply (auto simp add: augment-rels-def split: option.splits)
done
lemma sharing-consistent-shared-exchange:
assumes shared-eq: $\forall \mathrm{a} \in$ all-acquired sb. $\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows sharing-consistent $\mathcal{S}^{\prime} \mathcal{O}$ sb
using shared-eq consis
proof (induct sb arbitrary: $\mathcal{S} \mathcal{S}^{\prime} \mathcal{O}$ )
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
shared-eq: $\forall \mathrm{a} \in \mathrm{A} \cup$ all-acquired sb. $\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
by (clarsimp simp add: Write sb $^{\text {True }}$ )
from shared-eq
have shared-eq': $\forall \mathrm{a} \in$ all-acquired sb. $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$
by (auto simp add: augment-shared-def restrict-shared-def)
from Cons.hyps [OF shared-eq' consis']
have sharing-consistent $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb.
thus ? thesis
using A-shared-owns L-A A-R R-owns shared-eq
by (auto simp add: Write sb True domIff)
next
case False with Cons show ?thesis
by (auto simp add: Writesb
qed
next
case $\operatorname{Read}_{\text {sb }}$ with Cons show ?thesis
by auto
next
case $\operatorname{Prog}_{\text {sb }}$ with Cons show ?thesis
by auto
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
from Cons.prems obtain
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
shared-eq: $\forall \mathrm{a} \in \mathrm{A} \cup$ all-acquired sb. $\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
from shared-eq
have shared-eq': $\forall \mathrm{a} \in$ all-acquired sb. $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$
by (auto simp add: augment-shared-def restrict-shared-def)
from Cons.hyps [OF shared-eq' consis']
have sharing-consistent $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb.
thus ?thesis
using A-shared-owns L-A A-R R-owns shared-eq
by (auto simp add: Ghostsb domIff)
qed
qed
lemma release-shared-exchange:
assumes shared-eq: $\forall \mathrm{a} \in \mathcal{O} \cup$ all-acquired $\mathrm{sb} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows release sb $\left(\operatorname{dom} \mathcal{S}^{\prime}\right) \mathcal{R}=$ release sb $(\operatorname{dom} \mathcal{S}) \mathcal{R}$
using shared-eq consis
proof (induct sb arbitrary: $\mathcal{S} \mathcal{S}^{\prime} \mathcal{O} \mathcal{R}$ )
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True

## from Cons.prems obtain

A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and L-A: $\mathrm{L} \subseteq \mathrm{A}$ and A -R: $\mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
shared-eq: $\forall \mathrm{a} \in \mathcal{O} \cup \mathrm{A} \cup$ all-acquired sb. $\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S}$ a
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True )
from shared-eq
have shared-eq': $\forall \mathrm{a} \in \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb. $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R}\right.$
$\theta_{\mathrm{A}} \mathrm{L}$ ) a
by (auto simp add: augment-shared-def restrict-shared-def)
from Cons.hyps [OF shared-eq' consis']
have release sb $\left(\operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$ Map.empty $=$ release sb $\left(\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}}\right.\right.$
L)) Map.empty .
then show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True domIff)
next
case False with Cons show ?thesis
by (auto simp add: Write ${ }_{\mathbf{s b}}$ )
qed
next
case $\operatorname{Read}_{\mathrm{sb}}$ with Cons show ?thesis
by auto
next
case Prog $_{\text {sb }}$ with Cons show ?thesis
by auto
next
case ( Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and L-A: $\mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and shared-eq: $\forall \mathrm{a} \in \mathcal{O} \cup \mathrm{A} \cup$ all-acquired $\mathrm{sb} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
by (clarsimp simp add: Ghostsb )
from shared-eq
have shared-eq': $\forall \mathrm{a} \in \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb. $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}}\right.$
L) a
by (auto simp add: augment-shared-def restrict-shared-def)

```
    from A-shared-owns shared-eq R-owns have }\forall\textrm{a}\in\textrm{R}.(\textrm{a}\in\operatorname{dom}\mathcal{S})=(\textrm{a}\in\operatorname{dom}\mp@subsup{\mathcal{S}}{}{\prime}
        by (auto simp add: domIff)
    from augment-rels-shared-exchange [OF this]
    have (augment-rels (dom \mathcal{S})R\mathcal{R})=(\mathrm{ augment-rels (dom S ) R R}).
    with Cons.hyps [OF shared-eq' consis']
    have release sb (dom (\mathcal{S}}\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{\prime}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}))\mathrm{ ) (augment-rels (dom }\mp@subsup{\mathcal{S}}{}{\prime})\textrm{R}\mathcal{R})
                release sb (dom (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})) (augment-rels (dom S ) R \mathcal{R}) by simp
    then show ?thesis
        by (clarsimp simp add: Ghostsb domIff)
    qed
qed
lemma release-append:
\\mathcal{S}\mathcal{R}. release (sb@xs) (dom S)}\mathcal{R}=\mathrm{ release xs (dom (share sb S )) (release sb (dom (S))
R)
proof (induct sb)
    case Nil thus ?case by auto
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            from Cons.hyps [of (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mathrm{ Map.empty]
                show ?thesis
                    by (clarsimp simp add: Write sb True)
        next
            case False with Cons show ?thesis by (auto simp add: Write (s)
        qed
    next
        case Read sb with Cons show ?thesis
            by auto
    next
        case Progsb with Cons show ?thesis
                by auto
    next
        case (Ghostsb A L R W)
        with Cons.hyps [of (\mathcal{S}\oplus\textrm{W}}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mathrm{ augment-rels (dom S S R R
        show ?thesis
            by (clarsimp simp add: Ghostsb)
        qed
qed
locale xvalid-program = valid-program +
    fixes valid
    assumes valid-implies-valid-prog:
        |i < length ts;
```

$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$ valid $\mathrm{ts} \rrbracket \Longrightarrow$ valid-prog p
assumes valid-implies-valid-prog-hd:
$\llbracket \mathrm{i}<$ length ts;
ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$ valid $\mathrm{ts} \rrbracket \Longrightarrow$ valid-prog (hd-prog p sb)
assumes distinct-load-tmps-prog-step:
$\llbracket \mathrm{i}<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}\right) ;$ valid $\mathrm{ts} \rrbracket$
$\Longrightarrow$
distinct-load-tmps is ${ }^{\prime} \wedge$
(load-tmps is ${ }^{\prime} \cap$ load-tmps is $\left.=\{ \}\right) \wedge$
$\left(\right.$ load-tmps is ${ }^{\prime} \cap$ read-tmps sb) $=\{ \}$
assumes valid-data-dependency-prog-step:
$\llbracket \mathrm{i}<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}\right) ;$ valid $\mathrm{ts} \rrbracket$
$\Longrightarrow$
data-dependency-consistent-instrs (dom $\vartheta \cup$ load-tmps is) is ${ }^{\prime} \wedge$
load-tmps is' $\cap \bigcup$ (fst ' store-sops is) $=\{ \} \wedge$
load-tmps is ${ }^{\prime} \cap \bigcup($ fst ' write-sops sb) $=\{ \}$
assumes load-tmps-fresh-prog-step:
$\llbracket i<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}\right) ;$ valid $\mathrm{ts} \rrbracket$
$\Longrightarrow$
load-tmps is ${ }^{\prime} \cap \operatorname{dom} \vartheta=\{ \}$
assumes valid-sops-prog-step:
$\llbracket \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $)$; valid-prog $\mathrm{p} \rrbracket \Longrightarrow \forall$ sop $\in$ store-sops is ${ }^{\prime}$. valid-sop sop
assumes prog-step-preserves-valid:
$\llbracket i<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ; \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}\right) ;$ valid $\mathrm{ts} \rrbracket \Longrightarrow$ $\operatorname{valid}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is} @ i \mathrm{~s}^{\prime}, \vartheta, \mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime}\right.\right.\right.\right.$ is $\left.\left.\left.], \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]\right)$
assumes flush-step-preserves-valid:
$\llbracket i<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right) ;$ valid $\mathrm{ts} \rrbracket \Longrightarrow$ $\operatorname{valid}\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$
assumes sbh-step-preserves-valid:
$\llbracket i<$ length ts;
$\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$
$($ is $, \vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$;
valid ts】
$\Longrightarrow$
valid $\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$
lemma refl': $\mathrm{x}=\mathrm{y} \Longrightarrow \mathrm{r}^{\wedge} * * \mathrm{x} y$
by auto

```
lemma no-volatile-Read sb-volatile-reads-consistent:
    \m. outstanding-refs is-volatile-Read sb sb ={}\Longrightarrow volatile-reads-consistent m sb
    apply (induct sb)
    apply simp
    subgoal for a sb m
    apply (case-tac a)
    apply (auto split: if-split-asm)
    done
    done
```

theorem (in program) flush-store-buffer-append:
shows $\wedge$ ts pm $\vartheta \mathcal{O} \mathcal{R} \mathcal{D}$ is $\mathcal{O}^{\prime}$.
[i $<$ length ts;
instrs (sb@sb) @ is $\mathrm{sb}_{\mathrm{sb}}=$ is @ prog-instrs (sb@sb);
causal-program-history is $\mathrm{s}_{\mathrm{sb}}$ (sb@sb);
$\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is},\left.\vartheta\right|^{*}(\operatorname{dom} \vartheta-\right.$ read-tmps $\left.(\mathrm{sb@sb})), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right) ;$
$\mathrm{p}=$ hd-prog $\mathrm{p}_{\mathrm{sb}}($ sb@sb );
$\left(\operatorname{last}-\mathrm{prog} \mathrm{p}_{\mathrm{sb}}\left(\right.\right.$ sb@sb$\left.\left.^{\prime}\right)\right)=\mathrm{p}_{\mathrm{sb}} ;$
reads-consistent True $\mathcal{O}^{\prime} \mathrm{m}$ sb;
history-consistent $\vartheta$ p (sb@sb);
$\forall$ sop $\in$ write-sops sb. valid-sop sop;
distinct-read-tmps (sb@sb);
volatile-reads-consistent m sb
】
$\Longrightarrow$
$\exists$ is'. instrs sb' @ is $s_{\text {sb }}=$ is ${ }^{\prime} @$ prog-instrs sb ${ }^{\prime} \wedge$
$(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}$

( $\mathcal{D} \vee$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}$ ),
acquired True sb $\mathcal{O}$, release sb $(\operatorname{dom} \mathcal{S}) \mathcal{R})$ ], flush sb m,share sb $\mathcal{S}$ )
proof (induct sb)
case Nil
thus ?case by (auto simp add: list-update-id' split: if-split-asm)
next
case (Cons r sb)
interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda p p^{\prime}$ is sb. sb.
have ts-i:
ts! $\mathrm{i}^{=}=\left(\mathrm{p}, \mathrm{is},\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-\right.$ read-tmps $\left.((\mathrm{r} \# \mathrm{sb}) @ s b \prime)), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by fact
have is: instrs ((r \# sb) @ sb) @ is $\mathrm{s}_{\mathrm{sb}}=$ is @ prog-instrs ((r \# sb) @ sb') by fact
have i-bound: i < length ts by fact
have causal：causal－program－history is $\mathrm{sb}^{\text {（ }}(\mathrm{r} \# \mathrm{sb})$＠sb$\left.{ }^{\prime}\right)$ by fact hence causal＇：causal－program－history is ${ }_{\text {sb }}$（ sb ＠sb＇）
by（auto simp add：causal－program－history－def）
note reads－consis $=\left\langle\right.$ reads－consistent $\left.\operatorname{True} \mathcal{O}^{\prime} \mathrm{m}(\mathrm{r} \# \mathrm{sb})\right\rangle$
note $p=\left\langle p=\right.$ hd－prog $\left.p_{\text {sb }}((r \# s b) @ s b)^{\prime}\right\rangle$
note $\mathrm{p}_{\mathrm{sb}}=$ \｛last－prog $\mathrm{p}_{\mathrm{sb}}\left((\mathrm{r} \# \mathrm{sb}) @ \mathrm{sb}^{\prime}\right)=\mathrm{p}_{\mathrm{sb}}$ 〉
note hist－consis $=\langle$ history－consistent $\vartheta \mathrm{p}((\mathrm{r} \# \mathrm{sb}) @ s b)\rangle$
note valid－sops $=\langle\forall$ sop $\in$ write－sops（ $\mathrm{r} \# \mathrm{sb}$ ）．valid－sop sop $\rangle$
note dist $=$ 〈distinct－read－tmps（（r\＃sb）＠sb）${ }^{\prime}$ 〉
note vol－read－consis $=$ 〈volatile－reads－consistent $\mathrm{m}(\mathrm{r} \# \mathrm{sb})$ ）

```
show ?case
proof (cases r)
    case (Progsb p1 p2 pis)
    from vol-read-consis
    have vol-read-consis': volatile-reads-consistent m sb
        by (auto simp add: Progsb)
    from hist-consis obtain
    prog-step: \vartheta|` (dom \vartheta - read-tmps (sb @ sb
    hist-consis': history-consistent \vartheta p}\mp@subsup{\textrm{p}}{2}{(sb}@ sb
    by (auto simp add: Progsb)
from p obtain p: p = p1
    by (simp add: Progsb)
from history-consistent-hd-prog [OF hist-consis']
have hist-consis'": history-consistent v (hd-prog p2 (sb @ sb}))\mathrm{ (sb @ sb).
from is
have is: instrs (sb @ sb') @ is sb = (is @ pis) @ prog-instrs (sb @ sb')
    by (simp add: Prog
```

from ts-i is have
ts-i: ts! $i=\left(\mathrm{p}\right.$, is $,\left.\vartheta\right|^{6}(\operatorname{dom} \vartheta-$ read-tmps $\left.(\mathrm{sb} @ \mathrm{sb})), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by (simp add: Prog $_{\text {sb }}$ )
let ${ }^{2} \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{2}\right.\right.$, is@pis, $\left.\vartheta\right|^{6}(\operatorname{dom} \vartheta-$ read-tmps $\left.\left.(\mathrm{sb} @ \mathrm{sb})), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$
from direct-computation.Program [OF i-bound ts-i prog-step [simplified p[symmetric]]]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}(? \mathrm{ts}, \mathrm{m}, \mathcal{S})$ by simp
also
from i-bound have i-bound': i < length ?ts'
by auto
from i-bound
have ts' $-\mathrm{i}: ~ ? \mathrm{ts} \mathrm{s}^{\prime} \mathrm{i}=\left(\mathrm{p}_{2}\right.$, is $@$ pis $,\left(\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta-\right.$ read-tmps $\left.\left.(\mathrm{sb} @ \operatorname{sb}))\right), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by auto
from history-consistent-hd-prog-p [OF hist-consis']
have $\mathrm{p}_{2}$-hd-prog: $\mathrm{p}_{2}=h d-\operatorname{prog} \mathrm{p}_{2}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)$.
from reads-consis have reads-consis': reads-consistent True $\mathcal{O}^{\prime} \mathrm{m}$ sb by (simp add: $\operatorname{Prog}_{\text {sb }}$ )
from valid-sops have valid-sops': $\forall$ sop $\in$ write-sops sb. valid-sop sop by (simp add: $\operatorname{Prog}_{\text {sb }}$ )
from dist have dist': distinct-read-tmps (sb@sb')
by ( $\operatorname{simp}$ add: Prog $_{\text {sb }}$ )
from $\mathrm{p}_{\mathrm{sb}}$ have last-prog- $\mathrm{p}_{2}$ : last-prog $\mathrm{p}_{2}\left(\mathrm{sb}^{(\mathrm{s}}\right.$ @ $\left.\mathrm{sb}^{\prime}\right)=\mathrm{p}_{\mathrm{sb}}$
by (simp add: Prog $_{\text {sb }}$ )
from hd-prog-last-prog-end [OF p2-hd-prog this]
have $\mathrm{p}_{2}$-hd-prog' $\mathrm{p}_{2}=$ hd-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)$.
from last-prog- $\mathrm{p}_{2}$ [symmetric] have last-prog': last-prog $\mathrm{p}_{\text {sb }}\left(\mathrm{sb}\right.$ @ sb') $=\mathrm{p}_{\mathrm{sb}}$ by (simp add: last-prog-idem)
from Cons.hyps [OF i-bound' is causal' ts'i $^{\prime}$ i $\mathrm{p}_{2}$-hd-prog' last-prog' reads-consis ${ }^{\prime}$ hist-consis' valid-sops' dist' vol-read-consis'] i-bound
obtain is' where
is': instrs sb ${ }^{\prime} @$ is $_{\text {sb }}=$ is' @ prog-instrs sb' and
step: (?ts $\left.{ }^{\prime}, \mathrm{m}, \mathcal{S}\right) \Rightarrow{ }_{\mathrm{d}}{ }^{*}$
(ts[i := (last-prog (hd-prog $\left.\left.\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\right)^{\prime}\right) \mathrm{sb}$, is ${ }^{\prime}$,
$\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta$ - read-tmps sb$), \mathrm{x}, \mathcal{D} \vee$ outstanding-refs is-volatile-Write $\mathrm{e}_{\mathrm{sb}} \mathrm{sb} \neq\{ \}$, acquired True sb $\mathcal{O}$,release sb $(\operatorname{dom} \mathcal{S}) \mathcal{R})$ ], flush sb m,share sb $\mathcal{S}$ )
by (auto)
from $p_{2}$-hd-prog ${ }^{\prime}$
have last-prog-eq: last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}$ ) sb $=$ last-prog $\mathrm{p}_{2} \mathrm{sb}$
by (simp add: last-prog-hd-prog-append)
note step
finally show? thesis
using is'
by (simp add: Prog $_{\text {sb }}$ last-prog-eq)
next
case (Write ${ }_{\text {sb }}$ volatile a sop v A L R W)
obtain D f where sop: sop=(D,f)
by (cases sop)
from vol-read-consis
have vol-read-consis': volatile-reads-consistent (m(a:=v)) sb by (auto simp add: Write ${ }_{\text {sb }}$ )

## from hist-consis obtain

D-tmps: $\mathrm{D} \subseteq \operatorname{dom} \vartheta$ and
$\mathrm{f}-\mathrm{v}: \mathrm{f} \vartheta=\mathrm{v}$ and
dep: $\mathrm{D} \cap$ read-tmps $\left(\mathrm{sb}_{\mathrm{B}} \mathrm{sb}^{\prime}\right)=\{ \}$ and
hist-consis': history-consistent $\vartheta \mathrm{p}$ (sb@sb')
by (simp add: Write ${ }_{\text {sb }}$ sop split: option.splits)
from dist have dist': distinct-read-tmps (sb@sb') by (auto simp add: Write ${ }_{\text {sb }}$ )
from valid-sops obtain valid-sop sop and
valid-sops ${ }^{\prime}: ~ \forall$ sop $\in$ write-sops sb. valid-sop sop
by (simp add: Write ${ }_{\text {sb }}$ )
interpret valid-sop sop by fact
from valid-sop [OF sop D-tmps]
have $\mathrm{f} \vartheta=\mathrm{f}\left(\left.\vartheta\right|^{\prime} \mathrm{D}\right)$.
moreover
from dep D-tmps have D-subset: $\mathrm{D} \subseteq\left(\operatorname{dom} \vartheta\right.$ - read-tmps $\left.\left(\operatorname{sb} @ s b^{\prime}\right)\right)$
by auto
moreover from D-subset have $\left(\left.\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta\right.$ - read-tmps $\left.(\operatorname{sb@sb}))\right|^{6} \mathrm{D}\right)=\left.\vartheta\right|^{6} \mathrm{D}$
apply -
apply (rule ext)
apply (auto simp add: restrict-map-def)
done
moreover from D-subset D-tmps have $\mathrm{D} \subseteq \operatorname{dom}\left(\left.\vartheta\right|^{\prime}\left(\operatorname{dom} \vartheta-\right.\right.$ read-tmps $\left.\left.\left(\operatorname{sb@sb}^{\prime}\right)\right)\right)$
by simp
moreover
note valid-sop [OF sop this]
ultimately have $\mathrm{f}-\mathrm{v}^{\prime}: \mathrm{f}\left(\left.\vartheta\right|^{\cdot}(\operatorname{dom} \vartheta-r e a d-t m p s(\operatorname{sb@sb}))\right)=\mathrm{v}$
by (simp add: f-v)
interpret causal': causal-program-history is sb $_{\text {sb }}$ sb@sb ${ }^{\prime}$ by fact
from is
have Write volatile a sop A L R W\# instrs (sb @ sb) @ is $\mathrm{s}_{\mathrm{sb}}=$ is @ prog-instrs (sb @ sb)
by (simp add: Write ${ }_{\text {sb }}$ )
with causal'.causal-program-history [of [], simplified, OF refl]
obtain is' where is: is=Write volatile a sop A L R W\#is' and
is': instrs (sb @ sb') @ is sb $=$ is ${ }^{\prime} @$ prog-instrs (sb @ sb')
by auto
from ts- i is
have ts- $\mathrm{i}: \mathrm{ts}!\mathrm{i}=\left(\mathrm{p}\right.$, Write volatile a sop A L R W\#is ${ }^{\prime}$,
$\left.\vartheta\right|^{\prime}\left(\operatorname{dom} \vartheta-\right.$ read-tmps $\left(\right.$ sb@sb'$\left.\left.\left.^{\prime}\right)\right), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by (simp add: Write ${ }_{\text {sb }}$ )
from p have $\mathrm{p}^{\prime}: \mathrm{p}=$ hd- $\operatorname{prog} \mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb@sb}{ }^{\prime}\right)$
by (auto simp add: Write ${ }_{\text {sb }}$ hd-prog-idem)
from $\mathrm{p}_{\mathrm{sb}}$ have $\mathrm{p}_{\mathrm{sb}}{ }^{\prime}$ : last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)=\mathrm{p}_{\mathrm{sb}}$
by (simp add: Write ${ }_{\text {sb }}$ )
show ?thesis
proof (cases volatile)
case False
have memop-step:
(Write volatile a sop A L R W\#is', $\left.\vartheta\right|^{\cdot}(\operatorname{dom} \vartheta-r e a d-t m p s(s b @ s b))$, $\mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow$
$\left(\right.$ is $\left.^{\prime},\left.\vartheta\right|^{\cdot}(\operatorname{dom} \vartheta-\operatorname{read}-\mathrm{tmps}(\mathrm{sb} @ \mathrm{sb})), \mathrm{x}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
using D-subset
apply (simp only: sop $\mathrm{f}-\mathrm{v}^{\prime}$ [symmetric] False)
apply (rule direct-memop-step.WriteNonVolatile)
done
let $? \mathrm{ts}^{\prime}=\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.$, is $\left.\left.^{\prime},\left.\vartheta\right|^{\cdot}(\operatorname{dom} \vartheta-\operatorname{read}-\mathrm{tmps}(\mathrm{sb} @ \operatorname{sb})), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$
from direct-computation.Memop [OF i-bound ts-i memop-step]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(? \mathrm{ts}^{\prime}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{S}\right)$.
also
from reads-consis have reads-consis': reads-consistent $\operatorname{True} \mathcal{O}^{\prime}(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{sb}$
by (auto $\operatorname{simp}$ add: Write sb $_{\text {b }}$ False)
from i-bound have i-bound': i < length ?ts ${ }^{\prime}$
by auto
from i-bound
have $\mathrm{ts}^{\prime}-\mathrm{i}: ~ ? \mathrm{ts}{ }^{\prime}!\mathrm{i}=\left(\mathrm{p}\right.$, is $\left.{ }^{\prime},\left.\vartheta\right|^{‘}\left(\operatorname{dom} \vartheta-\operatorname{read}-\operatorname{tmps}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by simp
from Cons.hyps [OF i-bound ${ }^{\prime}$ is $^{\prime}$ causal $^{\prime}$ ts' ${ }^{-i} \mathrm{p}^{\prime} \mathrm{p}_{\mathrm{sb}}{ }^{\prime}$ reads-consis ${ }^{\prime}$ hist-consis ${ }^{\prime}$ valid-sops' dist $^{\prime}$ vol-read-consis ] i-bound
obtain is ${ }^{\prime \prime}$ where
is ${ }^{\prime \prime}$ : instrs sb' @ is $s_{\text {sb }}=$ is ${ }^{\prime \prime} @$ prog-instrs sb' and
steps: $\left(? \mathrm{ts}^{\prime}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{S}\right) \Rightarrow_{\mathrm{d}}{ }^{*}$
( $\mathrm{ts}\left[\mathrm{i}:=\right.$ (last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\right) \mathrm{sb}$, is $^{\prime \prime}$,
$\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta-r e a d-t m p s ~ s b '), ~ x$, $\mathcal{D} \vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \neq\{ \}$, acquired True sb $\mathcal{O}$, release sb
$(\operatorname{dom} \mathcal{S}) \mathcal{R})]$,
flush $\operatorname{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$, share $\mathrm{sb} \mathcal{S})$
by (auto simp del: fun-upd-apply)
note steps
finally
show ?thesis
using is "
by (simp add: Write ${ }_{\text {sb }}$ False)
next
case True
have memop-step:
(Write volatile a sop A L R W\#is', $\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta$ - read-tmps $(\operatorname{sb@sb}))$,

$$
\mathrm{x}, \mathrm{~m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow
$$


$\left.\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
using D-subset
apply (simp only: sop f-v' [symmetric] True)
apply (rule direct-memop-step.WriteVolatile)
done
let $? \mathrm{ts}^{\prime}=\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime},\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-\operatorname{read}-\operatorname{tmps}(\mathrm{sb} @ \mathrm{sb})), \mathrm{x}\right.\right.$, True, $\mathcal{O} \cup \mathrm{A}-$ R,Map.empty)]
from direct-computation.Memop [OF i-bound ts-i memop-step]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(? \mathrm{ts}^{\prime}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
also
from reads-consis have reads-consis': reads-consistent $\operatorname{True}\left(\mathcal{O}^{\prime} \cup \mathrm{A}-\mathrm{R}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$
sb
by (auto simp add: Write sb $^{\text {True) }}$
from i-bound have i-bound': i < length ?ts ${ }^{\prime}$
by auto
from i-bound
have $\mathrm{ts}^{\prime}-\mathrm{i}: ~ ? \mathrm{ts}^{\prime}!\mathrm{i}=\left(\mathrm{p}\right.$, is ${ }^{\prime},\left.\vartheta\right|^{‘}\left(\operatorname{dom} \vartheta-r e a d-t m p s\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right)$, x , True, $\mathcal{O} \cup \mathrm{A}-$ R,Map.empty)
by simp
from Cons.hyps [OF i-bound ${ }^{\prime}$ is ${ }^{\prime}$ causal $^{\prime} \mathrm{ts}^{\prime}-\mathrm{i} \mathrm{p}^{\prime} \mathrm{p}_{\mathrm{sb}}{ }^{\prime}$ reads-consis ${ }^{\prime}$ hist-consis ${ }^{\prime}$
valid-sops' dist $^{\prime}$ vol-read-consis ${ }^{\prime}$, of $\left.\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right]$ i-bound
obtain is" where
is ${ }^{\prime \prime}$ : instrs $\mathrm{sb}^{\prime} @ \mathrm{is}_{\mathrm{sb}}=$ is ${ }^{\prime \prime} @$ prog-instrs sb ${ }^{\prime}$ and
steps: $\left(? \mathrm{ts}^{\prime}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \Rightarrow_{\mathrm{d}}{ }^{*}$
( $\mathrm{ts}\left[\mathrm{i}:=\left(\right.\right.$ last- prog (hd-prog $\left.\mathrm{p}_{\mathrm{sb}} \mathrm{sb}{ }^{\prime}\right) \mathrm{sb}, \mathrm{is}^{\prime \prime}$,
$\left.\vartheta\right|^{\prime}\left(\operatorname{dom} \vartheta-r e a d-t m p s b^{\prime}\right)$, $x$,
True, acquired True sb $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$, release $\operatorname{sb}\left(\operatorname{dom}\left(\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$ Map.empty)],
flush $\operatorname{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$, share $\left.\mathrm{sb}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$
by (auto simp del: fun-upd-apply)
note steps
finally
show ?thesis
using is"
by (simp add: Writesb True)
qed
next
case $\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v$)$
from vol-read-consis reads-consis obtain v : $\mathrm{v}=\mathrm{m}$ a and r -consis: reads-consistent True $\mathcal{O}^{\prime} \mathrm{m} \mathrm{sb}$ and
vol-read-consis': volatile-reads-consistent m sb
by (cases volatile) (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
from valid-sops have valid-sops ${ }^{\prime}: \forall$ sop $\in$ write-sops sb. valid-sop sop
by ( $\operatorname{simp}$ add: $\operatorname{Read}_{\mathrm{sb}}$ )
from hist-consis obtain $\vartheta: \vartheta \mathrm{t}=$ Some v and
hist-consis': history-consistent $\vartheta \mathrm{p}\left(\mathrm{sb@sb}{ }^{\prime}\right)$
by (simp add: Read sb split: option.splits)
from dist obtain t-notin: $\mathrm{t} \notin$ read-tmps ( $\mathrm{sb} @ \mathrm{sb}^{\prime}$ ) and
dist $^{\prime}$ : distinct-read-tmps (sb@sb') by (simp add: Read ${ }_{\text {sb }}$ )
from $\vartheta$ t-notin have restrict-commute:
$\left(\left.\vartheta\right|^{\bullet}(\operatorname{dom} \vartheta-\operatorname{read}-\operatorname{tmps}(\operatorname{sb} @ \operatorname{sb}))\right)(\mathrm{t} \mapsto \mathrm{v})=$ $\left.\vartheta\right|^{‘}\left(\operatorname{dom} \vartheta-\right.$ read-tmps $\left.\left(\operatorname{sb@sb}^{\prime}\right)\right)$
apply -
apply (rule ext)
apply (auto simp add: restrict-map-def domIff)
done
from $\vartheta$ t-notin
have restrict-commute':

```
\(\left(\left(\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-\right.\right.\) insert t \((\) read-tmps \(\left.\left.(\operatorname{sb@sb})))\right)(\mathrm{t} \mapsto \mathrm{v})\right)=\)
    \(\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-r e a d-t m p s(s b @ s b))\)
apply -
apply (rule ext)
apply (auto simp add: restrict-map-def domIff)
done
```

interpret causal': causal-program-history is $_{\text {sb }}$ sb@sb $^{\prime}$ by fact
from is
have Read volatile a t \# instrs ( sb @ $\mathrm{sb}^{\prime}$ ) @ is $\mathrm{sb}_{\mathrm{sb}}=$ is @ prog-instrs ( $\mathrm{sb} @ \mathrm{sb}^{\prime}$ )
by ( $\operatorname{simp}$ add: $\operatorname{Read}_{\mathrm{sb}}$ )
with causal'.causal-program-history [of [], simplified, OF refl]
obtain is' where is: is=Read volatile a t\#is' and
is': instrs (sb @ sb') @ is sb $^{\prime}=$ is $^{\prime} @$ prog-instrs (sb @ sb')
by auto
from ts-i is
have ts-i: $\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}\right.$, Read volatile a $\mathrm{t} \# \mathrm{is}{ }^{\prime}$,
$\left.\vartheta\right|^{‘}\left(\operatorname{dom} \vartheta-\right.$ insert t (read-tmps $\left.\left(\operatorname{sb}^{@} \mathrm{sb}^{\prime}\right)\right)$ ) $\left.\mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by ( $\operatorname{simp}$ add: $\operatorname{Read}_{\text {sb }}$ )
from direct-memop-step.Read [of volatile a t is $\left.{ }^{\prime} \vartheta\right|^{\cdot}$ (dom $\vartheta$ - insert t (read-tmps $\left.\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right)$ ) x m $\left.\mathcal{D} \mathcal{O} \mathcal{R} \mathcal{S}\right]$
have memop-step:
(Read volatile a t \# is ${ }^{\prime}$,
$\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-$ insert $\mathrm{t}($ read-tmps $\left.(\mathrm{sb} @ \mathrm{sb}))), \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow$ (is',
$\left.\left.\vartheta\right|^{\prime}\left(\operatorname{dom} \vartheta-\left(\operatorname{read}-\operatorname{tmps}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right)\right), \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
by (simp add: v [symmetric] restrict-commute restrict-commute')
let ${ }^{2} \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.$, is ${ }^{\prime}$,
$\left.\left.\left.\vartheta\right|^{6}\left(\operatorname{dom} \vartheta-\operatorname{read}-\mathrm{tmps}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$
from direct-computation.Memop [OF i-bound ts-i memop-step]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(? \mathrm{ts}^{\prime}, \mathrm{m}, \mathcal{S}\right)$.
also
from i-bound have i-bound': i < length ?ts'
by auto
from i-bound
have ts' $-\mathrm{i}: ? \mathrm{ts}!^{\prime} \mathrm{i}=\left(\mathrm{p}, \mathrm{is}^{\prime},\left(\left.\vartheta\right|^{6}\left(\operatorname{dom} \vartheta-\right.\right.\right.$ read-tmps $\left.\left.\left.\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)\right)\right), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)$
by auto
from $p$ have $p^{\prime}: p=\operatorname{hd}-\operatorname{prog} p_{s b}\left(\right.$ sb@sb $\left.^{\prime}\right)$
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ hd-prog-idem)
from $\mathrm{p}_{\mathrm{sb}}$ have $\mathrm{p}_{\mathrm{sb}}{ }^{\prime}$ : last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb}\right.$ @ $\left.\mathrm{sb}^{\prime}\right)=\mathrm{p}_{\mathrm{sb}}$
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
from Cons.hyps [OF i-bound ${ }^{\prime}$ is' causal' ts' $^{\prime}$ - $\mathrm{p}^{\prime} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{r}$-consis hist-consis ${ }^{\prime}$
valid-sops' dist' vol-read-consis']

## obtain is" where

is ${ }^{\prime \prime}$ : instrs $\mathrm{sb}^{\prime} @$ is $\mathrm{s}_{\mathrm{sb}}=$ is" ${ }^{\prime \prime}$ @ prog-instrs sb' and
steps: $\left(? \mathrm{ts}^{\prime}, \mathrm{m}, \mathcal{S}\right) \Rightarrow \mathrm{d}^{*}$
(ts[i := (last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}$ ) sb, is ${ }^{\prime \prime}$,
$\left.\vartheta\right|^{\bullet}\left(\right.$ dom $\vartheta-$ read-tmps sb $\left.{ }^{\prime}\right), \mathrm{x}, \mathcal{D} \vee$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb} \neq$
\{\}, acquired True sb $\mathcal{O}$, release sb $(\operatorname{dom} \mathcal{S}) \mathcal{R})$ ], flush sb m,share sb $\mathcal{S}$ )
by (auto simp del: fun-upd-apply)
note steps
finally
show ?thesis
using is"
by (simp add: $\operatorname{Read}_{\text {sb }}$ )
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
from vol-read-consis
have vol-read-consis': volatile-reads-consistent m sb by (auto simp add: Ghost ${ }_{\text {sb }}$ )
from reads-consis have r-consis: reads-consistent $\operatorname{True}\left(\mathcal{O}^{\prime} \cup \mathrm{A}-\mathrm{R}\right) \mathrm{msb}$ by (auto simp add: Ghost ${ }_{\mathbf{s b}}$ )
from valid-sops have valid-sops': $\forall$ sop $\in$ write-sops sb. valid-sop sop
by (simp add: Ghost ${ }_{\text {sb }}$ )
from hist-consis obtain
hist-consis': history-consistent $\vartheta$ p (sb@sb)
by (simp add: Ghost ${ }_{\text {sb }}$ )

## from dist obtain

dist $^{\prime}$ : distinct-read-tmps (sb@sb') by (simp add: Ghost ${ }_{\mathbf{s b}}$ )
interpret causal': causal-program-history is sb sb@sb' $^{\prime}$ by fact
from is
have Ghost A L R W\# instrs ( sb @ $\mathrm{sb}^{\prime}$ ) @ is $\mathrm{sb}_{\mathrm{s}}=$ is @ prog-instrs ( sb @ $\mathrm{sb}^{\prime}$ )
by (simp add: Ghost ${ }_{\text {sb }}$ )
with causal'.causal-program-history [of [], simplified, OF refl]
obtain is' where is: is=Ghost A L R W\#is' and is': instrs (sb @ sb') @ is $\mathrm{sb}_{\mathrm{sb}}=\mathrm{is}^{\prime} @$ prog-instrs (sb @ sb')
by auto
from ts-i is
have ts-i: ts $!\mathrm{i}=\left(\mathrm{p}\right.$, Ghost A L R W\#is ${ }^{\prime}$,
$\left.\vartheta\right|^{6}(\operatorname{dom} \vartheta-($ read-tmps $(\mathrm{sb} @ \mathrm{sb} \mathcal{)})), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by ( $\operatorname{simp}$ add: Ghost ${ }_{\text {sb }}$ )
from direct-memop-step.Ghost [of A L R W is ${ }^{\prime}$
$\left.\vartheta\right|^{\bullet}\left(\operatorname{dom} \vartheta-\left(\right.\right.$ read-tmps $\left(\right.$ sb@sb' $\left.\left.^{\prime}\right)\right)$ ) $\left.\mathrm{m} \mathcal{D} \mathcal{O} \mathcal{R} \mathcal{S}\right]$
have memop-step:
(Ghost A L R W\# is $\left.{ }^{\prime},\left.\vartheta\right|^{‘}(\operatorname{dom} \vartheta-r e a d-t m p s(s b @ \operatorname{sb})), \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)$
$\rightarrow\left(\right.$ is $^{\prime},\left.\vartheta\right|^{\prime}\left(\operatorname{dom} \vartheta-r e a d-t m p s\left(s b @ b^{\prime}\right)\right), \mathrm{x}, \mathrm{m}, \mathcal{D}, \mathcal{O} \cup \mathrm{A}-\mathrm{R}$, augment-rels (dom
$\mathcal{S}) \mathrm{R} \mathcal{R}$,
$\left.\mathcal{S} \oplus \mathrm{W} \quad \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
let $? \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}\right.\right.$, $\left.\vartheta\right|^{6}\left(\operatorname{dom} \vartheta-\right.$ read-tmps $\left.\left(\mathrm{sb}^{@} \mathrm{sb}^{\prime}\right)\right), \mathrm{x}, \mathcal{D}, \mathcal{O} \cup \mathrm{A}-\mathrm{R}$, augment-rels (dom
$\mathcal{S}) \mathrm{R} \mathcal{R})]$
from direct-computation.Memop [OF i-bound ts-i memop-step]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(? \mathrm{ts}^{\prime}, \mathrm{m}, \mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
also
from i-bound have i-bound': i < length ?ts ${ }^{\prime}$
by auto
from i-bound
have $\mathrm{ts}^{\prime}-\mathrm{i}: ~ ? \mathrm{ts}{ }^{\prime}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}^{\prime},\left(\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta-\operatorname{read}-\operatorname{tmps}(\mathrm{sb} @ \mathrm{sb}))\right), \mathrm{x}, \mathcal{D}, \mathcal{O} \cup \mathrm{A}-\right.$ R,augment-rels (dom $\mathcal{S}) \mathrm{R} \mathcal{R}$ )
by auto
from p have $\mathrm{p}^{\prime}: \mathrm{p}=$ hd- $\operatorname{prog} \mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb@sb}{ }^{\prime}\right)$
by (auto simp add: Ghost ${ }_{\text {sb }}$ hd-prog-idem)
from $p_{s b}$ have $\mathrm{p}_{\mathrm{sb}}{ }^{\prime}$ : last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @ \mathrm{sb}^{\prime}\right)=\mathrm{p}_{\mathrm{sb}}$
by ( $\operatorname{simp}$ add: Ghost ${ }_{\text {sb }}$ )

```
        from Cons.hyps [OF i-bound' is' causal' ts'-i p' p ps ' r-consis hist-consis'
            valid-sops' dist' vol-read-consis', of S S \oplusW R }\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
    obtain is" where
        is'": instrs sb'@ is isb = is" @ prog-instrs sb' and
        steps: (?ts',m,\mathcal{S}\oplus\textrm{w}}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})=>\mp@subsup{|}{\textrm{d}}{}\mp@subsup{}{}{*
            (ts[i := (last-prog (hd-prog posb sb) sb, is'",
                    \vartheta |` (dom \vartheta - read-tmps sb),x,
                \mathcal{D}\vee outstanding-refs is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\textrm{sb}\not={}}\mathrm{ , acquired True sb (O }\cup\textrm{O}-\textrm{R})
                release sb (dom (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})) (augment-rels (dom \mathcal{S})R\mathcal{R}))],
                flush sb m,share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{\prime}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
        by (auto simp add: list-update-overwrite simp del: fun-upd-apply)
    note steps
    finally
    show ?thesis
        using is"
        by (simp add: Ghostsb
    qed
qed
corollary (in program) flush-store-buffer:
assumes i-bound: \(\mathrm{i}<\) length ts
assumes instrs: instrs sb @ is \(\mathrm{s}_{\text {sb }}=\) is @ prog-instrs sb
assumes cph: causal-program-history is isb sb
assumes ts-i: ts! \(\mathrm{i}=\left(\mathrm{p}, \mathrm{is},\left.\vartheta\right|^{\prime}(\operatorname{dom} \vartheta-\operatorname{read}-\mathrm{tmps} \mathrm{sb}), \mathrm{x}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\)
assumes p: \(\mathrm{p}=\mathrm{hd}\)-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\)
assumes last-prog: (last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\) ) \(=\mathrm{p}_{\mathrm{sb}}\)
assumes reads-consis: reads-consistent \(\operatorname{True} \mathcal{O}^{\prime} \mathrm{m}\) sb
assumes hist-consis: history-consistent \(\vartheta \mathrm{p} \mathrm{sb}\)
assumes valid-sops: \(\forall\) sop \(\in\) write-sops sb. valid-sop sop
assumes dist: distinct-read-tmps sb
assumes vol-read-consis: volatile-reads-consistent m sb
shows ( \(\mathrm{ts}, \mathrm{m}, \mathcal{S}\) ) \(\Rightarrow_{\mathrm{d}}{ }^{*}\)
(ts \(\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{i}_{\mathrm{sb}}, \vartheta, \mathrm{x}\right.\right.\),
\(\mathcal{D} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb} \neq\{ \}\),acquired True \(\mathrm{sb} \mathcal{O}\), release sb \((\operatorname{dom} \mathcal{S}) \mathcal{R})]\),
flush sb m,share sb \(\mathcal{S}\) )
using flush-store-buffer-append [where \(\mathrm{sb}^{\prime}=[]\), simplified, OF i-bound instrs cph ts-i [simplified] p last-prog reads-consis hist-consis valid-sops dist vol-read-consis] last-prog by simp
```

```
lemma last-prog-same-append: \(\Lambda\) xs \(\mathrm{p}_{\mathrm{sb}}\). last-prog \(\mathrm{p}_{\mathrm{sb}}(\mathrm{sb} @ \mathrm{xs})=\mathrm{p}_{\mathrm{sb}} \Longrightarrow\) last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{xs}\)
\(=\mathrm{p}_{\mathrm{sb}}\)
    apply (induct sb)
    apply simp
    subgoal for a sb xs \(p_{s b}\)
    apply (case-tac a)
    apply simp
    apply simp
```

```
apply simp
apply (drule last-prog-to-last-prog-same)
apply simp
apply simp
done
done
```

lemma reads-consistent-drop-volatile-writes-no-volatile-reads:
^pending-write $\mathcal{O} \mathrm{m}$. reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \Longrightarrow$
outstanding-refs is-volatile-Read $_{\text {sb }}\left(\left(\right.\right.$ dropWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right)\right) \mathrm{sb}\right)=\{ \}$
apply (induct sb)
apply (auto split: memref.splits)
done
lemma reads-consistent-flush-other:
assumes no-volatile-Write ${ }_{\text {sb }}$-sb: outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb}=\{ \}$
shows $\bigwedge \mathrm{m}$ pending-write $\mathcal{O}$.
【outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{\mathbf{s b}}$ ) xs $\cap$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ $\mathrm{sb}=\{ \}$;
reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{xs} \rrbracket \Longrightarrow$ reads-consistent pending-write $\mathcal{O}$ (flush
sb m) xs
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
note no-inter $=\lessdot$ outstanding-refs $\left(\right.$ Not $\circ$ is-volatile-Read $\left.{ }_{\text {sb }}\right)(\mathrm{x} \# \mathrm{xs}) \cap$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$,
hence no-inter': outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{\text {sb }}$ ) xs $\cap$ outstanding-refs
is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$
by (auto)
note consis $=$ reads-consistent pending-write $\mathcal{O} \mathrm{m}(\mathrm{x} \# \mathrm{xs})$ 〉
show ?case
proof (cases x)
case (Write sb volatile a sop v A L R)
show ?thesis
proof (cases volatile)
case False
from consis obtain consis': reads-consistent pending-write $\mathcal{O}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ xs
by (simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF no-inter ' consis']
have reads-consistent pending-write $\mathcal{O}$ (flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ ) xs.
moreover
from no-inter have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb
by (auto simp add: Write ${ }_{\text {sb }}$ split: if-split-asm)
from flush-update-other ${ }^{\prime}$ [OF this no-volatile-Write $\left.{ }_{s b}-\mathrm{sb}\right]$
have $($ flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v})))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$.
ultimately
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ False)
next
case True
from consis obtain consis': reads-consistent True $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ xs and
no-read: (outstanding-refs is-volatile-Read ${ }_{\text {sb }} \mathrm{xs}=\{ \}$ )
by (simp add: Write ${ }_{\text {sb }}$ True)
from Cons.hyps [OF no-inter' consis']
have reads-consistent True $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ (flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ ) xs.
moreover
from no-inter have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}$
by (auto simp add: Write ${ }_{\text {sb }}$ split: if-split-asm)
from flush-update-other ${ }^{\prime}$ [OF this no-volatile-Write ${ }_{\text {sb }}$-sb]
have (flush sb $(\mathrm{m}(\mathrm{a}:=\mathrm{v})))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$.
ultimately
show ?thesis
using no-read
by (simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case ( $\operatorname{Read}_{\mathrm{sb}}$ volatile at v )
from consis obtain val: $(\neg$ volatile $\longrightarrow$ (pending-write $\vee \mathrm{a} \in \mathcal{O}) \longrightarrow \mathrm{v}=\mathrm{m}$ a) and consis': reads-consistent pending-write $\mathcal{O} \mathrm{m}$ xs
by ( simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
from Cons.hyps [OF no-inter' consis']
have hyp: reads-consistent pending-write $\mathcal{O}$ (flush sb m) xs by simp
show ?thesis
proof (cases volatile)
case False
from no-inter False have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ split: if-split-asm)
with no-volatile-Write ${ }_{s b}$-sb
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb
apply (clarsimp simp add: outstanding-refs-conv is-Write ${ }_{\text {sb }}$-def split: memref.splits)
apply force
done
with hyp val flush-unchanged-addresses [OF this]
show ?thesis
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
next
case True
with hyp val show ?thesis
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
qed
next
case Prog $_{\text {sb }}$ with Cons show ?thesis by auto

```
    next
    case Ghost sb with Cons show ?thesis by auto
    qed
qed
lemma reads-consistent-flush-independent:
    assumes no-volatile-Write }\mp@subsup{\mp@code{sb}}{}{\prime}\mathrm{ -sb: outstanding-refs is-Write sb sb }\cap\mathrm{ outstanding-refs
is-non-volatile-Read
    assumes consis: reads-consistent pending-write }\mathcal{O}\textrm{m}\mathrm{ xs
    shows reads-consistent pending-write \mathcal{O}}\mathrm{ (flush sb m) xs
proof -
    from flush-unchanged-addresses [where sb=sb and m=m] no-volatile-Write sb
    have }\forall\textrm{a}\in\mathrm{ outstanding-refs is-non-volatile-Read
        by auto
    from reads-consistent-mem-eq-on-non-volatile-reads [OF this subset-refl consis]
    show ?thesis .
qed
```

lemma reads-consistent-flush-all-until-volatile-write-aux:
assumes no-reads: outstanding-refs is-volatile-Read $\operatorname{Reb}_{\text {xb }}=\{ \}$
shows $\bigwedge \mathrm{m}$ pending-write $\mathcal{O}^{\prime}$. 【reads-consistent pending-write $\mathcal{O}^{\prime} \mathrm{m} \mathrm{xs} ; \forall \mathrm{i}<$ length ts. let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in
outstanding-refs (Not $\circ$ is-volatile-Read $\operatorname{Rb}_{\text {b }}$ ) xs $\cap$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}\left(\right.$ takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb) $=$ \{\}]
$\Longrightarrow$ reads-consistent pending-write $\mathcal{O}^{\prime}$ (flush-all-until-volatile-write ts m) xs
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
have consis: reads-consistent pending-write $\mathcal{O}^{\prime} \mathrm{m}$ xs by fact
obtain $p_{\mathrm{t}} \mathrm{is}_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$
where $\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t)
from Cons.prems t obtain
no-inter: outstanding-refs (Not $\circ$ is-volatile- $\operatorname{Read}_{\text {sb }}$ ) xs $\cap$
outstanding-refs is-non-volatile-Write ${ }_{s b}\left(\right.$ takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\operatorname{sb}_{\mathrm{t}}$ ) $=$
\{\} and
no-inter': $\forall \mathrm{i}<$ length ts.
let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in
outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{\text {sb }}$ ) xs $\cap$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}\left(\right.$ takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)=$
\{\}
by (force simp add: Let-def simp del: o-apply)

```
    have out1: outstanding-refs is-volatile-Write sb
    (takeWhile (Not ○ is-volatile-Write sb})\mp@subsup{\textrm{sb}}{\textrm{t}}{})={
{}
    by auto
    from reads-consistent-flush-other [OF out1 this consis]
m) xs.
    from Cons.hyps [OF this no-inter']
    show ?case
        by (simp add: t)
qed
```

    by (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
    from no-inter have outstanding-refs (Not \(\circ\) is-volatile-Read \({ }_{s b}\) ) xs \(\cap\)
    outstanding-refs is-non-volatile-Write \({ }_{s b} \quad\left(\right.\) takeWhile \(^{(N o t} \circ\) is-volatile-Write \(\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{t}}\right)=\)
    have reads-consistent pending-write \(\mathcal{O}^{\prime}\) (flush (takeWhile (Not o is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{t}}\) )
    lemma reads－consistent－flush－other ${ }^{\prime}$ ：
assumes no－volatile－Write ${ }_{\mathbf{s b}}$－sb：outstanding－refs is－volatile－Write ${ }_{\mathbf{s b}} \mathrm{sb}=\{ \}$
shows $\bigwedge \mathrm{m} \mathcal{O}$ ．
【outstanding－refs is－non－volatile－Write ${ }_{\text {sb }} \mathrm{sb} \cap$
（outstanding－refs is－volatile－Write ${ }_{\text {sb }}$ xs $\cup$
outstanding－refs is－non－volatile－Write ${ }_{s b}$ xs $\cup$
outstanding－refs is－non－volatile－Read ${ }_{\text {sb }}\left(\operatorname{dropWhile}^{(N o t} \circ\right.$ is－volatile－Write $\left.{ }_{\text {sb }}\right) \mathrm{xs}$ ）
$\cup$
（outstanding－refs is－non－volatile－Read ${ }_{\mathbf{s b}}$（takeWhile（Not $\circ$ is－volatile－Write ${ }_{\text {sb }}$ ）xs）
$-\mathrm{RO}) \cup$
$\left(\mathcal{O} \cup\right.$ all－acquired（takeWhile（Not $\circ$ is－volatile－Write $\left.\left.{ }_{s b}\right) \mathrm{xs}\right)$ ）
）$=\{ \}$ ；
reads－consistent False $\mathcal{O}$ m xs；
read－only－reads $\mathcal{O}$（takeWhile（Not $\circ$ is－volatile－Write ${ }_{\text {sb }}$ ）xs）$\subseteq \mathrm{RO} \rrbracket$
$\Longrightarrow$ reads－consistent False $\mathcal{O}$（flush sb m）xs
proof（induct xs）
case Nil thus ？case by simp
next
case（Cons x xs）
note no－inter $=$ Cons．prems（1）
note consis $=$ 〈reads－consistent False $\mathcal{O} \mathrm{m}(\mathrm{x} \# \mathrm{xs})\rangle$
have aargh：（Not $\circ$ is－volatile－Write $\left.{ }_{\mathbf{s b}}\right)=\left(\lambda a . \neg\right.$ is－volatile－Write $\left._{\mathbf{s b}} \mathrm{a}\right)$
by（rule ext）auto
note $\mathrm{RO}=$ 〔read－only－reads $\mathcal{O}$（takeWhile（Not $\circ$ is－volatile－Write $\left.\left.{ }_{\mathrm{sb}}\right)(\mathrm{x} \# \mathrm{xs})\right) \subseteq \mathrm{RO}$ 〉

```
show ?case
proof (cases x)
    case (Writesb volatile a sop v A L R)
    show ?thesis
    proof (cases volatile)
    case False
    from consis obtain consis': reads-consistent False \mathcal{O}}(\textrm{m}(\textrm{a}:=\textrm{v})) x
by (simp add: Writesb False)
    from no-inter
    have no-inter': outstanding-refs is-non-volatile-Write 
    (outstanding-refs is-volatile-Write }\mp@subsup{\mathrm{ sb }}{}{xs}
        outstanding-refs is-non-volatile-Write
        outstanding-refs is-non-volatile-Read
U
        (outstanding-refs is-non-volatile-Read }\mp@subsup{\mp@code{sb}}{}{(takeWhile (Not o is-volatile-Write
- RO)
    (\mathcal{O}\cup all-acquired (takeWhile (Not \circ is-volatile-Write 
    )={}
by (clarsimp simp add: Write sb False aargh)
```

from RO
have $\mathrm{RO}^{\prime}$ : read-only-reads $\mathcal{O}$ (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{xs}\right) \subseteq \mathrm{RO}$
by (auto simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF no-inter' consis' RO ]
have reads-consistent False $\mathcal{O}$ (flush sb (m(a $:=\mathrm{v}))$ ) xs.
moreover
from no-inter have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb
by (auto simp add: Write ${ }_{\text {sb }}$ split: if-split-asm)
from flush-update-other ${ }^{\prime}$ [OF this no-volatile-Write sb $^{\mathbf{s b}}$-sb]
have (flush $\operatorname{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v})))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$.
ultimately
show ?thesis
by (simp add: Write sb $^{\text {b }}$ False)
next
case True
from consis obtain consis': reads-consistent $\operatorname{True}(\mathcal{O} \cup A-R)(m(a:=v))$ xs and no-read: (outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{xs}=\{ \}$ )
by (simp add: Write ${ }_{\text {sb }}$ True)

## from no-inter obtain

a-notin: a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb and
disj: (outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{s b}$ ) xs) $\cap$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$
by (auto simp add: Write ${ }_{\text {sb }}$ True aargh misc-outstanding-refs-convs)
from reads-consistent-flush-other [OF no-volatile-Write ${ }_{\text {sb }}$-sb disj consis ${ }^{\prime}$ ]
have reads-consistent $\operatorname{True}(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ (flush $\mathrm{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ ) xs.
moreover
note a-notin
from flush-update-other ${ }^{\prime}$ [OF this no-volatile-Write sb $^{\mathbf{s}}$-sb]
have (flush $\operatorname{sb}(\mathrm{m}(\mathrm{a}:=\mathrm{v})))=($ flush sb m$)(\mathrm{a}:=\mathrm{v})$.
ultimately
show ?thesis
using no-read
by (simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case $\left(\operatorname{Read}_{\text {sb }}\right.$ volatile at v$)$
from consis obtain val: $(\neg$ volatile $\longrightarrow \mathrm{a} \in \mathcal{O} \longrightarrow \mathrm{v}=\mathrm{m}$ a) and
consis': reads-consistent False $\mathcal{O}$ m xs
by ( $\operatorname{simp}$ add: $\operatorname{Read}_{\mathrm{sb}}$ )
from RO
have $\mathrm{RO}^{\prime}$ : read-only-reads $\mathcal{O}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) xs) $\subseteq \mathrm{RO}$
by (auto simp add: Read ${ }_{\text {sb }}$ )

## from no-inter

have no-inter': outstanding-refs is-non-volatile-Write ${ }_{s b} \mathrm{sb} \cap$
(outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Write ${ }_{s b}$ xs $\cup$

(outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}\left(\right.$ takeWhile (Not $\circ$ is-volatile-Write $\left.\left._{\text {sb }}\right) \mathrm{xs}\right)-$
RO) $\cup$
$\left(\mathcal{O} \cup\right.$ all-acquired $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \mathrm{xs}\right)\right)$
) $=\{ \}$
by (fastforce simp add: Read ${ }_{\text {sb }}$ aargh)
show ?thesis
proof (cases volatile) case True
from Cons.hyps [OF no-inter ${ }^{\prime}$ consis ${ }^{\prime} \mathrm{RO}^{\prime}$ ]
show ?thesis
by (simp add: Read sb $_{\text {b }}$ True)
next
case False
note non-volatile=this
from Cons.hyps [OF no-inter' consis' $\left.{ }^{\prime} \mathrm{RO}^{\dagger}\right]$
have hyp: reads-consistent False $\mathcal{O}$ (flush sb m) xs.
show ?thesis
proof (cases a $\in \mathcal{O}$ )
case False
with hyp show? thesis by (simp add: $\operatorname{Read}_{\text {sb }}$ non-volatile False)
next
case True
from no-inter True have a-notin: a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb by blast
with no-volatile-Write ${ }_{s b}$-sb
have a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb
apply (clarsimp simp add: outstanding-refs-conv is-Write ${ }_{\text {sb }}$-def split: memref.splits)
apply force
done
from flush-unchanged-addresses [OF this] hyp val
show ?thesis
by (simp add: Read $_{\mathrm{sb}}$ non-volatile True)
qed
qed
next
case $\operatorname{Prog}_{s b}$ with Cons show ?thesis
by auto
next
case ( Ghost $_{\text {sb }}$ A L R W)
from consis obtain consis': reads-consistent False $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{m}$ xs
by (simp add: Ghost ${ }_{\text {sb }}$ )
from RO
have $\mathrm{RO}^{\prime}$ : read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{xs}\right) \subseteq$ RO
by (auto simp add: Ghost $_{\mathbf{s b}}$ )
from no-inter
have no-inter': outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \cap$
(outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Write $_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) xs)
$\cup$ (outstanding-refs is-non-volatile-Read ${ }_{\mathbf{s b}}\left(\operatorname{takeWhile}^{(N o t} \circ\right.$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{xs}\right)$

- RO)
$\left(\mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup\right.$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) xs))
) $=\{ \}$
by (fastforce simp add: Ghost sb $_{\text {b }}$ aargh)
from Cons.hyps [OF no-inter' consis' $\mathrm{RO}^{\prime}$ ]
show ?thesis
by (clarsimp simp add: Ghostsb)

```
    qed
```

qed
lemma reads-consistent-flush-all-until-volatile-write-aux':
assumes no-reads: outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{xs}=\{ \}$
assumes read-only-reads-RO: read-only-reads $\mathcal{O}^{\prime}$ (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{xs}) \subseteq \mathrm{RO}$
shows $\wedge \mathrm{m}$. $\llbracket$ reads-consistent False $\mathcal{O}^{\prime} \mathrm{m}$ xs; $\forall \mathrm{i}<$ length ts.
let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O})=\mathrm{ts}!\mathrm{i}$ in
outstanding-refs is-non-volatile-Write ${ }_{\mathbf{s b}}\left(\right.$ takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right) \cap$
(outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}\left(\operatorname{dropWhile}^{(N o t} \circ\right.$ is-volatile-Write $\left.{ }_{\text {sb }}\right) \mathrm{xs}$ )
$\cup$
(outstanding-refs is-non-volatile-Read ${ }_{\mathbf{s b}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) xs)
$-\mathrm{RO}) \cup$
$\left(\mathcal{O}^{\prime} \cup\right.$ all-acquired $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \mathrm{xs}\right)\right)$
)
$=\{ \}$
】
$\Longrightarrow$ reads-consistent False $\mathcal{O}^{\prime}$ (flush-all-until-volatile-write ts m) xs
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
have consis: reads-consistent False $\mathcal{O}^{\prime} \mathrm{m}$ xs by fact
obtain $p_{\mathrm{t}} \mathrm{is}_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$
where $\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t)

## obtain

no-inter: outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\left.\mathrm{sb}_{\mathrm{t}}\right) \cap$
(outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Read ${ }_{s b}$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) xs)
$\cup$
(outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) xs)
$-\mathrm{RO}) \cup$
$\left(\mathcal{O}^{\prime} \cup\right.$ all-acquired $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{s b}\right) \mathrm{xs}\right)\right)$
)
$=\{ \}$ and
no-inter ${ }^{\prime}: \forall \mathrm{i}<$ length ts.
let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O})=\mathrm{ts}!\mathrm{i}$ in
outstanding-refs is-non-volatile-Write ${ }_{s b}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap$ (outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs $\cup$
outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ xs $\cup$

```
            outstanding-refs is-non-volatile-Read sb (dropWhile (Not ० is-volatile-Write sb) xs)
U
            (outstanding-refs is-non-volatile-Read sb (takeWhile (Not O is-volatile-Write sb
- RO)
            (\mp@subsup{\mathcal{O}}{}{\prime}\cup\mathrm{ all-acquired (takeWhile (Not o is-volatile-Write sb})\textrm{xs}))
        )
    ={}
    proof -
    show ?thesis
        apply (rule that)
        using Cons.prems (2) [rule-format, of 0]
        apply (clarsimp simp add: t)
        apply clarsimp
        using Cons.prems (2)
        apply -
        subgoal for i
        apply (drule-tac x=Suc i in spec)
        apply (clarsimp simp add: Let-def simp del: o-apply)
        done
        done
    qed
    have out1: outstanding-refs is-volatile-Write}\mp@subsup{e}{\mathrm{ sb}}{
        (takeWhile (Not ○ is-volatile-Write sb
        by (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
    from reads-consistent-flush-other' [OF out1 no-inter consis read-only-reads-RO ]
    have reads-consistent False }\mp@subsup{\mathcal{O}}{}{\prime}\mathrm{ (flush (takeWhile (Not o is-volatile-Writesb}\mp@subsup{)}{sb}{})\mp@subsup{\textrm{sb}}{\textrm{t}}{}\mathrm{ ) m) xs.
    from Cons.hyps [OF this no-inter]
    show ?case
    by (simp add: t)
qed
```

lemma in-outstanding-refs-cases [consumes 1, case-names Write ${ }_{\text {sb }} \operatorname{Read}_{\text {sb }}$ ]:
$\mathrm{a} \in$ outstanding-refs $\mathrm{P} \mathrm{xs} \Longrightarrow$
( $\bigwedge$ volatile sop v A L R W. $\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $\in$ set xs $\Longrightarrow P$
$\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W) $\left.\Longrightarrow \mathrm{C}\right) \Longrightarrow$
$\left(\bigwedge\right.$ volatile $\mathrm{t} \mathrm{v} .\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v$) \in$ set $\mathrm{xs} \Longrightarrow \mathrm{P}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v$\left.) \Longrightarrow \mathrm{C}\right)$
$\Longrightarrow$ C
apply (clarsimp simp add: outstanding-refs-conv)
subgoal for x
apply (case-tac x)
apply fastforce+

## done

## done

lemma dropWhile-Cons: $($ dropWhile $\mathrm{P} x s)=\mathrm{x} \# \mathrm{ys} \Longrightarrow \neg \mathrm{P} \mathrm{x}$
apply (induct xs)
apply (auto split: if-split-asm)
done
lemma reads-consistent-dropWhile:
reads-consistent pending-write $\mathcal{O} \mathrm{m}$ (dropWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)=$ reads-consistent True $\mathcal{O} \mathrm{m}$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (case-tac (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb))
apply (simp only:)
apply simp
apply (frule dropWhile-Cons)
apply (auto split: memref.splits)
done

## theorem

reads-consistent-flush-all-until-volatile-write:
$\bigwedge \mathrm{i}$ m pending-write. $\llbracket$ valid-ownership-and-sharing $\mathcal{S}$ ts;
$\mathrm{i}<$ length ts; ts!i $=(\mathrm{p}$, is $, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$
reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb} \rrbracket$
$\Longrightarrow$ reads-consistent True (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O}$ )
(flush-all-until-volatile-write ts m) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb)
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
note i -bound $=\langle i<$ length $(\mathrm{t} \# \mathrm{ts})\rangle$
note $\mathrm{ts}-\mathrm{i}=\langle(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}$, is $, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\rangle$
note consis $=\langle$ reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb}\rangle$
note valid $=$ 〈valid-ownership-and-sharing $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$ )
then interpret valid-ownership-and-sharing $\mathcal{S}$ t\#ts.
from valid-ownership-and-sharing-tl [OF valid] have valid': valid-ownership-and-sharing $\mathcal{S}$ ts.
obtain $\mathrm{p}_{\mathrm{t}}$ is $_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where t : $\mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t )
show ?case
proof (cases i)
case 0
with ts-i t have sb-eq: $s b=s b_{t}$
by simp
let ?take-sb $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$
let ?drop-sb $=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$
from reads-consistent-append [of pending-write $\mathcal{O} \mathrm{m}$ ?take-sb ?drop-sb] consis
have consis': reads-consistent True (acquired True ?take-sb $\mathcal{O}$ ) (flush ?take-sb m) ?drop-sb
apply (cases outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ )
sb) $\neq\{ \}$ )
apply clarsimp
apply clarsimp
apply (simp add: reads-consistent-dropWhile [of pending-write]) done
from reads-consistent-drop-volatile-writes-no-volatile-reads [OF consis]
have no-vol-Read ${ }_{\text {sb }}$ : outstanding-refs is-volatile-Read ${ }_{\mathrm{sb}}$ (dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) $=\{ \}$.
hence outstanding-refs (Not o is-volatile-Read ${ }_{\mathbf{s b}}$ ) (dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb)
$=$
outstanding-refs ( $\lambda \mathrm{s}$. True) (dropWhile (Not $\circ$ is-volatile-Write $\mathrm{e}_{\mathrm{sb}}$ ) sb)
by (auto simp add: outstanding-refs-conv)
have $\forall \mathrm{i}<$ length ts.
let $\left(\mathrm{p}\right.$, is $\left., \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)=\mathrm{ts}!\mathrm{i}$
in outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{s b}$ ) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap$
outstanding-refs is-non-volatile-Write $\mathrm{s}_{\mathrm{sb}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)
$=\{ \}$
proof -
\{
fix $\mathrm{jp}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}$
assume j-bound: $\mathrm{j}<$ length ts
assume ts-j: ts! $=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{isj}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume x-in-sb: x $\in$ outstanding-refs (Not $\circ$ is-volatile-Read ${ }_{s b}$ ) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
assume x -in- $\mathrm{j}: \mathrm{x} \in$ outstanding-refs is-non-volatile-Write ${ }_{\mathrm{sb}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $_{j}$ )
have False
proof -
from outstanding-non-volatile-write-not-volatile-read-disj [rule-format, of Suc j 0, simplified, OF j-bound ts-j t]
sb-eq $x$-in-sb $x-i n-j$
show ?thesis
by auto
qed
\}
thus ?thesis
by (auto simp add: Let-def)
qed
from reads-consistent-flush-all-until-volatile-write-aux [OF no-vol-Read ${ }_{\text {sb }}$ consis' this] show ?thesis
by (simp add: t sb-eq del: o-apply)

## next

```
    case (Suc k)
```

    with i-bound have k -bound: \(\mathrm{k}<\) length ts
        by auto
    from ts-i Suc have ts-k: ts \(!\mathrm{k}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
        by simp
    ```
    have reads-consistent False \(\mathcal{O}\) (flush (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{t}}\) ) m ) sb
```

    proof -
    have no-vW:
    outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ) $=\{ \}$
apply (clarsimp simp add: outstanding-refs-conv )
apply (drule set-takeWhileD)
apply simp
done
from consis have consis': reads-consistent False $\mathcal{O} \mathrm{m} \mathrm{sb}$
by (cases pending-write) (auto intro: reads-consistent-pending-write-antimono)
note disj $=$ outstanding-non-volatile-write-disj [where $\mathrm{i}=0$, OF - i-bound [simplified Suc], simplified, OF t ts-k ]
from reads-consistent-flush-other ${ }^{\prime}$ [OF no-vW disj consis' subset-refl] show ?thesis .
qed
from Cons.hyps [OF valid' k-bound ts-k this]
show ?thesis
by (simp add: t)
qed
qed
lemma split-volatile-Write ${ }_{\text {sb }}$-in-outstanding-refs:
$\mathrm{a} \in$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{xs} \Longrightarrow\left(\exists\right.$ sop v ys zs ALRW. xs $=$ ys@ $\left(\right.$ Write $_{\text {sb }}$
True a sop v A L R W\#zs))
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
have a-in: a $\in$ outstanding-refs is-volatile-Write ${ }_{s b}$ ( $\mathrm{x} \# \mathrm{xs}$ ) by fact
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case False
from a-in have $a \in$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs by (auto simp add: False Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop " $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where $\mathrm{xs}=\mathrm{ys} @ W$ Write $_{\text {sb }}$ True a sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ W r i t e_{\text {sb }}$ True a sop" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
by blast
next
case True
note volatile $=$ this
show ?thesis
proof (cases $\mathrm{a}^{\prime}=\mathrm{a}$ )
case False
with a-in have a $\in$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ xs by (auto simp add: volatile Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop " $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where xs=ys@Write ${ }_{\text {sb }}$ True a sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ W r i t e_{\text {sb }}$ True a sop" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ? thesis
by blast
next
case True
then have $\mathrm{x} \# \mathrm{xs}=[] @\left(\right.$ Write $_{\text {sb }}$ True a sop v A L R W\#xs)
by (simp add: Write ${ }_{\text {sb }}$ volatile True)
thus ?thesis
by blast
qed
qed
next
case Read ${ }_{\text {sb }}$
from a-in have $a \in$ outstanding-refs is-volatile-Write $_{\text {sb }}$ xs by (auto simp add: Read ${ }_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where xs=ys@Write ${ }_{\text {sb }}$ True a sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ W r i t e_{\text {sb }}$ True a sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
thus ?thesis
by blast
next
case Prog $_{\text {sb }}$
from a-in have a $\in$ outstanding-refs is-volatile-Write $_{\text {sb }}$ xs by (auto simp add: Prog $_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where $\mathrm{xs}=\mathrm{ys} @ W$ Write $_{\text {sb }}$ True a sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto

```
    hence x#xs = (x#ys)@Write sb True a sop" v" A" L" R" W"#zs
        by auto
    thus ?thesis
        by blast
    next
    case Ghostsb
    from a-in have a \in outstanding-refs is-volatile-Write sb xs
        by (auto simp add: Ghostsb)
    from Cons.hyps [OF this] obtain sop" v" A" L" R" W" ys zs
        where xs=ys@Write sb True a sop" v" A" L" R" W"#zs
        by auto
    hence x#xs = (x#ys)@Write sb True a sop" v" A" L" R" W"#zs
        by auto
    thus ?thesis
        by blast
    qed
qed
lemma sharing-consistent-mono-shared:
\(\wedge \mathcal{S} \mathcal{S}^{\prime} \mathcal{O}\).
dom \(\mathcal{S} \subseteq \operatorname{dom} \mathcal{S}^{\prime} \Longrightarrow\) sharing-consistent \(\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow\) sharing-consistent \(\mathcal{S}^{\prime} \mathcal{O}\) sb
apply (induct sb)
apply simp
subgoal for \(\operatorname{abb} \mathcal{S} \mathcal{S}^{\prime} \mathcal{O}\)
apply (case-tac a)
apply clarsimp
subgoal for volatile a D f v A L R W
apply (frule-tac \(\mathrm{A}=\mathcal{S}\) and \(\mathrm{B}=\mathcal{S}^{\prime}\) and \(\mathrm{C}=\mathrm{R}\) and \(\mathrm{x}=\mathrm{W}\) in augment-mono-aux)
apply (frule-tac \(\mathrm{A}=\mathcal{S} \oplus \mathrm{w} \mathrm{R}\) and \(\mathrm{B}=\mathcal{S}^{\prime} \oplus \mathrm{w} \quad \mathrm{R}\) and \(\mathrm{C}=\mathrm{L}\) in restrict-mono-aux)
apply blast
done
apply clarsimp
apply clarsimp
apply clarsimp
subgoal for A L R W
apply (frule-tac \(\mathrm{A}=\mathcal{S}\) and \(\mathrm{B}=\mathcal{S}^{\prime}\) and \(\mathrm{C}=\mathrm{R}\) and \(\mathrm{x}=\mathrm{W}\) in augment-mono-aux)
apply (frule-tac \(\mathrm{A}=\mathcal{S} \oplus \mathrm{w} \mathrm{R}\) and \(\mathrm{B}=\mathcal{S}^{\prime} \oplus \mathrm{w} \quad \mathrm{R}\) and \(\mathrm{C}=\mathrm{L}\) in restrict-mono-aux)
apply blast
done
done
done
lemma sharing-consistent-mono-owns:
\(\wedge \mathcal{O} \mathcal{O}^{\prime} \mathcal{S}\).
\(\mathcal{O} \subseteq \mathcal{O}^{\prime} \Longrightarrow\) sharing-consistent \(\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow\) sharing-consistent \(\mathcal{S} \mathcal{O}^{\prime}\) sb
apply (induct sb)
apply simp
subgoal for a \(\operatorname{sb} \mathcal{O} \mathcal{O}^{\prime} \mathcal{S}\)
apply (case-tac a)
apply clarsimp
```

```
    subgoal for volatile a D f v A L R W
    apply (frule-tac }\textrm{A}=\mathcal{O}\mathrm{ and }\textrm{B}=\mp@subsup{\mathcal{O}}{}{\prime}\mathrm{ and }\textrm{C}=\textrm{A}\mathrm{ in union-mono-aux)
    apply (frule-tac }\textrm{A}=\mathcal{O}\cup\textrm{A}\mathrm{ and }\textrm{B}=\mp@subsup{\mathcal{O}}{}{\prime}\cup\textrm{A}\mathrm{ and }\textrm{C}=\textrm{R}\mathrm{ in set-minus-mono-aux)
    apply fastforce
    done
apply clarsimp
apply clarsimp
apply clarsimp
subgoal for A L R W
apply (frule-tac }\textrm{A}=\mathcal{O}\mathrm{ and }\textrm{B}=\mp@subsup{\mathcal{O}}{}{\prime}\mathrm{ and }\textrm{C}=\textrm{A}\mathrm{ in union-mono-aux)
apply (frule-tac }\textrm{A}=\mathcal{O}\cup\textrm{A}\mathrm{ and }\textrm{B}=\mp@subsup{\mathcal{O}}{}{\prime}\cup\textrm{A}\mathrm{ and }\textrm{C}=\textrm{R}\mathrm{ in set-minus-mono-aux)
apply fastforce
done
done
done
primrec all-shared :: 'a memref list }=>\mathrm{ addr set
where
    all-shared [] = {}
| all-shared (i#is) =
    (case i of
        Write sb volatile -- A L R W }=>\mathrm{ (if volatile then R U all-shared is else all-shared is)
        |host sb A L R W m R U all-shared is
        |- }=>\mathrm{ all-shared is)
lemma sharing-consistent-all-shared:
    \bigwedge\mathcal{S}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\textrm{sb}\Longrightarrow\mathrm{ all-shared sb }\subseteq\operatorname{dom S}\cup\mathcal{O}
    apply (induct sb)
    apply clarsimp
    subgoal for a
    apply (case-tac a)
    apply (fastforce split: memref.splits if-split-asm)
    apply clarsimp
    apply clarsimp
    apply fastforce
    done
    done
lemma sharing-consistent-share-all-shared:
    \bigwedge\mathcal{S}.dom (share sb S S dom S \cup all-shared sb
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop t A L R W)
    show ?thesis
```

```
    proof (cases volatile)
        case True
        from Cons.hyps [of (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
        show ?thesis
            by (auto simp add: Write
    next
        case False with Cons Writesb show ?thesis by auto
    qed
next
    case Read
next
    case Prog
next
    case (Ghost sb A L R W)
    from Cons.hyps [of (\mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
    show ?thesis
        by (auto simp add: Ghostsb)
    qed
qed
```

primrec all-unshared :: 'a memref list $\Rightarrow$ addr set where
all-unshared [] $=\{ \}$
| all-unshared (i\#is) =
(case i of
Write $_{\text {sb }}$ volatile -- A L R W $\Rightarrow$ (if volatile then $\mathrm{L} \cup$ all-unshared is else all-unshared
is)
| Ghost ${ }_{\text {sb }} \mathrm{A} L \mathrm{R} \mathrm{W} \Rightarrow \mathrm{L} \cup$ all-unshared is
$\mid-\Rightarrow$ all-unshared is)
lemma all-unshared-append: all-unshared (xs @ ys) = all-unshared xs $\cup$ all-unshared ys
apply (induct xs)
apply simp
subgoal for a
apply (case-tac a)
apply auto
done
done
lemma freshly-shared-owned:
$\bigwedge \mathcal{S} \mathcal{O}$. sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow \operatorname{dom}($ share $\operatorname{sb} \mathcal{S})-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case

```
proof (cases x)
    case (Write sb volatile a sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case False
        with Cons Writesb show ?thesis by auto
    next
        case True
        from Cons.hyps [where \mathcal{S}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]\mathrm{ Cons.prems}
        show ?thesis
by (auto simp add: Writesb True)
    qed
next
    case Read
next
    case Progsb with Cons show ?thesis by auto
next
    case (Ghost sb A L R W)
    with Cons.hyps [where S = (\mathcal{S}\oplus\textrm{w}R}\mp@subsup{\textrm{R}}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]\mathrm{ Cons.prems show
?thesis by auto
    qed
qed
lemma unshared-all-unshared:
    \bigwedge\mathcal{S}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\textrm{sb}\Longrightarrow\operatorname{dom}\mathcal{S}-\operatorname{dom}(\mathrm{ share sb S S }\subseteq\mathrm{ all-unshared sb}
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
        case False
        with Cons Writesb show ?thesis by auto
    next
        case True
        from Cons.hyps [where \mathcal{S}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]}\mathrm{ Cons.prems
        show ?thesis
    by (auto simp add: Writesb True)
        qed
    next
        case Read
    next
        case Prog
    next
        case (Ghost sb A L R W)
        with Cons.hyps [where S}=(\mathcal{S}\oplus\textrm{w}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and }\mathcal{O}=(\mathcal{O}\cup\textrm{A}-\textrm{R})]\mathrm{ Cons.prems show
?thesis by auto
```

```
    qed
```

qed
lemma unshared-acquired-or-owned:
$\bigwedge \mathcal{S} \mathcal{O}$. sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow$ all-unshared sb $\subseteq$ all-acquired sb $\cup \mathcal{O}$
apply (induct sb)
apply simp
subgoal for a
apply (case-tac a)
apply auto+
done
done
lemma all-shared-acquired-or-owned:

```
\mathcal{S}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\textrm{sb}\Longrightarrow\mathrm{ all-shared sb }\subseteq\mathrm{ all-acquired sb }\cup\mathcal{O}
apply (induct sb)
apply simp
subgoal for a
apply (case-tac a)
apply auto+
done
done
```

lemma sharing-consistent-preservation:
$\bigwedge \mathcal{S} \mathcal{S}^{\prime} \mathcal{O}$.
«sharing-consistent $\mathcal{S} \mathcal{O}$ sb;
all-acquired $\mathrm{sb} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} \mathcal{S}^{\prime}=\{ \} ;$
all-unshared $\mathrm{sb} \cap \operatorname{dom} \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \} \rrbracket$
$\Longrightarrow$ sharing-consistent $\mathcal{S}^{\prime} \mathcal{O}$ sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
have consis: sharing-consistent $\mathcal{S} \mathcal{O}$ (x \# sb) by fact
have removed-cond: all-acquired $(\mathrm{x} \# \mathrm{sb}) \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} \mathcal{S}^{\prime}=\{ \}$ by fact
have new-cond: all-unshared $(\mathrm{x} \# \mathrm{sb}) \cap \operatorname{dom} \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$ by fact
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case False with Write $_{\text {sb }}$ Cons show ?thesis
by auto
next
case True
from consis obtain
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and
$\mathrm{L}: \mathrm{L} \subseteq \mathrm{A}$ and
$A-R: A \cap R=\{ \}$ and
$\mathrm{R}: \mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from removed-cond obtain rem-cond: (A $\cup$ all-acquired sb$) \cap \operatorname{dom} \mathcal{S} \subseteq \operatorname{dom} \mathcal{S}^{\prime}$ by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
hence rem-cond': all-acquired $\mathrm{sb} \cap \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=$ \{\}
by auto
from new-cond obtain (L $\cup$ all-unshared sb$) \cap \operatorname{dom} \mathcal{S}^{\prime} \subseteq \operatorname{dom} \mathcal{S}$ by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
hence new-cond': all-unshared $\operatorname{sb} \cap \operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$ by auto
from Cons.hyps [OF consis' rem-cond' new-cond']
have sharing-consistent $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$.
moreover
from A rem-cond have $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S}^{\prime} \cup \mathcal{O}$
by auto
moreover note L A-R R
ultimately show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case (Ghost sb $_{\text {sb }}$ A R W)
from consis obtain
$\mathrm{A}: \mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and
$\mathrm{L}: \mathrm{L} \subseteq \mathrm{A}$ and
$\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
$\mathrm{R}: \mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{W} R \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
from removed-cond obtain rem-cond: (A $\cup$ all-acquired sb$) \cap \operatorname{dom} \mathcal{S} \subseteq \operatorname{dom} \mathcal{S}^{\prime}$ by (clarsimp simp add: Ghostsb)
hence rem-cond': all-acquired $\mathrm{sb} \cap \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\{ \}$
by auto
from new-cond obtain $(\mathrm{L} \cup$ all-unshared sb$) \cap \operatorname{dom} \mathcal{S}^{\prime} \subseteq \operatorname{dom} \mathcal{S}$ by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
hence new-cond': all-unshared $\mathrm{sb} \cap \operatorname{dom}\left(\mathcal{S}^{\prime} \oplus \mathrm{w} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=$ \{\}
by auto
from Cons.hyps [OF consis' rem-cond' new-cond ']
have sharing-consistent $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$.

```
    moreover
    from A rem-cond have A}\subseteq\operatorname{dom}\mp@subsup{\mathcal{S}}{}{\prime}\cup\mathcal{O
        by auto
    moreover note L A-R R
    ultimately show ?thesis
    by (auto simp add: Ghostsb)
qed (insert Cons, auto)
qed
lemma (in sharing-consis) sharing-consis-preservation:
assumes dist:
    i < length ts. let (-,-,-,sb,-,-,-) = ts!i in
        all-acquired sb \cap dom S - dom S'S
{}
shows sharing-consis }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts
proof
    fix i p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\textrm{sb
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show sharing-consistent }\mp@subsup{\mathcal{S}}{}{\prime}\mathcal{O}\mathrm{ sb
    proof -
        from sharing-consis [OF i-bound ts-i]
        have consis: sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb.
        from dist [rule-format, OF i-bound] ts-i
        obtain
            acq: all-acquired sb \cap dom S - dom }\mp@subsup{\mathcal{S}}{}{\prime}={}\mathrm{ and
            uns: all-unshared sb \cap dom S'S
            by auto
        from sharing-consistent-preservation [OF consis acq uns]
        show ?thesis .
    qed
qed
lemma (in sharing-consis) sharing-consis-shared-exchange:
assumes dist:
    \foralli < length ts. let (-,-,-,sb,-,-,-) = ts!i in
    \foralla}\in\mathrm{ all-acquired sb. S'S
shows sharing-consis }\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ ts
proof
    fix i p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\textrm{sb
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
show sharing-consistent }\mp@subsup{\mathcal{S}}{}{\prime}\mathcal{O}\mathrm{ sb
proof -
    from sharing-consis [OF i-bound ts-i]
    have consis: sharing-consistent }\mathcal{S}\mathcal{O}\textrm{sb}
    from dist [rule-format, OF i-bound] ts-i
    obtain
        dist-sb: }\forall\textrm{a}\in\mathrm{ all-acquired sb. }\mp@subsup{\mathcal{S}}{}{\prime}\textrm{a}=\mathcal{S}\textrm{S
        by auto
```

```
    from sharing-consistent-shared-exchange [OF dist-sb consis]
    show ?thesis .
    qed
qed
```

lemma all-acquired-takeWhile: all-acquired (takeWhile P sb) $\subseteq$ all-acquired sb
proof -
from all-acquired-append [of takeWhile P sb dropWhile P sb]
show ?thesis
by auto
qed
lemma all-acquired-dropWhile: all-acquired (dropWhile P sb) $\subseteq$ all-acquired sb
proof -
from all-acquired-append [of takeWhile P sb dropWhile P sb]
show ?thesis
by auto
qed
lemma acquired-share-owns-shared:
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows acquired pending-write $\operatorname{sb} \mathcal{O} \cup \operatorname{dom}(\operatorname{share} \operatorname{sb} \mathcal{S}) \subseteq \mathcal{O} \cup \operatorname{dom} \mathcal{S}$
proof -
from acquired-all-acquired have acquired pending-write sb $\mathcal{O} \subseteq \mathcal{O} \cup$ all-acquired sb.
moreover
from sharing-consistent-all-acquired [OF consis] have all-acquired sb $\subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$.
moreover
from sharing-consistent-share-all-shared have $\operatorname{dom}($ share sb $\mathcal{S}) \subseteq \operatorname{dom} \mathcal{S} \cup$ all-shared
sb.
moreover
from sharing-consistent-all-shared [OF consis] have all-shared sb $\subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$.
ultimately
show ?thesis
by blast
qed
lemma acquired-owns-shared:
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows acquired True sb $\mathcal{O} \subseteq \mathcal{O} \cup \operatorname{dom} \mathcal{S}$
using acquired-share-owns-shared [OF consis]
by blast
lemma share-owns-shared:
assumes consis: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
shows dom (share sb $\mathcal{S}$ ) $\subseteq \mathcal{O} \cup$ dom $\mathcal{S}$
using acquired-share-owns-shared [OF consis]
by blast
lemma all-shared-append: all-shared (xs@ys) $=$ all-shared xs $\cup$ all-shared ys by (induct xs) (auto split: memref.splits)
lemma acquired-union-notin-first:
$\bigwedge$ pending-write AB . $\mathrm{a} \in$ acquired pending-write $\mathrm{sb}(\mathrm{A} \cup \mathrm{B}) \Longrightarrow \mathrm{a} \notin \mathrm{A} \Longrightarrow \mathrm{a} \in$ acquired pending-write sb B
proof (induct sb)
case Nil thus ?case by (auto split: if-split-asm)
next
case (Cons x sb)
then obtain a-notin-A: a $\notin \mathrm{A}$ and
a-acq: a $\in$ acquired pending-write $(x \# s b)(A \cup B)$
by blast
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A' L R W)
show ?thesis
proof (cases volatile)
case False
with Write ${ }_{\text {sb }}$ Cons show ?thesis by simp
next
case True
note volatile $=$ this
show ?thesis
proof (cases pending-write)
case True
from a-acq have a-acq': a $\in$ acquired True $s b\left(A \cup B \cup A^{\prime}-R\right)$ by (simp add: Write ${ }_{\text {sb }}$ volatile True)
have $\left(A \cup B \cup A^{\prime}-R\right) \subseteq\left(A \cup\left(B \cup A^{\prime}-R\right)\right)$ by auto
from acquired-mono-in [OF a-acq' this]
have $a \in$ acquired True sb $\left(A \cup\left(B \cup A^{\prime}-R\right)\right)$.
from Cons.hyps [OF this a-notin-A]
have $a \in$ acquired True sb $\left(B \cup A^{\prime}-R\right)$.
then
show ?thesis by (simp add: Write ${ }_{\text {sb }}$ volatile True)
next
case False
from a-acq have a-acq': a $\in$ acquired True sb ( $\mathrm{A}^{\prime}-\mathrm{R}$ ) by (simp add: Write ${ }_{\text {sb }}$ volatile False)
then
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ volatile False)
qed
qed
next
case (Ghost ${ }_{\text {sb }}$ A $^{\prime}$ L R W)
show ?thesis
proof (cases pending-write)

```
    case True
    from a-acq have a-acq': a }\in\mathrm{ acquired True sb (A }\cup\textrm{B}\cup\mp@subsup{A}{}{\prime}-R
        by (simp add: Ghostsb True)
    have }(A\cupB\cup\mp@subsup{A}{}{\prime}-R)\subseteq(A\cup(B\cup\mp@subsup{A}{}{\prime}-R)
        by auto
    from acquired-mono-in [OF a-acq' this]
    have a }\in\mathrm{ acquired True sb (A U (B U A' - R)).
    from Cons.hyps [OF this a-notin-A]
    have a }\in\mathrm{ acquired True sb (B U A'}-R)
    then
    show ?thesis by (simp add: Ghostsb True)
    next
    case False
    from a-acq have a-acq': a }\in\mathrm{ acquired False sb (A U B)
by (simp add: Ghostsb False)
    from Cons.hyps [OF this a-notin-A]
    show ?thesis
by (simp add: Ghostsb False)
    qed
    qed (insert Cons, auto)
qed
```

lemma split-all-acquired-in:
$\mathrm{a} \in$ all-acquired $\mathrm{xs} \Longrightarrow$
$\left(\exists\right.$ sop $a^{\prime}$ v ys zs A L R W. xs = ys @ Write ${ }_{\text {sb }}$ True $a^{\prime}$ sop v A L R W\# zs $\left.\wedge a \in A\right) \vee$
$\left(\exists \mathrm{A} L \mathrm{R} \mathrm{W}\right.$ ys zs. xs $=\mathrm{ys} @$ Ghost $\left._{\text {sb }} \mathrm{A} L \mathrm{R} \mathrm{W} \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}\right)$
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
have a-in: a $\in$ all-acquired ( $\mathrm{x} \# \mathrm{xs}$ ) by fact
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case False
from a-in have a $\in$ all-acquired xs
by (auto simp add: False Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF this]
have ( $\exists$ sop $\mathrm{a}^{\prime}$ v ys zs A L R W. xs $=$ ys @ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v A L R W\# zs $\wedge$ a
$\in A) V$
$\left(\exists \mathrm{A} L \mathrm{R} \mathrm{W}\right.$ ys zs. xs $=$ ys @ $\left.\mathrm{Ghost}_{\mathrm{sb}} \mathrm{A} L \mathrm{R} W \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}\right)$ (is ?write $\vee$ ?ghst). then
show ?thesis
proof
assume ? write
then
obtain sop " $a^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs
where xs=ys@Writesb True $a^{\prime \prime} \operatorname{sop}^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime} \# z s$ and $a-i n: a \in A^{\prime \prime}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in
by blast
next
assume ?ghst
then obtain $\mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where
xs=ys@Ghost ${ }_{\text {sb }} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime} \# z s$ and $a-i n: a \in A^{\prime \prime}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \operatorname{Ghost}_{\mathrm{sb}} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ? thesis
using a-in
by blast
qed
next
case True
note volatile $=$ this
show ?thesis
proof (cases a $\in A$ )
case False
with a-in have $a \in$ all-acquired xs
by (auto simp add: volatile Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF this]
have $\left(\exists\right.$ sop $\mathrm{a}^{\prime}$ v ys zs A L R W. xs = ys @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v A L R W \# zs $\wedge \mathrm{a} \in$ A) $\vee$
$\left(\exists \mathrm{A} L \mathrm{R} W\right.$ ys zs. xs $=\mathrm{ys} @$ Ghost $\left._{\mathrm{sb}} \mathrm{A} L \mathrm{R} W \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}\right)(\mathrm{is}$ ?write $\vee$
?ghst).
then
show ?thesis
proof
assume ?write
then
obtain sop" $a^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs
where xs=ys@Write sb True $a^{\prime \prime} \operatorname{sop}^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime}$ \#zs and $a-i n: ~ a \in A^{\prime \prime}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in

```
    by blast
next
    assume ?ghst
    then obtain }\mp@subsup{A}{}{\prime\prime}\mp@subsup{L}{}{\prime\prime}\mp@subsup{R}{}{\prime\prime}\mp@subsup{W}{}{\prime\prime}\mathrm{ ys zs where
```



```
        by auto
    hence x#xs = (x#ys)@Ghostsb A" L'/ R" W'#zs
    by auto
    thus ?thesis
        using a-in
        by blast
qed
    next
case True
then have x#xs=[]@(Write sb True a'sop v A L R W#xs)
    by (simp add: Write sb volatile True)
thus ?thesis
    using True
    by blast
        qed
    qed
    next
    case Readsb
    from a-in have a }\in\mathrm{ all-acquired xs
        by (auto simp add: Readsb)
    from Cons.hyps [OF this]
    have (\existssop a'v ys zs A L R W. xs = ys @ Writesb True a'sop v A L R W# zs ^a G
A) V
            (\exists A L R W ys zs. xs = ys @ Ghostsb A L R W# zs ^a\inA) (is ?write \vee ?ghst).
    then
    show ?thesis
    proof
        assume ?write
        then
        obtain sop" a" v" A " L" R" W" ys zs
where xs=ys@Writesb True a'" sop" }\mp@subsup{}{}{\prime\prime}\mp@subsup{v}{}{\prime\prime}\mp@subsup{A}{}{\prime\prime}\mp@subsup{L}{}{\prime\prime}\mp@subsup{R}{}{\prime\prime}\mp@subsup{W}{}{\prime\prime}#zs and a-in: a \in A"
by auto
    hence x#xs = (x#ys)@Writesb True a" sop" v" A" L" R" W"#zs
by auto
    thus ?thesis
using a-in
by blast
    next
    assume ?ghst
    then obtain A" L" R R" W}\mp@subsup{}{}{\prime\prime}\mathrm{ ys zs where
xs=ys@Ghostsb A /' L" R" W'#zs and a-in: a }\in\mp@subsup{A}{}{\prime\prime
by auto
    hence x#xs = (x#ys)@Ghost }\mp@subsup{\textrm{mb}}{\textrm{sb}}{}\mp@subsup{\textrm{A}}{}{\prime\prime}\mp@subsup{L}{}{\prime\prime}\mp@subsup{\textrm{R}}{}{\prime\prime}\mp@subsup{\textrm{W}}{}{\prime\prime}#\textrm{zs
by auto
    thus ?thesis
```

using a-in
by blast
qed
next
case Prog $_{\text {sb }}$
from $a$-in have a $\in$ all-acquired xs
by (auto simp add: Progsb $_{\text {b }}$ )
from Cons.hyps [OF this]
have ( $\exists$ sop a' v ys zs A L R W. xs = ys @ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v A L R W\# zs $\wedge \mathrm{a} \in$
A) $v$
$\left(\exists \mathrm{ALR} W\right.$ ys zs. xs $=\mathrm{ys} @$ Ghost $\left._{\text {sb }} \mathrm{ALRW} \mathrm{W} \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}\right)$ (is ?write $\vee$ ?ghst).
then
show ?thesis
proof
assume ?write
then
obtain sop" $a^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime}$ ys zs
where $\mathrm{xs}=\mathrm{ys} @ W$ rite $\mathrm{e}_{\mathrm{sb}}$ True $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ and $\mathrm{a}-\mathrm{in}: \mathrm{a} \in \mathrm{A}^{\prime \prime}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in
by blast
next
assume ?ghst
then obtain $A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime}$ ys zs where
xs=ys@Ghost ${ }_{\text {sb }} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime} \# z s$ and $a-i n: a \in A^{\prime \prime}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Ghost $_{\mathrm{sb}} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in
by blast
qed
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
show ?thesis
proof (cases a $\in A$ )
case False
with a-in have a $\in$ all-acquired xs
by (auto simp add: Ghost ${ }_{\text {sb }}$ )
from Cons.hyps [OF this]
have ( $\exists$ sop $\mathrm{a}^{\prime}$ v ys zs A L R W. xs $=$ ys @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v A LR W \# zs $\wedge$ a $\in$ A) $v$
$\left(\exists \mathrm{A} L \mathrm{R} W\right.$ ys zs. xs $=\mathrm{ys} @$ Ghost $\left._{\text {sb }} \mathrm{A} L \mathrm{R} \mathrm{W} \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}\right)$ (is ?write $\vee$ ?ghst).
then
show ?thesis
proof
assume ?write
then
obtain sop" $a^{\prime \prime} v^{\prime \prime} A^{\prime \prime} L^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs
where xs=ys@Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ and $\mathrm{a}-\mathrm{in}: \mathrm{a} \in \mathrm{A}^{\prime \prime}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in
by blast
next
assume ?ghst
then obtain $A^{\prime \prime} L^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where
xs=ys@Ghost ${ }_{\text {sb }} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime} \# z s$ and $a-i n: a \in A^{\prime \prime}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Ghost $_{\mathrm{sb}} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ?thesis
using a-in
by blast
qed
next
case True
then have $\mathrm{x} \# \mathrm{xs}=[] @\left(\right.$ Ghost $_{\text {sb }}$ A L R W\#xs)
by (simp add: Ghost ${ }_{\text {sb }}$ True)
thus ? thesis
using True
by blast
qed
qed
qed
lemma split-Write ${ }_{\text {sb }}$-in-outstanding-refs:
$\mathrm{a} \in$ outstanding-refs is-Write $\mathrm{s}_{\mathrm{sb}} \mathrm{xs} \Longrightarrow\left(\exists\right.$ sop volatile v ys zs A L R W. xs $=$ ys $@\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W\#zs))
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
have a-in: a $\in$ outstanding-refs is-Write ${ }_{s b}(x \neq x s)$ by fact
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases a'=a)
case False
with a-in have $a \in$ outstanding-refs is-Write $_{\text {sb }}$ xs
by (auto simp add: Writesb)
from Cons.hyps [OF this] obtain sop" volatile" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs
where $\mathrm{xs}=\mathrm{ys} @ W$ Write $\mathrm{sb}_{\mathrm{s}}$ volatile" $\mathrm{a} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \mathrm{Write}_{\text {sb }}$ volatile ${ }^{\prime \prime} \mathrm{a} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
thus ? thesis
by blast
next
case True
then have $\mathrm{x} \# \mathrm{xs}=[] @\left(\right.$ Write $_{\text {sb }}$ volatile a sop v A L R W\#xs)
by (simp add: Write sb True)
thus ?thesis
by blast
qed
next
case Read ${ }_{\text {sb }}$
from a-in have $a \in$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by (auto simp add: Read ${ }_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop" volatile ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where xs=ys@Write ${ }_{\text {sb }}$ volatile" a sop" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ W$ rite $_{\text {sb }}$ volatile ${ }^{\prime \prime} \mathrm{a} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
thus ?thesis
by blast
next
case Prog $_{\text {sb }}$
from a-in have $a \in$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs by (auto simp add: Prog $_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop" volatile" $\mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where $\mathrm{xs}=\mathrm{ys} @ W r i t e_{s b}$ volatile" $a s o p^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$
by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \mathrm{Write}_{\text {sb }}$ volatile ${ }^{\prime \prime} \mathrm{a} \mathrm{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
thus ?thesis
by blast
next
case Ghost ${ }_{\text {sb }}$
from a-in have $a \in$ outstanding-refs is-Write $_{\text {sb }}$ xs by (auto simp add: Ghost ${ }_{\text {sb }}$ )
from Cons.hyps [OF this] obtain sop " volatile ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime}$ ys zs where $\mathrm{xs}=\mathrm{ys} @ W r i t e_{\text {sb }}$ volatile" $\mathrm{a}^{\prime} \operatorname{sop}^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
hence $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Write $_{\text {sb }}$ volatile ${ }^{\prime \prime} \mathrm{a}$ sop ${ }^{\prime \prime} \mathrm{v}^{\prime \prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{zs}$ by auto
thus ?thesis by blast
qed
qed
lemma outstanding-refs-is-Write ${ }_{\text {sb }}$-union:

```
    outstanding-refs is-Write 
    (outstanding-refs is-volatile-Write sb xs U outstanding-refs is-non-volatile-Write }\mp@subsup{e}{sb}{}\mathrm{ xs)
apply (induct xs)
apply simp
subgoal for a
apply (case-tac a)
apply auto
done
done
lemma rtranclp-r-rtranclp: \llbracketr** x y; r y z\rrbracket\Longrightarrow ( r** x z
    by auto
lemma r-rtranclp-rtranclp: \llbracketr x y; r** y z\rrbracket\Longrightarrow ( r** x z
    by auto
lemma unshared-is-non-volatile-Write }\mp@subsup{\textrm{sb}}{\textrm{s}}{}:\bigwedge\mathcal{S}\mathrm{ .
    \llbracketnon-volatile-writes-unshared \mathcal{S sb; a }\in\mathrm{ dom S S a }\not\in\mathrm{ all-unshared sb】 #}
    a & outstanding-refs is-non-volatile-Write }\mp@subsup{\mathrm{ sb }}{\mathrm{ sb}}{\mathrm{ sb}
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case False
            with Cons Write sb show ?thesis by auto
        next
            case True
            from Cons.hyps [where S = (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})]}\mathrm{ ] Cons.prems
            show ?thesis
    by (auto simp add: Writesb True)
        qed
    next
        case Readsb with Cons show ?thesis by auto
    next
        case Prog
    next
        case (Ghost sb A L R W)
        with Cons.hyps [where S}=(\mathcal{S}\oplus\textrm{w}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})]\mathrm{ Cons.prems show ?thesis by auto
    qed
qed
lemma outstanding-non-volatile-Read \({ }_{s b}\)-acquired-or-read-only-reads:
\(\wedge \mathcal{O} \mathcal{S}\) pending-write.
【non-volatile-owned-or-read-only pending-write \(\mathcal{S} \mathcal{O}\) sb;
```

```
a \in outstanding-refs is-non-volatile-Read sb sb\rrbracket
\Longrightarrow a \in ~ a c q u i r e d - r e a d s ~ T r u e ~ s b \mathcal { O } \vee ~ a ~ \in ~ r e a d - o n l y - r e a d s ~ \mathcal { O ~ s b }
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a'sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            with Write sb Cons.hyps [of True (\mathcal{S}\oplusw R }\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ ] Cons.prems
            show ?thesis by auto
    next
            case False
            with Cons show ?thesis
    by (auto simp add: Writesb)
        qed
    next
        case (Read sb volatile a't v)
        show ?thesis
        proof (cases volatile)
            case False with Read sb Cons show ?thesis by auto
        next
            case True
            with Read
        qed
    next
        case Prog
    next
        case (Ghost sb A L R W) with Cons.hyps [of pending-write (\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{*}R
R] Cons.prems
        show ?thesis
            by auto
    qed
qed
lemma acquired-reads-union: \bigwedgepending-writes A B.
    \llbracket a \in ~ a c q u i r e d - r e a d s ~ p e n d i n g - w r i t e s ~ s b ~ ( A \cup B ) ; a \notin A \rrbracket \Longrightarrow a \in ~ a c q u i r e d - r e a d s ~ p e n d -
ing-writes sb B
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a'sop v A' L' R' W')
    show ?thesis
    proof (cases volatile)
```

```
    case True
    note volatile=this
    show ?thesis
    proof (cases pending-writes)
case True
from Cons.prems obtain
    a-in: a }\in\mathrm{ acquired-reads True sb (A }\cup\textrm{B}\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})\mathrm{ and
    a-notin: a }\not\in\textrm{A
    by (simp add: Write sb volatile True)
have }(A\cupB\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})\subseteq(A\cup(B\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})
    by auto
from acquired-reads-mono [OF this ] a-in
have a }\in\mathrm{ acquired-reads True sb (A }\cup(B\cup\mp@subsup{A}{}{\prime}-R')
    by auto
from Cons.hyps [OF this a-notin]
have a }\in\mathrm{ acquired-reads True sb (B U A' - R').
then show ?thesis
    by (simp add: Write sb volatile True)
        next
case False
with Cons show ?thesis
    by (auto simp add: Write sb volatile False)
        qed
    next
        case False
        with Cons show ?thesis
by (auto simp add: Writesb False)
        qed
    next
        case Read
        by (auto split: if-split-asm)
    next
        case Prog
            by (auto)
    next
        case (Ghost sb A' L' R' W')
        show ?thesis
        proof -
            from Cons.prems obtain
a-in:a}\in\mathrm{ acquired-reads pending-writes sb (A B B A A'}-\mp@subsup{R}{}{\prime})\mathrm{ and
a-notin: a }\not\in\textrm{A
        by (simp add: Ghostsb )
        have }(A\cupB\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})\subseteq(A\cup(B\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})
        by auto
        from acquired-reads-mono [OF this ] a-in
        have a }\in\mathrm{ acquired-reads pending-writes sb (A }\cup(B\cup\mp@subsup{A}{}{\prime}-\mp@subsup{R}{}{\prime})
        by auto
    from Cons.hyps [OF this a-notin]
```

```
        have a }\in\mathrm{ acquired-reads pending-writes sb (B U A' - R').
        then show ?thesis
        by (simp add: Ghostsb
    qed
    qed
qed
```

lemma non-volatile-writes-unshared-no-outstanding-non-volatile-Write ${ }_{\text {sb }}: \bigwedge \mathcal{S} \mathcal{S}^{\prime}$.
【non-volatile-writes-unshared $\mathcal{S}$ sb;
$\forall \mathrm{a} \in \operatorname{dom} \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} . \mathrm{a} \notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb $\rrbracket$
$\Longrightarrow$ non-volatile-writes-unshared $\mathcal{S}^{\prime}$ sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
unshared-sb: non-volatile-writes-unshared $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ sb and
no-refs-sb: $\forall \mathrm{a} \in \operatorname{dom} \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}$. a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb
by ( $\operatorname{simp}$ add: Write ${ }_{\text {sb }}$ True)
from no-refs-sb have $\forall \mathrm{a} \in \operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb
by auto
from Cons.hyps [OF unshared-sb this]
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ True)
next
case False
with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case Read ${ }_{\text {sb }}$ with Cons show ?thesis
by (auto)
next
case $\operatorname{Prog}_{\text {sb }}$ with Cons show ?thesis by (auto)
next
case (Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain
unshared-sb: non-volatile-writes-unshared $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ sb and
no-refs-sb: $\forall \mathrm{a} \in \operatorname{dom} \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}$. a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb by ( $\operatorname{simp}$ add: Ghost ${ }_{\text {sb }}$ )
from no-refs-sb have $\forall \mathrm{a} \in \operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.

```
        a & outstanding-refs is-non-volatile-Write sb sb
    by auto
    from Cons.hyps [OF unshared-sb this]
    show ?thesis
        by (simp add: Ghostsb
    qed
qed
```

theorem sharing-consis-share-all-until-volatile-write:
$\bigwedge \mathcal{S}$ ts'. ${ }^{\prime}$ ownership-distinct ts; sharing-consis $\mathcal{S}$ ts; length ts ${ }^{\prime}=$ length ts;
$\forall \mathrm{i}<$ length ts.

$$
\begin{aligned}
& \text { (let }\left(-,-,-, \mathrm{sb},-, \mathcal{O}^{2},-\right)=\mathrm{ts}!\mathrm{i} ; \\
& \quad\left(-,-,-, \mathrm{sb}^{\prime},-, \mathcal{O}^{\prime},-\right)=\mathrm{ts}!\mathrm{i}
\end{aligned}
$$

in $\mathcal{O}^{\prime}=$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O} \wedge$

$$
\left.\mathrm{sb}^{\prime}=\operatorname{drop}^{2} \text { hile }\left(\text { Not } \circ \text { is-volatile-Write }_{\text {sb }}\right) \mathrm{sb}\right) \rrbracket \Longrightarrow
$$

sharing-consis (share-all-until-volatile-write ts $\mathcal{S}$ ) ts ${ }^{\prime} \wedge$
dom (share-all-until-volatile-write ts $\mathcal{S}$ ) - $\operatorname{dom} \mathcal{S} \subseteq$

$$
\bigcup((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot \text { set ts }) \wedge
$$

dom $\mathcal{S}$ - dom (share-all-until-volatile-write ts $\mathcal{S}) \subseteq$
$\bigcup((\lambda(-,-,-, s b,-, \mathcal{O},-)$. all-acquired $\mathrm{sb} \cup \mathcal{O})$ ' set ts)
proof (induct ts)
case Nil thus ?case by auto
next
case (Cons t ts)
have leq: length $\mathrm{ts}^{\prime}=$ length ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
have $\operatorname{sim}$ : $\forall \mathrm{i}<$ length ( $\mathrm{t} \# \mathrm{ts}$ ).
(let $(-,-,-, \mathrm{sb},-, \mathcal{O},-)=(\mathrm{t} \# \mathrm{ts})!$ i;
$\left(-,-,-, \mathrm{sb}^{\prime},-, \mathcal{O}^{\prime},-\right)=\mathrm{ts}!$ i
in $\mathcal{O}^{\prime}=$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O} \wedge$
$\mathrm{sb}^{\prime}=$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
by fact
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$
where $\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases t )
from leq obtain $\mathrm{t}^{\prime} \mathrm{ts}^{\prime \prime}$ where $\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{t}^{\prime} \# \mathrm{ts}{ }^{\prime \prime}$ and leq': length $\mathrm{ts}^{\prime \prime}=$ length ts
by (cases $\mathrm{ts}^{\prime}$ ) force+
obtain $\mathrm{p}^{\prime}$ is $\mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{D}^{\prime} \vartheta^{\prime} \mathrm{sb}^{\prime}$
where $\mathrm{t}^{\prime}: \mathrm{t}^{\prime}=\left(\mathrm{p}^{\prime}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)$
by ( cases t')
from $\operatorname{sim}$ [rule-format, of 0$] \mathrm{t} \mathrm{t}^{\prime} \mathrm{ts}^{\prime}$
obtain $\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O}$ and $\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb
by auto
from sim ts ${ }^{\prime}$
have $\operatorname{sim}^{\prime}: \forall \mathrm{i}<$ length ts.

$$
\begin{aligned}
& \text { (let }(-,-,-, s b,-, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i} ; \\
& \quad\left(-,-,-, \mathrm{sb}^{\prime},-, \mathcal{O}^{\prime}, \mathcal{R}\right)=\mathrm{ts}^{\prime \prime}!\mathrm{i} \\
& \text { in } \left.\left.\mathcal{O}^{\prime}=\text { acquired True (takeWhile (Not } \circ \text { is-volatile-Write }{ }_{\mathrm{sb}}\right) \mathrm{sb}\right) \mathcal{O} \wedge \\
& \mathrm{sb}^{\prime}=\text { dropWhile (Not } \circ \text { is-volatile-Write } \\
& \text { sb }) \mathrm{sb})
\end{aligned}
$$

by auto
have consis: sharing-consis $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$ by fact
then interpret sharing-consis $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$.
from sharing-consis [of 0] t
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}$ sb
by fastforce
from sharing-consistent-takeWhile [OF this]
have consis': sharing-consistent $\mathcal{S} \mathcal{O}$ (takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb)
by simp
let $? \mathcal{S}^{\prime}=\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right) \mathcal{S}\right)$
from freshly-shared-owned [OF consis']
have fresh-owned: dom $\mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}$.
from unshared-all-unshared [OF consis'] unshared-acquired-or-owned [OF consis']
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}$
$\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right) \cup \mathcal{O}$
by $\operatorname{simp}$
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
from ownership-distinct-tl [OF this]
have dist $^{\prime}$ : ownership-distinct ts .
from sharing-consis-tl [OF consis]
interpret consis': sharing-consis $\mathcal{S}$ ts.
from dist interpret ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ).

```
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=\) ts!i in
        all-acquired \(\mathrm{sb}^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
        all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(\mathrm{p}_{\mathrm{i}} \mathrm{is}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i-bound: \(\mathrm{i}<\) length ts
    assume ts-i: ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
    have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
        proof -
```

from ownership-distinct [of 0 Suc i] ts-i t i-bound
have dist: $(\mathcal{O} \cup$ all-acquired sb$) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}$
by force
from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb] have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$ by blast

## moreover

from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}}$ sb $_{\mathrm{i}}$ by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$ by blast
ultimately show?thesis by simp qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF sep]
have consis-ts: sharing-consis ? $\mathcal{S}^{\prime}$ ts.
from Cons.hyps [OF dist' this leq' sim]
obtain consis-ts":
sharing-consis (share-all-until-volatile-write ts ? $\mathcal{S}^{\prime}$ ) ts ${ }^{\prime \prime}$ and
fresh: dom (share-all-until-volatile-write ts $? \mathcal{S}^{\prime}$ ) - dom $? \mathcal{S}^{\prime} \subseteq$ $\cup((\lambda(-,-,-,-,-\mathcal{O}, \mathcal{R}) . \mathcal{O})$ ' set ts) and
unshared: dom ? $\mathcal{S}^{\prime}-\operatorname{dom}$ (share-all-until-volatile-write ts $\left.? \mathcal{S}^{\prime}\right) \subseteq$ $\cup\left((\lambda(-,-,-, s b,-, \mathcal{O}, \mathcal{R}) \text {. all-acquired } \mathrm{sb} \cup \mathcal{O})^{‘}\right.$ set ts $)$
by auto
from sharing-consistent-append [of - (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb)] consis-sb
have consis-t': sharing-consistent ? $\mathcal{S}^{\prime} \mathcal{O}^{\prime}$ sb $^{\prime}$
by ( simp add: $\mathcal{O}^{\prime}$ sb ${ }^{\prime}$ )
have fresh-dist: all-acquired $\mathrm{sb}^{\prime} \cap$ dom $? \mathcal{S}^{\prime}-$ dom (share-all-until-volatile-write ts $? \mathcal{S}^{\prime}$ ) $=\{ \}$
proof -
have all-acquired $\mathrm{sb}^{\prime} \cap \bigcup\left((\lambda(-,-,-, \mathrm{sb},-, \mathcal{O},-) \text {. all-acquired } \mathrm{sb} \cup \mathcal{O})^{‘}\right.$ set ts $)=\{ \}$ proof \{
fix $x$
assume x -sb': $\mathrm{x} \in$ all-acquired $\mathrm{sb}^{\prime}$
assume x -ts: $\mathrm{x} \in \bigcup\left((\lambda(-,-,-, \mathrm{sb},-, \mathcal{O},-)\right.$. all-acquired $\mathrm{sb} \cup \mathcal{O}){ }^{‘}$ set ts $)$
have False
proof -
from $x$-ts
obtain i $p_{i}$ is $\mathcal{O}_{i} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$ where
i-bound: i < length ts and
ts-i: ts! $1=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$ and
x-in: $\mathrm{x} \in$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{i}$
by (force simp add: in-set-conv-nth)
from ownership-distinct [of 0 Suc i] ts-i t i-bound
have dist: $(\mathcal{O} \cup$ all-acquired sb$) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}$
by force
with x -sb' x -in all-acquired-dropWhile [of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb] show False by (auto simp add: sb')
qed \} thus ?thesis by blast
qed
with unshared show? thesis
by blast
qed
have unshared-dist: all-unshared $\mathrm{sb}^{\prime} \cap \operatorname{dom}$ (share-all-until-volatile-write ts $? \mathcal{S}^{\prime}$ ) - dom $? \mathcal{S}^{\prime}=\{ \}$
proof -
from unshared-acquired-or-owned [OF consis-t']
have all-unshared $\mathrm{sb}^{\prime} \subseteq$ all-acquired $\mathrm{sb}^{\prime} \cup \mathcal{O}^{\prime}$.
also
from all-acquired-dropWhile [of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb]
acquired-all-acquired [of True takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $\mathcal{O}$ ]
all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb]
have all-acquired $\mathrm{sb}^{\prime} \cup \mathcal{O}^{\prime} \subseteq$ all-acquired $\mathrm{sb} \cup \mathcal{O}$
by (auto simp add: $\mathrm{sb}^{\prime} \mathcal{O}^{\prime}$ )
finally
have all-unshared $\mathrm{sb}^{\prime} \subseteq$ (all-acquired $\left.\mathrm{sb} \cup \mathcal{O}\right)$.

## moreover

have (all-acquired $\mathrm{sb} \cup \mathcal{O}) \cap \bigcup((\lambda(-,-,-,-,, \mathcal{O},-) . \mathcal{O})$ 'set ts $)=\{ \}$
proof -
\{
fix $x$

```
assume x -sb': \(\mathrm{x} \in\) all-acquired \(\mathrm{sb} \cup \mathcal{O}\)
assume \(\mathrm{x}-\mathrm{ts}: \mathrm{x} \in \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})^{〔}\right.\) set ts)
have False
proof -
    from x -ts
    obtain i \(\mathrm{p}_{\mathrm{i}}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\) where
        i-bound: \(\mathrm{i}<\) length ts and
                    ts- \(\mathrm{i}: \mathrm{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\) and
        x-in: \(x \in \mathcal{O}_{i}\)
        by (force simp add: in-set-conv-nth)
    from ownership-distinct [of 0 Suc i] ts-i t i-bound
    have dist: \((\mathcal{O} \cup\) all-acquired sb\() \cap\left(\mathcal{O}_{\mathbf{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
        by force
    with x -sb' x -in show False
        by (auto simp add: sb')
qed
        \}
        thus ?thesis by blast
    qed
    ultimately show ?thesis
        using fresh by fastforce
qed
from sharing-consistent-preservation [OF consis-t' fresh-dist unshared-dist]
have consis-ts: sharing-consistent (share-all-until-volatile-write ts ? \(\mathcal{S}^{\prime}\) ) \(\mathcal{O}^{\prime} \mathrm{sb}^{\prime}\).
note sharing-consis-Cons [OF consis-ts \({ }^{\prime \prime}\) consis-ts, of \(\mathrm{p}^{\prime}\) is \({ }^{\prime} \vartheta^{\prime} \mathcal{D}^{\prime}\) ]
moreover
from fresh fresh-owned
have dom (share-all-until-volatile-write ts \({ }^{\prime} \mathcal{S}^{\prime}\) ) \(-\operatorname{dom} \mathcal{S} \subseteq\)
                                    \(\mathcal{O} \cup \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}){ }^{\prime}\right.\) set ts \()\)
    by auto
moreover
from unshared unshared-acq-owned all-acquired-takeWhile [of (Not o is-volatile-Write \({ }_{\text {sb }}\) )
sb]
    have \(\operatorname{dom} \mathcal{S}\) - dom (share-all-until-volatile-write ts \(? \mathcal{S}^{\prime}\) ) \(\subseteq\)
                all-acquired \(\mathrm{sb} \cup \mathcal{O} \cup \bigcup((\lambda(-,-,-, \mathrm{sb},-, \mathcal{O},-)\). all-acquired \(\mathrm{sb} \cup \mathcal{O})\) ' set ts \()\)
    by auto
    ultimately
    show ?case
    by (auto simp add: \(\mathrm{t} \mathrm{ts}^{\prime} \mathrm{t}^{\prime}\) )
qed
corollary sharing-consistent-share-all-until-volatile-write:
assumes dist: ownership-distinct ts
assumes consis: sharing-consis \(\mathcal{S}\) ts
assumes i-bound: i < length ts
assumes ts- i : \(\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
shows sharing-consistent (share-all-until-volatile-write ts \(\mathcal{S}\) )
```

```
(acquired True (takeWhile (Not ○ is-volatile-Write sb
(dropWhile (Not \circ is-volatile-Write sb})\textrm{sb}
```

```
proof -
    define ts' where ts' == map ( }\lambda(\textrm{p},\textrm{is},\vartheta,\textrm{vb},\mathcal{D},\mathcal{O},\mathcal{R})
                (p,is,\vartheta,
                            dropWhile (Not o is-volatile-Write sb
(Not ○ is-volatile-Write sb
    have leq: length ts' = length ts
        by (simp add: ts'-def)
    have flush: }\forall\textrm{i}< length ts
                (let (-,-,-,sb,-,\mathcal{O,-) = ts!i;}
                        (-,-,-,sb
                in \mathcal{O}
                    sb}\mp@subsup{}{}{\prime}=\mathrm{ dropWhile (Not ○ is-volatile-Write sb}) sb
    by (auto simp add: ts'-def Let-def)
    from sharing-consis-share-all-until-volatile-write [OF dist consis leq flush]
    interpret sharing-consis (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ) ts' by simp
    from i-bound leq ts-i sharing-consis [of i]
    show ?thesis
    by (force simp add: ts'-def)
qed
lemma restrict-map-UNIV [simp]: S |}\mathrm{ | UNIV = S
    by (auto simp add: restrict-map-def)
lemma share-all-until-volatile-write-Read-commute:
    shows \bigwedgeS i. \llbracketi < length ls; ls!i=(p,Read volatile a t#is,\vartheta,sb,\mathcal{D},\mathcal{O})
    】
    share-all-until-volatile-write
        (ls[i := (p,is,\vartheta(t\mapstov), sb @ [Readsb volatile a t v ],\mathcal{D}
    share-all-until-volatile-write ls S
proof (induct ls)
    case Nil thus ?case
        by simp
next
    case (Cons l ls)
    note i-bound = <i < length (l#ls)\rangle
    note ith = <(l#ls)!i = (p,Read volatile a t#is,\vartheta,sb,\mathcal{D},\mathcal{O})\rangle
    show ?case
    proof (cases i)
    case 0
```

```
    from ith 0 have l: l = (p,Read volatile a t#is,\vartheta,sb,\mathcal{D},\mathcal{O})
        by simp
    thus ?thesis
        by (simp add: 0 share-append-Readsb del: fun-upd-apply )
    next
    case (Suc n)
    obtain plis is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{D}}{l}{}\mp@subsup{\vartheta}{l}{}\mathrm{ sb m
        by (cases l)
    from i-bound ith
    have share-all-until-volatile-write
        (1s[n := (p,is,\vartheta(t\mapstov), sb @ [Read sb volatile a t v ],\mathcal{D}
        (share (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\textrm{sb}}{}\mathrm{ ) sb l})\textrm{S})
        share-all-until-volatile-write ls (share (takeWhile (Not o is-volatile-Write sb) sbl
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc l)
        done
    then
    show ?thesis
        by (simp add: Suc l del: fun-upd-apply)
    qed
qed
lemma share-all-until-volatile-write-Write-commute:
    shows \S i. \llbracketi < length ls; ls!i=(p,Write volatile a (D,f) A L R W#is,\vartheta,sb,\mathcal{D},\mathcal{O})
    】
    \Longrightarrow
    share-all-until-volatile-write
        (ls[i := (p,is,\vartheta, sb @ [Write sb volatile a t (f \vartheta) A L R W], D
    share-all-until-volatile-write ls S
proof (induct ls)
    case Nil thus ?case
    by simp
next
    case (Cons l ls)
    note i-bound = {i < length (l#ls)
    note ith = <(l#ls)!i = (p,Write volatile a (D,f) A L R W#is,\vartheta,sb,\mathcal{D},\mathcal{O})
    show ?case
    proof (cases i)
    case 0
    from ith 0 have l: l = (p,Write volatile a (D,f) A L R W#is,\vartheta,sb,\mathcal{D},\mathcal{O})
        by simp
    thus ?thesis
        by (simp add: 0 share-append-Write}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ del: fun-upd-apply )
    next
    case (Suc n)
```



```
        by (cases l)
    from i-bound ith
```

have share-all-until-volatile-write

```
(ls[n := (p,is, \vartheta, sb @ [Write esb volatile a t (f \vartheta) A L R W],\mathcal{D}
    (share (takeWhile (Not o is-volatile-Write esb) sbl})\textrm{S})
    share-all-until-volatile-write ls (share (takeWhile (Not o is-volatile-Write sb) sbl})\textrm{S}
    apply -
    apply (rule Cons.hyps)
    apply (auto simp add: Suc l)
    done
```

    then
    show ?thesis
        by (simp add: Suc l del: fun-upd-apply)
    qed
    qed
lemma share-all-until-volatile-write-RMW-commute:
shows $\wedge$ S i. $\llbracket i<\operatorname{length} 1 \mathrm{l} ; \mathrm{ls}!\mathrm{i}=(\mathrm{p}, \mathrm{RMW}$ at (D,f) cond ret A L R W\#is, $,[], \mathcal{D}, \mathcal{O})$
』
share-all-until-volatile-write ( $1 \mathrm{ss}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.$, is, $\left.\left.\left.\vartheta^{\prime},[], \mathcal{D}^{\prime}, \mathcal{O}^{\prime}\right)\right]\right) \mathrm{S}=$
share-all-until-volatile-write is S
proof (induct ls)
case Nil thus ?case
by simp
next
case (Cons 1 ls)
note i -bound $=$ i $<$ length ( $\mathrm{l} \# \mathrm{l} \mathrm{s}$ ) $\rangle$
note ith $=\langle(\mathrm{l} \# \mathrm{l})!\mathrm{i}=(\mathrm{p}, \mathrm{RMW}$ at $(\mathrm{D}, \mathrm{f})$ cond ret A L R W\#is, $\vartheta,[], \mathcal{D}, \mathcal{O})\rangle$
show ?case
proof (cases i)
case 0
from ith 0 have $1: 1=\left(p, R M W\right.$ at $(D, f)$ cond ret A L R W\#is, $\left., \frac{1}{}[], \mathcal{D}, \mathcal{O}\right)$
by simp
thus ?thesis
by (simp add: 0 share-append-Write ${ }_{\text {sb }}$ del: fun-upd-apply )
next
case (Suc n)
obtain $\mathrm{p}_{\mathrm{l}}$ is $\mathcal{O}_{\mid} \mathcal{D}_{\mid} \vartheta_{\mid} \mathrm{sb}_{\mid}$where $\mathrm{l}: 1=\left(\mathrm{p} \mid\right.$, is $\left.\mid, \vartheta_{\mid}, \mathrm{sb}_{\mid}, \mathcal{D}_{1}, \mathcal{O}_{\mid}\right)$
by (cases l)
from i-bound ith
have share-all-until-volatile-write
$\left(\mathrm{ls}\left[\mathrm{n}:=\left(\mathrm{p}^{\prime}, \mathrm{is}, \vartheta^{\prime},[], \mathcal{D}^{\prime}, \mathcal{O}^{\prime}\right)\right]\right)$
$\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\mathbf{s b}}\right) \operatorname{sb}_{\boldsymbol{l}}\right) \mathrm{S}\right)=$
share-all-until-volatile-write ls (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sblı) S)
apply -
apply (rule Cons.hyps)
apply (auto simp add: Suc l)
done
then

```
        show ?thesis
            by (simp add: Suc l del: fun-upd-apply)
    qed
qed
lemma share-all-until-volatile-write-Fence-commute:
    shows }\bigwedge\textrm{S}\mathrm{ i. }\llbracket\textrm{i}<\mathrm{ length ls; ls!i=(p,Fence#is,},\vartheta,[],\mathcal{D},\mathcal{O},\mathcal{R}
        】
    \Longrightarrow
        share-all-until-volatile-write (ls[i := (p,is,\vartheta,[], \mathcal{D}
        share-all-until-volatile-write ls S
proof (induct ls)
    case Nil thus ?case
        by simp
next
    case (Cons l ls)
    note i-bound = <i < length (l#ls)〉
    note ith = <(l#ls)!i = (p,Fence#is,\vartheta,[],\mathcal{D},\mathcal{O},\mathcal{R})\rangle
    show ?case
    proof (cases i)
        case 0
        from ith 0 have l: l = (p,Fence#is,\vartheta,[],\mathcal{D},\mathcal{O},\mathcal{R})
            by simp
        thus ?thesis
            by (simp add: 0 share-append-Write sb del: fun-upd-apply )
    next
        case (Suc n)
        obtain p pl is|}\mp@subsup{\mathcal{O}}{|}{}\mp@subsup{\mathcal{R}}{l}{}\mp@subsup{\mathcal{D}}{l}{}\mp@subsup{\vartheta}{l}{}\mathrm{ sbl
            by (cases l)
    from i-bound ith
    have share-all-until-volatile-write
        (ls[n := (p,is,\vartheta,[],\mp@subsup{\mathcal{D}}{}{\prime},\mathcal{O},\mp@subsup{\mathcal{R}}{}{\prime})])
        (share (takeWhile (Not ○ is-volatile-Write }\mp@subsup{\textrm{sb}}{}{\prime}\mathrm{ ) sb }\mp@subsup{\textrm{b}}{\textrm{l}}{})\textrm{S})
        share-all-until-volatile-write ls (share (takeWhile (Not o is-volatile-Write sb
        apply -
        apply (rule Cons.hyps)
        apply (auto simp add: Suc 1)
        done
    then
    show ?thesis
            by (simp add: Suc l del: fun-upd-apply)
    qed
qed
```

```
lemma unshared-share-in: \(\bigwedge \mathrm{S} . \mathrm{a} \in \operatorname{dom} \mathrm{S} \Longrightarrow \mathrm{a} \notin\) all-unshared \(\mathrm{sb} \Longrightarrow \mathrm{a} \in \operatorname{dom}\) (share
sb S)
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
    case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        show ?thesis
        proof -
    from Cons.prems obtain a-S: \(a \in \operatorname{dom} S\) and \(a-L: ~ a \notin L\) and \(a-s b: ~ a \notin\) all-unshared sb
        by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
from a-S a-L have \(a \in \operatorname{dom}\left(S \oplus_{W} R \ominus_{A} L\right)\)
    by auto
from Cons.hyps [OF this a-sb]
show ?thesis
    by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
            qed
        next
            case False
            with Cons show ?thesis
by (auto simp add: Write \({ }_{\text {sb }}\) False)
        qed
    next
        case Read \(_{\text {sb }}\)
        with Cons show ?thesis
            by (auto simp add: Read sb )
    next
        case \(\operatorname{Prog}_{\text {sb }}\)
        with Cons show ? thesis
            by (auto simp add: Read sb )
    next
        case Ghost \({ }_{\text {sb }}\)
        with Cons show ?thesis
            by (auto simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
```

lemma dom-eq-dom-share-eq: $\bigwedge S^{\prime} S^{\prime} . \operatorname{dom} S=\operatorname{dom} S^{\prime} \Longrightarrow \operatorname{dom}($ share $\operatorname{sb} S)=\operatorname{dom}$ (share sb $\mathrm{S}^{\prime}$ )
proof (induct sb)

```
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A' L R W)
        show ?thesis
        proof (cases volatile)
            case True
            from Cons.prems
            have \(\operatorname{dom}\left(S \oplus_{W} R \ominus_{A^{\prime}} L\right)=\operatorname{dom}\left(S^{\prime} \oplus_{W} R \ominus_{A^{\prime}} L\right)\)
by auto
            from Cons.hyps [OF this]
            show ?thesis
by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
        next
            case False with Cons.hyps [of S S ] Cons.prems Write \({ }_{\text {sb }}\) show ?thesis by auto
        qed
    next
        case Read \(_{\text {sb }}\) with Cons.hyps [of S S \({ }^{\prime}\) ] Cons.prems show ?thesis by auto
    next
        case Prog \(_{\text {sb }}\) with Cons.hyps [of S S \(]\) Cons.prems show ?thesis by auto
    next
        case ( Ghost \(_{\text {sb }}\) A \(^{\prime}\) L R W)
        from Cons.prems
        have \(\operatorname{dom}\left(S \oplus_{W} R \quad \ominus_{A^{\prime}} L\right)=\operatorname{dom}\left(S^{\prime} \oplus_{W} R \ominus_{A^{\prime}} L\right)\)
            by auto
    from Cons.hyps [OF this]
    show ?thesis
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
lemma share-union:
    \(\bigwedge A B . \llbracket a \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(A \oplus_{z} B\right)\right) ; a \notin \operatorname{dom} A \rrbracket \Longrightarrow a \in \operatorname{dom}\) (share sb (Map.empty
\(\left.\oplus_{\mathbf{z}} \mathrm{B}\right)\) )
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A' L R W)
        show ?thesis
        proof (cases volatile)
            case True
            from Cons.prems
            obtain a-in: \(a \in \operatorname{dom}\left(\operatorname{share} s b\left(\left(A \oplus_{z} B\right) \oplus_{W} R \ominus_{A^{\prime}} L\right)\right)\) and a-A: a \(\notin \operatorname{dom} A\)
    by (clarsimp simp add: Write sb \(_{\text {b }}\) True)
            have \(\operatorname{dom}\left(\left(A \oplus_{z} B\right) \oplus_{W} R \ominus_{A^{\prime}} L\right) \subseteq \operatorname{dom}\left(A \oplus_{z}(B \cup R-L)\right)\)
```

```
by auto
    from share-mono [OF this] a-in
    have \(\mathrm{a} \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathrm{A} \oplus_{\mathrm{z}}(\mathrm{B} \cup R-L)\right)\right)\)
by blast
    from Cons.hyps [OF this] a-A
    have \(\mathrm{a} \in \operatorname{dom}\) (share sb (Map.empty \(\oplus_{\mathrm{z}}(\mathrm{B} \cup \mathrm{R}-\mathrm{L})\) ))
by blast
    moreover
    have dom (Map.empty \(\left.\oplus_{\mathbf{z}} \mathrm{B} \cup \mathrm{R}-\mathrm{L}\right)=\operatorname{dom}\left(\left(\right.\right.\) Map.empty \(\left.\left.\oplus_{\mathbf{z}} \mathrm{B}\right) \oplus_{\mathrm{W}} \mathrm{R} \oplus_{A^{\prime}} \mathrm{L}\right)\)
by auto
    note dom-eq-dom-share-eq [OF this, of sb]
    ultimately
    show ?thesis
by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
    next
        case False
        with Cons show ?thesis
by (auto simp add: Write \({ }_{\text {sb }}\) False)
        qed
    next
        case \(\operatorname{Read}_{\text {sb }}\)
        with Cons show?thesis
        by (auto simp add: \(\operatorname{Read}_{\text {sb }}\) )
    next
        case Prog \(_{\text {sb }}\)
        with Cons show ?thesis
        by (auto simp add: \(\operatorname{Read}_{\text {sb }}\) )
    next
        case (Ghost \({ }_{\text {sb }}\) A \(^{\prime}\) L R W)
        from Cons.prems
        obtain a-in: \(\mathrm{a} \in \operatorname{dom}\left(\right.\) share sb \(\left.\left(\left(\mathrm{A} \oplus_{\mathrm{z}} \mathrm{B}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}\right)\right)\) and a-A: a \(\notin \operatorname{dom} \mathrm{A}\)
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
    have \(\operatorname{dom}\left(\left(A \oplus_{\mathbf{z}} B\right) \oplus_{W} R \ominus_{A^{\prime}} L\right) \subseteq \operatorname{dom}\left(A \oplus_{\mathbf{z}}(B \cup R-L)\right)\)
        by auto
    from share-mono [OF this] a-in
    have \(\mathrm{a} \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathrm{A} \oplus_{\mathrm{z}}(\mathrm{B} \cup \mathrm{R}-\mathrm{L})\right)\right)\)
        by blast
    from Cons.hyps [OF this] a-A
    have \(\mathrm{a} \in \operatorname{dom}\) (share sb (Map.empty \(\oplus_{\mathrm{z}}(\mathrm{B} \cup \mathrm{R}-\mathrm{L})\) ))
        by blast
    moreover
    have dom (Map.empty \(\left.\oplus_{\mathbf{z}} \mathrm{B} \cup \mathrm{R}-\mathrm{L}\right)=\operatorname{dom}\left(\left(\right.\right.\) Map.empty \(\left.\left.\oplus_{\mathbf{z}} \mathrm{B}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{A^{\prime}} \mathrm{L}\right)\)
        by auto
    note dom-eq-dom-share-eq [OF this, of sb]
    ultimately
    show ?thesis
        by (clarsimp simp add: Ghostsb)
    qed
qed
```

lemma share-unshared-in:

```
    \(\wedge\) S. a \(\in \operatorname{dom}(\) share \(s b S) \Longrightarrow a \in \operatorname{dom}\) (share sb Map.empty) \(\vee(a \in \operatorname{dom} S \wedge a \notin\)
```

all-unshared sb)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems
have a-in: $\mathrm{a} \in \operatorname{dom}\left(\right.$ share sb $\left(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ )
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in$ dom S)
case True
from Cons.hyps [OF a-in]
have $\mathrm{a} \in \operatorname{dom}$ (share sb Map.empty) $\vee \mathrm{a} \in \operatorname{dom}\left(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \wedge \mathrm{a} \notin$ all-unshared sb.
then show ?thesis
proof
assume a $\in$ dom (share sb Map.empty)
from share-mono-in [OF this]
have $\mathrm{a} \in \operatorname{dom}$ (share sb (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )) by auto
then show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile True)
next
assume a $\in \operatorname{dom}\left(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \wedge \mathrm{a} \notin$ all-unshared sb
then obtain a $\notin \mathrm{L}$ a $\notin$ all-unshared sb
by auto
then show ?thesis by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile True)
qed
next
case False
have dom $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \subseteq \operatorname{dom}\left(\mathrm{S} \oplus_{\mathrm{W}}(\mathrm{R}-\mathrm{L})\right)$
by auto
from share-mono [OF this] a-in
have $\mathrm{a} \in \operatorname{dom}($ share $\mathrm{sb}(\mathrm{S} \oplus \mathrm{w}(\mathrm{R}-\mathrm{L})))$ by blast
from share-union [OF this False]
have a $\in \operatorname{dom}$ (share sb (Map.empty $\oplus \mathrm{w}(\mathrm{R}-\mathrm{L})$ )).
moreover
have dom (Map.empty $\left.\oplus_{\mathrm{W}}(\mathrm{R}-\mathrm{L})\right)=$ dom (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
by auto
note dom-eq-dom-share-eq [OF this, of sb]
ultimately
show ?thesis

```
    by (clarsimp simp add: Writesb True)
        qed
    next
        case False
        with Cons show ?thesis
by (auto simp add: Write sb False)
    qed
next
    case Read sb
    with Cons show ?thesis
        by (auto simp add: Read sb
next
    case Progsb
    with Cons show ?thesis
        by (auto simp add: Readsb
next
    case (Ghostsb A L R W)
    from Cons.prems
    have a-in: a }\in\operatorname{dom (share sb (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
        by (clarsimp simp add: Ghostsb)
    show ?thesis
    proof (cases a }\in\operatorname{dom}S\mathrm{ )
        case True
        from Cons.hyps [OF a-in]
    have a }\in\operatorname{dom (share sb Map.empty) \vee a \in dom (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{A}{}\textrm{L})\wedge\textrm{a}\not\in\mathrm{ all-unshared
sb.
        then show ?thesis
        proof
assume a }\in\mathrm{ dom (share sb Map.empty)
from share-mono-in [OF this]
have a }\in\operatorname{dom (share sb (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mathrm{ ) by auto
            then show ?thesis
    by (clarsimp simp add: Ghost sb True)
        next
assume a }\in\operatorname{dom}(\textrm{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\wedge\textrm{a}\not\in\mathrm{ all-unshared sb
then obtain a }\not\in\textrm{L}\mathrm{ a }\not\in\mathrm{ all-unshared sb
    by auto
then show ?thesis by (clarsimp simp add: Ghost sb True)
        qed
    next
        case False
        have dom (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\subseteq\operatorname{dom}(\textrm{S}\oplus\textrm{W}(\textrm{R}-\textrm{L})
            by auto
        from share-mono [OF this] a-in
        have a }\in\operatorname{dom}(\mathrm{ share sb (S }\oplus\textrm{w}(\textrm{R}-\textrm{L})))\mathrm{ by blast
        from share-union [OF this False]
        have a }\in\mathrm{ dom (share sb (Map.empty }\oplus\textrm{w}(\textrm{R}-\textrm{L}))\mathrm{ ).
        moreover
        have dom (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{(}(\textrm{R}-\textrm{L}))=\operatorname{dom (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
            by auto
```

```
        note dom-eq-dom-share-eq [OF this, of sb]
        ultimately
        show ?thesis
        by (clarsimp simp add: Ghostsb False)
    qed
    qed
qed
```

lemma dom-augment-rels-shared-eq: dom (augment-rels $\mathrm{S} R \mathcal{R}$ ) $=\operatorname{dom}$ (augment-rels $\mathrm{S}^{\prime}$ $\mathrm{R} \mathcal{R}$ ) by (auto simp add: augment-rels-def domIff split: option.splits if-split-asm)
lemma dom-eq-SomeD1: dom $\mathrm{m}=\operatorname{dom} \mathrm{n} \Longrightarrow \mathrm{mx}=$ Some $\mathrm{y} \Longrightarrow \mathrm{n} \mathrm{x} \neq$ None by (auto simp add: dom-def)
lemma dom-eq-SomeD2: dom $m=\operatorname{dom} \mathrm{n} \Longrightarrow \mathrm{n} x=$ Some $\mathrm{y} \Longrightarrow \mathrm{mx} \neq$ None by (auto simp add: dom-def)
lemma dom-augment-rels-rels-eq: dom $\mathcal{R}^{\prime}=\operatorname{dom} \mathcal{R} \Longrightarrow \operatorname{dom}$ (augment-rels $\mathrm{S} R \mathcal{R}^{\prime}$ ) $=$ dom (augment-rels S R $\mathcal{R}$ )
by (auto simp add: augment-rels-def domIff split: option.splits if-split-asm dest: dom-eq-SomeD1 dom-eq-SomeD2)

```
lemma dom-release-rels-eq: \(\wedge \mathcal{S} \mathcal{R} \mathcal{R}^{\prime}\). \(\operatorname{dom} \mathcal{R}^{\prime}=\operatorname{dom} \mathcal{R} \Longrightarrow\)
    \(\operatorname{dom}\left(\right.\) release \(\left.\operatorname{sb} \mathcal{S} \mathcal{R}^{\prime}\right)=\operatorname{dom}(\) release \(\operatorname{sb} \mathcal{S} \mathcal{R})\)
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    hence dr: \(\operatorname{dom} \mathcal{R}^{\prime}=\operatorname{dom} \mathcal{R}\)
        by simp
    show ?case
    proof (cases x)
        case Write \({ }_{\text {sb }}\) with Cons.hyps [OF dr] show ?thesis by (clarsimp)
    next
        case \(\operatorname{Read}_{\text {sb }}\) with Cons.hyps [OF dr] show ?thesis by (clarsimp)
    next
        case Prog \(_{\text {sb }}\) with Cons.hyps [OF dr] show ?thesis by (clarsimp)
    next
        case (Ghost sb A L R W)
        from Cons.hyps [OF dom-augment-rels-rels-eq [OF dr]]
        show ?thesis
        by ( \(\operatorname{simp}\) add: Ghost \(\mathrm{sb}_{\mathrm{sb}}\) )
    qed
qed
```

```
lemma dom-release-shared-eq: \(\bigwedge \mathcal{S} \mathcal{S}^{\prime} \mathcal{R}\). dom (release sb \(\left.\mathcal{S}^{\prime} \mathcal{R}\right)=\operatorname{dom}(\) release sb \(\mathcal{S} \mathcal{R})\)
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case Write \({ }_{\text {sb }}\) with Cons.hyps show ?thesis by (clarsimp)
    next
        case Read \(_{\text {sb }}\) with Cons.hyps show ?thesis by (clarsimp)
    next
        case \(\operatorname{Prog}_{\text {sb }}\) with Cons.hyps show ?thesis by (clarsimp)
    next
        case (Ghost \({ }_{\text {sb }}\) A L R W)
        have dr: dom (augment-rels \(\mathcal{S}^{\prime} \mathrm{R} \mathcal{R}\) ) \(=\operatorname{dom}\) (augment-rels \(\mathcal{S} \mathrm{R} \mathcal{R}\) )
                by(rule dom-augment-rels-shared-eq)
    have dom (release sb \(\left(\mathcal{S}^{\prime} \cup \mathrm{R}-\mathrm{L}\right)\left(\right.\) augment-rels \(\left.\left.\mathcal{S}^{\prime} \mathrm{R} \mathcal{R}\right)\right)=\)
                dom (release sb \((\mathcal{S} \cup \mathrm{R}-\mathrm{L})\) (augment-rels \(\left.\mathcal{S}^{\prime} \mathrm{R} \mathcal{R}\right)\) )
        by (rule Cons.hyps)
    also have \(\ldots=\operatorname{dom}(\) release \(\operatorname{sb}(\mathcal{S} \cup \mathrm{R}-\mathrm{L})\) (augment-rels \(\mathcal{S} \mathrm{R} \mathcal{R})\) )
        by (rule dom-release-rels-eq [OF dr])
    finally show ?thesis
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
lemma share-other-untouched:
\(\bigwedge \mathcal{O} \mathcal{S}\). sharing-consistent \(\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow \mathrm{a} \notin \mathcal{O} \cup\) all-acquired \(\mathrm{sb} \Longrightarrow\) share \(\mathrm{sb} \mathcal{S} \mathrm{a}=\mathcal{S} \mathrm{a}\) proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
A-shared-owns: \(\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}\) and \(\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}\) and \(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\) and R-owns: \(\mathrm{R} \subseteq \mathcal{O}\) and
consis': sharing-consistent \(\left(\mathcal{S} \oplus \mathrm{W} R \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) sb and a-owns: \(\mathrm{a} \notin \mathcal{O}\) and \(\mathrm{a}-\mathrm{A}\) : \(\mathrm{a} \notin \mathrm{A}\) and a-sb: a \(\notin\) all-acquired sb by ( simp add: Write \({ }_{\text {sb }}\) True )
from a-owns a-A a-sb
have \(\mathrm{a} \notin \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup\) all-acquired sb by auto
```

```
    from Cons.hyps [OF consis' this]
    have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}.
    moreover have (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{a}
    using L-A A-R R-owns a-owns a-A
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    ultimately show ?thesis
        by (simp add: Write sb True)
    next
    case False with Cons show ?thesis
        by (auto simp add: Write sb False)
    qed
next
    case Read
    show ?thesis
        by (auto)
next
    case Progsb with Cons
    show ?thesis
        by (auto)
next
    case (Ghostsb A L R W)
    from Cons.prems obtain
    A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns:
R}\subseteq\mathcal{O}\mathrm{ and
    consis': sharing-consistent (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and
        a-owns: a }\not\in\mathcal{O}\mathrm{ and a-A: a }\not\in\textrm{A}\mathrm{ and a-sb: a }\not\in\mathrm{ all-acquired sb
        by ( simp add: Ghostsb )
    from a-owns a-A a-sb
    have a }\not\in\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb
        by auto
    from Cons.hyps [OF consis' this]
    have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}.
    moreover have (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{a}
    using L-A A-R R-owns a-owns a-A
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    ultimately show ?thesis
        by (simp add: Ghostsb)
    qed
qed
```

lemma shared-owned: $\bigwedge \mathcal{O} \mathcal{S}$. sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow \mathrm{a} \notin \operatorname{dom} \mathcal{S} \Longrightarrow \mathrm{a} \in \operatorname{dom}$ (share sb $\mathcal{S}$ ) $\Longrightarrow$
$\mathrm{a} \in \mathcal{O} \cup$ all-acquired sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)

```
case (Write sb volatile a'sop v A L R W)
```

show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-notin: a $\notin \operatorname{dom} \mathcal{S}$ and a-in: a $\in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$
by ( simp add: Write sb True )
show ?thesis
proof (cases a $\in \mathcal{O}$ )
case True thus ?thesis by auto
next
case False
with a-notin R-owns A-shared-owns L-A A-R have a $\notin \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by (auto)
from Cons.hyps [OF consis' this a-in]
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case False with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case Read ${ }_{\text {sb }}$ with Cons
show ?thesis
by (auto)
next
case $\operatorname{Prog}_{s b}$ with Cons
show ?thesis
by (auto)
next
case (Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns:
$\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-notin: $\mathrm{a} \notin \operatorname{dom} \mathcal{S}$ and a-in: $\mathrm{a} \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$
by (simp add: Ghost ${ }_{\text {sb }}$ )
show ?thesis
proof (cases a $\in \mathcal{O}$ )
case True thus ?thesis by auto
next
case False
with a-notin R-owns A-shared-owns L-A A-R have a $\notin \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$

```
            by (auto)
            from Cons.hyps [OF consis' this a-in]
            show ?thesis
            by (auto simp add: Ghostsb)
    qed
qed
qed
```

lemma share-all-shared-in: $\mathrm{a} \in \operatorname{dom}$ (share sb $\mathcal{S}) \Longrightarrow \mathrm{a} \in \operatorname{dom} \mathcal{S} \vee \mathrm{a} \in$ all-shared sb using sharing-consistent-share-all-shared [of sb $\mathcal{S}$ ]
by auto
lemma share-all-until-volatile-write-unowned:
assumes dist: ownership-distinct ts
assumes consis: sharing-consis $\mathcal{S}$ ts
assumes other: $\forall \mathrm{i}$ p is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R}$. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ a $\notin \mathcal{O} \cup$ all-acquired sb
shows share-all-until-volatile-write ts $\mathcal{S}$ a $=\mathcal{S}$ a
using dist consis other
proof (induct ts arbitrary: $\mathcal{S}$ )
case Nil thus ?case by simp
next
case (Cons t ts)
obtain $\mathrm{p}_{\mathrm{t}}$ is $\mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where
$\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by ( cases t )
from Cons.prems t obtain
other': $\forall \mathrm{i}$ p is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R}$. $\mathrm{i}<$ length ts $\longrightarrow$ ts! $!=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ a $\notin \mathcal{O} \cup$ all-acquired sb and
a-notin: $\mathrm{a} \notin \mathcal{O}_{\mathrm{t}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{t}}$
apply -
apply (rule that)
apply clarsimp
subgoal for ip is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O}$
apply (drule-tac $x=S u c i$ in spec)
apply clarsimp
done
apply (drule-tac $x=0$ in spec)
apply clarsimp
done
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact then interpret ownership-distinct t \#ts.
have consis: sharing-consis $\mathcal{S}$ ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret sharing-consis $\mathcal{S}$ t\#ts.
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then
interpret consis': sharing-consis $\mathcal{S}$ ts.
let $\boldsymbol{S}^{\prime}=\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{s b}\right) \mathrm{sb}_{\mathrm{t}}\right) \mathcal{S}\right)$
from sharing-consis [of 0 , simplified, OF t]
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{t}}$ ).
from freshly-shared-owned [OF consis-sb]
have fresh-owned: dom ? $\mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}$.
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}$ - dom ? $\mathcal{S}^{\prime}$ $\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ) $\cup \mathcal{O}_{\mathrm{t}}$
by simp
have sep:
$\forall \mathrm{i}<$ length ts. let $\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=$ ts!i in
all-acquired sb' $\cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge$
all-unshared $\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
proof -
\{
fix i $p_{i}$ is $\mathcal{O}_{i} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$
assume i-bound: $\mathrm{i}<$ length ts
assume ts- i : ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge$
all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
proof -
from ownership-distinct [of 0 Suc i] ts-i t i-bound
have dist: $\left(\mathcal{O}_{\mathrm{t}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}$
by force
from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ] have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast

## moreover

from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$ by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned

```
have all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
    by blast
ultimately show ?thesis by simp
        qed
    \}
    thus ?thesis
        by (fastforce simp add: Let-def)
    qed
    from consis \({ }^{\prime}\).sharing-consis-preservation [OF this]
    have sharing-consis ? \(\mathcal{S}^{\prime}\) ts.
    from Cons.hyps [OF dist' this other']
    have share-all-until-volatile-write ts ? \(\mathcal{S}^{\prime} \mathrm{a}=\)
    share (takeWhile (Not o is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{t}}\) ) \(\mathcal{S} \mathrm{a}\).
moreover
from share-other-untouched [OF consis-sb] a-notin
    all-acquired-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \({ }_{t}\) ) (dropWhile (Not \(\circ\)
is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{t}}\) )]
    have share (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{t}}\) ) \(\mathcal{S} \mathrm{a}=\mathcal{S} \mathrm{a}\)
    by auto
    ultimately
    show ?case
    by (simp add: t)
qed
lemma share-shared-eq: \(\bigwedge \mathcal{S}^{\prime} \mathcal{S} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a} \Longrightarrow\) share \(\mathrm{sb} \mathcal{S}^{\prime} \mathrm{a}=\) share \(\mathrm{sb} \mathcal{S} \mathrm{a}\)
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    have eq: \(\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S}\) a by fact
    show ?case
    proof (cases x)
    case (Write \({ }_{\text {sb }}\) volatile a'sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        have \(\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}\)
        using eq by (auto simp add: augment-shared-def restrict-shared-def)
        from Cons.hyps [of \(\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\), OF this]
        show ?thesis
            by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
    next
        case False
        with Cons.hyps [of \(\mathcal{S}^{\prime} \mathcal{S}\) ] Cons.prems show ?thesis
by (auto simp add: Write \({ }_{\text {sb }}\) False)
    qed
```

```
next
    case Readsb
    with Cons.hyps [of S'S}\mathcal{S}]\mathrm{ Cons.prems show ?thesis
        by (auto simp add: Readsb)
    next
    case Progsb
    with Cons.hyps [of S'S
        by (auto simp add: Readsb
next
    case (Ghost sb A L R W)
    have (S'S
    using eq by (auto simp add: augment-shared-def restrict-shared-def)
    from Cons.hyps [of (\mathcal{S}
    show ?thesis
        by (clarsimp simp add: Ghost }\mp@subsup{\mathrm{ sb }}{\mathrm{ )}}{
    qed
qed
lemma share-all-until-volatile-write-thread-local:
    assumes dist: ownership-distinct ts
    assumes consis: sharing-consis }\mathcal{S}\mathrm{ ts
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-owned: a }\in\mathcal{O}\cup\mathrm{ all-acquired sb
    shows share-all-until-volatile-write ts \mathcal{S a}=\mathrm{ share (takeWhile (Not o is-volatile-Write sb})
sb)\mathcal{S a}
using dist consis i-bound ts-i
proof (induct ts arbitrary: }\mathcal{S}\mathrm{ i)
    case Nil thus ?case by simp
next
    case (Cons t ts)
obtain p pis is }\mp@subsup{\mathcal{O}}{\textrm{t}}{}\mp@subsup{\mathcal{R}}{\textrm{t}}{}\mp@subsup{\mathcal{D}}{\textrm{t}}{}\mp@subsup{\vartheta}{\textrm{t}}{}\mp@subsup{\textrm{sb}}{\textrm{t}}{}\mathrm{ where
    t: t=( p
    by (cases t)
have dist: ownership-distinct (t#ts) by fact
then interpret ownership-distinct t#ts.
have consis: sharing-consis }\mathcal{S}(\textrm{t}#\textrm{ts})\mathrm{ by fact
then interpret sharing-consis }\mathcal{S}\textrm{t}#\textrm{ts}
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis }\mathcal{S}\mathrm{ ts.
then
interpret consis': sharing-consis }\mathcal{S}\mathrm{ ts.
let ?S'S
```

```
from sharing-consis [of 0 , simplified, OF t ]
have sharing-consistent \(\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}\).
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent \(\mathcal{S} \mathcal{O}_{\mathrm{t}}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{t}}\) ).
from freshly-shared-owned [OF consis-sb]
have fresh-owned: \(\operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}\).
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: \(\operatorname{dom} \mathcal{S}\) - \(\operatorname{dom}\) ? \(\mathcal{S}^{\prime}\)
    \(\subseteq\) all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\left.\mathrm{sb}_{\mathrm{t}}\right) \cup \mathcal{O}_{\mathrm{t}}\)
    by simp
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=\) ts!i in
        all-acquired sb' \(\cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
        all-unshared sb' \(\cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(p_{i}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i-bound: \(\mathrm{i}<\) length ts
        assume ts-i: ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \mathcal{Y}_{\mathrm{i}}, \mathrm{Sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
        have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
    proof -
from ownership-distinct [of 0 Suc i] ts-i ti-bound
have dist: \(\left(\mathcal{O}_{\mathrm{t}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathbf{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
    by force
from dist unshared-acq-owned all-acquired-takeWhile [of (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{t}}\) ]
have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}\)
    by blast
moreover
from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent \(\mathcal{S} \mathcal{O}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
    by force
from unshared-acquired-or-owned [OF this]
have all-unshared \(\mathrm{sb}_{\mathrm{i}} \subseteq\) all-acquired \(\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}\).
with dist fresh-owned
have all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
    by blast
ultimately show ?thesis by simp
        qed
    \}
    thus ?thesis
        by (fastforce simp add: Let-def)
qed
```

from consis'.sharing-consis-preservation [OF this]
have consis-shared': sharing-consis ? $\mathcal{S}^{\prime}$ ts.

```
have aargh: \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.{ }_{s b}\right)=\left(\lambda a . \neg\right.\) is-volatile-Write \(_{\text {sb }}\) a \()\)
    by (rule ext) auto
show ?case
proof (cases i)
    case 0
    with Cons.prems
    have \(\mathrm{t}^{\prime}: \mathrm{t}=(\mathrm{p}\), is, \(\mathcal{\vartheta}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
        by simp
    \{
        fix j \(p_{j}\) is \(\vartheta_{j} \vartheta_{j} \operatorname{sb}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}}\)
        assume j -bound: \(\mathrm{j}<\) length ts
        assume ts-j: ts ! \(\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}\right.\), is \(\left.\mathrm{s}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
        have a \(\notin \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
        proof -
            from ownership-distinct [of 0 Suc j , simplified, OF j -bound t ts-j] t a-owned \(\mathrm{t}^{\prime} 0\)
            show ?thesis
                by auto
    qed
    \}
```

    with share-all-until-volatile-write-unowned [OF dist' consis-shared', of a]
    have share-all-until-volatile-write ts ? \(\mathcal{S}^{\prime} \mathrm{a}=? \mathcal{S}^{\prime} \mathrm{a}\)
        by fastforce
    then show ?thesis
    using \(\mathrm{t} \mathrm{t}^{\prime} 0\)
        by (auto simp add: Cons t aargh)
    next
case (Suc n)
with Cons.prems obtain $n$-bound: $\mathrm{n}<$ length ts and ts-n: $\operatorname{ts!n}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by auto
from Cons.hyps [OF dist' consis-shared' n-bound ts-n]
have share-all-until-volatile-write ts $?^{\prime} \mathcal{S}^{\prime} \mathrm{a}=$
share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) ? $\mathcal{S}^{\prime}$ a.
moreover
from ownership-distinct [of 0 Suc n] t a-owned ts-n n-bound
have a $\notin \mathcal{O}_{\mathrm{t}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{t}}$
by fastforce
with share-other-untouched [OF consis-sb, of a]
all-acquired-append [of (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) sb $_{\mathrm{t}}$ ) (dropWhile (Not o
is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ )]
have $\mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
by auto
from share-shared-eq [of ? $\mathcal{S}^{\prime}$ a $\mathcal{S}, \mathrm{OF}$ this ]

```
    have share (takeWhile (Not o is-volatile-Write sb) sb) ? '\mathcal{S}
        share (takeWhile (Not o is-volatile-Write sb) sb) \mathcal{S a .}
    ultimately show ?thesis
    using t Suc
        by (auto simp add: aargh)
    qed
qed
lemma share-all-until-volatile-write-thread-local':
    assumes dist: ownership-distinct ts
    assumes consis: sharing-consis }\mathcal{S}\mathrm{ ts
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-owned: a }\in\mathcal{O}\cup\mathrm{ all-acquired sb
    shows share (dropWhile (Not o is-volatile-Write sb) sb) (share-all-until-volatile-write ts
S) a =
    share sb S a
proof -
    let ?take = takeWhile (Not % is-volatile-Write sb})\textrm{sb
    let ?drop = dropWhile (Not o is-volatile-Writesb) sb
    from share-all-until-volatile-write-thread-local [OF dist consis i-bound ts-i a-owned]
    have share-all-until-volatile-write ts }\mathcal{S}\textrm{a}=\mathrm{ share ?take }\mathcal{S}\textrm{a}
    moreover
    from share-shared-eq [of share-all-until-volatile-write ts }\mathcal{S}\mathrm{ a share ?take }\mathcal{S}\mathrm{ , OF this]
    have share ?drop (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ) a = share ?drop (share ?take S a a.
    thus ?thesis
    using share-append [of ?take ?drop S
        by simp
qed
lemma (in ownership-distinct) in-shared-sb-share-all-until-volatile-write:
    assumes consis: sharing-consis }\mathcal{S}\mathrm{ ts
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-owned: a }\in\mathcal{O}\cup\mathrm{ all-acquired sb
    assumes a-share: a }\in\operatorname{dom (share sb }\mathcal{S}\mathrm{ )
    shows a }\in\mathrm{ dom (share (dropWhile (Not 0 is-volatile-Write sb) sb)
                            (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ))
proof -
    have dist: ownership-distinct ts
    using assms ownership-distinct
        apply -
        apply (rule ownership-distinct.intro)
        apply auto
        done
    from share-all-until-volatile-write-thread-local' [OF dist consis i-bound ts-i a-owned]
        a-share
    show ?thesis
        by (auto simp add: domIff)
qed
```

lemma owns-unshared-share-acquired:
$\wedge \mathcal{S} \mathcal{O}$. [sharing-consistent $\mathcal{S} \mathcal{O}$ sb; $\mathrm{a} \in \mathcal{O} ; \mathrm{a} \notin$ all-unshared $\mathrm{sb} \rrbracket$
$\Longrightarrow \mathrm{a} \in \operatorname{dom}$ (share sb $\mathcal{S}$ ) $\cup$ acquired True sb $\mathcal{O}$
proof (induct sb)
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
a-owns: $\mathrm{a} \in \mathcal{O}$ and A -shared-onws: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and
a-L: a $\notin \mathrm{L}$ and a-unsh: a $\notin$ all-unshared sb and L-A: $\mathrm{L} \subseteq \mathrm{A}$ and
A -R: $\mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
have $\mathrm{a} \in \operatorname{dom}\left(\right.$ share $\left.\mathrm{sb}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \cup$ acquired True sb $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$
proof (cases a $\in R$ )
case True
with a-L have a $\in \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ by auto
from unshared-share-in [OF this a-unsh]
show ?thesis by blast next
case False
hence $\mathrm{a} \in \mathcal{O} \cup \mathrm{A}-\mathrm{R}$
using a-owns
by auto
from Cons.hyps [OF consis' this a-unsh]
show ?thesis .
qed
then
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
next
case False
with Cons
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case $\operatorname{Read}_{\text {sb }}$
with Cons show ?thesis
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
next

```
case Progsb
with Cons show ?thesis
    by (auto simp add: Readsb
next
case (Ghostsb A L R W)
from Cons.prems obtain
a-owns: a }\in\mathcal{O}\mathrm{ and A-shared-onws: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and
a-L: a & L and a-unsh: a & all-unshared sb and L-A: L }\subseteqA\mathrm{ and
A-R: A \cap R = {} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
consis': sharing-consistent (\mathcal{S}\oplus\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb}
by (clarsimp simp add: Ghostsb)
have a }\in\operatorname{dom}(\mathrm{ share sb (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}))\cup\mathrm{ acquired True sb (O}\cup\textrm{O}\cup\textrm{A}-\textrm{R}
proof (cases a }\inR\mathrm{ )
case True
with a-L have a }\in\operatorname{dom}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
by auto
from unshared-share-in [OF this a-unsh]
show ?thesis by blast
next
    case False
    hence a }\in\mathcal{O}\cup\textrm{A}-\textrm{R
        using a-owns
by auto
        from Cons.hyps [OF consis' this a-unsh]
        show ?thesis .
    qed
    then show ?thesis
        by (auto simp add: Ghostsb)
    qed
qed
lemma shared-share-acquired: }\\mathcal{S}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb }
    a}\in\operatorname{dom}\mathcal{S}\Longrightarrow\textrm{a}\in\operatorname{dom}(\mathrm{ share sb S})\cup\mathrm{ acquired True sb }\mathcal{O
proof (induct sb)
    case Nil thus ?case by auto
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a' sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
        from Cons.prems obtain
    a-shared: a }\in\operatorname{dom}\mathcal{S}\mathrm{ and A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and
L-A: L \subseteq A and A-R: A \cap R ={} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
            consis': sharing-consistent (\mathcal{S}\oplus\textrm{w}}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\textrm{sb
by (clarsimp simp add: Write sb True)
        show ?thesis
```

proof (cases a $\in L$ )
case False with a-shared
have $\mathrm{a} \in \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by auto
from Cons.hyps [OF consis' this]
show ?thesis
by (clarsimp simp add: Write sb $^{\text {s }}$ volatile)
next
case True
with L-A have $a-A: a \in A$
by blast
from sharing-consistent-mono-shared [OF - consis', where $\mathcal{S}^{\prime}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R}\right)$ ]
have sharing-consistent $(\mathcal{S} \oplus \mathrm{w})(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by auto
from Cons.hyps [OF this] a-shared
have hyp: $\mathrm{a} \in \operatorname{dom}(\operatorname{share} \operatorname{sb}(\mathcal{S} \oplus \mathrm{w})) \cup \operatorname{acquired} \operatorname{True} \mathrm{sb}(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ by auto
\{
assume $\mathrm{a} \in \operatorname{dom}(\operatorname{share} \operatorname{sb}(\mathcal{S} \oplus \mathrm{w} \mathrm{R}))$
from share-unshared-in [OF this]
have $\mathrm{a} \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \cup \operatorname{acquired} \operatorname{True} \operatorname{sb}(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$
proof
assume a $\in$ dom (share sb Map.empty)
from share-mono-in [OF this]
have $\mathrm{a} \in \operatorname{dom}\left(\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$
by auto
thus ?thesis by blast
next
assume $\mathrm{a} \in \operatorname{dom}(\mathcal{S} \oplus \mathrm{w} \mathrm{R}) \wedge \mathrm{a} \notin$ all-unshared sb
hence a-unsh: a $\notin$ all-unshared sb by blast
from $a-A A-R$ have $a \in \mathcal{O} \cup A-R$
by auto
from owns-unshared-share-acquired [OF consis' this a-unsh]
show ?thesis .
qed
\}
with hyp show ?thesis
by (auto simp add: Write sb volatile)
qed
next
case False
with Cons
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case Read ${ }_{\text {sb }}$
with Cons show ?thesis
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
next

```
    case Progsb
    with Cons show ?thesis
    by (auto simp add: Readsb
next
    case (Ghostsb A L R W)
    from Cons.prems obtain
        a-shared: a }\in\operatorname{dom}\mathcal{S}\mathrm{ and A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and
        L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns: R}\subseteq\mathcal{O}\mathrm{ and
        consis': sharing-consistent (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb
        by (clarsimp simp add: Ghostsb)
    show ?thesis
    proof (cases a }\inL\mathrm{ L)
    case False with a-shared
    have a }\in\operatorname{dom}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
        by auto
    from Cons.hyps [OF consis' this]
    show ?thesis
        by (clarsimp simp add: Ghostsb
    next
    case True
    with L-A have a-A: a }\in\textrm{A
        by blast
    from sharing-consistent-mono-shared [OF - consis', where }\mp@subsup{\mathcal{S}}{}{\prime}=(\mathcal{S}\oplus\textrm{w}R)
    have sharing-consistent (\mathcal{S}\oplus\textrm{w}}\textrm{R})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb
        by auto
    from Cons.hyps [OF this] a-shared
    have hyp: a }\in\operatorname{dom}(\mathrm{ share sb (S }\oplus\textrm{w}R))\cup\mathrm{ acquired True sb (O }\cup\textrm{O
        by auto
    {
assume a }\in\operatorname{dom}(\mathrm{ share sb (S }\mathcal{S}\textrm{w}R)
from share-unshared-in [OF this]
have a }\in\operatorname{dom}(\mathrm{ share sb (S ( }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}))\cup\mathrm{ acquired True sb (O }\cup\textrm{O}\cup\textrm{A}-\textrm{R}
proof
assume a }\in\mathrm{ dom (share sb Map.empty)
from share-mono-in [OF this]
have a }\in\operatorname{dom}(\operatorname{share sb (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
    by auto
thus ?thesis by blast
        next
assume a }\in\operatorname{dom}(\mathcal{S}\oplus\textrm{w}R)\wedge \textrm{a}\not\in\mathrm{ all-unshared sb
hence a-unsh: a }\not\in\mathrm{ all-unshared sb by blast
from a-A A-R have a }\in\mathcal{O}\cup\textrm{A}-\textrm{R
        by auto
from owns-unshared-share-acquired [OF consis' this a-unsh]
show ?thesis.
        qed
    }
        with hyp show ?thesis
        by (auto simp add: Ghostsb)
qed
```


## qed

qed
lemma dom-release-takeWhile:
$\wedge \mathrm{S} \mathcal{R}$.
dom (release (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathrm{S} \mathcal{R}$ ) $=$
dom $\mathcal{R} \cup$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (induct sb)
apply (clarsimp)
subgoal for $\operatorname{asb} \mathrm{S} \mathcal{R}$
apply (case-tac a)
apply (auto simp add: augment-rels-def domIff split: if-split-asm option.splits)
done
done
lemma share-all-until-volatile-write-share-acquired:
assumes dist: ownership-distinct ts
assumes consis: sharing-consis $\mathcal{S}$ ts
assumes a-notin: a $\notin \operatorname{dom} \mathcal{S}$
assumes a-in: a $\in$ dom (share-all-until-volatile-write ts $\mathcal{S}$ )
shows $\exists \mathrm{i}<$ length ts.
let $(-,-,-, \mathrm{sb},-,-,-)=\mathrm{ts}!\mathrm{i}$
in $\mathrm{a} \in$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)
using dist consis a-notin a-in
proof (induct ts arbitrary: $\mathcal{S}$ i)
case Nil thus ?case by simp
next
case (Cons t ts)
have a-notin: a $\notin \operatorname{dom} \mathcal{S}$ by fact
obtain $p_{t}$ is $_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where
$\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t)
let ? take $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}_{\mathrm{t}}\right)$
from $t$ Cons.prems
have a-in: a $\in \operatorname{dom}$ (share-all-until-volatile-write ts (share ?take $\mathcal{S}$ )) by auto
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
have consis: sharing-consis $\mathcal{S}$ ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret sharing-consis $\mathcal{S} \mathrm{t} \# \mathrm{ts}$.
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then

```
interpret consis': sharing-consis \(\mathcal{S}\) ts.
let \(? \mathcal{S}^{\prime}=(\) share ?take \(\mathcal{S})\)
from sharing-consis [of 0 , simplified, OF t ]
have sharing-consistent \(\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}\).
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent \(\mathcal{S} \mathcal{O}_{\mathrm{t}}\) ?take.
from freshly-shared-owned [OF consis-sb]
have fresh-owned: dom ? \(\mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}\).
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: \(\operatorname{dom} \mathcal{S}\) - \(\operatorname{dom}\) ? \(\mathcal{S}^{\prime}\)
    \(\subseteq\) all-acquired ?take \(\cup \mathcal{O}_{\mathrm{t}}\)
    by simp
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=\) ts! \(!\) in
        all-acquired \(\mathrm{sb}^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
        all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(p_{i}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i-bound: \(\mathrm{i}<\) length ts
        assume ts- i : ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
        have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
        proof -
from ownership-distinct [of 0 Suc i] ts-i ti-bound
have dist: \(\left(\mathcal{O}_{\mathrm{t}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
    by force
```

from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ]
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast

## moreover

from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}}$ sb $_{\mathrm{i}}$ by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$ by blast
ultimately show ?thesis by simp qed
\}
thus ?thesis

```
    by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF this]
have consis-shared': sharing-consis ?S' ts.
```



```
    by (rule ext) auto
show ?case
proof (cases a }\in\mathrm{ all-shared ?take)
    case True
    thus ?thesis
    apply -
    apply (rule-tac x=0 in exI)
    apply (auto simp add: t aargh)
    done
next
    case False
    have a-notin': a }\not\in\mathrm{ dom ?S'
    proof
        assume a }\in\operatorname{dom}?\mp@subsup{\mathcal{S}}{}{\prime
        from share-all-shared-in [OF this] False a-notin
        show False
            by auto
    qed
    from Cons.hyps [OF dist' consis-shared' a-notin' a-in]
    obtain i where i < length ts and
        rel: let (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})= ts!i
            in a }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write sb
        by (auto simp add: Let-def aargh)
    then show ?thesis
        apply -
        apply (rule-tac x = Suc i in exI)
        apply (auto simp add: Let-def aargh)
        done
    qed
qed
```

lemma all-shared-share-acquired: $\wedge \mathcal{S} \mathcal{O}$. sharing-consistent $\mathcal{S} \mathcal{O}$ sb $\Longrightarrow$ $\mathrm{a} \in$ all-shared $\mathrm{sb} \Longrightarrow \mathrm{a} \in \operatorname{dom}$ (share sb $\mathcal{S}$ ) $\cup$ acquired True sb $\mathcal{O}$ proof (induct sb)
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)

## show ?thesis

proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
a-shared: $\mathrm{a} \in \mathrm{R} \cup$ all-shared sb and A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in$ all-shared sb)
case True
from Cons.hyps [OF consis' True]
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
next
case False
with a-shared have $a \in R$
by auto
with L-A A-R R-owns have a $\in \operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by auto
from shared-share-acquired [OF consis' this]
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
qed
next
case False
with Cons
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case $\operatorname{Read}_{\text {sb }}$
with Cons show ?thesis
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
next
case Prog $_{\text {sb }}$
with Cons show?thesis
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
next
case (Ghostsb A L R W)
from Cons.prems obtain
a-shared: $\mathrm{a} \in \mathrm{R} \cup$ all-shared sb and A -shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Ghostsb ${ }_{\text {sb }}$ )
show ?thesis
proof (cases a $\in$ all-shared sb)
case True
from Cons.hyps [OF consis' True]

```
        show ?thesis
            by (clarsimp simp add: Ghostsb
    next
        case False
        with a-shared have a }\in
            by auto
        with L-A A-R R-owns have a }\in\operatorname{dom}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
            by auto
        from shared-share-acquired [OF consis' this]
        show ?thesis
            by (clarsimp simp add: Ghostsb
        qed
    qed
qed
lemma (in ownership-distinct) share-all-until-volatile-write-share-acquired:
    assumes consis: sharing-consis }\mathcal{S}\mathrm{ ts
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-in: a }\in\mathrm{ dom (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ )
    shows a }\in\mathrm{ dom (share sb }\mathcal{S})\vee\textrm{a}\in\mathrm{ acquired True sb }\mathcal{O}
        ( }\exists\textrm{j}<\mathrm{ length ts. }\textrm{j}\not=\textrm{i}
        (let (-,-,-,施,,-,-,) = ts!j
        in a }\in\mathrm{ all-shared (takeWhile (Not * is-volatile-Write sb
proof -
    from assms ownership-distinct have dist: ownership-distinct ts
        apply -
        apply (rule ownership-distinct.intro)
        apply simp
        done
from consis
interpret sharing-consis S ts .
from sharing-consis [OF i-bound ts-i]
have consis-sb: sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb.
let ?take-sb = takeWhile (Not o is-volatile-Write sb) sb
let ?drop-sb = dropWhile (Not ० is-volatile-Writesb) sb
show ?thesis
proof (cases a }\in\operatorname{dom}\mathcal{S}
    case True
    from shared-share-acquired [OF consis-sb True]
    have a }\in\operatorname{dom}(\mathrm{ share sb S})\cup\mathrm{ acquired True sb }\mathcal{O}\mathrm{ .
    thus ?thesis by auto
next
    case False
    from share-all-until-volatile-write-share-acquired [OF dist consis False a-in]
    obtain j where j-bound: j < length ts and
        rel: let (-,-,,,sbj,,-,,-) = ts!j
            in a }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write sb ) sb 
```

```
        by auto
    show ?thesis
    proof (cases j=i)
        case False
        with j-bound rel
        show ?thesis
        by blast
    next
        case True
        with rel ts-i have a }\in\mathrm{ all-shared ?take-sb
        by (auto simp add: Let-def)
    hence a }\in\mathrm{ all-shared sb
        using all-shared-append [of ?take-sb ?drop-sb]
            by auto
        from all-shared-share-acquired [OF consis-sb this]
        have a }\in\operatorname{dom}(\operatorname{share sb S})\cup\mathrm{ acquired True sb }\mathcal{O}\mathrm{ .
        thus ?thesis
        by auto
    qed
    qed
qed
```

lemma acquired-all-shared-in:
$\bigwedge$ A. $\mathrm{a} \in$ acquired True $\mathrm{sb} \mathrm{A} \Longrightarrow \mathrm{a} \in$ acquired True $\mathrm{sb}\} \vee(\mathrm{a} \in \mathrm{A} \wedge \mathrm{a} \notin$ all-shared sb$)$
proof (induct sb)

```
        case Nil thus ?case by simp
```

next
case (Cons x sb)
show ?case
proof (cases x)
case $\left(\right.$ Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop $\mathrm{v} \mathrm{A}^{\prime} \mathrm{L} R$ )
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems
have a-in: $a \in$ acquired True $\operatorname{sb}\left(A \cup A^{\prime}-R\right)$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in A$ )
case True
from Cons.hyps [OF a-in]
have $a \in$ acquired True $s b\left\} \vee a \in A \cup A^{\prime}-R \wedge a \notin\right.$ all-shared sb.
then show ?thesis
proof

```
    assume a }\in\mathrm{ acquired True sb {}
    from acquired-mono-in [OF this]
    have a }\in\mathrm{ acquired True sb (A' - R) by auto
    then show ?thesis
        by (clarsimp simp add: Write
next
    assume a }\inA\cup\mp@subsup{A}{}{\prime}-R\wedgea\not\in all-shared sb
    then obtain a }\not\in\textrm{R}\mathrm{ a & all-shared sb
        by blast
    then show ?thesis by (clarsimp simp add: Write sb volatile True)
qed
        next
case False
have (A\cup\mp@subsup{A}{}{\prime}-R)\subseteqA\cup(\mp@subsup{A}{}{\prime}-R)
    by blast
from acquired-mono [OF this] a-in
have a }\in\mathrm{ acquired True sb (A U (A' - R)) by blast
from acquired-union-notin-first [OF this False]
have a }\in\mathrm{ acquired True sb (A' - R).
then show ?thesis
    by (clarsimp simp add: Write sb True)
        qed
        next
            case False
            with Cons show ?thesis
by (auto simp add: Write sb False)
        qed
next
        case Read
        with Cons show ?thesis
            by (auto simp add: Readsb
next
    case Progsb
    with Cons show ?thesis
            by (auto simp add: Readsb
next
    case (Ghostsb A'L R W)
    from Cons.prems
    have a-in: a }\in\mathrm{ acquired True sb (A }\cup\mp@subsup{A}{}{\prime}-R
        by (clarsimp simp add: Ghostsb)
    show ?thesis
    proof (cases a G A)
        case True
        from Cons.hyps [OF a-in]
        have a }\in\mathrm{ acquired True sb {} }\vee\textrm{a}\in\textrm{A}\cup\mp@subsup{\textrm{A}}{}{\prime}-\textrm{R}\wedge\textrm{a}\not\in\mathrm{ all-shared sb.
        then show ?thesis
        proof
assume a \in acquired True sb {}
from acquired-mono-in [OF this]
have a }\in\mathrm{ acquired True sb (A' - R) by auto
```

then show ?thesis

```
    by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) True)
        next
assume \(a \in A \cup A^{\prime}-R \wedge a \notin\) all-shared sb
then obtain \(\mathrm{a} \notin \mathrm{R}\) a \(\notin\) all-shared sb
    by blast
then show ?thesis by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) True)
        qed
    next
        case False
        have \(\left(A \cup A^{\prime}-R\right) \subseteq A \cup\left(A^{\prime}-R\right)\)
            by blast
        from acquired-mono [OF this] a-in
        have \(a \in\) acquired True sb \(\left(\mathrm{A} \cup\left(\mathrm{A}^{\prime}-\mathrm{R}\right)\right)\) by blast
        from acquired-union-notin-first [OF this False]
        have \(a \in\) acquired True sb ( \(A^{\prime}-R\) ).
        then show ?thesis
            by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
        qed
    qed
qed
```

lemma all-shared-acquired-in: $\bigwedge \mathrm{A} . \mathrm{a} \in \mathrm{A} \Longrightarrow \mathrm{a} \notin$ all-shared $\mathrm{sb} \Longrightarrow \mathrm{a} \in$ acquired True sb A
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case $\left(\right.$ Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A' L R W)
show ?thesis
proof (cases volatile)
case True
show ?thesis
proof -
from Cons.prems obtain $a-A: a \in A$ and $a-R: a \notin R$ and $a-s b: a \notin$ all-shared sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from $a-A$ a-R have $a \in A \cup A^{\prime}-R$
by blast
from Cons.hyps [OF this a-sb]
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case False
with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ False)
qed

```
    next
        case Readsb
        with Cons show ?thesis
        by (auto simp add: Readsb
    next
        case Progsb
        with Cons show ?thesis
        by (auto simp add: Readsb
    next
        case Ghost sb
        with Cons show ?thesis
        by (auto simp add: Ghostsb)
    qed
qed
lemma owned-share-acquired: }\bigwedge\mathcal{S}\mathcal{O}\mathrm{ . sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb }
    a }\in\mathcal{O}\Longrightarrowa\in\operatorname{dom}(\mathrm{ share sb }\mathcal{S})\cup\mathrm{ acquired True sb }\mathcal{O
proof (induct sb)
    case Nil thus ?case by auto
next
    case (Cons x sb)
    show ?case
    proof (cases x)
    case (Write sb volatile a' sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        note volatile=this
        from Cons.prems obtain
    a-owned: a }\in\mathcal{O}\mathrm{ and A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and
    L-A: L\subseteq A and A-R:A }\cap\textrm{R}={}\mathrm{ and R-owns: R}\subseteq\mathcal{O}\mathrm{ and
        consis': sharing-consistent (\mathcal{S}\mp@subsup{\oplus}{W}{*}R\mp@subsup{\ominus}{\textrm{A}}{\prime}L)(\mathcal{O}\cup\textrm{A}
    by (clarsimp simp add: Write sb True)
        show ?thesis
        proof (cases a }\inR\mathrm{ )
    case False with a-owned
    have a }\in\mathcal{O}\cup\textrm{A}-\textrm{R
        by auto
    from Cons.hyps [OF consis' this]
    show ?thesis
        by (clarsimp simp add: Write sb volatile)
            next
    case True
    from True L-A A-R have a }\in\operatorname{dom}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
        by auto
    from shared-share-acquired [OF consis' this]
    show ?thesis
        by (clarsimp simp add: Write sb volatile True)
            qed
    next
```

```
    case False
    with Cons
    show ?thesis
by (auto simp add: Writesb)
    qed
    next
        case Read
        with Cons show ?thesis
        by (auto simp add: Read
    next
        case Progsb
        with Cons show ?thesis
            by (auto simp add: Readsb)
    next
        case (Ghost sb A L R W)
        from Cons.prems obtain
            a-owned: a }\in\mathcal{O}\mathrm{ and A-shared-owns: A }\subseteq\operatorname{dom}\mathcal{S}\cup\mathcal{O}\mathrm{ and
            L-A: L \subseteq A and A-R: A \cap R = {} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
            consis': sharing-consistent (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb
            by (clarsimp simp add: Ghost }\mp@subsup{\textrm{sb}}{\mathbf{b}}{}\mathrm{ )
    show ?thesis
    proof (cases a }\inR\mathrm{ )
            case False with a-owned
            have a }\in\mathcal{O}\cup\textrm{A}-\textrm{R
            by auto
        from Cons.hyps [OF consis' this]
        show ?thesis
            by (clarsimp simp add: Ghost
        next
            case True
            from True L-A A-R have a }\in\operatorname{dom}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
            by auto
        from shared-share-acquired [OF consis' this]
        show ?thesis
            by (clarsimp simp add: Ghost sb True)
        qed
    qed
qed
lemma outstanding-refs-non-volatile-Read sb -all-acquired:
\(\bigwedge \mathrm{m} \mathcal{S} \mathcal{O}\) pending-write.
【reads-consistent pending-write \(\mathcal{O} \mathrm{m} \mathrm{sb}\);non-volatile-owned-or-read-only pending-write \(\mathcal{S} \mathcal{O} \mathrm{sb} ;\)
\(\mathrm{a} \in\) outstanding-refs is-non-volatile-Read sb \(\mathrm{sb} \rrbracket\)
\(\Longrightarrow \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in\) all-acquired \(\mathrm{sb} \vee\)
a \(\in\) read-only-reads \(\mathcal{O}\) sb
proof (induct sb)
case Nil thus ?case by simp
next
```

```
case (Cons x sb)
show ?case
proof (cases x)
    case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A L R W)
    show ?thesis
    proof (cases volatile)
    case True
    note volatile=this
    from Cons.prems obtain
non-vo: non-volatile-owned-or-read-only \(\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
                \((\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}\) and
            out-vol: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}} \mathrm{sb}=\{ \}\) and
out: \(\mathrm{a} \in\) outstanding-refs is-non-volatile-Read \({ }_{\mathrm{sb}} \mathrm{sb}\)
by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
    show ?thesis
    proof (cases \(\mathrm{a} \in \mathcal{O}\) )
case True
show ?thesis
    by (clarsimp simp add: Write sb \(^{\text {b }}\) True volatile)
        next
case False
from outstanding-non-volatile-Read sb -acquired-or-read-only-reads [OF non-vo out]
have a-in: \(a \in\) acquired-reads True \(\operatorname{sb}(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \vee\)
                \(\mathrm{a} \in\) read-only-reads \((\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}\)
    by auto
with acquired-reads-all-acquired [of True sb \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) ]
show ?thesis
    by (auto simp add: Write \({ }_{\text {sb }}\) volatile)
        qed
        next
        case False
        with Cons show ?thesis
by (auto simp add: Write \({ }_{\text {sb }}\) False)
    qed
next
    case Read \({ }_{\text {sb }}\)
    with Cons show ?thesis
        apply (clarsimp simp del: o-apply simp add: Read \({ }_{\text {sb }}\)
acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) split: if-split-asm)
        apply auto
        done
next
    case Prog \(_{\text {sb }}\)
    with Cons show ?thesis
        by (auto simp add: Read \({ }_{\text {sb }}\) )
next
    case (Ghost \(_{\text {sb }}\) A L)
    with Cons show ?thesis
        by (auto simp add: Ghost \({ }_{\text {sb }}\) )
qed
```

qed
lemma outstanding-refs-non-volatile-Read ${ }_{\text {sb }}$-all-acquired-dropWhile:
assumes consis: reads-consistent pending-write $\mathcal{O} \mathrm{m}$ sb
assumes nvo: non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ sb
assumes out: a $\in$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
shows $\mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in$ all-acquired $\mathrm{sb} \vee$ a $\in$ read-only-reads $\mathcal{O}$ sb
using outstanding-refs-append [of is-non-volatile-Read ${ }_{\text {sb }}$ takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb
dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb]
outstanding-refs-non-volatile-Read sbb $^{\text {-all-acquired }}$ [OF consis nvo, of a] out
by (auto)
lemma share-commute:
$\wedge \mathrm{L} \mathrm{R} \mathcal{S} \mathcal{O}$. [sharing-consistent $\mathcal{S} \mathcal{O}$ sb;
all-shared $\mathrm{sb} \cap \mathrm{L}=\{ \}$; all-shared $\mathrm{sb} \cap \mathrm{A}=\{ \}$; all-acquired $\mathrm{sb} \cap \mathrm{R}=\{ \}$;
all-unshared sb $\cap \mathrm{R}=\{ \}$; all-shared $\mathrm{sb} \cap \mathrm{R}=\{ \} \rrbracket \Longrightarrow$
$\left(\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)=$
(share sb $\mathcal{S}) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a sop v $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ )
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
L-prop: $\left(\mathrm{R}^{\prime} \cup\right.$ all-shared sb) $\cap \mathrm{L}=\{ \}$ and
A-prop: $\left(R^{\prime} \cup\right.$ all-shared sb) $\cap A=\{ \}$ and
R-acq-prop: (A' $\cup$ all-acquired sb) $\cap \mathrm{R}=\{ \}$ and
R-prop: $\left(L^{\prime} \cup\right.$ all-unshared sb) $\cap \mathrm{R}=\{ \}$ and
R-prop-sh: $\left(R^{\prime} \cup\right.$ all-shared sb) $\cap R=\{ \}$ and
$\mathrm{A}^{\prime}$-shared-owns: $\mathrm{A}^{\prime} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}^{\prime}-\mathrm{A}^{\prime}: \mathrm{L}^{\prime} \subseteq \mathrm{A}^{\prime}$ and $\mathrm{A}^{\prime}-\mathrm{R}^{\prime}: \mathrm{A}^{\prime} \cap \mathrm{R}^{\prime}=\{ \}$ and
$\mathrm{R}^{\prime}$-owns: $\mathrm{R}^{\prime} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right) \mathrm{sb}$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
from L-prop obtain $R^{\prime}-L: R^{\prime} \cap L=\{ \}$ and acq-L: all-shared $s b \cap L=\{ \}$
by blast
from A-prop obtain $R^{\prime}-A: R^{\prime} \cap A=\{ \}$ and acq-A: all-shared $s b \cap A=\{ \}$
by blast
from $R$-acq-prop obtain $A^{\prime}-R: A^{\prime} \cap R=\{ \}$ and acq-R:all-acquired $s b \cap R=\{ \}$ by blast
from R-prop obtain $L^{\prime}-R: L^{\prime} \cap R=\{ \}$ and unsh-R: all-unshared $s b \cap R=\{ \}$ by blast
from $R$-prop-sh obtain $R^{\prime}-R: R^{\prime} \cap R=\{ \}$ and sh-R: all-shared sb $\cap R=\{ \}$
by blast
from Cons.hyps [OF consis' acq-L acq-A acq-R unsh-R sh-R ] have share sb $\left(\left(\mathcal{S} \oplus_{W^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{W}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R}$ $\theta_{\mathrm{A}} \mathrm{L}$.

## moreover

from $R^{\prime}-L L^{\prime}-R R^{\prime}-R R^{\prime}-A^{\prime}-R$
have $\left(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \oplus_{\mathrm{W}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)=\left(\left(\mathcal{S} \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
apply -
apply (rule ext)
apply (clarsimp simp add: augment-shared-def restrict-shared-def)
apply (auto split: if-split-asm option.splits)
done
ultimately
have share sb $\left(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \oplus_{\mathrm{W}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)=\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{W}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R}$ $\ominus_{\mathrm{A}} \mathrm{L}$
by simp
then
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
next
case False with Cons show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case Read $_{\text {sb }}$ with Cons show ?thesis
by (clarsimp simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
next
case Prog $_{\text {sb }}$ with Cons show ?thesis
by (clarsimp simp add: Prog $_{\mathrm{sb}}$ )
next
case ( Ghost $_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ )
from Cons.prems obtain
L-prop: $\left(\mathrm{R}^{\prime} \cup\right.$ all-shared sb$) \cap \mathrm{L}=\{ \}$ and
A-prop: $\left(\mathrm{R}^{\prime} \cup\right.$ all-shared sb$) \cap \mathrm{A}=\{ \}$ and
R-acq-prop: ( $A^{\prime} \cup$ all-acquired sb) $\cap \mathrm{R}=\{ \}$ and
R-prop:( $L^{\prime} \cup$ all-unshared sb) $\cap \mathrm{R}=\{ \}$ and
R-prop-sh: $\left(R^{\prime} \cup\right.$ all-shared sb) $\cap \mathrm{R}=\{ \}$ and
$\mathrm{A}^{\prime}$-shared-owns: $\mathrm{A}^{\prime} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}^{\prime}-\mathrm{A}^{\prime}: \mathrm{L}^{\prime} \subseteq \mathrm{A}^{\prime}$ and $\mathrm{A}^{\prime}-\mathrm{R}^{\prime}: \mathrm{A}^{\prime} \cap \mathrm{R}^{\prime}=\{ \}$ and
$\mathrm{R}^{\prime}$-owns: $\mathrm{R}^{\prime} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)$ sb
by ( clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
from $L$-prop obtain $R^{\prime}-L: R^{\prime} \cap L=\{ \}$ and acq-L: all-shared sb $\cap \mathrm{L}=\{ \}$ by blast
from $A$-prop obtain $R^{\prime}-A: R^{\prime} \cap A=\{ \}$ and acq-A: all-shared $s b \cap A=\{ \}$ by blast
from R -acq-prop obtain $\mathrm{A}^{\prime}-\mathrm{R}: \mathrm{A}^{\prime} \cap \mathrm{R}=\{ \}$ and acq-R:all-acquired $\mathrm{sb} \cap \mathrm{R}=\{ \}$ by blast
from $R$-prop obtain $L^{\prime}-R: L^{\prime} \cap R=\{ \}$ and unsh-R: all-unshared $\operatorname{sb} \cap \mathrm{R}=\{ \}$ by blast
from $R$-prop-sh obtain $R^{\prime}-R: R^{\prime} \cap \mathrm{R}=\{ \}$ and sh-R: all-shared sb $\cap \mathrm{R}=\{ \}$ by blast
from Cons.hyps [OF consis' acq-L acq-A acq-R unsh-R sh-R ]
have share sb $\left(\left(\mathcal{S} \oplus_{W^{\prime}} R^{\prime} \ominus_{A^{\prime}} L^{\prime}\right) \oplus W^{\prime} R \ominus_{A} L\right)=\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{w^{\prime}} R^{\prime} \ominus_{A^{\prime}} L^{\prime}\right) \oplus{ }_{w} R$ $\ominus_{\mathrm{A}}$ L.
moreover
from $\mathrm{R}^{\prime}-\mathrm{L} \mathrm{L}^{\prime}-\mathrm{R} \mathrm{R}^{\prime}-\mathrm{R} \mathrm{R}^{\prime}-\mathrm{A} \mathrm{A}^{\prime}-\mathrm{R}$
have $\left(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)=\left(\left(\mathcal{S} \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
apply -
apply (rule ext)
apply (clarsimp simp add: augment-shared-def restrict-shared-def)
apply (auto split: if-split-asm option.splits)
done

## ultimately

have share sb $\left(\left(\mathcal{S} \oplus_{W} R \ominus_{\mathrm{A}} \mathrm{L}\right) \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right)=\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{w}^{\prime}} \mathrm{R}^{\prime} \ominus_{\mathrm{A}^{\prime}} \mathrm{L}^{\prime}\right) \oplus_{\mathrm{W}} \mathrm{R}$ $\ominus_{\mathrm{A}} \mathrm{L}$
by simp
then
show ?thesis
by ( clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
lemma share-all-until-volatile-write-commute:
$\bigwedge \mathcal{S} \mathrm{R}$ L. [ownership-distinct ts; sharing-consis $\mathcal{S}$ ts;
$\forall \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) $\cap \mathrm{L}=\{ \}$;
$\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{A}=\{ \}$;
$\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$;
$\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb) $\cap \mathrm{R}=\{ \}$;
$\forall \mathrm{i} p$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} . \mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$

## $\Longrightarrow$

share-all-until-volatile-write ts $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}=$ share-all-until-volatile-write ts $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R}\right.$ $\ominus_{\mathrm{A}} \mathrm{L}$ )
proof (induct ts)
case Nil
thus ?case by simp

## next

case (Cons t ts)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb where
$\mathrm{t}: \mathrm{t}=(\mathrm{p}$, is $, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases t)
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
have consis: sharing-consis $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$ by fact
then interpret sharing-consis $\mathcal{S} \mathrm{t} \# \mathrm{ts}$.
have L-prop: $\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{L}=\{ \}$ by fact
hence L-prop': $\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i $<$ length $(\mathrm{ts}) \longrightarrow(\mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{L}=\{ \}$
by force
have A-prop: $\forall$ i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<\operatorname{length}(\mathrm{t} \# \mathrm{ts}) \longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{A}=\{ \}$ by fact
hence A-prop': $\forall$ i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i < length $(\mathrm{ts}) \longrightarrow(\mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb) $\cap \mathrm{A}=\{ \}$
by force
have R-prop-acq: $\forall$ i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<\operatorname{length}(\mathrm{t} \# \mathrm{ts}) \longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
$\qquad$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$ by fact
hence R-prop-acq $: ~ \forall i \operatorname{is} \mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i $<$ length $(\mathrm{ts}) \longrightarrow(\mathrm{ts})!i=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
$\qquad$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$
by force
have R-prop: $\forall \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$. $\mathrm{i}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
$\qquad$
all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ) $\cap \mathrm{R}=\{ \}$ by fact hence R-prop': $\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i < length $(\mathrm{ts}) \longrightarrow(\mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$
by force
have R-prop-sh: $\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i $<\operatorname{length}(\mathrm{t} \# \mathrm{ts}) \longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ $\longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$ by fact
hence R-prop-sh': $\forall$ i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i $<$ length $(\mathrm{ts}) \longrightarrow(\mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$
by force
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then
interpret consis': sharing-consis $\mathcal{S}$ ts.
from L-prop [rule-format, of 0 p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O}$ ] t
have sh-L: all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{L}=\{ \}$
by simp
from A-prop [rule-format, of 0 p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O}$ ] t
have sh-A: all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{A}=\{ \}$
by simp
from R-prop-acq [rule-format, of 0 p is $\vartheta$ sb $\mathcal{D} \mathcal{O}$ ] t
have acq-R: all-acquired (takeWhile (Not $\circ$ is-volatile-Write $_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$ by simp
from R-prop [rule-format, of 0 p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O}] t$
have unsh-R: all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap R=\{ \}$ by simp
from R-prop-sh [rule-format, of 0 p is $\vartheta$ sb $\mathcal{D} \mathcal{O}$ ] t
have sh-R: all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap R=\{ \}$
by simp
from sharing-consis [of 0 , simplified, OF t]
have sharing-consistent $\mathcal{S} \mathcal{O}$ sb.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb).
from share-commute [OF consis-sb sh-L sh-A acq-R unsh-R sh-R]
have share-eq:
$\left(\operatorname{share}\left(\right.\right.$ takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}\right)\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)=$
(share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) $\left.\mathcal{S}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$.
let $\boldsymbol{S}^{\prime}=\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb) $\left.\mathcal{S}\right)$
from freshly-shared-owned [OF consis-sb]
have fresh-owned: dom ? $\mathcal{S}^{\prime}$ - $\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}$.
from unshared-all-unshared [ OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}$ - $\operatorname{dom}$ ? $\mathcal{S}^{\prime}$
$\subseteq$ all-acquired (takeWhile (Not o is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right) \cup \mathcal{O}$
by simp

```
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-,, \mathrm{sb}^{\prime},-,-,-\right)=\) ts!i in
        all-acquired \(\mathrm{sb}^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
        all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
    fix i \(p_{i}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
    assume i -bound: \(\mathrm{i}<\) length ts
    assume ts- i : ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{Sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
    have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
                all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
    proof -
from ownership-distinct [of 0 Suc i] ts-i ti-bound
have dist: \((\mathcal{O} \cup\) all-acquired sb\() \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
    by force
```

from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb]
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast
moreover
from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{i}$ sb $_{i}$
by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
by blast
ultimately show ?thesis by simp
qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF sep]
have sharing-consis': sharing-consis (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) $\mathcal{S}$ )
ts.
from Cons.hyps [OF dist' sharing-consis' L-prop' A-prop' R-prop-acq' R-prop' R-prop-sh']
have share-all-until-volatile-write ts $? \mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}=$
share-all-until-volatile-write ts $\left(? \mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
then
have share-all-until-volatile-write ts

$$
? \mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}=
$$

share-all-until-volatile-write ts
(share (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$
by (simp add: share-eq)

## then

show ?case
by (simp add: t)
qed
lemma share-append-Ghost ${ }_{\text {sb }}$ :
$\wedge \mathcal{S}$. outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \} \Longrightarrow\left(\right.$ share $\left(\mathrm{sb} @\left[\right.\right.$ Ghost $_{\text {sb }}$ A L R W] $)$
$\mathcal{S})=($ share sb $\mathcal{S}) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$
apply (induct sb)
apply simp
subgoal for a sb $\mathcal{S}$
apply (case-tac a)
apply auto
done
done
lemma share-append-Ghostsb ${ }^{\prime}$ :
$\wedge \mathcal{S}$. outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb} \neq\{ \} \Longrightarrow$
 (share (takeWhile (Not $\circ$ is-volatile-Write $\mathrm{e}_{\mathrm{sb}}$ ) sb) $\mathcal{S}$ )
apply (induct sb)
apply simp
subgoal for a sb $\mathcal{S}$
apply (case-tac a)
apply force+
done
done
lemma share-all-until-volatile-write-append-Ghost ${ }_{\mathrm{sb}}$ :
assumes no-out-VWrite ${ }_{\text {sb }}$ : outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$
shows $\wedge \mathcal{S}$ i. 【ownership-distinct ts; sharing-consis $\mathcal{S}$ ts;
$\mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {bb }}$ ) sb) $\cap \mathrm{L}=\{ \}$;
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} . \mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {bb }}$ ) sb) $\cap \mathrm{A}=\{ \}$;
$\forall \mathrm{jp}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} . \mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$;
$\forall \mathrm{j}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length ts $\longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$;
$\forall \mathrm{j}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length ts $\longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \} \rrbracket$
$\Longrightarrow$
share-all-until-volatile-write ( $\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.$, is ${ }^{\prime}, \boldsymbol{\vartheta}^{\prime}$, sb @ $\left[\mathrm{Ghost}_{\text {sb }}\right.$ A L R W], $\left.\left.\left.\mathcal{D}^{\prime}, \mathcal{O}^{\prime}\right)\right]\right) \mathcal{S}$ $=$ share-all-until-volatile-write ts $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$

```
proof (induct ts)
    case Nil
    thus ?case by simp
next
    case (Cons t ts)
    obtain p pis ist}\mp@subsup{\mathcal{O}}{\textrm{t}}{}\mp@subsup{\mathcal{R}}{\textrm{t}}{}\mp@subsup{\mathcal{D}}{\textrm{t}}{* acq}\mp@subsup{\textrm{t}}{\textrm{t}}{}\mp@subsup{\vartheta}{\textrm{t}}{}\mp@subsup{\textrm{sb}}{\textrm{t}}{}\mathrm{ where
        t: t= (p
        by (cases t)
    have dist: ownership-distinct (t#ts) by fact
    then interpret ownership-distinct t#ts.
    have consis: sharing-consis }\mathcal{S}\mathrm{ (t#ts) by fact
    then interpret sharing-consis }\mathcal{S}\textrm{t}#\textrm{ts}
```

    have L-prop: \(\forall \mathrm{j} \mathrm{p}\) is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{j}<\) length \((\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow\)
    $(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{L}=\{ \}$ by fact
have A-prop: $\forall \mathrm{j} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$ $(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{A}=\{ \}$ by fact
have R-prop-acq: $\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$ $(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$ by fact
have R-prop: $\forall \mathrm{j} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$ $(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$ by fact
have R-prop-sh: $\forall \mathrm{j} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$ $(\mathrm{t} \# \mathrm{ts})!j=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$ by fact
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then
interpret consis': sharing-consis $\mathcal{S}$ ts.
from sharing-consis [of 0 , simplified, OF t]
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{t}}$ ).
let $? \mathcal{S}^{\prime}=\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{t}}\right) \mathcal{S}\right)$
from freshly-shared-owned [OF consis-sb]
have fresh-owned: $\operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}$.
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}$ - $\operatorname{dom}$ ? $\mathcal{S}^{\prime}$
$\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write $\mathrm{sb}_{\mathrm{sb}}$ ) $\left.\mathrm{sb}_{\mathrm{t}}\right) \cup \mathcal{O}_{\mathrm{t}}$
by simp

```
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-,, \mathrm{sb}^{\prime},-,-,-\right)=\) ts!i in
        all-acquired sb \(b^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(\mathrm{p}_{\mathrm{i}}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}}\) acqii \(\boldsymbol{\vartheta}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i -bound: \(\mathrm{i}<\) length ts
        assume ts- i : ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
        have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
        proof -
from ownership-distinct [of 0 Suc i] ts-i ti-bound
have dist: \(\left(\mathcal{O}_{\mathrm{t}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
    by force
```

from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write $\mathbf{s b}_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{t}}$ ]
have all-acquired sbi $\cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast

## moreover

from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$ by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
by blast
ultimately show?thesis by simp
qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF sep]
have sharing-consis': sharing-consis (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $_{\mathrm{t}}$ ) $\mathcal{S}$ ) ts.
show ?case
proof (cases i)
case 0
with t Cons.prems have eqs: $\mathrm{p}_{\mathrm{t}}=\mathrm{p}$ is $=$ is $\mathcal{O}_{\mathrm{t}}=\mathcal{O} \quad \mathcal{R}_{\mathrm{t}}=\mathcal{R} \quad \vartheta_{\mathrm{t}}=\vartheta \mathrm{sb}_{\mathrm{t}}=\mathrm{sb} \mathcal{D}_{\mathrm{t}}=\mathcal{D}$ by auto
from no-out-VWrite ${ }_{\text {sb }}$
have flush-all: takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}=\mathrm{sb}$
by (auto simp add: outstanding-refs-conv)
from no-out-VWrite ${ }_{\text {sb }}$
have flush-all': takeWhile (Not o is-volatile-Write ${ }_{\mathbf{s b}}$ ) ( sb@[Ghost $_{\mathrm{sb}}$ A L R W]) $=$ sb@[Ghost ${ }_{\text {sb }}$ A L R W]
by (auto simp add: outstanding-refs-conv)

## have share-eq:

 (share (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) sb) $\left.\mathcal{S}\right) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$
apply (simp only: flush-all flush-all')
apply (rule share-append-Ghost ${ }_{\mathbf{s b}}$ [OF no-out-VWrite $\left.{ }_{\text {sb }}\right]$ )
done
from L-prop 0 have L-prop':
$\forall \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\mathrm{i}<$ length ts $\longrightarrow$
ts ! i $=(\mathrm{p}$, is,$\vartheta$, sb, $\mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb) $\cap \mathrm{L}=\{ \}$
apply clarsimp
subgoal for il p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$
apply (drule-tac $x=$ Suc i1 in spec)
apply auto
done
done
from A-prop 0 have A-prop':
$\forall \mathrm{ip}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$.
$\mathrm{i}<$ length ts $\longrightarrow$
ts ! $\mathrm{i}=(\mathrm{p}$, is, $, \mathrm{\imath}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{A}=\{ \}$
apply clarsimp
subgoal for il p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$
apply (drule-tac $x=S u c$ i1 in spec)
apply auto
done
done
from R-prop-acq 0 have R-prop-acq':
$\forall \mathrm{ip}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<$ length ts $\longrightarrow$ ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ) $\cap \mathrm{R}=\{ \}$
apply clarsimp
subgoal for il p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb

```
    apply (drule-tac x=Suc i1 in spec)
    apply auto
    done
    done
    from R-prop 0
    have R-prop':
        i p is \mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. i < length ts }\longrightarrow\textrm{ts}!\textrm{i}=(\textrm{p},\textrm{is},\vartheta,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})
        all-unshared (takeWhile (Not ○ is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb) }\cap\textrm{R}={
    apply clarsimp
    subgoal for i1 p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\mathrm{ sb
    apply (drule-tac x=Suc i1 in spec)
    apply auto
    done
    done
from R-prop-sh 0 have R-prop-sh':
    i p is \mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. i < length ts }\longrightarrow\textrm{ts}!\textrm{i}=(\textrm{p},\textrm{is},\vartheta,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})
    all-shared (takeWhile (Not ○ is-volatile-Write sb})\textrm{sb})\cap\textrm{R}={
    apply clarsimp
    subgoal for i1 p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta s
    apply (drule-tac x=Suc i1 in spec)
    apply auto
    done
    done
```

from share-all-until-volatile-write-commute [OF dist' sharing-consis' L-prop' A-prop ${ }^{\prime}$ R-prop-acq ${ }^{\prime}$ R-prop ${ }^{\prime}$

R-prop-sh $]$
have share-all-until-volatile-write ts (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb)
$\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=$
share-all-until-volatile-write ts (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ )
$\mathcal{S}) \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$
by (simp add: eqs)
with share-eq
show ?thesis
by (clarsimp simp add: 0 t)
next
case (Suc k)
from L-prop Suc
have L-prop': $\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{ts}) \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow(\mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
$\qquad$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{L}=\{ \}$ by force
from A-prop Suc
have A-prop': $\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{ts}) \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow(\mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{A}=\{ \}$ by force from R-prop-acq Suc have R-prop-acq':
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} . \mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cap \mathrm{R}=\{ \}$ by force
from R-prop Suc
have R-prop':
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{R}=\{ \}$ by force
from R-prop-sh Suc have R-prop-sh':
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ) $\cap \mathrm{R}=\{ \}$ by force
from Cons.prems Suc obtain k -bound: $\mathrm{k}<$ length ts and ts-k: ts!k $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}$, $\mathcal{O}, \mathcal{R})$
by auto
from Cons.hyps [OF dist' sharing-consis' $k$-bound ts-k L-prop' A-prop' R-prop-acq' R-prop' R-prop-sh']
show ?thesis
by (clarsimp simp add: t Suc)
qed
qed
lemma share-domain-changes:
$\Lambda \mathcal{S} \mathcal{S}^{\prime} . \mathrm{a} \in$ all-shared $\mathrm{sb} \cup$ all-unshared $\mathrm{sb} \Longrightarrow$ share $\mathrm{sb} \mathcal{S}^{\prime} \mathrm{a}=\operatorname{share} \mathrm{sb} \mathcal{S} \mathrm{a}$ proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain a-in: $a \in R \cup$ all-shared $s b \cup L \cup$ all-unshared sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in R$ )
case True
from True have $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$ by (auto simp add: augment-shared-def restrict-shared-def split: option.splits) from share-shared-eq [where $\mathcal{S}^{\prime}=\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ and $\mathcal{S}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$, OF this] have share sb $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\operatorname{share} \operatorname{sb}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}$
by auto

```
    then show ?thesis
    by (clarsimp simp add: Writesb volatile)
    next
    case False
    note not-R = this
    show ?thesis
    proof (cases a }\inL\mathrm{ L)
        case True
        from not-R True have (S'S
            by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L},O
this]
            have share sb (S'S
                by auto
            then show ?thesis
                by (clarsimp simp add: Write sb volatile)
    next
            case False
            with not-R a-in have a }\in\mathrm{ all-shared sb U all-unshared sb
                by auto
            from Cons.hyps [OF this]
            show ?thesis by (clarsimp simp add: Write sb volatile)
            qed
    qed
next
                            case False with Cons show ?thesis by (auto simp add: Write sb)
qed
next
    case Read}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ with Cons show ?thesis by (auto)
next
    case Progsb with Cons show ?thesis by (auto)
next
    case (Ghost sb A L R W)
    from Cons.prems obtain a-in: a }\inR\cupR\cup\mathrm{ all-shared sb UL U all-unshared sb
        by (clarsimp simp add: Ghost sb)
show ?thesis
    proof (cases a }\inR\mathrm{ )
        case True
    from True have ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L},\mathrm{ ,OF this]
    have share sb (\mathcal{S}
        by auto
    then show ?thesis
    by (clarsimp simp add: Ghost sb)
next
    case False
    note not-R = this
    show ?thesis
    proof (cases a }\inL\mathrm{ L)
```

```
            case True
            from not-R True have ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
                by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ , OF this]
            have share sb (\mathcal{S}}\mp@subsup{\mathcal{\prime}}{}{\prime}\mp@subsup{\textrm{w}}{}{\prime}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}=\operatorname{share sb}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a
                by auto
            then show?thesis
                by (clarsimp simp add: Ghostsb
        next
            case False
            with not-R a-in have a }\in\mathrm{ all-shared sb U all-unshared sb
                by auto
            from Cons.hyps [OF this]
            show ?thesis by (clarsimp simp add: Ghost sb
        qed
    qed
    qed
qed
lemma share-domain-changesX:
\(\Lambda \mathcal{S} \mathcal{S}^{\prime} \mathrm{X} . \forall \mathrm{a} \in \mathrm{X} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}\)
\(\Longrightarrow \mathrm{a} \in\) all-shared \(\mathrm{sb} \cup\) all-unshared \(\mathrm{sb} \cup \mathrm{X} \Longrightarrow\) share \(\mathrm{sb} \mathcal{S}^{\prime} \mathrm{a}=\) share \(\mathrm{sb} \mathcal{S} \mathrm{a}\)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
then have shared-eq: \(\forall \mathrm{a} \in \mathrm{X} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}\) by auto
show ?case
proof (cases x)
case ( Write \(_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain a-in: \(a \in R \cup\) all-shared sb \(\cup L \cup\) all-unshared \(s b \cup X\)
by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
show ?thesis
proof (cases a \(\in R\) )
case True
from True have \(\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}\)
by (auto simp add: augment-shared-def restrict-shared-def split: option.splits) from share-shared-eq [where \(\mathcal{S}^{\prime}=\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) and \(\mathcal{S}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\), OF this] have share sb \(\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\operatorname{share} \mathrm{sb}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}\)
by auto
then show ?thesis
by (clarsimp simp add: Write \({ }_{\text {sb }}\) volatile)
next
case False
note not-R = this
```

```
    show ?thesis
    proof (cases a }\inL\mathrm{ )
        case True
        from not-R True have ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
            by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ , OF
this]
            have share sb (\mathcal{S}}\mp@subsup{}{\prime}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}=\operatorname{share sb}(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{\prime}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a
                by auto
            then show ?thesis
                by (clarsimp simp add: Write sb volatile)
            next
            case False
            from shared-eq have shared-eq': }\forall\textrm{a}\in\textrm{X}.(\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
                by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            from False not-R a-in have a }\in\mathrm{ all-shared sb }\cup\mathrm{ all-unshared sb }\cup\textrm{X
                by auto
            from Cons.hyps [OF shared-eq' this]
            show ?thesis by (clarsimp simp add: Writesb volatile)
            qed
        qed
    next
        case False with Cons show ?thesis by (auto simp add: Write sb
    qed
next
    case Read
next
    case Progsb with Cons show ?thesis by (auto)
next
    case (Ghostsb A L R W)
    from Cons.prems obtain a-in: a }\in\textrm{R}\cup\mathrm{ all-shared sb }\cup\textrm{L}\cup\mathrm{ all-unshared sb }\cup\textrm{X
    by (clarsimp simp add: Ghostsb)
    show ?thesis
    proof (cases a G R)
        case True
        from True have (S'S
            by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L},\mathrm{ ,OF this]
    have share sb (\mathcal{S}
        by auto
    then show ?thesis
    by (clarsimp simp add: Ghostsb
next
    case False
    note not-R = this
    show ?thesis
    proof (cases a \in L)
        case True
        from not-R True have ( }\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
            by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
```

```
            from share-shared-eq [where }\mp@subsup{\mathcal{S}}{}{\prime}=\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}\mathrm{ and }\mathcal{S}=\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ , OF this]
            have share sb (\mathcal{S}
                by auto
            then show ?thesis
                by (clarsimp simp add: Ghostsb)
        next
            case False
            from shared-eq have shared-eq':}\forall\textrm{a}\in\textrm{X}.(\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a
                by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            from False not-R a-in have a }\in\mathrm{ all-shared sb }\cup\mathrm{ all-unshared sb }\cup\textrm{X
                by auto
            from Cons.hyps [OF shared-eq' this]
            show ?thesis by (clarsimp simp add: Ghostsb)
        qed
    qed
    qed
qed
lemma share-unchanged:
    \mathcal{S}.a|}\mathrm{ all-shared sb }\cup\mathrm{ all-unshared sb }\cup\mathrm{ all-acquired sb }\Longrightarrow\mathrm{ share sb }\mathcal{S}\textrm{a}=\mathcal{S}\mathrm{ a
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a' sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
            from Cons.prems obtain a-R: a & R and a-L: a }\not=\textrm{L}\mathrm{ and a-A: a & A
                and a': a & all-shared sb \cup all-unshared sb U all-acquired sb
                by (clarsimp simp add: Write sb True)
            from Cons.hyps [OF a]
            have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}.
            moreover
            from a-R a-L a-A have (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{S
            by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
            ultimately
            show ?thesis
            by (clarsimp simp add: Write sb True)
    next
        case False with Cons show ?thesis by (auto simp add: Write sb)
        qed
    next
        case Read
    next
        case Progsb with Cons show ?thesis by (auto)
    next
```

```
    case (Ghostsb A L R W)
    from Cons.prems obtain a-R: a & R and a-L: a }\not\in\textrm{L}\mathrm{ and a-A: a & A
        and a': a & all-shared sb \cup all-unshared sb }\cup\mathrm{ all-acquired sb
        by (clarsimp simp add: Ghostsb)
    from Cons.hyps [OF a']
    have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a}.
    moreover
    from a-R a-L a-A have (\mathcal{S }\mp@subsup{\oplus}{\textrm{W}}{\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{S}
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    ultimately
    show ?thesis
        by (clarsimp simp add: Ghostsb)
    qed
qed
lemma share-augment-release-commute:
assumes dist: (R L L U A) \cap (all-shared sb \cup all-unshared sb \cup all-acquired sb)={}
shows (share sb \mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})=\operatorname{share sb (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
proof -
    from dist have shared-eq: }\forall\textrm{a}\in\mathrm{ all-acquired sb. (S }\mp@subsup{\mathcal{W}}{\textrm{W}}{\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathcal{S}\textrm{a
    by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
{
    fix a
    assume a-in: a }\in\mathrm{ all-shared sb }\cup\mathrm{ all-unshared sb }\cup\mathrm{ all-acquired sb
    from share-domain-changesX [OF shared-eq this]
    have share sb (\mathcal{S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=\mathrm{ share sb }\mathcal{S}\textrm{a}.
    also
    from dist a-in have ... = (share sb S S }\mp@subsup{\textrm{w}}{\textrm{W}}{\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    finally have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathrm{ share sb S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}.
}
moreover
{
    fix a
    assume a-notin: a & all-shared sb \cup all-unshared sb \cup all-acquired sb
    from share-unchanged [OF a-notin]
    have share sb (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}.
    moreover
    from share-unchanged [OF a-notin]
    have share sb S a = \mathcal{S a.}
    hence (share sb \mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\textrm{a
        by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
    ultimately have share sb (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a}=(\mathrm{ share sb }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\textrm{a
        by simp
}
ultimately show ?thesis
    apply -
    apply (rule ext)
    subgoal for }\textrm{x
```

```
    apply (case-tac x }\in\mathrm{ all-shared sb U all-unshared sb U all-acquired sb)
    apply auto
    done
    done
qed
lemma share-append-commute:
    yss S.(all-shared xs \cup all-unshared xs \cup all-acquired xs) }
                            (all-shared ys }\cup\mathrm{ all-unshared ys }\cup\mathrm{ all-acquired ys) ={}
"share xs (share ys \mathcal{S})=\mathrm{ share ys (share xs }\mathcal{S})
proof (induct xs)
    case Nil thus ?case by simp
next
    case (Cons x xs)
    show ?case
    proof (cases x)
    case (Write sb volatile a sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        note volatile=this
        from Cons.prems have
            dist':}(\mathrm{ (all-shared xs }\cup\mathrm{ all-unshared xs }\cup\mathrm{ all-acquired xs) }
                        (all-shared ys }\cup\mathrm{ all-unshared ys }\cup\mathrm{ all-acquired ys) ={}
                apply (clarsimp simp add: Writesb True)
                apply blast
                done
        from Cons.prems have
            dist: (R\cupL\cupA)\cap(all-shared ys \cup all-unshared ys \cup all-acquired ys)={}
                apply (clarsimp simp add: Writesb True)
                apply blast
                done
            from share-augment-release-commute [OF dist]
            have (share ys S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})=\operatorname{share ys (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
            with Cons.hyps [OF dist']
            show ?thesis
                by (clarsimp simp add: Writesb True)
        next
            case False with Cons show ?thesis
                by (clarsimp simp add: Write sb False)
        qed
    next
        case Read
    next
        case Prog
    next
        case (Ghost sb A L R W)
        from Cons.prems have
            dist':(all-shared xs \cup all-unshared xs \cup all-acquired xs) \cap
```

(all-shared ys $\cup$ all-unshared ys $\cup$ all-acquired ys) $=\{ \}$
apply (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
apply blast done
from Cons.prems have
dist: $(\mathrm{R} \cup \mathrm{L} \cup \mathrm{A}) \cap($ all-shared ys $\cup$ all-unshared ys $\cup$ all-acquired ys $)=\{ \}$
apply (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
apply blast
done
from share-augment-release-commute [OF dist]
have (share ys $\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\operatorname{share}$ ys $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
with Cons.hyps [OF dist']
show ?thesis by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
lemma share-append-commute':
assumes dist: (all-shared xs $\cup$ all-unshared xs $\cup$ all-acquired xs) $\cap$
(all-shared ys $\cup$ all-unshared ys $\cup$ all-acquired ys) $=\{ \}$
shows share (ys@xs) $\mathcal{S}=$ share $(\mathrm{xs} @ y s) \mathcal{S}$
proof -
from share-append-commute [OF dist] share-append [of xs ys] share-append [of ys xs]
show ?thesis
by simp
qed
lemma share-all-until-volatile-write-share-commute:
shows $\bigwedge \mathcal{S}$ (sb'::'a memref list). [ownership-distinct ts; sharing-consis $\mathcal{S}$ ts;
$\forall$ i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta($ sb::'a memref list). i $<$ length ts
$\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$ all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) $\cap$ (all-shared $\mathrm{sb}^{\prime} \cup$ all-unshared $\mathrm{sb}^{\prime} \cup$ all-acquired $\mathrm{sb}^{\prime}$ ) $=\{ \} \rrbracket$

## $\Longrightarrow$

share-all-until-volatile-write ts $\left(\right.$ share $\left.\mathrm{sb}^{\prime} \mathcal{S}\right)=$
share $\mathrm{sb}^{\prime}$ (share-all-until-volatile-write ts $\mathcal{S}$ )
proof (induct ts)
case Nil
thus ?case by simp
next
case (Cons t ts)
obtain $p_{\mathrm{t}}$ is $_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where
$\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t )
let ?take $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}_{\mathrm{t}}\right)$
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
have consis: sharing-consis $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$ by fact
then interpret sharing-consis $\mathcal{S}$ t\#ts .
have dist-prop: $\forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{i}<$ length ( $\mathrm{t} \# \mathrm{ts}$ )
$\longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cup$
all-unshared (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) $\cap$
(all-shared $\mathrm{sb}^{\prime} \cup$ all-unshared $\mathrm{sb}^{\prime} \cup$ all-acquired $\left.\mathrm{sb}^{\prime}\right)=\{ \}$ by fact
from dist-prop [rule-format, of 0] t
have dist-t: (all-shared ?take $\cup$ all-unshared ?take $\cup$ all-acquired ?take) $\cap$
(all-shared $\mathrm{sb}^{\prime} \cup$ all-unshared $\mathrm{sb}^{\prime} \cup$ all-acquired $\left.\mathrm{sb}^{\prime}\right)=\{ \}$
apply clarsimp
done
from dist-prop have
dist-prop' $: \forall \mathrm{i}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. i < length ts $\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cup$
all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) $\cap$
(all-shared $\mathrm{sb}^{\prime} \cup$ all-unshared $\mathrm{sb}^{\prime} \cup$ all-acquired $\left.\mathrm{sb}^{\prime}\right)=\{ \}$
apply (clarsimp)
subgoal for i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb
apply (drule-tac $x=$ Suc i in spec)
apply clarsimp
done
done
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then
interpret consis': sharing-consis $\mathcal{S}$ ts .
from sharing-consis [of 0 , simplified, OF t ]
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}}$ ?take.
let $?^{\prime} \mathcal{S}^{\prime}=($ share ?take $\mathcal{S})$
from freshly-shared-owned [OF consis-sb]
have fresh-owned: dom ${ }^{\mathcal{S}^{\prime}}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}$.
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}$
$\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{t}}\right) \cup \mathcal{O}_{\mathrm{t}}$

```
by simp
```

```
have sep:
    \(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=\) ts! \(!\) in
        all-acquired \(\mathrm{sb}^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
        all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(p_{i}\) is \(\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}}\) acqio \(\vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i-bound: \(\mathrm{i}<\) length ts
        assume ts-i: ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \mathcal{Y}_{\mathrm{i}}, \mathrm{Sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
        have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
        proof -
from ownership-distinct [of 0 Suc i] ts-i ti-bound
have dist: \(\left(\mathcal{O}_{\mathrm{t}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}\)
by force
```

from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ]
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast
moreover
from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}}$ sb $_{\mathrm{i}}$
by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
by blast
ultimately show?thesis by simp
qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF sep]
have sharing-consis': sharing-consis ? $\mathcal{S}^{\prime}$ ts.
have share-all-until-volatile-write ts (share ?take $($ share sb' $\mathcal{S})$ ) $=$
share $\mathrm{sb}^{\prime}$ (share-all-until-volatile-write ts (share ?take $\mathcal{S}$ ))
proof -
from share-append-commute [OF dist-t]
have $\left(\right.$ share ?take $\left(\right.$ share $\left.\left.\mathrm{sb}^{\prime} \mathcal{S}\right)\right)=\left(\right.$ share $\mathrm{sb}^{\prime}($ share ?take $\left.\mathcal{S})\right)$.

```
    then
    have share-all-until-volatile-write ts (share ?take (share sb'}\mathcal{S})\mathrm{ ) =
        share-all-until-volatile-write ts (share sb' (share ?take \mathcal{S})
        by (simp)
    also
    from Cons.hyps [OF dist' sharing-consis' dist-prop']
    have ... = share sb' (share-all-until-volatile-write ts (share ?take \mathcal{S})).
    finally show ?thesis .
    qed
    then show ?case
    by (clarsimp simp add: t)
qed
```

lemma all-shared-takeWhile-subset: all-shared (takeWhile P sb) $\subseteq$ all-shared sb using all-shared-append [of (takeWhile P sb) (dropWhile P sb)] by auto
lemma all-shared-dropWhile-subset: all-shared (dropWhile P sb) $\subseteq$ all-shared sb using all-shared-append [of (takeWhile P sb) (dropWhile P sb)]
by auto
lemma all-unshared-takeWhile-subset: all-unshared (takeWhile P sb) $\subseteq$ all-unshared sb using all-unshared-append [of (takeWhile P sb) (dropWhile P sb)] by auto
lemma all-unshared-dropWhile-subset: all-unshared (dropWhile P sb ) $\subseteq$ all-unshared sb using all-unshared-append [of (takeWhile P sb) (dropWhile P sb)] by auto
lemma all-acquired-takeWhile-subset: all-acquired (takeWhile $\mathrm{P} s \mathrm{sb}) \subseteq$ all-acquired sb using all-acquired-append [of (takeWhile P sb) (dropWhile P sb)] by auto
lemma all-acquired-dropWhile-subset: all-acquired (dropWhile P sb) $\subseteq$ all-acquired sb using all-acquired-append [of (takeWhile P sb) (dropWhile P sb)]
by auto
lemma share-all-until-volatile-write-flush-commute:
assumes takeWhile-empty: (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $=[]$
shows $\wedge \mathcal{S}$ R L W A i. [ownership-distinct ts; sharing-consis $\mathcal{S}$ ts; i < length ts; $\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) ;$
$\forall \mathrm{i} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ (sb::'a memref list). $\mathrm{i}<$ length ts $\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cup$
all-unshared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$
all-acquired (takeWhile (Noto is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)) $\cap$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$
all-unshared (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb') $\cup$
all-acquired ( takeWhile $\left.^{(\text {Not } \circ \text { is-volatile-Write }}{ }_{\text {sb }}\right)$ sb $\left.\left.{ }^{\prime}\right)\right)=\{ \}$;
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ (sb::'a memref list). $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{i} \neq \mathrm{j}$
$\longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$

```
\(\Longrightarrow\)
share-all-until-volatile-write ( \(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.\), is \(\left.\left.\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\)
share (takeWhile (Not o is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb') (share-all-until-volatile-write ts \(\mathcal{S} \oplus \mathrm{w} \mathrm{R}\)
\(\ominus_{\mathrm{A}} \mathrm{L}\) )
proof (induct ts)
    case Nil
    thus ?case by simp
next
    case (Cons t ts)
    obtain \(p_{t} \operatorname{is}_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}\) where
        \(\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)\)
        by (cases t )
    let ? take \(=\left(\right.\) takeWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{t}}\right)\)
    let ?take-sb' \(=\left(\right.\) takeWhile \(^{(N o t} \circ\) is-volatile-Write \(\left.\left.{ }_{s b}\right) \mathrm{sb}^{\prime}\right)\)
    let \(?\) drop \(=\left(\right.\) dropWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{t}}\) )
    have dist: ownership-distinct ( \(\mathrm{t} \# \mathrm{ts}\) ) by fact
    then interpret ownership-distinct \(\mathrm{t} \# \mathrm{ts}\).
    have consis: sharing-consis \(\mathcal{S}\) ( \(\mathrm{t} \# \mathrm{ts}\) ) by fact
    then interpret sharing-consis \(\mathcal{S}\) t\#ts .
    have dist-prop: \(\forall \mathrm{i}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{i}<\) length ( \(\mathrm{t} \# \mathrm{ts}\) )
        \(\longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
                            (all-shared (takeWhile (Not o is-volatile-Write \({ }_{s b}\) ) sb) \(\cup\)
                            all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb) \(\cup\)
                            all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb)) \(\cap\)
                            (all-shared ?take-sb \({ }^{\prime} \cup\) all-unshared ?take-sb' \(\cup\) all-acquired ?take-sb') \(=\)
\{\} by fact
    from dist-prop [rule-format, of 0 ] t
    have dist-t: (all-shared ?take \(\cup\) all-unshared ?take \(\cup\) all-acquired ?take) \(\cap\)
        (all-shared ?take-sb' \(\cup\) all-unshared ?take-sb \({ }^{\prime} \cup\) all-acquired ?take-sb \()=\{ \}\)
    apply clarsimp
    done
from dist-prop have
dist-prop': \(\forall \mathrm{i}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. i \(<\) length ts
        \(\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
            (all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb) \(\cup\)
            all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb) \(\cup\)
            all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb)) \(\cap\)
            (all-shared ?take-sb \({ }^{\prime} \cup\) all-unshared ?take-sb \({ }^{\prime} \cup\) all-acquired ?take-sb \(\left.{ }^{\prime}\right)=\{ \}\)
    apply (clarsimp)
    subgoal for i p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb
    apply (drule-tac \(x=\) Suc i in spec)
    apply clarsimp
    done
    done
have dist-prop-R-L-A: \(\forall \mathrm{j}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{j}<\) length ( \(\mathrm{t} \# \mathrm{ts}\) ) \(\longrightarrow \mathrm{i} \neq \mathrm{j}\)
        \(\longrightarrow(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
                            (all-shared sb \(\cup\) all-unshared sb \(\cup\) all-acquired sb) \(\cap\)
```

$$
(\mathrm{R} \cup \mathrm{~L} \cup \mathrm{~A})=\{ \} \text { by fact }
$$

from ownership-distinct-tl [OF dist] have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis] have consis': sharing-consis $\mathcal{S}$ ts. then
interpret consis': sharing-consis $\mathcal{S}$ ts .
from sharing-consis [of 0 , simplified, OF t]
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{t}}$ ).
have aargh: $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)=\left(\lambda a . \neg\right.$ is-volatile-Write $_{s b}$ a)
by (rule ext) auto

```
show ?case
proof (cases i)
    case 0
```

    with t Cons.prems have eqs: \(\mathrm{p}_{\mathrm{t}}=\mathrm{p}\) is \(=\mathrm{is} \mathcal{O}_{\mathrm{t}}=\mathcal{O} \quad \mathcal{R}_{\mathrm{t}}=\mathcal{R} \quad \vartheta_{\mathrm{t}}=\vartheta \mathrm{sb}_{\mathrm{t}}=\mathrm{sb} \quad \mathcal{D}_{\mathrm{t}}=\mathcal{D}\)
        by auto
    let \({ }^{\mathcal{S}} \mathcal{S}^{\prime}=\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\)
    from dist-prop-R-L-A 0 have
        dist-prop-R-L-A \(: \forall\) i p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{i}<\) length ts
            \(\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
                                    (all-shared sb \(\cup\) all-unshared \(s b \cup\) all-acquired sb)
                                    \((R \cup L \cup A)=\{ \}\)
    apply (clarsimp)
    subgoal for il p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}\)
    apply (drule-tac \(x=\) Suc i1 in spec)
    apply clarsimp
    done
    done
    then
    have dist-prop-R-L-A \({ }^{\prime \prime}: \forall \mathrm{i}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{i}<\) length ts
        \(\longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
    (all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cup\) all-unshared (takeWhile (Not
    $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$
all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)) $\cap$
$(\mathrm{R} \cup \mathrm{L} \cup \mathrm{A})=\{ \}$
apply (clarsimp)
subgoal for ip is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb
apply (cut-tac $\mathrm{sb}=\mathrm{sb}$ in all-shared-takeWhile-subset [where $\mathrm{P}=$ Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ])
apply (cut-tac $\mathrm{sb}=\mathrm{sb}$ in all-unshared-takeWhile-subset [where $\mathrm{P}=$ Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right]$ )
apply (cut-tac $\mathrm{sb}=\mathrm{sb}$ in all-acquired-takeWhile-subset [where $\mathrm{P}=$ Not $\circ$ is-volatile-Write $_{\text {sb }}$ ])
apply fastforce
done
done
have sep: $\forall \mathrm{i}<$ length ts.
let $(-,-,-$, sb,,,---$)=$ ts!i
in $\forall \mathrm{a} \in$ all-acquired $\mathrm{sb} . ? \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
proof -
\{
fix i $p_{i}$ is $\mathcal{O}_{i} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}}$ acq $_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$ a
assume i-bound: i < length ts
assume ts- i : ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
assume a-in: a $\in$ all-acquired $\mathrm{sb}_{\mathrm{i}}$
have $?^{\prime} \mathrm{a}=\mathcal{S} \mathrm{a}$
proof -
from dist-prop-R-L-A' [rule-format, OF i-bound ts-i] a-in
show ?thesis
by (auto simp add: augment-shared-def restrict-shared-def split: option.splits)
qed
\}
thus ?thesis by auto
qed
from consis'.sharing-consis-shared-exchange [OF sep]
have sharing-consis': sharing-consis ? $\mathcal{S}^{\prime}$ ts.
from share-all-until-volatile-write-share-commute [of ts $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ), OF dist' sharing-consis' dist-prop $]$
have share-all-until-volatile-write ts (share ?take-sb $\left.{ }^{\prime} ? \mathcal{S}^{\prime}\right)=$ share ?take-sb' (share-all-until-volatile-write ts ? $\mathcal{S}^{\prime}$ ) .

## moreover

from dist-prop-R-L-A ${ }^{\prime \prime}$
have (share-all-until-volatile-write ts $\left.\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)=$ (share-all-until-volatile-write ts $\mathcal{S} \oplus \mathrm{w}_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
apply -
apply (rule share-all-until-volatile-write-commute [OF dist' consis', of L A R W,symmetric])
apply (clarsimp,blast) +
done
ultimately
show ?thesis

```
    using takeWhile-empty
    by (clarsimp simp add: t 0 aargh eqs)
    next
    case (Suc k)
    from Cons.prems Suc obtain k-bound: k < length ts and ts-k: ts!k = (p, is,\vartheta, sb, \mathcal{D}
O,\mathcal{R}
by auto
let ?\mp@subsup{\mathcal{S}}{}{\prime}=(\mathrm{ share (takeWhile (Not o is-volatile-Write sb})\mp@subsup{\textrm{sb}}{\textrm{t}}{})\mathcal{S})
from freshly-shared-owned [OF consis-sb]
have fresh-owned: dom ?S'S
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: dom S - dom ? S'
    \subseteqall-acquired (takeWhile (Not o is-volatile-Write sb})\mp@subsup{)}{\mathrm{ sb}}{\textrm{t}}\mathrm{ ) }\cup\mp@subsup{\mathcal{O}}{\textrm{t}}{
    by simp
```

from freshly-shared-owned [OF consis-sb]
have fresh-owned: $\operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}$.
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb]
have unshared-acq-owned: $\operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}$
$\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\left.\mathrm{sb}_{\mathrm{t}}\right) \cup \mathcal{O}_{\mathrm{t}}$
by simp
have sep:
$\forall \mathrm{i}<$ length ts. let $\left(-,,-,-, \mathrm{sb}^{\prime},-,-,-\right)=$ ts! $!$ in
all-acquired sb' $\cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge$
all-unshared $\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
proof -
\{
fix i $\mathrm{p}_{\mathrm{i}}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}}$ acq $_{\mathrm{i}} \boldsymbol{\vartheta}_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}$
assume i-bound: i $<$ length ts
assume ts-i: ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge$
all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$
proof -
from ownership-distinct [of 0 Suc i] ts-i t i-bound
have dist: $\left(\mathcal{O}_{\mathrm{t}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}$
by force
from dist unshared-acq-owned all-acquired-takeWhile [of (Not o is-volatile-Write ${ }_{s b}$ ) sb ${ }_{\mathrm{t}}$ ]
have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$
by blast
moreover
from sharing-consis [of Suc i] ts-i i-bound

```
    have sharing-consistent }\mathcal{S}\mp@subsup{\mathcal{O}}{\textrm{i}}{}\mp@subsup{\textrm{sb}}{\textrm{i}}{
        by force
    from unshared-acquired-or-owned [OF this]
    have all-unshared sb}\mp@subsup{\textrm{i}}{\textrm{i}}{}\subseteq\mathrm{ all-acquired sb}\mp@subsup{\textrm{si}}{\textrm{i}}{}\cup\mp@subsup{\mathcal{O}}{\textrm{i}}{}\mathrm{ .
    with dist fresh-owned
    have all-unshared sbi}\mp@subsup{\textrm{i}}{\textrm{i}}{}\cap\operatorname{dom ?\mp@subsup{\mathcal{S}}{}{\prime}-\operatorname{dom}\mathcal{S}={}
        by blast
    ultimately show ?thesis by simp
        qed
        }
        thus ?thesis
        by (fastforce simp add: Let-def)
    qed
    from consis'.sharing-consis-preservation [OF sep]
    have sharing-consis': sharing-consis ?S' ts.
    from dist-prop-R-L-A [rule-format, of 0] Suc t
    have dist-t-R-L-A: (all-shared }\mp@subsup{\textrm{sb}}{\textrm{t}}{}\cup\mathrm{ all-unshared sb
        (R\cupL\cupA)={}
        apply clarsimp
        done
    from dist-t-R-L-A
    have (R\cupL\cupA) \cap (all-shared ?take U all-unshared ?take \cup all-acquired ?take) = {}
        using all-shared-append [of ?take ?drop] all-unshared-append [of ?take ?drop]
all-acquired-append [of ?take ?drop]
        by auto
    from share-augment-release-commute [OF this]
    have share ?take S 利 R }\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}=\mathrm{ share ?take (S }\mp@subsup{\mathcal{W}}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
    moreover
    from dist-prop-R-L-A Suc
    have }\forall\textrm{j p}\mathrm{ is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\mathrm{ sb. j < length (ts) }\longrightarrow\textrm{k}\not=\textrm{j
        (ts)!j=(p,is,\vartheta,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow
                            (all-shared sb \cup all-unshared sb \cup all-acquired sb) \cap
                    (R\cupL\cupA)={}
        apply (clarsimp)
        subgoal for jp is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\mathrm{ sb
        apply (drule-tac x=Suc j in spec)
        apply clarsimp
        done
        done
    note Cons.hyps [OF dist' sharing-consis' k-bound ts-k dist-prop' this, of W]
    ultimately
    show ?thesis
        by (clarsimp simp add: t Suc )
    qed
qed
```

lemma share-all-until-volatile-write-Ghost ${ }_{\text {sb }}$-commute:
shows $\bigwedge \mathcal{S}$ i. 【ownership-distinct ts; sharing-consis $\mathcal{S}$ ts; i < length ts; ts!i $=\left(\mathrm{p}, \mathrm{is}, \vartheta\right.$, Ghost $_{\mathrm{sb}}$ A L R W\#sb, $\left.\mathcal{D}, \mathcal{O}, \mathcal{R}\right)$;
$\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cup$ all-unshared (takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb) $\cup$ all-acquired $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb) $) \cap$

$$
(\mathrm{R} \cup \mathrm{~L} \cup \mathrm{~A})=\{ \} \rrbracket
$$

$\Longrightarrow$ share-all-until-volatile-write (ts $[\mathrm{i}:$
share-all-until-volatile-write ts $\mathcal{S}$
proof (induct ts)
case Nil
thus ?case by simp
next
case (Cons t ts)
obtain $\mathrm{p}_{\mathrm{t}}$ is $\mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where
$\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t )
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct t\#ts .
have consis: sharing-consis $\mathcal{S}$ ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret sharing-consis $\mathcal{S} \mathrm{t} \# \mathrm{ts}$.
have dist-prop: $\forall \mathrm{j} \mathrm{p}$ is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $(\mathrm{t} \# \mathrm{ts}) \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$ $(\mathrm{t} \# \mathrm{ts})!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
(all-shared (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\cup$ all-unshared (takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb) $\cup$ all-acquired $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb) $) \cap$

$$
(\mathrm{R} \cup \mathrm{~L} \cup \mathrm{~A})=\{ \} \text { by fact }
$$

from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
from sharing-consis-tl [OF consis]
have consis': sharing-consis $\mathcal{S}$ ts.
then
interpret consis': sharing-consis $\mathcal{S}$ ts .
from sharing-consis [of 0 , simplified, OF t]
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$.
from sharing-consistent-takeWhile [OF this]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{t}}\left(\right.$ takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{t}}$ ).
let $? \mathcal{S}^{\prime}=\left(\right.$ share $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{t}}\right) \mathcal{S}\right)$
from freshly-shared-owned [OF consis-sb]
have fresh-owned: $\operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S} \subseteq \mathcal{O}_{\mathrm{t}}$.
from unshared-all-unshared [OF consis-sb] unshared-acquired-or-owned [OF consis-sb] have unshared-acq-owned: dom $\mathcal{S}$ - dom ? $\mathcal{S}^{\prime}$ $\subseteq$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\left.\mathrm{sb}_{\mathrm{t}}\right) \cup \mathcal{O}_{\mathrm{t}}$ by simp

## have sep

```
\(\forall \mathrm{i}<\) length ts. let \(\left(-,-,-, \mathrm{sb}^{\prime},-,-,-\right)=\) ts!i in
all-acquired \(\mathrm{sb}^{\prime} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
all-unshared \(\mathrm{sb}^{\prime} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
proof -
    \{
        fix i \(\mathrm{p}_{\mathrm{i}} \mathrm{is}_{\mathrm{i}} \mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \vartheta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}}\)
        assume i-bound: \(\mathrm{i}<\) length ts
        assume ts-i: ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \vartheta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)\)
        have all-acquired \(\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \} \wedge\)
            all-unshared \(\mathrm{sb}_{\mathrm{i}} \cap\) dom \(? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}\)
        proof -
```

from ownership-distinct [of 0 Suc i] ts-i t i-bound
have dist: $\left(\mathcal{O}_{\mathrm{t}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right)=\{ \}$
by force
from dist unshared-acq-owned all-acquired-takeWhile [of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ] have all-acquired $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} \mathcal{S}-\operatorname{dom} ? \mathcal{S}^{\prime}=\{ \}$ by blast

## moreover

from sharing-consis [of Suc i] ts-i i-bound
have sharing-consistent $\mathcal{S} \mathcal{O}_{\mathrm{i}}$ sb $_{\mathrm{i}}$ by force
from unshared-acquired-or-owned [OF this]
have all-unshared $\mathrm{sb}_{\mathrm{i}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{i}} \cup \mathcal{O}_{\mathrm{i}}$.
with dist fresh-owned
have all-unshared $\mathrm{sb}_{\mathrm{i}} \cap \operatorname{dom} ? \mathcal{S}^{\prime}-\operatorname{dom} \mathcal{S}=\{ \}$ by blast
ultimately show ?thesis by simp qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from consis'.sharing-consis-preservation [OF sep]
have sharing-consis ${ }^{\prime}$ : sharing-consis (share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }_{\mathrm{t}}$ ) $\mathcal{S}$ )
ts.
show ?case

```
proof (cases i)
```

    case 0
    with t Cons.prems have eqs: $\mathrm{p}_{\mathrm{t}}=\mathrm{p}$ is $=\mathrm{is} \mathcal{O}_{\mathrm{t}}=\mathcal{O} \quad \mathcal{R}_{\mathrm{t}}=\mathcal{R} \quad \vartheta_{\mathrm{t}}=\vartheta \mathrm{sb}_{\mathrm{t}}=$ Ghost $_{\mathrm{sb}}$ A L R $\mathrm{W} \#$ sb $\mathcal{D}_{\mathrm{t}}=\mathcal{D}$
by auto
show ?thesis
by (clarsimp simp add: 0 t eqs)
next
case (Suc k)
from Cons.prems Suc obtain k-bound: $\mathrm{k}<$ length ts and ts-k: ts!k $=$ ( $\mathrm{p}, \mathrm{is}, \vartheta$, Ghost ${ }_{\text {sb }}$ A L R W\#sb, $\mathcal{D}, \mathcal{O}, \mathcal{R})$
by auto
from dist-prop Suc
have dist-prop': $\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts} \longrightarrow \mathrm{k} \neq \mathrm{j} \longrightarrow \mathrm{ts}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
$\qquad$
(all-shared (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) sb) $\cup$ all-unshared (takeWhile $\left(\right.$ Not $^{\circ}$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb) $\cup$ all-acquired (takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)\right) \cap$

$$
(\mathrm{R} \cup \mathrm{~L} \cup \mathrm{~A})=\{ \}
$$

apply (clarsimp)
subgoal for j p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}$
apply (drule-tac $x=$ Suc $j$ in spec)
apply auto
done
done
from Cons.hyps [OF dist $^{\prime}$ sharing-consis ${ }^{\prime} \mathrm{k}$-bound ts-k dist-prop $]$
have share-all-until-volatile-write $\left(\operatorname{ts}\left[\mathrm{k}:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is $\left.\left.\left.{ }^{\prime}, \vartheta^{\prime}, \operatorname{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$ (share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\left.\left.\mathrm{sb}_{\mathrm{t}}\right) \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=$ share-all-until-volatile-write ts
(share (takeWhile (Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{t}}\right) \mathcal{S}$ ).

## moreover

from dist-prop [rule-format, of $0 \mathrm{p}_{\mathrm{t}} \mathrm{is}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}}$ ] t Suc
have $(\mathrm{R} \cup \mathrm{L} \cup \mathrm{A}) \cap$ (all-shared (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ) $\cup$ all-unshared (takeWhile (Not o is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{t}}$ ) $\cup$ all-acquired (takeWhile (Not - is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{t}}\right)$ ) $=\{ \}$
apply clarsimp
apply blast
done
from share-augment-release-commute [OF this]
have share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{t}}$ ) $\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}=$
share (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\left.\operatorname{sb}_{t}\right)\left(\mathcal{S} \oplus_{W} R \ominus_{\mathrm{A}} \mathrm{L}\right)$.
ultimately
show ?thesis
by (clarsimp simp add: Suc t)
qed
qed
lemma share-all-until-volatile-write-update-sb:
assumes congr: $\wedge \mathrm{S}$. share (takeWhile (Not o is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb') $\mathrm{S}=$ share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb) S
shows $\wedge \mathcal{S}$ i. $\llbracket \mathrm{i}<$ length ts; ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket$

$$
\Longrightarrow
$$

share-all-until-volatile-write ts $\mathcal{S}=$
share-all-until-volatile-write $\left(\operatorname{ts}\left[i:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is $\left.\left.\left.{ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right)\right]\right) \mathcal{S}$
proof (induct ts)
case Nil
thus ?case by simp
next
case (Cons t ts)
obtain $\mathrm{p}_{\mathrm{t}}$ is $\mathrm{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$ where
$\mathrm{t}: \mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by ( cases t )
show ?case
proof (cases i)
case 0
with t Cons.prems have eqs: $\mathrm{p}_{\mathrm{t}}=\mathrm{p}$ is $=\mathrm{is} \mathcal{O}_{\mathrm{t}}=\mathcal{O} \quad \mathcal{R}_{\mathrm{t}}=\mathcal{R} \quad \vartheta_{\mathrm{t}}=\vartheta \mathrm{sb}_{\mathrm{t}}=\mathrm{sb} \quad \mathcal{D}_{\mathrm{t}}=\mathcal{D}$
by auto
show ?thesis
by (clarsimp simp add: 0 t eqs congr)
next
case (Suc k)
from Cons.prems Suc obtain k -bound: $\mathrm{k}<$ length ts and ts-k: ts!k $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}$,
$\mathcal{O}, \mathcal{R})$
by auto
from Cons.hyps [OF k-bound ts-k ]
show ?thesis
by (clarsimp simp add: t Suc)
qed
qed
lemma share-all-until-volatile-write-append-Ghost ${ }_{\text {sb }}$ ':
assumes out-VWrite ${ }_{\mathbf{s b}}$ : outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb} \neq\{ \}$
assumes i-bound: i < length ts
assumes ts-i: ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
shows share-all-until-volatile-write ts $\mathcal{S}=$
share-all-until-volatile-write
$\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.\right.$, is $\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb} @\left[\right.$ Ghost $_{\mathrm{sb}}$ A L R W], $\left.\left.\left.\mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right)\right]\right) \mathcal{S}$
proof -
from out-VWrite ${ }_{s b}$
have $\wedge$ S. share (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) (sb @ [Ghost ${ }_{\text {sb }}$ A L R W])) $\mathrm{S}=$ share (takeWhile (Not o is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) S
by (simp add: outstanding-vol-write-takeWhile-append)
from share-all-until-volatile-write-update-sb [OF this i-bound ts-i]

```
    show ?thesis
    by simp
qed
```

lemma acquired-append-Prog ${ }_{\text {sb }}$ :
$\bigwedge \mathrm{S}$. (acquired pending-write (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) ( $\mathrm{sb} @\left[\operatorname{Prog}_{s b} \mathrm{p}_{1} \mathrm{p}_{2}\right.$ mis]) $S$ S $=$
(acquired pending-write (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) S)
by (induct sb) (auto split: memref.splits)
lemma outstanding-refs-non-empty-dropWhile:
outstanding-refs P xs $\neq\{ \} \Longrightarrow$ outstanding-refs $\mathrm{P}($ dropWhile (Not $\circ \mathrm{P}) \mathrm{xs}) \neq\{ \}$
apply (induct xs)
apply simp
apply (simp split: if-split-asm)
done
lemma ex-not: Ex Not
by blast
lemma (in computation) concurrent-step-append:
assumes step: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
shows $(\mathrm{xs} @ \mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow\left(\mathrm{xs}^{\mathrm{C}} \mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)$
using step
proof (cases)
case (Program i p is $\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R} \mathrm{p}^{\prime}$ is ${ }^{\prime}$ )
then obtain
i-bound: i < length ts and
ts-i: ts!i $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
prog-step: $\quad \vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{p}}\left(\mathrm{p}^{\prime}\right.$, is $)$ and
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is} @ i \mathrm{~s}^{\prime}, \vartheta\right.\right.$, record $\mathrm{p} \mathrm{p}{ }^{\prime}$ is $\left.\left.{ }^{\prime} \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$
by auto
from i-bound have i-bound': i + length xs $<$ length (xs@ts)
by auto
from ts-i i-bound have ts-i': (xs@ts)!(i + length xs) $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (auto simp add: nth-append)
from concurrent-step. Program [OF i-bound' ts-i' prog-step, of $m \mathcal{S}]$ ts $^{\prime}$ i-bound show ?thesis
by (auto simp add: $\mathrm{ts}^{\prime}$ list-update-append $\mathcal{S}^{\prime} \mathrm{m}$ )
next
case (Memop i p is $\vartheta$ sb $\mathcal{D} \mathcal{O} \mathcal{R}$ is $\vartheta^{\prime} \operatorname{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}$ )
then obtain
i-bound: i < length ts and
ts-i: ts!i $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
memop-step: (is, $\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{m}}\left(\mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$ and
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]$
by auto
from i-bound have i-bound': i + length xs < length (xs@ts)
by auto
from ts-i i-bound have ts-i': (xs@ts)!(i + length xs $)=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (auto simp add: nth-append)
from concurrent-step. Memop [OF i-bound ${ }^{\prime}$ ts-i ${ }^{\prime}$ memop-step] ts ${ }^{\prime}$ i-bound show ?thesis
by (auto simp add: ts' list-update-append)
next
case (StoreBuffer i p is $\left.\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R} \mathrm{sb}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\right)$
then obtain
i-bound: i < length ts and
ts-i: ts!i $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ and
sb-step: $(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$ and
$\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]$
by auto
from i-bound have i-bound': i + length xs $<$ length (xs@ts)
by auto
from ts-i i-bound have ts-i': (xs@ts)!(i + length xs $)=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$ by (auto simp add: nth-append)
from concurrent-step.StoreBuffer [OF i-bound' ts-i' sb-step] ts' i-bound show ?thesis
by (auto simp add: ts' list-update-append)
qed
primrec weak-sharing-consistent:: owns $\Rightarrow$ 'a memref list $\Rightarrow$ bool
where
weak-sharing-consistent $\mathcal{O}[]=$ True
| weak-sharing-consistent $\mathcal{O}(\mathrm{r} \# \mathrm{rs})=$
(case r of
Write $_{\text {sb }}$ volatile -- - A L R W $\Rightarrow$
(if volatile then $\mathrm{L} \subseteq \mathrm{A} \wedge \mathrm{A} \cap \mathrm{R}=\{ \} \wedge \mathrm{R} \subseteq \mathcal{O} \wedge$ weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ rs
else weak-sharing-consistent $\mathcal{O} \mathrm{rs}$ )
$\mid$ Ghost $_{\text {sb }} \mathrm{AL} \mathrm{R} W \Rightarrow \mathrm{~L} \subseteq \mathrm{~A} \wedge \mathrm{~A} \cap \mathrm{R}=\{ \} \wedge \mathrm{R} \subseteq \mathcal{O} \wedge$ weak-sharing-consistent $(\mathcal{O} \cup$ A -R ) rs
|- $\Rightarrow$ weak-sharing-consistent $\mathcal{O}$ rs)
lemma sharing-consistent-weak-sharing-consistent:
$\wedge \mathcal{S} \mathcal{O}$. sharing-consistent $\mathcal{S} \mathcal{O} \mathrm{sb} \Longrightarrow$ weak-sharing-consistent $\mathcal{O} \mathrm{sb}$
apply (induct sb)
apply (auto split: memref.splits)
done
lemma weak-sharing-consistent-append:
$\wedge \mathcal{O}$. weak-sharing-consistent $\mathcal{O}$ (xs @ ys) $=$
(weak-sharing-consistent $\mathcal{O}$ xs $\wedge$ weak-sharing-consistent (acquired True xs $\mathcal{O}$ ) ys)
apply (induct xs)
apply (auto split: memref.splits)
done
lemma read-only-share-unowned: $\wedge \mathcal{O} \mathcal{S}$.
【weak-sharing-consistent $\mathcal{O} \mathrm{sb}$; a $\notin \mathcal{O} \cup$ all-acquired sb; a $\in$ read-only (share sb $\mathcal{S}$ )】
$\Longrightarrow \mathrm{a} \in$ read-only $\mathcal{S}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis proof (cases volatile)
case False
with Cons Write ${ }_{\text {sb }}$ show ?thesis by auto next
case True
from Cons.hyps [where $\mathcal{S}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ and $\mathcal{O}=(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ ] Cons.prems show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True in-read-only-restrict-conv in-read-only-augment-conv
split: if-split-asm)
qed
next
case Read $_{\text {sb }}$ with Cons show ?thesis by auto
next
case $\operatorname{Prog}_{s b}$ with Cons show ?thesis by auto
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
with Cons.hyps [where $\mathcal{S}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ and $\mathcal{O}=(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ ] Cons.prems show
?thesis
by (auto simp add: in-read-only-restrict-conv in-read-only-augment-conv split: if-split-asm)
qed

```
lemma share-read-only-mono-in:
    assumes a-in: a }\in\mathrm{ read-only (share sb }\mathcal{S}\mathrm{ )
    assumes ss: read-only }\mathcal{S}\subseteq\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime
    shows a }\in\mathrm{ read-only (share sb S')
using share-read-only-mono [OF ss] a-in
by auto
```

lemma read-only-unacquired-share:
$\wedge \mathrm{S} \mathcal{O} . \llbracket \mathcal{O} \cap$ read-only $\mathrm{S}=\{ \} ;$ weak-sharing-consistent $\mathcal{O}$ sb; a $\in$ read-only S ;
a $\notin$ all-acquired sb 】
$\Longrightarrow \mathrm{a} \in$ read-only (share sb S)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems
obtain a-ro: a $\in$ read-only S and a-A: a $\notin \mathrm{A}$ and a-unacq: a $\notin$ all-acquired sb and owns-ro: $\mathcal{O} \cap$ read-only $\mathrm{S}=\{ \}$ and
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$
by (auto simp add: in-read-only-convs)
from a-ro a-A owns-ro R-owns L-A have a-ro': a $\in$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-ro' a-unacq]
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
next
case False
with Cons show ?thesis

```
by (clarsimp simp add: Write sb \(_{\text {b }}\) False)
    qed
    next
    case Read sb with Cons show ?thesis by (clarsimp)
next
    case Prog \(_{\text {sb }}\) with Cons show ?thesis by (clarsimp)
next
    case (Ghost \(_{\text {sb }}\) A L R W)
    from Cons.prems
    obtain a-ro: a \(\in\) read-only \(S\) and \(a-A\) : a \(\notin \mathrm{A}\) and a-unacq: a \(\notin\) all-acquired sb and
        owns-ro: \(\mathcal{O} \cap\) read-only \(S=\{ \}\) and
        \(\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}\) and \(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\) and R-owns: \(\mathrm{R} \subseteq \mathcal{O}\) and
        consis': weak-sharing-consistent \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) sb
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
    from owns-ro A-R R-owns have owns-ro': \((\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap\) read-only \(\left(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
\(=\{ \}\)
            by (auto simp add: in-read-only-convs)
    from a-ro a-A owns-ro R-owns L-A have a-ro': a \(\in\) read-only ( \(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) )
        by (auto simp add: in-read-only-convs)
    from Cons.hyps [OF owns-ro' consis' a-ro' a-unacq]
    show ?thesis
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
```

lemma read-only-share-unacquired: $\wedge \mathcal{O} \quad \mathrm{S} . \mathcal{O} \cap$ read-only $\mathrm{S}=\{ \} \Longrightarrow$ weak-sharing-consistent $\mathcal{O}$ sb $\Longrightarrow$
$\mathrm{a} \in$ read-only (share sb S ) $\Longrightarrow \mathrm{a} \notin$ acquired True sb $\mathcal{O}$
proof (induct sb)
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case False
with Cons Write $_{\text {sb }}$ show ?thesis by auto
next
case True
note volatile=this
from Cons.prems
obtain a-ro: $a \in$ read-only (share sb $\left(S \oplus_{W} R \ominus_{A} L\right)$ ) and
owns-ro: $\mathcal{O} \cap$ read-only $\mathrm{S}=\{ \}$ and
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathrm{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-ro]
show ?thesis
by (auto simp add: Write $_{\text {sb }}$ volatile)
qed
next
case $\operatorname{Read}_{\text {sb }}$ with Cons show ?thesis by auto
next
case $\operatorname{Prog}_{s b}$ with Cons show ?thesis by auto
next
case ( Ghost $_{\text {sb }}$ A L R W)
from Cons.prems
obtain a-ro: a $\in$ read-only (share sb $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ ) and owns-ro: $\mathcal{O} \cap$ read-only $\mathrm{S}=\{ \}$ and
L-A: $\mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$
by (clarsimp simp add: Ghostsb)
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-ro]
show ?thesis
by (auto simp add: Ghost ${ }_{\mathbf{s b}}$ )
qed
qed
lemma read-only-share-all-acquired-in:
$\wedge S \mathcal{O} . \llbracket \mathcal{O} \cap$ read-only $\mathrm{S}=\{ \} ;$ weak-sharing-consistent $\mathcal{O}$ sb; a $\in$ read-only (share sb S ) $\rrbracket$
$\Longrightarrow \mathrm{a} \in$ read-only (share sb Map.empty) $\vee$ ( $\mathrm{a} \in$ read-only $\mathrm{S} \wedge \mathrm{a} \notin$ all-acquired sb)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems
obtain a-in: a $\in$ read-only (share sb $\left(S \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ ) and
owns-ro: $\mathcal{O} \cap$ read-only $\mathrm{S}=\{ \}$ and
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Writesb True)
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-in]
have hyp: $\mathrm{a} \in$ read-only (share sb Map.empty) $\vee \mathrm{a} \in$ read-only $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \wedge \mathrm{a}$ $\notin$ all-acquired sb.
have $\mathrm{a} \in$ read-only $\left(\right.$ share sb $\left(\right.$ Map.empty $\left.\left.\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \vee(\mathrm{a} \in$ read-only $\mathrm{S} \wedge \mathrm{a} \notin \mathrm{A}$ $\wedge \mathrm{a} \notin$ all-acquired sb)
proof -
\{
assume a-emp: a $\in$ read-only (share sb Map.empty)
have read-only Map.empty $\subseteq$ read-only (Map.empty $\oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
by (auto simp add: in-read-only-convs)
from share-read-only-mono-in [OF a-emp this]
have $\mathrm{a} \in$ read-only (share sb (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )).
\}
moreover
\{
assume a-ro: a $\in$ read-only $\left(S \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right.$ ) and a-unacq: a $\notin$ all-acquired sb
have ?thesis
proof (cases a $\in$ read-only S)
case True
with a-ro obtain a $\notin \mathrm{A}$
by (auto simp add: in-read-only-convs)
with True a-unacq show ?thesis
by auto
next
case False
with a-ro have a-ro-empty: a $\in$ read-only (Map.empty $\oplus_{W} R \ominus_{A} L$ ) by (auto simp add: in-read-only-convs split: if-split-asm)
have read-only (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ) $\subseteq$ read-only $\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right.$ )
by (auto simp add: in-read-only-convs)
with owns-ro ${ }^{\prime}$
have owns-ro-empty: $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only (Map.empty $\left.\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\{ \}$
by blast
from read-only-unacquired-share [OF owns-ro-empty consis' a-ro-empty a-unacq]
have $\mathrm{a} \in$ read-only (share sb (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )).
thus? ?hesis
by simp

```
    qed
}
moreover note hyp
ultimately show ?thesis by blast
    qed
    then show ?thesis
by (clarsimp simp add: Writesb True)
    next
        case False with Cons show ?thesis
by (auto simp add: Write esb
    qed
next
    case Read}\mp@subsup{\mp@code{sb}}{}{\mathrm{ with Cons show ?thesis by auto}
next
    case Progsb with Cons show ?thesis by auto
next
    case (Ghostsb A L R W)
    from Cons.prems
    obtain a-in: a }\in\mathrm{ read-only (share sb (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}))\mathrm{ ) and
        owns-ro: }\mathcal{O}\cap\mathrm{ read-only S ={} and
        L-A: L\subseteqA and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns: R }\subseteq\mathcal{O}\mathrm{ and
        consis': weak-sharing-consistent (\mathcal{O}\cup\textrm{A}-\textrm{R})\textrm{sb}
        by (clarsimp simp add: Ghostsb)
```

    from owns-ro A-R R-owns have owns-ro': \((\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap\) read-only \(\left(\mathrm{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
    $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-in]
have hyp: $a \in$ read-only (share sb Map.empty) $\vee a \in \operatorname{read}$-only $\left(S \oplus_{W} R \ominus_{A} L\right) \wedge a$
$\notin$ all-acquired sb.
have a $\in$ read-only (share sb (Map.empty $\left.\left.\oplus_{W} R \ominus_{A} L\right)\right) \vee(a \in$ read-only $S \wedge a \notin A$
$\wedge \mathrm{a} \notin$ all-acquired sb)
proof -
\{
assume a-emp: a $\in$ read-only (share sb Map.empty)
have read-only Map.empty $\subseteq$ read-only (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
by (auto simp add: in-read-only-convs)
from share-read-only-mono-in [OF a-emp this]
have $\mathrm{a} \in$ read-only (share sb (Map.empty $\oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )).
\}
moreover
\{
assume a-ro: a $\in$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right.$ ) and a-unacq: a $\notin$ all-acquired sb
have ?thesis
proof (cases a $\in$ read-only S)
case True

```
    with a-ro obtain a & A
        by (auto simp add: in-read-only-convs)
    with True a-unacq show ?thesis
        by auto
next
    case False
    with a-ro have a-ro-empty: a f read-only (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R
        by (auto simp add: in-read-only-convs split: if-split-asm)
    have read-only (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\subseteq\mathrm{ read-only ( }\textrm{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ )
        by (auto simp add: in-read-only-convs)
    with owns-ro'
    have owns-ro-empty: (\mathcal{O \cup A - R ) \cap read-only (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})={}
        by blast
    from read-only-unacquired-share [OF owns-ro-empty consis' a-ro-empty a-unacq]
    have a }\in\mathrm{ read-only (share sb (Map.empty }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mathrm{ ).
    thus ?thesis
        by simp
qed
        }
            moreover note hyp
            ultimately show ?thesis by blast
    qed
    then show ?thesis
        by (clarsimp simp add: Ghostsb)
    qed
qed
```

lemma weak-sharing-consistent-preserves-distinct:
$\wedge \mathcal{O} \mathcal{S}$. weak-sharing-consistent $\mathcal{O}$ sb $\Longrightarrow \mathcal{O} \cap$ read-only $\mathcal{S}=\{ \} \Longrightarrow$ acquired True sb $\mathcal{O} \cap$ read-only (share sb $\mathcal{S}$ ) $=\{ \}$
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
owns-ro: $\mathcal{O} \cap$ read-only $\mathcal{S}=\{ \}$ and L-A: $\mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from owns-ro $\mathrm{A}-\mathrm{R}$ R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}}\right.$ L) $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF consis' owns-ro']
show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True)
next
case False with Cons Write $_{\text {sb }}$ show ?thesis by auto
qed
next
case Read ${ }_{\text {sb }}$ with Cons show ?thesis by auto
next
case $\operatorname{Prog}_{s b}$ with Cons show ?thesis by auto
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
from Cons.prems obtain
owns-ro: $\mathcal{O} \cap$ read-only $\mathcal{S}=\{ \}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
from owns-ro $\mathrm{A}-\mathrm{R}$ R-owns have owns-ro': $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \cap$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ $=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF consis' owns-ro']
show ?thesis
by (auto simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
locale weak-sharing-consis $=$
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes weak-sharing-consis:

```
\ip is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta\textrm{sb}
```



```
    \Longrightarrow
    weak-sharing-consistent }\mathcal{O}\mathrm{ sb
```

sublocale sharing-consis $\subseteq$ weak-sharing-consis
proof
fix i p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb
assume i-bound: $\mathrm{i}<$ length ts
assume ts- i : ts $!\mathrm{i}=(\mathrm{p}$, is, $\vartheta$, $\mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
from sharing-consistent-weak-sharing-consistent [OF sharing-consis [OF i-bound ts-i]]
show weak-sharing-consistent $\mathcal{O}$ sb.
qed

```
lemma weak-sharing-consis-tl: weak-sharing-consis (t#ts) \Longrightarrow weak-sharing-consis ts
apply (unfold weak-sharing-consis-def)
apply force
done
```

lemma read-only-share-all-until-volatile-write-unacquired:
$\wedge \mathcal{S}$. [ownership-distinct ts; read-only-unowned $\mathcal{S}$ ts; weak-sharing-consis ts;
$\forall \mathrm{i}<$ length ts. (let $(-,-,-$, sb,-, $\mathcal{O}, \mathcal{R})=$ ts! $\mathrm{i}^{\text {in }}$
a $\notin$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb));
a $\in$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow \mathrm{a} \in$ read-only (share-all-until-volatile-write ts $\mathcal{S}$ )
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb where
$\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases t )
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
have aargh: $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)=\left(\lambda a . \neg\right.$ is-volatile- Write $_{s b}$ a $)$
by (rule ext) auto
have a-ro: a $\in$ read-only $\mathcal{S}$ by fact
have ro-unowned: read-only-unowned $\mathcal{S}(\mathrm{t} \# \mathrm{ts})$ by fact
then interpret read-only-unowned $\mathcal{S}$ t\#ts .
have consis: weak-sharing-consis ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret weak-sharing-consis $\mathrm{t} \# \mathrm{ts}$.
note consis ${ }^{\prime}=$ weak-sharing-consis-tl [OF consis]
let ?take-sb $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$
let $?$ drop-sb $=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)$
from weak-sharing-consis [of 0] t
have consis-sb: weak-sharing-consistent $\mathcal{O}$ sb
by force
with weak-sharing-consistent-append [of $\mathcal{O}$ ?take-sb ?drop-sb]
have consis-take: weak-sharing-consistent $\mathcal{O}$ ?take-sb
by auto
have ro-unowned': read-only-unowned (share ?take-sb $\mathcal{S}$ ) ts proof

```
fix j
```



```
assume j-bound: j < length ts
assume jth: ts!j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{\textrm{i}},\mp@subsup{\textrm{i}}{\textrm{j}}{},\mp@subsup{\vartheta}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
show }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only (share ?take-sb S S)={}
proof -
    {
        fix a
        assume a-owns: a }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{
        assume a-ro: a }\in\mathrm{ read-only (share ?take-sb S S
        have False
        proof -
            from ownership-distinct [of 0 Suc j] j-bound jth t
            have dist: }(\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
                by fastforce
            from read-only-unowned [of Suc j] j-bound jth
            have dist-ro: }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only }\mathcal{S}={}\mathrm{ by force
            show ?thesis
            proof (cases a }\in(\mathcal{O}\cup\mathrm{ all-acquired sb))
                case True
                with dist a-owns show False by auto
            next
                case False
                hence a }\not\in(\mathcal{O}\cup\mathrm{ all-acquired ?take-sb)
                    using all-acquired-append [of ?take-sb ?drop-sb]
                by auto
                    from read-only-share-unowned [OF consis-take this a-ro]
            have a }\in\mathrm{ read-only }\mathcal{S}\mathrm{ .
            with dist-ro a-owns show False by auto
            qed
        qed
        }
        thus ?thesis by auto
    qed
qed
```

from Cons.prems
obtain unacq-ts: $\forall \mathrm{i}<$ length ts. (let $(-,-,-$, sb, $,-\mathcal{O},-)=$ ts!i in a $\notin$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)) and unacq-sb: a $\notin$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) by (force simp add: t aargh)

```
from read-only-unowned [of 0] t
have owns-ro: \(\mathcal{O} \cap\) read-only \(\mathcal{S}=\{ \}\)
    by force
from read-only-unacquired-share [OF owns-ro consis-take a-ro unacq-sb]
```

```
    have a }\in\mathrm{ read-only (share (takeWhile (Not o is-volatile-Write sb) sb) S S).
    from Cons.hyps [OF dist' ro-unowned' consis' unacq-ts this]
    show ?case
    by (simp add: t)
qed
lemma read-only-share-unowned-in:
【weak-sharing-consistent \(\mathcal{O}\) sb; a \(\in\) read-only (share sb \(\mathcal{S}\) )】
\(\Longrightarrow \mathrm{a} \in \operatorname{read}\)-only \(\mathcal{S} \cup \mathcal{O} \cup\) all-acquired sb
using read-only-share-unowned [of \(\mathcal{O}\) sb]
by auto
```

lemma read-only-shared-all-until-volatile-write-subset:
$\bigwedge \mathcal{S}$. $\llbracket$ ownership-distinct ts; weak-sharing-consis ts $\rrbracket$
read-only (share-all-until-volatile-write ts $\mathcal{S}$ ) $\subseteq$
read-only $\mathcal{S} \cup(\bigcup((\lambda(-,-,-$, sb, $-\mathcal{O},-) . \mathcal{O} \cup$ all-acquired (takeWhile (Not o
is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb)) ' set ts))
proof (induct ts)
case Nil thus ?case by simp
next
case (Cons t ts)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb where
$\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases t)
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
have consis: weak-sharing-consis ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret weak-sharing-consis $\mathrm{t} \# \mathrm{ts}$.
have aargh: (Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)=\left(\lambda a . \neg\right.$ is-volatile-Write $_{s b}$ a) by (rule ext) auto
note consis $^{\prime}=$ weak-sharing-consis-tl [OF consis]
let ? take-sb $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$
let ?drop-sb $=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$
from weak-sharing-consis [of 0] t
have consis-sb: weak-sharing-consistent $\mathcal{O}$ sb by force
with weak-sharing-consistent-append [of $\mathcal{O}$ ?take-sb ?drop-sb]
have consis-take: weak-sharing-consistent $\mathcal{O}$ ?take-sb
by auto

```
    {
    fix a
    assume a-in: a }\in\mathrm{ read-only
                    (share-all-until-volatile-write ts
                    (share ?take-sb S)) and
    a-notin-shared: a & read-only S and
        a-notin-rest: a }\not\in(\cup((\lambda(-,-,-, sb, -, \mathcal{O,-).\mathcal{O}\cup all-acquired (takeWhile (Not \circ
is-volatile-Write sb) sb)) ' set ts))
    have a }\in\mathcal{O}\cup\mathrm{ all-acquired (takeWhile (Not ० is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ ) sb)
    proof -
        from Cons.hyps [OF dist' consis', of (share ?take-sb S}\mathrm{ )] a-in a-notin-rest
        have a }\in\mathrm{ read-only (share ?take-sb }\mathcal{S}\mathrm{ )
            by (auto simp add: aargh)
        from read-only-share-unowned-in [OF consis-take this] a-notin-shared
        show ?thesis by auto
    qed
}
then show ?case
    by (auto simp add: t aargh)
qed
lemma weak-sharing-consistent-preserves-distinct-share-all-until-volatile-write:
    \mathcal{S}}\mathrm{ i. 【ownership-distinct ts; read-only-unowned }\mathcal{S}\mathrm{ ts;weak-sharing-consis ts;
i < length ts; ts!i = (p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\rrbracket
\Longrightarrow ~ a c q u i r e d ~ T r u e ~ ( t a k e W h i l e ~ ( N o t ~ o ~ i s - v o l a t i l e - W r i t e ~ s b ) ~ s b ) ~ \mathcal { O } \cap
    read-only (share-all-until-volatile-write ts S}\mathrm{ ) = {}
proof (induct ts)
    case Nil thus ?case by simp
next
    case (Cons t ts)
    note «read-only-unowned }\mathcal{S}(\textrm{t}#\textrm{ts})\mathrm{ )
    then interpret read-only-unowned }\mathcal{S}\textrm{t}#\textrm{ts}
    note i-bound = i < length (t # ts)>
    note ith = <(t # ts)!i=(p,is,\vartheta, sb,\mathcal{D},\mathcal{O},\mathcal{R})}
    have dist: ownership-distinct (t#ts) by fact
    then interpret ownership-distinct t#ts .
    from ownership-distinct-tl [OF dist]
    have dist': ownership-distinct ts.
    have consis: weak-sharing-consis (t#ts) by fact
    then interpret weak-sharing-consis t#ts .
    note consis' = weak-sharing-consis-tl [OF consis]
    let ?take-sb = (takeWhile (Not \circ is-volatile-Write sb ) sb)
    let ?drop-sb = (dropWhile (Not o is-volatile-Writesb) sb)
```

```
    have aargh: (Not \circ is-volatile-Write sb
        by (rule ext) auto
    show ?case
    proof (cases i)
    case 0
    from read-only-unowned [of 0] ith 0
    have owns-ro: }\mathcal{O}\cap\mathrm{ read-only }\mathcal{S}={
        by force
    from weak-sharing-consis [of 0] ith 0
    have weak-sharing-consistent }\mathcal{O}\mathrm{ sb
        by force
    with weak-sharing-consistent-append [of \mathcal{O ?take-sb ?drop-sb]}
    have consis-take: weak-sharing-consistent \mathcal{O}\mathrm{ ?take-sb}
        by auto
    from weak-sharing-consistent-preserves-distinct [OF this owns-ro]
    have dist-t: acquired True ?take-sb \mathcal{O}\cap read-only (share ?take-sb \mathcal{S}}={{}
    from read-only-shared-all-until-volatile-write-subset [OF dist' consis', of (share ?take-sb
S)]
    have ro-rest: read-only (share-all-until-volatile-write ts (share ?take-sb \mathcal{S}))\subseteq
                read-only (share ?take-sb \mathcal{S})\cup
                (U((\lambda(-,-,-, sb, -, \mathcal{O,-).\mathcal{O}\cup all-acquired (takeWhile (Not o is-volatile-Write sb})
sb))' set ts))
        by auto
    {
        fix a
        assume a-in-sb: a }\in\mathrm{ acquired True ?take-sb }\mathcal{O
        assume a-in-ro: a \in read-only (share-all-until-volatile-write ts (share ?take-sb \mathcal{S})
        have False
        proof -
```

            from Set.in-mono [rule-format, OF ro-rest a-in-ro] dist-t a-in-sb
            have \(a \in(\cup((\lambda(-,-,-\), sb, \(, \mathcal{O},-) . \mathcal{O} \cup\) all-acquired (takeWhile (Not \(\circ\)
    is-volatile-Write ${ }_{\text {sb }}$ ) sb)) ' set ts))
by auto
then obtain $\mathrm{jp}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}}$
where j -bound: j < length ts and ts- j : ts $!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
and a-in-j: $\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ )
by (fastforce simp add: in-set-conv-nth)
from ownership-distinct [of 0 Suc j] ith ts-j j-bound 0
have dist: $(\mathcal{O} \cup$ all-acquired sb$) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
by fastforce
moreover
from acquired-all-acquired [of True ?take-sb $\mathcal{O}$ ] a-in-sb all-acquired-append [of
?take-sb ?drop-sb]
have $\mathrm{a} \in \mathcal{O} \cup$ all-acquired sb
by auto
with a-in-j all-acquired-append [of (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb $_{\mathrm{j}}$ ) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ )]
dist
have False by fastforce
thus ?thesis ..
qed
\}
then show ?thesis
using 0 ith
by (auto simp add: aargh)
next
case (Suc k)
from i-bound Suc have k-bound: $k<$ length ts
by auto
from ith Suc have kth: ts!k $=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by auto
obtain $\mathrm{p}_{\mathrm{t}} \mathrm{is}_{\mathrm{t}} \mathcal{O}_{\mathrm{t}} \mathcal{R}_{\mathrm{t}} \mathcal{D}_{\mathrm{t}} \vartheta_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$
where t : $\mathrm{t}=\left(\mathrm{p}_{\mathrm{t}}, \mathrm{is}_{\mathrm{t}}, \vartheta_{\mathrm{t}}, \mathrm{sb}_{\mathrm{t}}, \mathcal{D}_{\mathrm{t}}, \mathcal{O}_{\mathrm{t}}, \mathcal{R}_{\mathrm{t}}\right)$
by (cases t)
let ?take-sb ${ }_{\mathrm{t}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}_{\mathrm{t}}\right)$
let ?drop-sb $=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{t}}\right)$
have ro-unowned': read-only-unowned (share ?take-sb ${ }_{t} \mathcal{S}$ ) ts
proof
fix j
fix $p_{j}$ is $_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j-bound: $\mathrm{j}<$ length ts
assume $\mathrm{jth}: \operatorname{ts}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\mathcal{O}_{\mathrm{j}} \cap$ read-only (share ?take-sb $\left.{ }_{\mathrm{t}} \mathcal{S}\right)=\{ \}$
proof -
from read-only-unowned [of Suc j] j-bound jth
have dist: $\mathcal{O}_{\mathrm{j}} \cap$ read-only $\mathcal{S}=\{ \}$ by force
from weak-sharing-consis [of 0] t
have weak-sharing-consistent $\mathcal{O}_{\mathrm{t}} \mathrm{sb}_{\mathrm{t}}$
by fastforce
with weak-sharing-consistent-append $\left[\right.$ of $\mathcal{O}_{\mathrm{t}}$ ?take-sb $\mathrm{ta}_{\mathrm{t}}$ ?drop-sb ${ }_{\mathrm{t}}$ ]
have consis-t: weak-sharing-consistent $\mathcal{O}_{\mathrm{t}}$ ?take-sb ${ }_{\mathrm{t}}$
by auto
\{
fix a
assume $a-i n-j: a \in \mathcal{O}_{j}$
assume a-ro: a $\in$ read-only (share ?take-sb ${ }_{\mathrm{t}} \mathcal{S}$ )
have False
proof -
from a-in-j ownership-distinct [of 0 Suc j] j-bound t jth have $\left(\mathcal{O}_{\mathrm{t}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{t}}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$ by fastforce

```
            with a-in-j all-acquired-append [of ?take-s\mp@subsup{b}{t}{} ?drop-s\mp@subsup{b}{t}{}}\mathrm{ ]
            have a }\not\in(\mp@subsup{\mathcal{O}}{\textrm{t}}{}\cup\mathrm{ all-acquired ?take-sb
                by fastforce
            from read-only-share-unowned [OF consis-t this a-ro]
            have a }\in\mathrm{ read-only }\mathcal{S}\mathrm{ .
            with a-in-j dist
            show False by auto
            qed
        }
        then
show ?thesis
    by auto
        qed
    qed
    from Cons.hyps [OF dist' ro-unowned' consis' k-bound kth]
    show ?thesis
        by (simp add: t)
    qed
qed
```

lemma in-read-only-share-all-until-volatile-write:
assumes dist: ownership-distinct ts
assumes consis: sharing-consis $\mathcal{S}$ ts
assumes ro-unowned: read-only-unowned $\mathcal{S}$ ts
assumes i-bound: i < length ts
assumes ts-i: ts $!i=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
assumes a-unacquired-others: $\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}!\mathrm{j}$ in
a $\notin$ all-acquired (takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\mathbf{s b}}\right)$ sb $_{\mathrm{j}}$ ))
assumes a-ro-share: a $\in$ read-only (share sb $\mathcal{S}$ )
shows a $\in$ read-only (share (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
(share-all-until-volatile-write ts $\mathcal{S})$ )
proof -
from consis
interpret sharing-consis $\mathcal{S}$ ts .
interpret read-only-unowned $\mathcal{S}$ ts by fact
from sharing-consis [OF i-bound ts-i]
have consis-sb: sharing-consistent $\mathcal{S} \mathcal{O}$ sb.
from sharing-consistent-weak-sharing-consistent [OF this]
have weak-consis: weak-sharing-consistent $\mathcal{O}$ sb.
from read-only-unowned [OF i-bound ts-i]
have owns-ro: $\mathcal{O} \cap$ read-only $\mathcal{S}=\{ \}$.
from read-only-share-all-acquired-in [OF owns-ro weak-consis a-ro-share]
have $\mathrm{a} \in$ read-only (share sb Map.empty) $\vee \mathrm{a} \in$ read-only $\mathcal{S} \wedge$ a $\notin$ all-acquired sb .
moreover
let ?take-sb $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.{ }_{s b}\right)$ sb)

```
let ?drop-sb = (dropWhile (Not \circ is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{sb}}{})\textrm{sb}
```

from weak-consis weak-sharing-consistent-append [of $\mathcal{O}$ ?take-sb ?drop-sb]
obtain weak-consis': weak-sharing-consistent (acquired True ?take-sb $\mathcal{O}$ ) ?drop-sb and weak-consis-take: weak-sharing-consistent $\mathcal{O}$ ?take-sb
by auto

## \{

assume a $\in$ read-only (share sb Map.empty)
with share-append [of ?take-sb ?drop-sb]
have a-in': a $\in$ read-only (share ?drop-sb (share ?take-sb Map.empty))
by auto
have owns-empty: $\mathcal{O} \cap$ read-only Map.empty $=\{ \}$
by auto
from weak-sharing-consistent-preserves-distinct [OF weak-consis-take owns-empty]
have acquired True ?take-sb $\mathcal{O} \cap$ read-only (share ?take-sb Map.empty) $=\{ \}$.
from read-only-share-all-acquired-in [OF this weak-consis' a -in']
have a $\in$ read-only (share ?drop-sb Map.empty) $\vee \mathrm{a} \in$ read-only (share ?take-sb
Map.empty) $\wedge$ a $\notin$ all-acquired ?drop-sb.

## moreover

\{
assume a-ro-drop: a $\in$ read-only (share ?drop-sb Map.empty)
have read-only Map.empty $\subseteq$ read-only (share-all-until-volatile-write ts $\mathcal{S}$ )
by auto
from share-read-only-mono-in [OF a-ro-drop this]
have ?thesis .
\}
moreover
\{
assume a-ro-take: a $\in$ read-only (share ?take-sb Map.empty)
assume a-unacq-drop: a $\notin$ all-acquired ?drop-sb
from read-only-share-unowned-in [OF weak-consis-take a-ro-take]
have $\mathrm{a} \in \mathcal{O} \cup$ all-acquired ?take-sb by auto
hence a $\in \mathcal{O} \cup$ all-acquired sb using all-acquired-append [of ?take-sb ?drop-sb]
by auto
from share-all-until-volatile-write-thread-local' ${ }^{\prime}$ [OF dist consis i-bound ts-i this] a-ro-share
have ?thesis by (auto simp add: read-only-def)
\}
ultimately have ?thesis by blast
\}

## moreover

## \{

assume a-ro: a $\in$ read-only $\mathcal{S}$
assume a-unacq: a $\notin$ all-acquired sb

```
    with all-acquired-append [of ?take-sb ?drop-sb]
    obtain a & all-acquired ?take-sb and a-notin-drop: a & all-acquired ?drop-sb
        by auto
    with a-unacquired-others i-bound ts-i
    have a-unacq: }\forall\textrm{j}<l\mathrm{ length ts.
            (let (-,-,-,sb⿱丶万⿱⿰㇒一丶⿴⿱冂一⿰丨丨丁心,-,-,-) = ts!j in
            a & all-acquired (takeWhile (Not o is-volatile-Write sb) sb j))
        by (auto simp add: Let-def)
    from local.weak-sharing-consis-axioms have weak-sharing-consis ts .
    from read-only-share-all-until-volatile-write-unacquired [OF dist ro-unowned
        «weak-sharing-consis ts` a-unacq a-ro]
    have a-ro-all: a }\in\mathrm{ read-only (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ).
    from weak-consis weak-sharing-consistent-append [of \mathcal{O ?take-sb ?drop-sb]}
    have weak-consis-drop: weak-sharing-consistent (acquired True ?take-sb \mathcal{O}) ?drop-sb
        by auto
    from weak-sharing-consistent-preserves-distinct-share-all-until-volatile-write [OF dist
        ro-unowned «weak-sharing-consis ts» i-bound ts-i]
    have acquired True ?take-sb }\mathcal{O}
        read-only (share-all-until-volatile-write ts }\mathcal{S})={}
    from read-only-unacquired-share [OF this weak-consis-drop a-ro-all a-notin-drop]
    have ?thesis .
}
ultimately show ?thesis by blast
qed
lemma all－acquired－dropWhile－in： \(\mathrm{x} \in\) all－acquired（dropWhile P sb\() \Longrightarrow \mathrm{x} \in\) all－acquired sb
using all－acquired－append［of takeWhile P sb dropWhile P sb］
by auto
```

lemma all－acquired－takeWhile－in： $\mathrm{x} \in$ all－acquired（takeWhile $\mathrm{P} s \mathrm{sb}) \Longrightarrow \mathrm{x} \in$ all－acquired sb
using all－acquired－append［of takeWhile P sb dropWhile P sb］
by auto
lemmas all－acquired－takeWhile－dropWhile－in $=$ all－acquired－takeWhile－in all－acquired－dropWhile－in
lemma split－in－read－only－reads：
$\wedge \mathcal{O}$. a $\in$ read－only－reads $\mathcal{O}$ xs $\Longrightarrow$
$\left(\exists \mathrm{t}\right.$ v ys zs．xs $=\mathrm{ys} @ \operatorname{Read}_{\mathrm{sb}}$ False atvazs $\wedge$ a $\notin$ acquired True ys $\left.\mathcal{O}\right)$
proof（induct xs）
case Nil thus ？case by simp
next
case (Cons x xs)
have a-in: a $\in$ read-only-reads $\mathcal{O}(\mathrm{x} \# \mathrm{xs})$ by fact
show ?case
proof (cases x)
case $\left(W_{r i t e}^{s b}\right.$ volatile a'sop v A L R W)
show ?thesis
proof (cases volatile)
case False
from a-in have a $\in$ read-only-reads $\mathcal{O}$ xs
by (auto simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF this] obtain $t$ v ys zs where
xs: xs=ys@Read ${ }_{\text {sb }}$ False a t v \# zs and a-notin: a $\notin$ acquired True ys $\mathcal{O}$
by auto
with xs a-notin obtain $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Read $_{\mathrm{sb}}$ False a $\mathrm{t} v \#$ zs a $\notin$ acquired True
(x\#ys) $\mathcal{O}$
by (simp add: Write ${ }_{\text {sb }}$ False)
then show ?thesis
by blast
next
case True
from a-in have $a \in$ read-only-reads $(\mathcal{O} \cup A-R)$ xs
by (auto simp add: Write ${ }_{\text {sb }}$ True)
from Cons.hyps [OF this] obtain $t$ v ys zs where
xs: xs $=$ ys@Read ${ }_{\text {sb }}$ False a t v $\#$ zs and a-notin: a $\notin$ acquired True ys $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$
by auto
with xs a-notin obtain $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @$ Read $_{\text {sb }}$ False a t v \# zs a $\notin$ acquired True
(x\#ys) $\mathcal{O}$
by ( $\operatorname{simp}$ add: Write ${ }_{\text {sb }}$ True)
then show ?thesis
by blast
qed
next
case $\left(\operatorname{Read}_{\text {sb }}\right.$ volatile $\left.\mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v}^{\prime}\right)$
show ?thesis
proof (cases $\neg$ volatile $\left.\wedge \mathrm{a} \notin \mathcal{O} \wedge \mathrm{a}^{\prime}=\mathrm{a}\right)$
case True
with Read $_{\text {sb }}$ show ?thesis
by fastforce
next
case False
with a-in have a $\in$ read-only-reads $\mathcal{O}$ xs
by (auto simp add: Read sb split: if-split-asm)
from Cons.hyps [OF this] obtain $t \mathrm{v}$ ys zs where
xs: xs=ys@Read ${ }_{\text {sb }}$ False a t v \# zs and a-notin: a $\notin$ acquired True ys $\mathcal{O}$
by auto
with xs a-notin obtain $\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \operatorname{Read}_{\mathrm{sb}}$ False a t v \# zs a $\notin$ acquired True
(x\#ys) $\mathcal{O}$
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
then show ?thesis

```
by blast
    qed
    next
    case Progsb \(_{\text {s }}\)
    with a-in have a \(\in\) read-only-reads \(\mathcal{O}\) xs
        by (auto)
    from Cons.hyps [OF this] obtain \(t \mathrm{v}\) ys zs where
            xs: xs=ys@Read \({ }_{\text {sb }}\) False a t v \(\#\) zs and a-notin: a \(\notin\) acquired True ys \(\mathcal{O}\)
            by auto
            with xs a-notin obtain \(\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \operatorname{Read}_{\mathrm{sb}}\) False a t v \(\#\) zs a \(\notin\) acquired True
(x\#ys) \(\mathcal{O}\)
        by (simp add: \(\operatorname{Prog}_{\text {sb }}\) )
        then show ?thesis
            by blast
next
    case (Ghost \(_{\text {sb }}\) A L R W)
    with a-in have \(a \in \operatorname{read-only-reads}(\mathcal{O} \cup \mathrm{~A}-\mathrm{R})\) xs
                by (auto)
    from Cons.hyps [OF this] obtain \(t \mathrm{v}\) ys zs where
            xs: xs=ys@Read \({ }_{\text {sb }}\) False a t v \# zs and a-notin: a \(\notin\) acquired True ys \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\)
            by auto
            with xs a-notin obtain \(\mathrm{x} \# \mathrm{xs}=(\mathrm{x} \# \mathrm{ys}) @ \operatorname{Read}_{\mathrm{sb}}\) False a \(\mathrm{t} v \# \mathrm{zs}\) a \(\notin\) acquired True
(x\#ys) \(\mathcal{O}\)
                by ( \(\operatorname{simp}\) add: Ghost \({ }_{\text {sb }}\) )
            then show ?thesis
                by blast
    qed
qed
```

lemma insert-monoD: $\mathrm{W} \subseteq \mathrm{W}^{\prime} \Longrightarrow$ insert a $\mathrm{W} \subseteq$ insert a $\mathrm{W}^{\prime}$
by blast
primrec unforwarded-non-volatile-reads:: 'a memref list $\Rightarrow$ addr set $\Rightarrow$ addr set
where
unforwarded-non-volatile-reads [] $\mathrm{W}=\{ \}$
| unforwarded-non-volatile-reads (x\#xs) W =
(case x of
Read $_{\text {sb }}$ volatile a - $\Rightarrow$ (if a $\notin \mathrm{W} \wedge \neg$ volatile
then insert a (unforwarded-non-volatile-reads xs W)
else (unforwarded-non-volatile-reads xs W))
| Write ${ }_{\text {sb }}$ - a---- $\Rightarrow$ unforwarded-non-volatile-reads xs (insert a W)
|- $\Rightarrow$ unforwarded-non-volatile-reads xs W)
lemma unforwarded-non-volatile-reads-non-volatile-Read ${ }_{\mathrm{sb}}$ :
$\bigwedge \mathrm{W}$. unforwarded-non-volatile-reads sb $\mathrm{W} \subseteq$ outstanding-refs is-non-volatile-Read $_{\text {sb }}$ sb apply (induct sb)
apply (auto split: memref.splits if-split-asm)
done
lemma in-unforwarded-non-volatile-reads-non-volatile-Read ${ }_{\mathrm{sb}}$ :
$\mathrm{a} \in$ unforwarded-non-volatile-reads $\mathrm{sb} \mathrm{W} \Longrightarrow \mathrm{a} \in$ outstanding-refs is-non-volatile-Read ${ }_{\mathrm{sb}}$ sb
using unforwarded-non-volatile-reads-non-volatile-Read ${ }_{s b}$ by blast
lemma unforwarded-non-volatile-reads-antimono:
$\wedge \mathrm{W} \mathrm{W}^{\prime} . \mathrm{W} \subseteq \mathrm{W}^{\prime} \Longrightarrow$ unforwarded-non-volatile-reads $\mathrm{xs} \mathrm{W}^{\prime} \subseteq$ unfor-warded-non-volatile-reads xs W
apply (induct xs)
apply (auto split: memref.splits dest: insert-monoD)
done
lemma unforwarded-non-volatile-reads-antimono-in:
$\mathrm{x} \in$ unforwarded-non-volatile-reads xs $\mathrm{W}^{\prime} \Longrightarrow \mathrm{W} \subseteq \mathrm{W}^{\prime}$
$\Longrightarrow \mathrm{x} \in$ unforwarded-non-volatile-reads xs W
using unforwarded-non-volatile-reads-antimono
by blast
lemma unforwarded-non-volatile-reads-append: $\wedge \mathrm{W}$. unforwarded-non-volatile-reads (xs@ys) W =
(unforwarded-non-volatile-reads xs W $\cup$
unforwarded-non-volatile-reads ys ( $\mathrm{W} \cup$ outstanding-refs is-Write ${ }_{\text {sb }} \mathrm{xs}$ ))
apply (induct xs)
apply clarsimp
apply (auto split: memref.splits)
done
lemma reads-consistent-mem-eq-on-unforwarded-non-volatile-reads:
assumes mem-eq: $\forall \mathrm{a} \in \mathrm{A} \cup \mathrm{W} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}$
assumes subset: unforwarded-non-volatile-reads sb $\mathrm{W} \subseteq \mathrm{A}$
assumes consis-m: reads-consistent pending-write $\mathcal{O} \mathrm{m}$ sb
shows reads-consistent pending-write $\mathcal{O} \mathrm{m}^{\prime} \mathrm{sb}$
using mem-eq subset consis-m
proof (induct sb arbitrary: A $\mathrm{W} \mathrm{m}^{\prime} \mathrm{m}$ pending-write $\mathcal{O}$ )
case Nil thus ?case by simp
next
case (Cons r sb)
note mem-eq $=\left\langle\forall \mathrm{a} \in \mathrm{A} \cup \mathrm{W} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}\right\rangle$
note subset $=\langle$ unforwarded-non-volatile-reads $(\mathrm{r} \# \mathrm{sb}) \mathrm{W} \subseteq \mathrm{A}$,
note consis-m $=\langle r e a d s-c o n s i s t e n t ~ p e n d i n g-w r i t e ~ \mathcal{O ~ m ~ ( r \# s b ) ~ 〉 ~}$
show ?case
proof (cases r)
case ( Write $_{\text {sb }}$ volatile a sop v A $\mathrm{A}^{\prime} \mathrm{R} \mathrm{W}^{\prime}$ )
from subset obtain
subset': unforwarded-non-volatile-reads sb (insert a W) $\subseteq$ A
by (auto simp add: Write ${ }_{\text {sb }}$ )
from mem-eq
have mem-eq ${ }^{\prime}$ :
$\forall \mathrm{a}^{\prime} \in(\mathrm{A} \cup($ insert $\mathrm{a} W)) .\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{a}^{\prime}=(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{a}^{\prime}$
by (auto)
show ?thesis
proof (cases volatile)
case True
from consis-m obtain
consis': reads-consistent $\operatorname{True}\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v}))$ sb and no-volatile-Read ${ }_{\mathrm{sb}}$ : outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{sb}=\{ \}$
by (simp add: Write sb True)
from Cons.hyps [OF mem-eq' subset ${ }^{\prime}$ consis']
have reads-consistent $\operatorname{True}\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{sb}$.
with no-volatile-Read ${ }_{\text {sb }}$
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ True)
next
case False
from consis-m obtain consis': reads-consistent pending-write $\mathcal{O}(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{sb}$
by (simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF mem-eq' subset ' consis']
have reads-consistent pending-write $\mathcal{O}\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{sb}$.
then
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case $\left(\operatorname{Read}_{\text {sb }}\right.$ volatile at v$)$
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in \mathrm{A} \cup W . \mathrm{m}^{\prime} \mathrm{a}^{\prime}=\mathrm{m} \mathrm{a}^{\prime}$
by (auto)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from consis-m obtain
consis': reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb}$
by (simp add: Read ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in W$ )
case False
from subset obtain
subset': unforwarded-non-volatile-reads sb $\mathrm{W} \subseteq \mathrm{A}$
using False
by (auto simp add: Read ${ }_{\text {sb }}$ True split: if-split-asm)
from Cons.hyps [OF mem-eq' subset ${ }^{\prime}$ consis ${ }^{\prime}$ ]

```
show ?thesis
    by (simp add: Readsb True)
        next
case True
from subset have
    subset': unforwarded-non-volatile-reads sb W \subseteq
            insert a A
    using True
    apply (auto simp add: Read sb volatile split: if-split-asm)
    done
from mem-eq True have mem-eq': }\forall\mp@subsup{\textrm{a}}{}{\prime}\in(\mathrm{ insert a A) }\cup\textrm{W}.\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\textrm{a}}{}{\prime}=m\mp@subsup{\textrm{m}}{}{\prime
    by auto
from Cons.hyps [OF mem-eq' subset' consis']
show ?thesis
    by (simp add: Read ds volatile)
        qed
    next
        case False
        note non-vol = this
        from consis-m obtain
consis': reads-consistent pending-write }\mathcal{O}\textrm{m sb}\mathrm{ and
v: (pending-write \vee a }\in\mathcal{O})\longrightarrow\textrm{v}=\textrm{m
by (simp add: Readsb False)
    show ?thesis
    proof (cases a }\in\textrm{W}\mathrm{ )
case True
from mem-eq subset Readsb True non-vol have m' a = m a
    by (auto simp add: False)
from subset obtain
    subset': unforwarded-non-volatile-reads sb W \subseteq insert a A
    using False
    by (auto simp add: Read sb non-vol split: if-split-asm)
from mem-eq True have mem-eq': }\forall\mp@subsup{\textrm{a}}{}{\prime}\in(\mathrm{ insert a A) }\cup\textrm{W}.\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\textrm{a}}{}{\prime}=m\mp@subsup{\textrm{m}}{}{\prime
    by auto
with Cons.hyps [OF mem-eq' subset' consis'] v
show ?thesis
    by (simp add: Readsb non-vol)
        next
case False
from mem-eq subset Readsb}\mathrm{ False non-vol have meq: m'a = m a
    by (clarsimp)
from subset obtain
    subset': unforwarded-non-volatile-reads sb W \subseteq A
    using non-vol False
    by (auto simp add: Read}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ non-vol split: if-split-asm)
from mem-eq non-vol have mem-eq': }\forall\mp@subsup{\textrm{a}}{}{\prime}\in\textrm{A}\cup\textrm{W}.\mp@subsup{\textrm{m}}{}{\prime}\mp@subsup{\textrm{a}}{}{\prime}=\textrm{m}\mp@subsup{\textrm{a}}{}{\prime
    by auto
with Cons.hyps [OF mem-eq' subset' consis'] v meq
show ?thesis
```

```
    by (simp add: Readsb non-vol False)
        qed
    qed
next
    case Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ with Cons show ?thesis by auto
next
    case Ghost sb with Cons show ?thesis by auto
qed
qed
```

lemma reads-consistent-mem-eq-on-unforwarded-non-volatile-reads-drop:
assumes mem-eq: $\forall \mathrm{a} \in \mathrm{A} \cup \mathrm{W} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}$
assumes subset: unforwarded-non-volatile-reads (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ )
sb) $\mathrm{W} \subseteq \mathrm{A}$
assumes subset-acq: acquired-reads True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O}$ $\subseteq \mathrm{A}$
assumes consis-m: reads-consistent False $\mathcal{O} \mathrm{m}$ sb
shows reads-consistent False $\mathcal{O}$ m' sb
using mem-eq subset subset-acq consis-m
proof (induct sb arbitrary: $\mathrm{A} \mathrm{Wm}^{\prime} \mathrm{m} \mathcal{O}$ )
case Nil thus ?case by simp
next
case (Cons r sb)
note mem-eq $=\left\langle\forall \mathrm{a} \in \mathrm{A} \cup \mathrm{W} . \mathrm{m}^{\prime} \mathrm{a}=\mathrm{m} \mathrm{a}\right\rangle$
note subset $=$ <unforwarded-non-volatile-reads
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) ( $\mathrm{r} \# \mathrm{sb}$ )) $\mathrm{W} \subseteq \mathrm{A}$,
note subset-acq $=\left\langle\right.$ acquired-reads True (takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)(\mathrm{r} \# \mathrm{sb})$ )
$\mathcal{O} \subseteq \mathrm{A}$,
note consis-m $=$ 〔reads-consistent False $\mathcal{O} \mathrm{m}(\mathrm{r} \# \mathrm{sb})$ 〉
show ?case
proof (cases r)
case ( Write $_{\text {sb }}$ volatile a sop v A' L R W')
show ?thesis
proof (cases volatile)
case True
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in(\mathrm{A} \cup($ insert $\mathrm{a} W)) .\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{a}^{\prime}=(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{a}^{\prime}$
by (auto)

## from consis-m obtain

consis': reads-consistent True $\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{sb}$ and no-volatile-Read ${ }_{\text {sb }}$ : outstanding-refs is-volatile-Read ${ }_{\text {sb }} \mathrm{sb}=\{ \}$
by (simp add: Write sb True)
from subset obtain unforwarded-non-volatile-reads sb (insert a W) $\subseteq$ A
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads [OF mem-eq' this consis']
have reads-consistent $\operatorname{True}\left(\mathcal{O} \cup \mathrm{A}^{\prime}-\mathrm{R}\right)\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{sb}$.
with no-volatile-Read ${ }_{\text {sb }}$
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ True)
next
case False
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in(\mathrm{A} \cup \mathrm{W}) .\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{a}^{\prime}=(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{a}^{\prime}$
by (auto)
from subset obtain
subset $^{\prime}:$ unforwarded-non-volatile-reads (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathrm{W} \subseteq$ A
by (auto simp add: Write sb $_{\text {sb }}$ False)
from subset-acq have
subset-acq': acquired-reads True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O} \subseteq \mathrm{A}$
by (auto simp add: Write sb $^{\text {F False) }}$
from consis-m obtain consis': reads-consistent False $\mathcal{O}(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{sb}$ by (simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF mem-eq' subset' subset-acq' consis']
have reads-consistent False $\mathcal{O}\left(\mathrm{m}^{\prime}(\mathrm{a}:=\mathrm{v})\right) \mathrm{sb}$.
then
show ?thesis
by (simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case $\left(\operatorname{Read}_{s b}\right.$ volatile a $\left.t \mathrm{v}\right)$
from mem-eq
have mem-eq':
$\forall \mathrm{a}^{\prime} \in \mathrm{A} \cup W . \mathrm{m}^{\prime} \mathrm{a}^{\prime}=\mathrm{m} \mathrm{a}^{\prime}$
by (auto)
from subset obtain
subset $^{\prime}$ : unforwarded-non-volatile-reads (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) W $\subseteq \mathrm{A}$
by (clarsimp simp add: Read $_{\text {sb }}$ )
from subset-acq obtain
$\mathrm{a}-\mathrm{A}: \neg$ volatile $\longrightarrow \mathrm{a} \in \mathcal{O} \longrightarrow \mathrm{a} \in \mathrm{A}$ and
subset-acq': acquired-reads True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O} \subseteq \mathrm{A}$
by (auto simp add: Read sb split: if-split-asm)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from consis-m obtain
consis': reads-consistent False $\mathcal{O}$ m sb
by (simp add: Read ${ }_{\text {sb }}$ True)

```
    from Cons.hyps [OF mem-eq' subset' subset-acq' consis']
    show ?thesis
by (simp add: Readsb True)
    next
        case False
        note non-vol = this
        from consis-m obtain
consis': reads-consistent False }\mathcal{O m sb and
v: a }\in\mathcal{O}\longrightarrow\textrm{v}=\textrm{m
by (simp add: Read
    from mem-eq a-A v have v': a }\in\mathcal{O}\longrightarrowv=\mp@subsup{m}{}{\prime}\textrm{a
by (auto simp add: non-vol)
    from Cons.hyps [OF mem-eq' subset' subset-acq' consis ] v'
    show ?thesis
by (simp add: Readsb False)
    qed
    next
    case Prog
    next
        case Ghostsb with Cons show ?thesis by auto
    qed
qed
```

lemma read-only-read-witness: $\wedge \mathcal{S} \mathcal{O}$.
$\llbracket$ non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb;
$\mathrm{a} \in$ read-only-reads $\mathcal{O} \mathrm{sb} \rrbracket$
$\Longrightarrow$
$\exists \mathrm{xs}$ ys t v. $\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False attv ys $\wedge \mathrm{a} \in \operatorname{read}-\mathrm{only}($ share $\mathrm{xs} \mathcal{S}) \wedge$ a $\notin$
read-only-reads $\mathcal{O}$ xs
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
a-ro: a $\in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
nvo': non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from Cons.hyps [OF nvo' a-ro]
obtain xs ys $\mathrm{t} v$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False atv$\#$ ys $\wedge \mathrm{a} \in \operatorname{read}$-only $\left(\operatorname{share} \mathrm{xs}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \wedge$ a $\notin \operatorname{read}-$ only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{xs}$
by blast
thus ?thesis
apply -
apply (rule-tac $x=(x \# x s)$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
done
next
case False
from Cons.prems obtain
a-ro: a $\in$ read-only-reads $\mathcal{O}$ sb and
nvo': non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb
by (clarsimp simp add: Writesb False)
from Cons.hyps [OF nvo' a-ro]
obtain xs ys $\mathrm{t} v$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False atv$\#$ ys $\wedge \mathrm{a} \in \operatorname{read}$-only (share xs $\left.\mathcal{S}\right) \wedge$ a $\notin$ read-only-reads $\mathcal{O}$ xs
by blast
thus ?thesis
apply -
apply (rule-tac $x=(x \# x s)$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
done
qed
next
case $\left(\operatorname{Read}_{\mathrm{sb}}\right.$ volatile $\left.\mathrm{a}^{\prime} \mathrm{t} \mathrm{v}\right)$
show ?thesis
proof (cases $\mathrm{a}^{\prime}=\mathrm{a} \wedge \mathrm{a} \notin \mathcal{O} \wedge \neg$ volatile)
case True
with Cons.prems have a $\in$ read-only $\mathcal{S}$
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
with True show ?thesis
apply -
apply (rule-tac $x=[]$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{sb}$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{t}$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
done
next
case False
with Cons.prems obtain
a-ro: a $\in$ read-only-reads $\mathcal{O}$ sb and
nvo': non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ split: if-split-asm)
from Cons.hyps [OF nvo' a-ro]
obtain xs ys t' $\mathrm{v}^{\prime}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False $\mathrm{at}^{\prime} \mathrm{v}^{\prime} \#$ ys $\wedge \mathrm{a} \in$ read-only (share xs $\mathcal{S}$ ) $\wedge \mathrm{a} \notin$ read-only-reads $\mathcal{O}$ xs
by blast
with False show ?thesis
apply -
apply (rule-tac $\mathrm{x}=(\mathrm{x} \# \mathrm{xs})$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t^{\prime}$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}^{\prime}$ in exI)
apply (clarsimp simp add: $\operatorname{Read}_{\text {sb }}$ )
done
qed
next
case Prog $_{\text {sb }}$
from Cons.prems obtain
a-ro: a $\in$ read-only-reads $\mathcal{O}$ sb and
nvo': non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb
by (clarsimp simp add: Prog $_{\text {sb }}$ )
from Cons.hyps [OF nvo' a-ro]
obtain xs ys $t \mathrm{v}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False atv$\#$ ys $\wedge \mathrm{a} \in$ read-only (share xs $\left.\mathcal{S}\right) \wedge \mathrm{a} \notin$ read-only-reads $\mathcal{O}$ xs
by blast
thus ?thesis
apply -
apply (rule-tac $\mathrm{x}=(\mathrm{x} \# \mathrm{xs})$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: Prog $_{\text {sb }}$ )
done
next
case ( Ghost $_{\text {sb }}$ A L R W)
from Cons.prems obtain
a-ro: a $\in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
nvo' $^{\prime}$ non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$

```
            by (clarsimp simp add: Ghost }\mp@subsup{\mathrm{ sb }}{\mathrm{ b }}{
    from Cons.hyps [OF nvo' a-ro]
    obtain xs ys t v where
    sb = xs @ Read
read-only-reads (\mathcal{O}\cup\textrm{A}-\textrm{R})\textrm{xs}
            by blast
    thus ?thesis
        apply -
        apply (rule-tac x=(x#xs) in exI)
        apply (rule-tac x=ys in exI)
        apply (rule-tac x=t in exI)
        apply (rule-tac x=v in exI)
        apply (clarsimp simp add: Ghost sb)
        done
    qed
qed
lemma read-only-read-acquired-witness: \(\bigwedge \mathcal{S} \mathcal{O}\).
【non-volatile-owned-or-read-only \(\operatorname{True} \mathcal{S} \mathcal{O}\) sb; sharing-consistent \(\mathcal{S} \mathcal{O}\) sb;
a \(\notin\) read-only \(\mathcal{S} ; \mathrm{a} \notin \mathcal{O} ;\) a \(\in\) read-only-reads \(\mathcal{O} \mathrm{sb} \rrbracket\)
\(\exists \mathrm{xs}\) ys \(\mathrm{t} v . \mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}\) False atv\(\#\) ys \(\wedge \mathrm{a} \in\) all-acquired \(\mathrm{xs} \wedge \mathrm{a} \in\) read-only (share xs \(\mathcal{S}) \wedge\)
a \(\notin\) read-only-reads \(\mathcal{O}\) xs
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write \({ }_{\text {sb }}\) volatile \(\mathrm{a}^{\prime}\) sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this from Cons.prems obtain
nvo': non-volatile-owned-or-read-only \(\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) sb and
a-nro: a \(\notin\) read-only \(\mathcal{S}\) and
a-unowned: a \(\notin \mathcal{O}\) and
a-ro': a \(\in\) read-only-reads \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) sb and
A-shared-owns: \(\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}\) and \(\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}\) and \(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\) and
R-owns: \(\mathrm{R} \subseteq \mathcal{O}\) and
consis': sharing-consistent \(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}\)
by (clarsimp simp add: Write \({ }_{\text {sb }}\) True)
from R-owns a-unowned
have \(a-R\) : \(a \notin R\)
```

by auto
show ?thesis
proof (cases a $\in A$ )
case True
from read-only-read-witness [OF nvo' a-ro']
obtain xs ys $t v^{\prime}$ where
$\mathrm{sb}: \mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False at v${ }^{\prime} \#$ ys and
ro: a $\in$ read-only (share xs $\left.\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$ and
a-ro-xs: a $\notin$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ xs
by blast
with True show ?thesis
apply -
apply (rule-tac $x=x \# x s$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
done
next
case False
with a-unowned R-owns a-nro L-A A-R
obtain a-nro': a $\notin$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ and a-unowned': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R}$ by (force simp add: in-read-only-convs)
from Cons.hyps [OF nvo' consis' a-nro' a-unowned' a-ro']
obtain xs ys $t v^{\prime}$ where $s b=x s @ \operatorname{Read}_{\text {sb }}$ False a $t v^{\prime} \#$ ys $\wedge$
$\mathrm{a} \in$ all-acquired xs $\wedge \mathrm{a} \in$ read-only (share xs $\left.\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \wedge$
a $\notin$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ xs
by blast
then show ?thesis
apply -
apply (rule-tac $x=x \# x s$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
done
qed
next
case False
from Cons.prems obtain
consis': sharing-consistent $\mathcal{S} \mathcal{O}$ sb and
a-nro': a $\notin$ read-only $\mathcal{S}$ and
a-unowned: a $\notin \mathcal{O}$ and
a-ro': a $\in$ read-only-reads $\mathcal{O}$ sb and
$a^{\prime} \in \mathcal{O}$ and
nvo': non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro']
obtain xs ys $\mathrm{t} \mathrm{v}^{\prime}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False a t $\mathrm{v}^{\prime} \#$ ys $\wedge$
$\mathrm{a} \in$ all-acquired xs $\wedge \mathrm{a} \in$ read-only (share xs $\mathcal{S}) \wedge \mathrm{a} \notin$ read-only-reads $\mathcal{O}$ xs by blast
then show ? thesis
apply -
apply (rule-tac $x=x \# x$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
done
qed
next
case $\left(\right.$ Read $_{\text {sb }}$ volatile $\left.\mathrm{a}^{\prime} \mathrm{t} v\right)$
from Cons.prems
obtain
consis': sharing-consistent $\mathcal{S} \mathcal{O}$ sb and
a-nro': a $\notin$ read-only $\mathcal{S}$ and
a-unowned: $\mathrm{a} \notin \mathcal{O}$ and
a-ro': a $\in$ read-only-reads $\mathcal{O}$ sb and
nvo': non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb
by (auto simp add: Read sb $^{\text {split: if-split-asm) }}$
from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro']
obtain xs ys $\mathrm{t} \mathrm{v}^{\prime}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False a $\mathrm{t} \mathrm{v}^{\prime} \#$ ys $\wedge$
$\mathrm{a} \in$ all-acquired xs $\wedge \mathrm{a} \in$ read-only (share xs $\mathcal{S}) \wedge \mathrm{a} \notin$ read-only-reads $\mathcal{O}$ xs
by blast
with Cons.prems show ?thesis
apply -
apply (rule-tac $x=x \# x$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v^{\prime}$ in exI)
apply (clarsimp simp add: Read ${ }_{\text {sb }}$ )
done
next
case $\operatorname{Prog}_{s b}$
from Cons.prems
obtain
consis': sharing-consistent $\mathcal{S} \mathcal{O}$ sb and
a-nro': a $\notin$ read-only $\mathcal{S}$ and
a-unowned: a $\notin \mathcal{O}$ and
a-ro': a $\in$ read-only-reads $\mathcal{O}$ sb and
nvo': non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb

```
    by (auto simp add: Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ )
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro']
    obtain xs ys t v where
    sb = xs @ Readsb False a t v # ys ^
    a}\in\mathrm{ all-acquired xs }\wedge\textrm{a}\in\mathrm{ read-only (share xs }\mathcal{S})\wedge\textrm{a}\not\in\mathrm{ read-only-reads }\mathcal{O}\textrm{xs
    by blast
    then show ?thesis
    apply -
    apply (rule-tac x=x#xs in exI)
    apply (rule-tac x=ys in exI)
    apply (rule-tac x=t in exI)
    apply (rule-tac x=v in exI)
    apply (clarsimp simp add: Progsb)
    done
next
    case (Ghostsb A L R W)
    from Cons.prems obtain
        nvo': non-volatile-owned-or-read-only True (\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and}
        a-nro: a & read-only }\mathcal{S}\mathrm{ and
        a-unowned: a }\not\in\mathcal{O}\mathrm{ and
        a-ro': a }\in\mathrm{ read-only-reads ( }\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and
```



```
        R-owns: R}\subseteq\mathcal{O}\mathrm{ and
        consis': sharing-consistent (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\textrm{sb
        by (clarsimp simp add: Ghostsb)
    from R-owns a-unowned
    have a-R: a & R
        by auto
    show ?thesis
    proof (cases a G A)
        case True
        from read-only-read-witness [OF nvo' a-ro']
        obtain xs ys t v' where
sb: sb = xs @ Readsb False a t v' # ys and
ro: a }\in\mathrm{ read-only (share xs ( }\mathcal{S}\oplus\textrm{W}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}))\mathrm{ ) and
a-ro-xs: a }\not=\mathrm{ read-only-reads (O) (OA - R) xs
        by blast
    with True show ?thesis
    apply -
    apply (rule-tac x=x#xs in exI)
    apply (rule-tac x=ys in exI)
    apply (rule-tac x=t in exI)
    apply (rule-tac x=v' in exI)
    apply (clarsimp simp add: Ghostsb)
    done
next
```

case False
with a-unowned R-owns a-nro L-A A-R
obtain a-nro': a $\notin$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ and a-unowned': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R}$ by (force simp add: in-read-only-convs)
from Cons.hyps [OF nvo' consis' ${ }^{\prime}$ a-nro' $\mathrm{a}-\mathrm{unowned}{ }^{\prime} \mathrm{a}$-ro $\left.{ }^{\prime}\right]$
obtain xs ys $t \mathrm{v}^{\prime}$ where $\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\text {sb }}$ False at v${ }^{\prime} \#$ ys $\wedge$
$\mathrm{a} \in$ all-acquired xs $\wedge \mathrm{a} \in$ read-only (share xs $\left.\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right) \wedge$ a $\notin$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ xs
by blast

## then show ?thesis <br> apply -

apply (rule-tac $x=x \# x s$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}^{\prime}$ in exI)
apply (clarsimp simp add: Ghostsb
done
qed
qed
qed
lemma unforwarded-not-written: $\wedge \mathrm{W} . \mathrm{a} \in$ unforwarded-non-volatile-reads sb $\mathrm{W} \Longrightarrow \mathrm{a} \notin$ W
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W')
from Cons.prems
have $\mathrm{a} \in$ unforwarded-non-volatile-reads sb (insert $\mathrm{a}^{\prime} \mathrm{W}$ )
by (clarsimp simp add: Write ${ }_{\text {sb }}$ )
from Cons.hyps [OF this]
have a $\notin$ insert $\mathrm{a}^{\prime} \mathrm{W}$.
then show ?thesis
by blast
next
case $\left(\operatorname{Read}_{\text {sb }}\right.$ volatile $\mathrm{a}^{\prime} \mathrm{t}$ v)
with Cons.hyps [of W] Cons.prems show ?thesis
by (auto split: if-split-asm)
next
case Progs $_{\text {sb }}$
with Cons.hyps [of W] Cons.prems show ?thesis
by (auto split: if-split-asm)
next

```
    case Ghost sb
    with Cons.hyps [of W] Cons.prems show ?thesis
        by (auto split: if-split-asm)
    qed
qed
```

lemma unforwarded-witness: $\bigwedge$ X.
$\llbracket \mathrm{a} \in$ unforwarded-non-volatile-reads sb X】
$\Longrightarrow$
$\exists \mathrm{xs}$ ys t v. sb=xs@ $\operatorname{Read}_{\text {sb }}$ False atv $\#$ ys $\wedge$ a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
a-unforw: $a \in$ unforwarded-non-volatile-reads sb (insert $\mathrm{a}^{\prime} \mathrm{X}$ )
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from unforwarded-not-written [OF a-unforw]
have $a^{\prime}-a$ : $a^{\prime} \neq a$
by auto
from Cons.hyps [OF a-unforw]
obtain xs ys t v where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False a t v \# ys $\wedge$
a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
thus ?thesis
using $\mathrm{a}^{\prime}$-a
apply -
apply (rule-tac $x=(x \# x s)$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
done
next
case False
from Cons.prems obtain
a-unforw: $\mathrm{a} \in$ unforwarded-non-volatile-reads sb (insert $\mathrm{a}^{\prime} \mathrm{X}$ )
by (clarsimp simp add: Write sb $_{\text {b }}$ False)
from unforwarded-not-written [OF a-unforw]
have $a^{\prime}-a: a^{\prime} \neq a$
by auto
from Cons.hyps [OF a-unforw]
obtain xs ys $t \mathrm{v}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False atveys $\wedge$
a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
thus ?thesis
using $\mathrm{a}^{\prime}-\mathrm{a}$
apply -
apply (rule-tac $\mathrm{x}=(\mathrm{x} \# \mathrm{xs})$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
done
qed
next
case ( $\operatorname{Read}_{\text {sb }}$ volatile $\mathrm{a}^{\prime} \mathrm{t}$ v)
show ?thesis
proof (cases $\mathrm{a}^{\prime}=\mathrm{a} \wedge \mathrm{a} \notin \mathrm{X} \wedge \neg$ volatile)
case True
with True show ?thesis
apply -
apply (rule-tac $x=[]$ in exI)
apply (rule-tac $x=s b$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}$ in exI)
apply (clarsimp simp add: Read $_{\text {sb }}$ )
done
next
case False
note not-ror $=$ this
with Cons.prems obtain a-unforw: a $\in$ unforwarded-non-volatile-reads sb X
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ split: if-split-asm)
from Cons.hyps [OF a-unforw]
obtain xs ys $t \mathrm{v}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False atvoys $\wedge$
a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
thus ?thesis
apply -
apply (rule-tac $\mathrm{x}=(\mathrm{x} \# \mathrm{xs})$ in exI)

```
apply (rule-tac \(x=y s\) in exI)
apply (rule-tac \(x=t\) in exI)
apply (rule-tac \(x=v\) in exI)
apply (clarsimp simp add: \(\operatorname{Read}_{\text {sb }}\) )
done
    qed
next
    case Prog \(_{\text {sb }}\)
    from Cons.prems obtain a-unforw: a \(\in\) unforwarded-non-volatile-reads sb X
            by (auto simp add: Prog \(_{\text {sb }}\) )
    from Cons.hyps [OF a-unforw]
    obtain xs ys t v where
            \(\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}\) False a t v \# ys \(\wedge\)
            a \(\notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) xs
            by blast
        thus ?thesis
            apply -
            apply (rule-tac \(x=(x \# x s)\) in exI)
            apply (rule-tac \(x=y s\) in exI)
            apply (rule-tac \(x=t\) in exI)
            apply (rule-tac \(\mathrm{x}=\mathrm{v}\) in exI)
            apply (clarsimp simp add: \(\operatorname{Prog}_{\text {sb }}\) )
            done
next
    case (Ghost \(_{\text {sb }}\) A L R W)
    from Cons.prems obtain a-unforw: a \(\in\) unforwarded-non-volatile-reads sb X
            by (auto simp add: Ghostsb)
    from Cons.hyps [OF a-unforw]
    obtain xs ys \(\mathrm{t} v\) where
        \(\mathrm{sb}=\mathrm{xs} @\) Read \(_{\mathrm{sb}}\) False at v \(\#\) ys \(\wedge\)
        a \(\notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) xs
        by blast
    thus ?thesis
        apply -
        apply (rule-tac \(x=(x \# x s)\) in exI)
        apply (rule-tac \(x=y s\) in exI)
        apply (rule-tac \(x=t\) in exI)
        apply (rule-tac \(\mathrm{x}=\mathrm{v}\) in exI)
        apply (clarsimp simp add: Ghost \({ }_{\text {sb }}\) )
        done
    qed
qed
```

lemma read-only-read-acquired-unforwarded-witness: $\bigwedge \mathcal{S} \mathcal{O}$ X.
【non-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb; sharing-consistent $\mathcal{S} \mathcal{O}$ sb;
a $\notin$ read-only $\mathcal{S} ; \mathrm{a} \notin \mathcal{O} ; \mathrm{a} \in$ read-only-reads $\mathcal{O}$ sb;
a $\in$ unforwarded-non-volatile-reads sb X 】
$\Longrightarrow$
$\exists \mathrm{xs}$ ys t v. $\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False at v $\#$ ys $\wedge \mathrm{a} \in$ all-acquired xs $\wedge$
$\mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
nvo': non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-nro: a $\notin$ read-only $\mathcal{S}$ and
a-unowned: a $\notin \mathcal{O}$ and
a-ro': a $\in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-unforw: $\mathrm{a} \in$ unforwarded-non-volatile-reads sb (insert $\mathrm{a}^{\prime} \mathrm{X}$ )
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from unforwarded-not-written [OF a-unforw]
have a-notin: a $\notin$ insert $\mathrm{a}^{\prime} \mathrm{X}$.
from R-owns a-unowned
have $a-R$ : a $\notin \mathrm{R}$
by auto
show ?thesis
proof (cases a $\in A$ )
case True
from unforwarded-witness [OF a-unforw]
obtain xs ys $t v^{\prime}$ where
$\mathrm{sb}: \mathrm{sb}=\mathrm{xs}$ @ Read $_{\mathrm{sb}}$ False a $\mathrm{t} \mathrm{v}^{\prime} \#$ ys and
a-xs: a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
with True a-notin show ?thesis
apply -
apply (rule-tac $\mathrm{x}=\mathrm{x} \# \mathrm{xs}$ in exI)
apply (rule-tac $x=y s$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{v}^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
done
next
case False
with a-unowned R-owns a-nro L-A A-R
obtain a-nro': a $\notin$ read-only $\left(\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right)$ and a-unowned': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R}$ by (force simp add: in-read-only-convs)
from Cons.hyps [OF nvo' consis' a-nro' a-unowned' a-ro' a-unforw]
obtain xs ys t v ${ }^{\prime}$ where $\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\text {sb }}$ False at v${ }^{\prime} \#$ ys $\wedge$
$a \in$ all-acquired xs $\wedge$
a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
with a-notin show ?thesis
apply -
apply (rule-tac $x=x \# x$ in exI)
apply (rule-tac $\mathrm{x}=\mathrm{ys}$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
done
qed
next
case False
from Cons.prems obtain
consis': sharing-consistent $\mathcal{S} \mathcal{O}$ sb and
a-nro': a $\notin$ read-only $\mathcal{S}$ and
a-unowned: a $\notin \mathcal{O}$ and
a-ro': a $\in$ read-only-reads $\mathcal{O}$ sb and
$a^{\prime} \in \mathcal{O}$ and
nvo': non-volatile-owned-or-read-only True $\mathcal{S} \mathcal{O}$ sb and
a-unforw': a $\in$ unforwarded-non-volatile-reads sb (insert a' X)
by (auto simp add: Write ${ }_{\text {sb }}$ False split: if-split-asm)
from unforwarded-not-written [OF a-unforw]
have a-notin: $\mathrm{a} \notin$ insert $\mathrm{a}^{\prime} \mathrm{X}$.
from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro' a-unforw']
obtain xs ys $t v^{\prime}$ where
$\mathrm{sb}=\mathrm{xs} @ \operatorname{Read}_{\mathrm{sb}}$ False at $\mathrm{v}^{\prime} \#$ ys $\wedge$
$a \in$ all-acquired xs $\wedge$ a $\notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ xs
by blast
with a-notin show ?thesis
apply -
apply (rule-tac $x=x \# x$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v^{\prime}$ in exI)
apply (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
done

```
qed
next
    case (Readsb volatile a't v)
    from Cons.prems
    obtain
        consis': sharing-consistent \mathcal{S O}}\mathrm{ sb and
        a-nro': a }\not\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
        a-unowned: a }\not\in\mathcal{O}\mathrm{ and
        a-ro': a }\in\mathrm{ read-only-reads }\mathcal{O}\mathrm{ sb and
        nvo': non-volatile-owned-or-read-only True S \mathcal{O sb and}
        a-unforw: a }\in\mathrm{ unforwarded-non-volatile-reads sb X
        by (auto simp add: Read sb split: if-split-asm)
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro' a-unforw]
    obtain xs ys t v' where
        sb = xs @ Readsb False a t v
        a}\in\mathrm{ all-acquired xs ^ a & outstanding-refs is-Write 
        by blast
    with Cons.prems show ?thesis
        apply -
        apply (rule-tac x=x#xs in exI)
        apply (rule-tac x=ys in exI)
        apply (rule-tac x=t in exI)
        apply (rule-tac x=v' in exI)
        apply (clarsimp simp add: Read
        done
next
    case Prog
    from Cons.prems
    obtain
        consis': sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb and
        a-nro': a }\not\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
        a-unowned: a }\not\in\mathcal{O}\mathrm{ and
        a-ro': a \in read-only-reads \mathcal{O sb and}
        nvo': non-volatile-owned-or-read-only True S \mathcal{O sb and}
        a-unforw: a \in unforwarded-non-volatile-reads sb X
        by (auto simp add: Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ )
from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-ro' a-unforw]
obtain xs ys t v where
    sb = xs @ Readsb False a t v # ys ^
    a \in all-acquired xs ^ a & outstanding-refs is-Write sb xs
    by blast
then show ?thesis
    apply -
    apply (rule-tac x=x#xs in exI)
    apply (rule-tac x=ys in exI)
    apply (rule-tac x=t in exI)
```

```
    apply (rule-tac x=v in exI)
    apply (clarsimp simp add: Prog}\mp@subsup{\textrm{sb}}{}{\mathrm{ )}
    done
next
    case (Ghost sb A L R W)
    from Cons.prems obtain
    nvo': non-volatile-owned-or-read-only True (\mathcal{S}\oplus\textrm{w}R
    a-nro: a }\not\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
    a-unowned: a }\not\in\mathcal{O}\mathrm{ and
    a-ro': a \in read-only-reads (\mathcal{O}\cup\textrm{A}-\textrm{R}) sb and
    A-shared-owns: A \subseteq dom S \cup\mathcal{O}\mathrm{ and L-A:L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and}
    R-owns: R \subseteq\mathcal{O}\mathrm{ and}
    consis': sharing-consistent (\mathcal{S}\oplus\textrm{W}R
    a-unforw: a \in unforwarded-non-volatile-reads sb (X)
    by (clarsimp simp add: Ghost sb)
    from unforwarded-not-written [OF a-unforw]
    have a-notin: a }\not\in\textrm{X}
    from R-owns a-unowned
    have a-R: a }\not\in
        by auto
    show ?thesis
    proof (cases a }\inA\mathrm{ A)
        case True
    from unforwarded-witness [OF a-unforw]
    obtain xs ys t v' where
sb: sb = xs @ Readsb False a t v' # ys and
a-xs: a & outstanding-refs is-Write sb xs
        by blast
    with True a-notin show ?thesis
        apply -
        apply (rule-tac x=x#xs in exI)
        apply (rule-tac x=ys in exI)
apply (rule-tac x=t in exI)
apply (rule-tac x=v' in exI)
apply (clarsimp simp add: Ghostsb
done
    next
        case False
        with a-unowned R-owns a-nro L-A A-R
        obtain a-nro': a }\not\in\operatorname{read-only (\mathcal{S}\oplus\textrm{W}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ and a-unowned': a }\not\in\mathcal{O}\cup\textrm{A}-\textrm{R
        by (force simp add: in-read-only-convs)
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned' a-ro' a-unforw]
    obtain xs ys t v' where sb = xs @ Readsb False at t v' # ys ^
a}\in\mathrm{ all-acquired xs }
a}\not\in\mp@subsup{outstanding-refs is-Write sb xs}{\mathrm{ sb}}{
        by blast
```

with a-notin show ?thesis
apply -
apply (rule-tac $x=x \# x$ in exI)
apply (rule-tac $x=y$ in exI)
apply (rule-tac $x=t$ in exI)
apply (rule-tac $x=v^{\prime}$ in exI)
apply (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
done
qed
qed
qed
lemma takeWhile-prefix: $\exists$ ys. takeWhile P xs @ ys = xs
apply (induct xs)
apply auto
done
lemma unforwarded-empty-extend:
$\bigwedge \mathrm{W} . \mathrm{x} \in$ unforwarded-non-volatile-reads $\mathrm{sb}\} \Rightarrow \mathrm{x} \notin \mathrm{W} \Longrightarrow \mathrm{x} \in$ unfor-warded-non-volatile-reads sb W
apply (induct sb)
apply clarsimp
subgoal for a sb W
apply (case-tac a)
apply clarsimp
apply (frule unforwarded-not-written)
apply (drule-tac $\mathrm{W}=\{ \}$ in unforwarded-non-volatile-reads-antimono-in)
apply blast
apply (auto split: if-split-asm)
done
done
lemma notin-unforwarded-empty:
^W. a $\notin$ unforwarded-non-volatile-reads $\mathrm{sb} \mathrm{W} \Longrightarrow \mathrm{a} \notin \mathrm{W} \Longrightarrow \mathrm{a} \notin$ unfor-warded-non-volatile-reads sb $\}$
using unforwarded-empty-extend
by blast

## lemma

assumes ro: a $\in$ read-only $\mathcal{S} \longrightarrow \mathrm{a} \in$ read-only $\mathcal{S}^{\prime}$
assumes a-in: a $\in$ read-only $(\mathcal{S} \oplus \mathrm{w} \mathrm{R})$
shows a $\in$ read-only $\left(\mathcal{S}^{\prime} \oplus \mathrm{w}\right.$ R $)$
using ro a-in
by (auto simp add: in-read-only-convs)

## lemma

assumes ro: a $\in$ read-only $\mathcal{S} \longrightarrow a \in$ read-only $\mathcal{S}^{\prime}$
assumes a-in: a $\in \operatorname{read}$-only $\left(\mathcal{S} \ominus_{\mathrm{A}} \mathrm{L}\right)$

```
shows a }\in\mathrm{ read-only (S S
```

using ro a-in
by (auto simp add: in-read-only-convs)
lemma non-volatile-owned-or-read-only-read-only-reads-eq:
$\bigwedge \mathcal{S} \mathcal{S}^{\prime} \mathcal{O}$ pending-write.
【non-volatile-owned-or-read-only pending-write $\mathcal{S} \mathcal{O}$ sb;
$\forall \mathrm{a} \in$ read-only-reads $\mathcal{O}$ sb. a $\in$ read-only $\mathcal{S} \longrightarrow \mathrm{a} \in$ read-only $\mathcal{S}^{\prime}$
】
$\Longrightarrow$ non-volatile-owned-or-read-only pending-write $\mathcal{S}^{\prime} \mathcal{O}$ sb
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
nvo': non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and ro': $^{\prime} \forall \mathrm{a} \in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb. a $\in \operatorname{read}$-only $\mathcal{S} \longrightarrow$ a $\in$ read-only $\mathcal{S}^{\prime}$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
from ro ${ }^{\prime}$
have ro ${ }^{\prime \prime}: \forall$ a $\in$ read-only-reads $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb.
$a \in \operatorname{read}$-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \longrightarrow \mathrm{a} \in \operatorname{read}$-only $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF nvo' ro']
show ?thesis
by (clarsimp simp add: Write sb $_{\text {sb }}$ volatile)
next
case False
with Cons.hyps [of pending-write $\mathcal{S} \mathcal{O} \mathcal{S}$ ] Cons.prems show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case $\left(\operatorname{Read}_{s b}\right.$ volatile at v)
show ?thesis
proof (cases volatile)
case True
with Cons.hyps [of pending-write $\mathcal{S} \mathcal{O} \mathcal{S}$ ] Cons.prems show ?thesis
by (auto simp add: Read sb )
next
case False
note non-vol $=$ this
show ?thesis
proof (cases a $\in \mathcal{O}$ )

```
case True
with Cons.hyps [of pending-write S O S S Cons.prems show ?thesis
    by (auto simp add: Readsb non-vol)
        next
case False
from Cons.prems Cons.hyps [of pending-write S O S J show ?thesis
    by (clarsimp simp add: Readsb non-vol False)
            qed
        qed
    next
        case Progsb
        with Cons.hyps [of pending-write S O-S \ Cons.prems show ?thesis
            by (auto)
    next
        case (Ghostsb A L R W)
            from Cons.hyps [of pending-write (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathcal{O}\cup\textrm{A}-\textrm{R}}\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
Cons.prems
        show ?thesis
            by (auto simp add: Ghostsb in-read-only-convs)
    qed
qed
lemma non-volatile-owned-or-read-only-read-only-reads-eq':
    \mathcal{S}\mp@subsup{\mathcal{S}}{}{\prime}\mathcal{O}.
    \llbracketnon-volatile-owned-or-read-only False \mathcal{S O sb;}
    \foralla\in read-only-reads (acquired True (takeWhile (Not o is-volatile-Write sb) sb)\mathcal{O})
            (dropWhile (Not o is-volatile-Write sb) sb). a }\in\mathrm{ read-only }\mathcal{S}\longrightarrow\textrm{a}\in\mathrm{ read-only S'S
    】
    \Longrightarrow \text { non-volatile-owned-or-read-only False } \mathcal { S } ^ { \prime } \mathcal { O } \text { sb}
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Writesb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
            from Cons.prems obtain
    nvo': non-volatile-owned-or-read-only True (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb and}
    ro':}\forall\textrm{a}\in\mathrm{ read-only-reads ( }\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb. a }\in\mathrm{ read-only }\mathcal{S}\longrightarrow\textrm{a}\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime
    by (clarsimp simp add: Write sb volatile)
        from ro'
        have ro":\forall a }\in\mathrm{ read-only-reads ( }\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ sb.
            a \in read-only (S }\mp@subsup{\mathcal{S}}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\longrightarrow\textrm{a}\in\mathrm{ read-only (S'S
by (auto simp add: in-read-only-convs)
```

from non-volatile-owned-or-read-only-read-only-reads-eq [OF nvo' ro'] show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ volatile)
next
case False
with Cons.hyps [of $\mathcal{S} \mathcal{O} \mathcal{S}$ ] Cons.prems show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case $\left(\right.$ Read $_{\text {sb }}$ volatile a t v$)$
show ?thesis
proof (cases volatile)
case True
with Cons.hyps [of $\mathcal{S} \mathcal{O} \mathcal{S}$ ] Cons.prems show ?thesis
by (auto simp add: $\operatorname{Read}_{\text {sb }}$ )
next
case False
note non-vol $=$ this
show ?thesis
proof (cases $\mathrm{a} \in \mathcal{O}$ )
case True
with Cons.hyps [of $\mathcal{S} \mathcal{O} \mathcal{S}^{\prime}$ ] Cons.prems show ?thesis
by (auto simp add: Read ${ }_{\text {sb }}$ non-vol)
next
case False
from Cons.prems Cons.hyps [of $\mathcal{S} \mathcal{O} \mathcal{S}$ ] show ?thesis
by (clarsimp simp add: Read ${ }_{\text {sb }}$ non-vol False)
qed
qed
next
case Prog $_{\text {sb }}$
with Cons.hyps [of $\mathcal{S} \mathcal{O} \mathcal{S}$ ] Cons.prems show ?thesis by (auto)
next
case ( Ghost $_{\text {sb }}$ A L R W)
from Cons.hyps [of $\left.\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathcal{O} \cup \mathrm{A}-\mathrm{R} \mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right]$ Cons.prems show ?thesis
by (auto simp add: Ghostsb in-read-only-convs)
qed
qed
lemma no-write-to-read-only-memory-read-only-reads-eq:
$\wedge \mathcal{S} \mathcal{S}^{\prime}$.
【no-write-to-read-only-memory $\mathcal{S}$ sb;
$\forall \mathrm{a} \in$ outstanding-refs is-Write ${ }_{\text {sb }}$ sb. $\mathrm{a} \in$ read-only $\mathcal{S}^{\prime} \longrightarrow \mathrm{a} \in$ read-only $\mathcal{S}$
】
$\Longrightarrow$ no-write-to-read-only-memory $\mathcal{S}^{\prime}$ sb
proof (induct sb)
case Nil thus ?case by simp

```
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
            from Cons.prems obtain
nvo': no-write-to-read-only-memory (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ sb and
ro':}\forall\textrm{a}\in\mathrm{ outstanding-refs is-Write sb sb. a }\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime}\longrightarrow\textrm{a}\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
not-ro: a }\not\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{}{\prime
by (auto simp add: Write sb volatile)
    from ro'
    have ro "':}\forall\mp@code{a}\inoutstanding-refs is-Write sb sb
                            a }\in\mathrm{ read-only (S'S
by (auto simp add: in-read-only-convs)
            from Cons.hyps [OF nvo' ro'\ not-ro
            show ?thesis
by (clarsimp simp add: Writesb volatile)
    next
        case False
        with Cons.hyps [of S\mathcal{S}] Cons.prems show ?thesis
by (auto simp add: Writesb
        qed
    next
        case (Read sb volatile a t v)
        with Cons.hyps [of S S S ] Cons.prems show ?thesis
            by (auto simp add: Readsb
    next
        case Prog
        with Cons.hyps [of S S S ] Cons.prems show ?thesis
            by (auto)
    next
        case (Ghostsb A L R W)
        from Cons.hyps [of (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}] Cons.prems
        show ?thesis
            by (auto simp add: Ghost sb in-read-only-convs)
    qed
qed
lemma reads-consistent-drop:
reads-consistent False \(\mathcal{O} \mathrm{m}\) sb
\(\Longrightarrow\) reads-consistent True
(acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) \(\mathcal{O}\) )
(flush (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb) m)
(dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb)
```

using reads-consistent-append [of False $\mathcal{O} \mathrm{m}$ (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)]
apply (cases outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends takeWhile-not-vol-write-outstanding-refs dropWhile-not-vol-write-empty)
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends takeWhile-not-vol-write-outstanding-refs dropWhile-not-vol-write-empty )
apply (case-tac (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb))
apply (fastforce simp add: outstanding-refs-conv)
apply (frule dropWhile-ConsD)
apply (clarsimp split: memref.splits)
done
lemma outstanding-refs-non-volatile-Read sb-all-acquired-dropWhile': $^{\prime}$
$\bigwedge \mathrm{m} \mathcal{S} \mathcal{O}$ pending-write.
$\llbracket$ reads-consistent pending-write $\mathcal{O} \mathrm{m} \mathrm{sb}$;non-volatile-owned-or-read-only pending-write
$\mathcal{S} \mathcal{O} \mathrm{sb} ;$
$\mathrm{a} \in$ outstanding-refs $^{\mathrm{s}}$-non-volatile-Read ${ }_{\mathbf{s b}}\left(\right.$ dropWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}\right) \rrbracket$
$\Longrightarrow \mathrm{a} \in \mathcal{O} \vee \mathrm{a} \in$ all-acquired $\mathrm{sb} \vee$
$a \in \operatorname{read}-o n l y-r e a d s$ (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O}$ )
(dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons x sb)
show ?case
proof (cases x)
case (Write ${ }_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
non-vo: non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
$(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \mathrm{sb}$ and
out-vol: outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{sb}=\{ \}$ and
out: a $\in$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ sb
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in \mathcal{O}$ )
case True
show ?thesis
by (clarsimp simp add: Write sb True volatile)
next
case False
from outstanding-non-volatile-Read ${ }_{s b}$-acquired-or-read-only-reads [OF non-vo out]
have a-in: $\mathrm{a} \in$ acquired-reads True $\mathrm{sb}(\mathcal{O} \cup \mathrm{A}-\mathrm{R}) \vee$

$$
a \in \text { read-only-reads }(\mathcal{O} \cup A-R) \text { sb }
$$

by auto

```
with acquired-reads-all-acquired [of True sb \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) ]
show ?thesis
    by (auto simp add: Write \(_{\text {sb }}\) volatile)
            qed
    next
        case False
        with Cons show ?thesis
by (auto simp add: Write \({ }_{\text {sb }}\) False)
    qed
next
    case Read \({ }_{\text {sb }}\)
    with Cons show ?thesis
        apply (clarsimp simp del: o-apply simp add: Read \({ }_{\text {sb }}\)
acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) split: if-split-asm)
        apply auto
        done
    next
        case Prog \(_{\text {sb }}\)
        with Cons show ?thesis
        by (auto simp add: Read \(_{\text {sb }}\) )
next
    case Ghost \(_{\text {sb }}\) A L R W)
            with Cons.hyps [of pending-write \(\left.\mathcal{O} \cup \mathrm{A}-\mathrm{R} \quad \mathrm{m} \boldsymbol{\mathcal { S }} \oplus_{\mathrm{w}} \mathrm{R} \quad \ominus_{\mathrm{A}} \mathrm{L}\right]\)
read-only-reads-antimono [of \(\mathcal{O} \mathcal{O} \cup \mathrm{A}-\mathrm{R}\) ]
        Cons.prems show ?thesis
        by (auto simp add: Ghost \({ }_{\text {sb }}\) )
    qed
qed
```

end
theory ReduceStoreBufferSimulation
imports ReduceStoreBuffer
begin
locale initial ${ }_{\text {sb }}=$ simple-ownership-distinct + read-only-unowned + unowned-shared +
constrains ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes empty-sb: $\llbracket \mathrm{i}<$ length ts; ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow \mathrm{sb}=[]$
assumes empty-is: $\llbracket \mathrm{i}$ < length ts; $\mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow \mathrm{is}=[]$
assumes empty-rels: $\llbracket \mathrm{i}<$ length ts; ts $!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow \mathcal{R}=$ Map.empty
sublocale initial ${ }_{\text {sb }} \subseteq$ outstanding-non-volatile-refs-owned-or-read-only
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show non-volatile-owned-or-read-only False $\mathcal{S} \mathcal{O}$ sb
using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ outstanding-volatile-writes-unowned-by-others

## proof

fix ij $p_{\mathrm{i}}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \theta_{\mathrm{i}} \mathrm{sb}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \theta_{\mathrm{j}} \mathrm{sb} \mathrm{b}_{\mathrm{j}}$
assume i -bound: $\mathrm{i}<$ length ts and
$j$-bound: $\mathrm{j}<$ length ts and
neq-i-j: $i \neq j$ and
ts-i: ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \theta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$ and
ts-j: ts ! $\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \theta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{i}}=\{ \}$
using empty-sb [OF i-bound ts-i] empty-sb [OF j-bound ts-j] by auto
qed
sublocale initialsb $\subseteq$ read-only-reads-unowned proof
fix $\mathrm{ij} \mathrm{p}_{\mathrm{i}}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \theta_{\mathrm{i}} \mathrm{sb} \mathrm{b}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}} \mathrm{is} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \theta_{\mathrm{j}} \mathrm{sb} \mathrm{b}_{\mathrm{j}}$
assume $i$-bound: $i<$ length ts and
$j$-bound: j < length ts and
neq-i-j: $\mathrm{i} \neq \mathrm{j}$ and
ts-i: ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \theta_{\mathrm{i}}, \mathrm{sb}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$ and
ts-j: ts ! $\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \theta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{j} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$
read-only-reads (acquired True
(takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }_{i}$ ) $\mathcal{O}_{\mathrm{i}}$ )
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }_{\mathrm{i}}$ ) $=\{ \}$
using empty-sb [OF i-bound ts-i] empty-sb [OF j-bound ts-j] by auto qed
sublocale initial ${ }_{\text {sb }} \subseteq$ ownership-distinct
proof
fix $\mathrm{ij} \mathrm{p}_{\mathrm{i}}$ is $\mathcal{O}_{\mathrm{i}} \mathcal{R}_{\mathrm{i}} \mathcal{D}_{\mathrm{i}} \theta_{\mathrm{i}} \mathrm{sb} \mathrm{b}_{\mathrm{i}} \mathrm{p}_{\mathrm{j}} \mathrm{is} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \theta_{\mathrm{j}} \mathrm{sb} \mathrm{b}_{\mathrm{j}}$
assume i -bound: $\mathrm{i}<$ length ts and
j -bound: j < length ts and
neq-i-j: $\mathrm{i} \neq \mathrm{j}$ and
ts-i: ts ! $\mathrm{i}=\left(\mathrm{p}_{\mathrm{i}}, \mathrm{is}_{\mathrm{i}}, \theta_{\mathrm{i}}\right.$, sb $\left.\mathrm{b}_{\mathrm{i}}, \mathcal{D}_{\mathrm{i}}, \mathcal{O}_{\mathrm{i}}, \mathcal{R}_{\mathrm{i}}\right)$ and
ts-j: ts! $\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \theta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{\mathrm{i}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{i}}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
using simple-ownership-distinct [OF i-bound j-bound neq-i-j ts-i ts-j] empty-sb [OF i-bound ts-i] empty-sb [OF j-bound ts-j]
by auto
qed
sublocale initial ${ }_{\mathrm{sb}} \subseteq$ valid-ownership ..
sublocale initial ${ }_{\text {sb }} \subseteq$ outstanding-non-volatile-writes-unshared
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show non-volatile-writes-unshared $\mathcal{S}$ sb
using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ sharing-consis
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i-bound: i < length ts
assume ts-i: ts!i $=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show sharing-consistent $\mathcal{S} \mathcal{O}$ sb
using empty-sb [OF i-bound ts-i] by auto qed
sublocale initial ${ }_{\text {sb }} \subseteq$ no-outstanding-write-to-read-only-memory
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show no-write-to-read-only-memory $\mathcal{S}$ sb
using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ valid-sharing ..
sublocale initial ${ }_{\text {sb }} \subseteq$ valid-ownership-and-sharing ..
sublocale initial ${ }_{\text {sb }} \subseteq$ load-tmps-distinct
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts! $!=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show distinct-load-tmps is
using empty-is [ OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ read-tmps-distinct
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts $!i=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show distinct-read-tmps sb
using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ load-tmps-read-tmps-distinct
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i -bound: $\mathrm{i}<$ length ts
assume ts-i: ts $!=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show load-tmps is $\cap$ read-tmps sb $=\{ \}$
using empty-sb [OF i-bound ts-i] empty-is [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ load-tmps-read-tmps-distinct ..
sublocale initial ${ }_{\text {sb }} \subseteq$ valid-write-sops
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb $p$
assume i-bound: $i<$ length ts
assume ts-i: ts! $!=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show $\forall$ sop $\in$ write-sops sb. valid-sop sop
using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial ${ }_{\text {sb }} \subseteq$ valid-store-sops
proof
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta \mathrm{sb} \mathrm{p}$
assume i-bound: $i<$ length ts
assume ts-i: ts! $!=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show $\forall$ sop $\in$ store-sops is. valid-sop sop
using empty-is [ OF i-bound ts-i] by auto
qed

```
sublocale initialsb }\subseteq\mathrm{ valid-sops ..
sublocale initialsb }\subseteq\mathrm{ valid-reads
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show reads-consistent False }\mathcal{O}\textrm{m}\mathrm{ sb
    using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initialsb}\subseteq\mathrm{ valid-history
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i=(p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show program.history-consistent program-step 0 (hd-prog p sb) sb
    using empty-sb [OF i-bound ts-i] by (auto simp add: program.history-consistent.simps)
qed
sublocale initial sb }\subseteq\mathrm{ valid-data-dependency
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i=(p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show data-dependency-consistent-instrs (dom 0) is
    using empty-is [OF i-bound ts-i] by auto
next
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i=(p,is, }0,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R}
    show load-tmps is \cap U(fst ' write-sops sb) = {}
    using empty-is [OF i-bound ts-i] empty-sb [OF i-bound ts-i] by auto
qed
sublocale initial sb }\subseteq\mathrm{ load-tmps-fresh
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show load-tmps is \cap dom 0={}
    using empty-is [OF i-bound ts-i] by auto
qed
sublocale initialsb }\subseteq\mathrm{ enough-flushs
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is, }0,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R}
    show outstanding-refs is-volatile-Write sb sb = {}
    using empty-sb [OF i-bound ts-i] by auto
qed
sublocale initialsb }\subseteq\mathrm{ valid-program-history
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\textrm{sb}p\mp@subsup{\textrm{sb}}{1}{}\mp@subsup{\textrm{sb}}{2}{
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assume sb: sb=sb
    show }\exists\mathrm{ isa. instrs sb 2 @ is = isa @ prog-instrs sb
```

using empty－sb［OF i－bound ts－i］empty－is［OF i－bound ts－i］sb by auto
next
fix i is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb p
assume i －bound： $\mathrm{i}<$ length ts
assume ts－i：ts！ $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
show last－prog p sb $=\mathrm{p}$
using empty－sb［OF i－bound ts－i］by auto
qed

## inductive

sim－config：：（＇p，＇p store－buffer，bool，owns，rels）thread－config list $\times$ memory $\times$ shared $\Rightarrow$
（＇p，unit，bool，owns，rels）thread－config list $\times$ memory $\times$ shared $\Rightarrow$ bool
（－～－［60，60］100）
where
【m $=$ flush－all－until－volatile－write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}$ ；
$\mathcal{S}=$ share－all－until－volatile－write $\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}} ;$
length $\mathrm{ts}_{\mathrm{sb}}=$ length ts ；
$\forall \mathrm{i}<$ length $\mathrm{ts}_{\mathrm{sb}}$ ．
let $\left(\mathrm{p}, \mathrm{is}_{\mathrm{sb}}, \theta, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}, \mathcal{R}\right)=\mathrm{ts}_{\mathrm{sb}}!$ ；
suspends $=$ dropWhile（Not $\circ$ is－volatile－Write ${ }_{\text {sb }}$ ）sb
in $\exists$ is $\mathcal{D}$ ．instrs suspends＠ $\mathrm{is}_{\mathrm{sb}}=$ is＠prog－instrs suspends $\wedge$
$\mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.$ outstanding－refs is－volatile－Write $\left.{ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}\right) \wedge$
ts $!i=(h d-p r o g ~ p$ suspends，
is，
$\left.\theta\right|^{‘}$（dom $\theta$－read－tmps suspends），（），
$\mathcal{D}$ ，
acquired True（takeWhile（Not $\circ$ is－volatile－Write ${ }_{\text {sb }}$ ）sb） $\mathcal{O}$ ， release（takeWhile（Not $\circ$ is－volatile－Write ${ }_{\text {sb }}$ ）sb）（dom $\left.\mathcal{S}_{\text {sb }}\right) \mathcal{R}$ ）
』
$\Longrightarrow$
$\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})$
The machine without history only stores writes in the store－buffer．inductive sim－history－config：：
（＇p，＇p store－buffer，＇dirty，＇owns，＇rels）thread－config list $\Rightarrow$（＇p，＇p store－buffer，bool，owns，rels）thread－config list $\Rightarrow$ bool
$\left(-\sim_{h}-[60,60] 100\right)$
where
【length ts＝length $\mathrm{ts}_{\mathrm{h}}$ ；
$\forall \mathrm{i}<$ length ts．
$\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.$ ．
let $(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{h}}!$ in ts $!i=\left(p\right.$, is,$\theta$ ，filter is－Write $\left.{ }_{\text {sb }} \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right) \wedge$ （filter is－Write ${ }_{\text {sb }} \mathrm{sb}=[] \longrightarrow \mathrm{sb}=[]$ ））
】
$\qquad$

$$
\text { ts } \sim_{h} t_{\mathrm{h}}
$$

lemma（in initial ${ }_{\text {sb }}$ ）history－refl：ts $\sim_{h}$ ts
apply－
apply（rule sim－history－config．intros）
apply simp
apply clarsimp
subgoal for i
apply（case－tac ts！i）
apply（drule－tac $\mathrm{i}=\mathrm{i}$ in empty－sb）
apply assumption
apply auto
done
done
lemma share-all-empty: $\forall \mathrm{i} \mathrm{p}$ is xs sb $\mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]$ $\Longrightarrow$ share-all-until-volatile-write ts $\mathcal{S}=\mathcal{S}$
apply (induct ts)
apply clarsimp
apply clarsimp
apply (frule-tac $x=0$ in spec)
apply clarsimp
apply force
done
lemma flush-all-empty: $\forall \mathrm{i} \mathrm{p}$ is xs sb $\mathcal{D} \mathcal{O} \mathcal{R}$. $\mathrm{i}<$ length $\mathrm{ts} \longrightarrow \mathrm{ts}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]$
$\Longrightarrow$ flush-all-until-volatile-write ts $\mathrm{m}=\mathrm{m}$
apply (induct ts)
apply clarsimp
apply clarsimp
apply (frule-tac $x=0$ in spec)
apply clarsimp
apply force
done
lemma sim-config-emptyE:
assumes empty:
$\forall \mathrm{i} \mathrm{p}$ is $\mathrm{xs} \mathrm{sb} \mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<$ length $\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]$
assumes sim: $\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})$
shows $\mathcal{S}=\mathcal{S}_{\mathrm{sb}} \wedge \mathrm{m}=\mathrm{m}_{\mathrm{sb}} \wedge$ length $\mathrm{ts}=$ length $\mathrm{ts}_{\mathrm{sb}} \wedge$
$\left(\forall \mathrm{i}<\right.$ length $\mathrm{ts}_{\mathrm{sb}}$.
let $(\mathrm{p}$, is, $\theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{s}}$ !i
in ts! $i=(p$, is, $\theta,(), \mathcal{D}, \mathcal{O}, \mathcal{R}))$
proof -

## from sim

show ?thesis
apply cases
apply (clarsimp simp add: flush-all-empty [OF empty] share-all-empty [OF empty])
subgoal for i
apply (drule-tac $x=i$ in spec)
apply (cut-tac $\mathrm{i}=\mathrm{i}$ in empty [rule-format])
apply clarsimp
apply assumption
apply (auto simp add: Let-def)
done
done
qed
lemma sim-config-emptyl:
assumes empty:
$\forall \mathrm{i} \mathrm{p}$ is $\mathrm{xs} \mathrm{sb} \mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<$ length $\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]$
assumes leq: length $\mathrm{ts}=$ length $\mathrm{ts}_{\mathrm{sb}}$
assumes ts: $\left(\forall \mathrm{i}<\right.$ length $\mathrm{ts}_{\mathrm{sb}}$.
let $(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}$
in ts! $\mathrm{i}=(\mathrm{p}$, is, $\theta,(), \mathcal{D}, \mathcal{O}, \mathcal{R}))$
shows ( $\left.\mathrm{tt}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right)$
apply (rule sim-config.intros)
apply (simp add: flush-all-empty [OF empty])
apply (simp add: share-all-empty [OF empty])
apply (simp add: leq)
apply (clarsimp)
apply (frule (1) empty [rule-format])
using ts
apply (auto simp add: Let-def)
done
lemma mem-eq-un-eq: 【length ts'=length ts; $\forall \mathrm{i}<$ length ts'. $\mathrm{P}\left(\mathrm{ts} s^{\prime}!\mathrm{i}\right)=\mathrm{Q}(\mathrm{ts}!\mathrm{i}) \rrbracket \Longrightarrow(\bigcup \mathrm{x} \in$ set ts'. $\mathrm{P} \times \mathrm{x})=$ ( $\bigcup x \in$ set ts. $Q x$ )
apply (auto simp add: in-set-conv-nth )
apply (force dest!: nth-mem)
apply (frule nth-mem)
subgoal for x i
apply (drule-tac $x=i$ in spec)
apply auto
done
done
lemma (in program) trace-to-steps:
assumes trace: trace c $0 k$
shows steps: c $0 \Rightarrow{ }_{\mathrm{d}}{ }^{*} \mathrm{ck}$
using trace
proof (induct k)
case 0
show $c 0 \Rightarrow d^{*}$ c 0
by auto
next
case (Suc k)
have prem: trace c 0 (Suc k) by fact
hence trace c $0 k$
by (auto simp add: program-trace-def)
from Suc.hyps [OF this]
have $c 0 \Rightarrow d^{*} c k$.
also
term program-trace
from prem interpret program-trace program-step c 0 Suc $k$.
from step [of $k$ ] have $c(k) \Rightarrow_{d} c$ (Suc $k$ )
by auto
finally show ?case .
qed
lemma (in program) safe-reach-to-safe-reach-upto:
assumes safe-reach: safe-reach-direct safe $c_{0}$
shows safe-reach-upto $n$ safe $c_{0}$
proof
fix kc
assume $k-n$ : $\mathrm{k} \leq \mathrm{n}$
assume trace: trace c 0 k
assume c-0: c $0=c_{0}$
assume l-k: $1 \leq k$
show safe (c I)
proof -
from trace k-n l-k have trace': trace c 0 I
by (auto simp add: program-trace-def)
from trace-to-steps [OF trace]
have c $0 \Rightarrow{ }_{\mathrm{d}}{ }^{*} \mathrm{c}$ I.
with safe-reach c-0 show safe (cl)
by (cases cl) (auto simp add: safe-reach-def)
qed
qed
lemma (in program-progress) safe-free-flowing-implies-safe-delayed':
assumes init: initial ${ }_{\text {sb }}$ ts $_{\text {sb }} \mathcal{S}_{\text {sb }}$
assumes sim: $\left(\mathrm{t}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})$
assumes safe-reach-ff: safe-reach-direct safe-free-flowing ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
shows safe-reach-direct safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
proof -

```
from init
interpret ini: initialsb ts sb }\mp@subsup{\mathcal{S}}{\textrm{sb}}{
from sim obtain
    m:m}=\mathrm{ flush-all-until-volatile-write ts mb m}\mp@subsup{m}{sb}{}\mathrm{ and
    S:S = share-all-until-volatile-write ts sb }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mathrm{ and
    leq: length ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}=\mathrm{ length ts and
    t-sim: }\forall\textrm{i}<<length ts sb
    let (p, is sb
    suspends = dropWhile (Not o is-volatile-Write sb) sb
in \existsis \mathcal{D}. instrs suspends @ is isb = is @ prog-instrs suspends }
            \mathcal{D}
        ts!i = (hd-prog p suspends,
            is,
                0 |` (dom 0 - read-tmps suspends),(),
                D
                acquired True (takeWhile (Not \circ is-volatile-Write sb) sb) }\mathcal{O}\mathrm{ ,
                release (takeWhile (Not o is-volatile-Write sb) sb) (dom }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{})\mathcal{R}
    by cases auto
from ini.empty-sb
have shared-eq: S = \mathcal{S}
    apply (simp only: S
    apply (rule share-all-empty)
    apply force
    done
have sd: simple-ownership-distinct ts
proof
```



```
    assume i-bound: i < length ts and
        j-bound: j < length ts and
        neq-i-j: i\not=j and
```



```
    ts-j: ts ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{i}}{\textrm{j}}{\textrm{j}},\mp@subsup{0}{\textrm{j}}{},\mp@code{sb
    show (\mathcal{O}}\mp@subsup{\boldsymbol{i}}{}{\prime})\cap(\mp@subsup{\mathcal{O}}{j}{})={
    proof -
        from t-sim [simplified leq, rule-format, OF i-bound] ini.empty-sb [simplified leq, OF i-bound]
        have ts-i: ts sb !i = (pi, is }\mp@subsup{\textrm{s}}{\textrm{i}}{},\mp@subsup{0}{\textrm{i}}{,},[],\mathcal{D},\mp@subsup{\mathcal{D}}{\textrm{i}}{},\mp@subsup{\mathcal{O}}{\textrm{i}}{},\mp@subsup{\mathcal{R}}{\textrm{i}}{}
        using ts-i
            by (force simp add: Let-def)
    from t-sim [simplified leq, rule-format, OF j-bound] ini.empty-sb [simplified leq, OF j-bound]
    have ts-j: ts sb !j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{i}}{\textrm{j}}{j},\mp@subsup{0}{\textrm{j}}{},[],\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
    using ts-j
        by (force simp add: Let-def)
    from ini.simple-ownership-distinct [simplified leq, OF i-bound j-bound neq-i-j ts-i ts-j]
    show ?thesis.
    qed
qed
have ro: read-only-unowned }\mathcal{S}\mathrm{ ts
proof
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is, , ,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    show }\mathcal{O}\cap\mathrm{ read-only }\mathcal{S}={
    proof -
        from t-sim [simplified leq, rule-format, OF i-bound] ini.empty-sb [simplified leq, OF i-bound]
        have ts-i: ts stb}!\textrm{i}=(\textrm{p},\textrm{is},0,[],\mathcal{D},\mathcal{O},\mathcal{R}
        using ts-i
            by (force simp add: Let-def)
    from ini.read-only-unowned [simplified leq, OF i-bound ts-i] shared-eq
    show ?thesis by simp
    qed
```

```
qed
have us: unowned-shared S}\mathrm{ ts
proof
    show - (\bigcup((\lambda(-, -, -, -, -,\mathcal{O},-).\mathcal{O})' set ts))\subseteq\operatorname{dom}\mathcal{S}
    proof -
        have (U((\lambda(-, -, -, -, -,\mathcal{O},-).\mathcal{O)' set ts ssb}))=(\bigcup((\lambda(-,-,-,-,-,\mathcal{O},-).\mathcal{O})' set ts))
            apply clarsimp
        apply (rule mem-eq-un-eq)
        apply (simp add: leq)
        apply clarsimp
        apply (frule t-sim [rule-format])
        apply (clarsimp simp add: Let-def)
        apply (drule (1) ini.empty-sb)
        apply auto
        done
        with ini.unowned-shared show ?thesis by (simp only: shared-eq)
    qed
qed
{
    fix i is }\mathcal{O}\mathcal{R}\mathcal{D}0\mathrm{ sb p
    assume i-bound: i < length ts
    assume ts-i: ts!i = (p,is, , sb,\mathcal{D},\mathcal{O},\mathcal{R})
    have }\mathcal{R}=\mathrm{ Map.empty
    proof -
        from t-sim [simplified leq, rule-format, OF i-bound] ini.empty-sb [simplified leq, OF i-bound]
        have ts-i: ts s⿱丶万⿱⿰㇒一乂⿴⿱冂一⿰丨丨丁心
        using ts-i
            by (force simp add: Let-def)
        from ini.empty-rels [simplified leq, OF i-bound ts-i]
        show ?thesis.
    qed
}
with us have initial: initial (ts, m, S
    by (fastforce simp add: initial-def)
{
    fix ts' S' }\mp@subsup{\mathbf{m}}{}{\prime
    assume steps: (ts,m,S) = }\mp@subsup{}{\textrm{d}}{}\mp@subsup{}{}{*}(\mp@subsup{\textrm{ts}}{}{\prime},\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime}
    have safe-delayed (ts',}\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime}
    proof -
        from steps-to-trace [OF steps] obtain c k
        where trace: trace c 0 k and c-0: c 0 = (ts,m,S) and c-k: c k = (ts', m
            by auto
        from safe-reach-to-safe-reach-upto [OF safe-reach-ff]
        have safe-upto-k: safe-reach-upto k safe-free-flowing (ts, m, S}\mathrm{ ).
        from safe-free-flowing-implies-safe-delayed [OF -- - safe-upto-k, simplified, OF initial sd ro us]
        have safe-reach-upto k safe-delayed (ts, m, S}\mathrm{ ).
        then interpret program-safe-reach-upto program-step k safe-delayed (ts,m,S).
        from safe-config [where c=c and k=k and I=k, OF - trace c-0] c-k show ?thesis by simp
    qed
}
then show ?thesis
    by (clarsimp simp add: safe-reach-def)
qed
```

lemma map-onws-sb-owned: $\backslash \mathrm{j} . \mathrm{j}$ < length ts $\Longrightarrow$ map $\mathcal{O}-\mathrm{sb}$ ts $!\mathrm{j}=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right) \Longrightarrow$ map owned ts $!\mathrm{j}=\mathcal{O}_{\mathrm{j}}$
apply (induct ts)
apply simp
subgoal for $t$ ts $j$
apply (case-tac j)
apply (case-tac t)
apply auto
done
done
lemma map-onws-sb-owned': $\wedge \mathrm{j} . \mathrm{j}<$ length $\mathrm{ts} \Longrightarrow \mathcal{O}$-sb $(\mathrm{ts}!\mathrm{j})=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right) \Longrightarrow$ owned (ts! j$)=\mathcal{O}_{\mathrm{j}}$
apply (induct ts)
apply simp
subgoal for t ts j
apply (case-tac j)
apply (case-tac t)
apply auto
done
done
lemma read-only-read-acquired-unforwarded-acquire-witness:
$\wedge \mathcal{S} \mathcal{O}$.[nnon-volatile-owned-or-read-only $\operatorname{True} \mathcal{S} \mathcal{O}$ sb;
sharing-consistent $\mathcal{S} \mathcal{O}$ sb; a $\notin$ read-only $\mathcal{S} ;$ a $\notin \mathcal{O}$;
$a \in$ unforwarded-non-volatile-reads sb $\mathrm{X} \rrbracket$
$\Longrightarrow(\exists$ sop a' v ys zs A L R W.
sb = ys @ Writesb True a' sop v A LR W \# zs ^
$a \in A \wedge a \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ ys $\left.\wedge a^{\prime} \neq a\right) \vee$
( $\exists \mathrm{A} L \mathrm{RW}$ ys zs. sb $=$ ys @ Ghost $_{\text {sb }} \mathrm{A} L \mathrm{RW} \mathrm{W} \# \mathrm{zs} \wedge a \in \mathrm{~A} \wedge \mathrm{a} \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ ys)
proof (induct sb)
case Nil thus ?case by simp
next
case (Cons $\times \mathrm{sb}$ )
show ?case
proof (cases $x$ )
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems obtain
nvo': non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-nro: a $\notin$ read-only $\mathcal{S}$ and
a-unowned: $a \notin \mathcal{O}$ and
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-unforw: a $\in$ unforwarded-non-volatile-reads sb (insert a' $X$ )
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
from unforwarded-not-written [OF a-unforw]
have a-notin: a $\notin$ insert $a^{\prime} X$.
hence $a^{\prime}-a$ : $a^{\prime} \neq a$
by simp
from R-owns a-unowned
have $a-R: a \notin R$
by auto
show ?thesis
proof (cases $a \in A$ )
case True
then show ?thesis
apply -
apply (rule disjl1)
apply (rule-tac $x=s o p$ in exl)
apply (rule-tac $x=a^{\prime}$ in exl)
apply (rule-tac $x=v$ in exl)

```
apply (rule-tac \(x=[]\) in exl)
apply (rule-tac \(x=s b\) in exl)
apply (simp add: Write \({ }_{s b}\) volatile True \(a^{\prime}-a\) )
done
    next
case False
with a-unowned R-owns a-nro L-A A-R
obtain a-nro': a \(\notin\) read-only \(\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) and a-unowned': a \(\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R}\)
    by (force simp add: in-read-only-convs)
```

from Cons.hyps [OF nvo' consis' a-nro' a-unowned' a-unforw]
have ( $\exists$ sop $a^{\prime}$ v ys zs A L R W.
sb = ys @ Writesb True a' sop v A LRW \# zs $\wedge$
$a \in A \wedge a \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ ys $\left.\wedge a^{\prime} \neq a\right) \vee$
( $\exists \mathrm{ALRW}$ ys zs. sb $=$ ys @ Ghostsb $\mathrm{ALRW} \mathrm{\#} \mathrm{zs} \wedge a \in \mathrm{~A} \wedge a \notin$ outstanding-refs is-Writesb ys)
(is ?write $\vee$ ?ghst)
by simp
then show ?thesis
proof
assume ?write
then obtain sop ${ }^{\prime} a^{\prime \prime} v^{\prime}$ ys zs $A^{\prime} L^{\prime} R^{\prime} W^{\prime}$ where
sb: sb = ys @ Write ${ }_{\text {sb }}$ True a" sop' $\mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ zs and
props: $a \in A^{\prime} a \notin$ outstanding-refs is-Write sb ys $\wedge a^{\prime \prime} \neq a$
by auto
show ?thesis
using props False a-notin sb
apply -
apply (rule disjl1)
apply (rule-tac $x=$ sop $^{\prime}$ in exl)
apply (rule-tac $x=a^{\prime \prime}$ in exl)
apply (rule-tac $x=v^{\prime}$ in exl)
apply (rule-tac $x=(x \# y s)$ in exl)
apply (rule-tac $x=z s$ in exl)
apply (simp add: Write $_{\text {sb }}$ volatile False $a^{\prime}-a$ )
done
next
assume ?ghst
then obtain ys zs $A^{\prime} L^{\prime} R^{\prime} W^{\prime}$ where
sb: sb = ys @ Ghostsb $A^{\prime} L^{\prime} R^{\prime} W^{\prime} \#$ zs and
props: $a \in A^{\prime} a \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ ys
by auto
show ?thesis
using props False a-notin sb
apply -
apply (rule disj12)
apply (rule-tac $x=\mathrm{A}^{\prime}$ in exl)
apply (rule-tac $x=L^{\prime}$ in exl)
apply (rule-tac $x=R^{\prime}$ in exl)
apply (rule-tac $x=W^{\prime}$ in exl)
apply (rule-tac $x=(x \# y s)$ in exl)
apply (rule-tac $x=z s$ in exl)
apply (simp add: Write sb volatile False $a^{\prime}-a$ )
done
qed
qed
next

```
    case False
    from Cons.prems obtain
consis': sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb and
a-nro': a \not\in read-only }\mathcal{S}\mathrm{ and
a-unowned: a \not\in\mathcal{O}\mathrm{ and}
a-ro': a' }\in\mathcal{O}\mathrm{ and
nvo': non-volatile-owned-or-read-only True \mathcal{S O}
a-unforw': a \in unforwarded-non-volatile-reads sb (insert a' X)
by (auto simp add: Writesb False split: if-split-asm)
    from unforwarded-not-written [OF a-unforw']
    have a-notin: a & insert a}\mp@subsup{a}{}{\prime}X\mathrm{ .
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-unforw']
    have ( }\exists\mathrm{ sop a' v ys zs A L R W.
            sb = ys @ Writesb True a' sop v A L R W # zs ^
            a}\inA\wedgea\not\in\mathrm{ outstanding-refs is-Write sb ys }\wedge\mp@subsup{a}{}{\prime}\not=a)
            (\existsALRW ys zs.sb = ys @ Ghostsb A LR W# zs ^a\inA ^a| outstanding-refs is-Writesb ys)
        (is ?write \vee ?ghst)
by simp
then show ?thesis
    proof
    assume ?write
    then obtain sop' ' '" v' ys zs A' L' R' W' where
        sb: sb = ys @ Write sb True a" sop' v' A' L' R' W' # zs and
        props: a }\in\mp@subsup{A}{}{\prime}\mathrm{ a }\not\in\mathrm{ outstanding-refs is-Write sb ys }\wedge\mp@subsup{a}{}{\prime\prime}\not=
    by auto
    show ?thesis
    using props False a-notin sb
        apply -
        apply (rule disjl1)
        apply (rule-tac x=sop' in exl)
        apply (rule-tac x=a"' in exl)
        apply (rule-tac }x=\mp@subsup{v}{}{\prime}\mathrm{ in exl)
        apply (rule-tac x=(x#ys) in exl)
        apply (rule-tac x=zs in exl)
        apply (simp add: Write sb False )
        done
next
    assume ?ghst
    then obtain ys zs A' L' R' }\mp@subsup{R}{}{\prime}\mathrm{ where
        sb: sb = ys @ Ghostsb A' L' R' W' # zs and
        props: a }\in\mp@subsup{A}{}{\prime}a\not\in\mathrm{ outstanding-refs is-Write sb ys
    by auto
show ?thesis
using props False a-notin sb
    apply -
    apply (rule disj12)
    apply (rule-tac x=A' in exl)
    apply (rule-tac x=L' in exl)
    apply (rule-tac x=R' in exl)
    apply (rule-tac }x=\mp@subsup{W}{}{\prime}\mathrm{ in exl)
    apply (rule-tac x=(x#ys) in exl)
    apply (rule-tac x=zs in exl)
    apply (simp add: Write sb False )
    done
```

```
qed
    qed
    next
    case (Readsb volatile a't v)
    from Cons.prems
    obtain
        consis': sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb and
        a-nro': a \not\in read-only }\mathcal{S}\mathrm{ and
        a-unowned: a }\not=\mathcal{O}\mathrm{ and
        nvo': non-volatile-owned-or-read-only True S\mathcal{O}
        a-unforw: a \in unforwarded-non-volatile-reads sb X
        by (auto simp add: Readsb split: if-split-asm)
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-unforw]
    have (\exists sop a' v ys zs A L R W.
                    sb = ys @ Writesb True a' sop v A L R W # zs ^
                    a\inA ^a & outstanding-refs is-Write sb ys }\wedge\mp@subsup{a}{}{\prime}\not=a)
                    (\existsA LR W ys zs. sb = ys @ Ghostsb A LR W# zs ^a\inA A a & outstanding-refs is-Writesb ys)
        (is ?write \vee ?ghst)
        by simp
    then show ?thesis
    proof
        assume ?write
        then obtain sop' 'a" v' ys zs A' L' R' W' where
        sb: sb = ys @ Write sb True a" sop' v' A' L' R' W' # zs and
        props: a }\in\mp@subsup{A}{}{\prime}\mathrm{ a & outstanding-refs is-Write sb ys }\wedge a\mp@subsup{a}{}{\prime\prime}\not=
        by auto
    show ?thesis
    using props sb
        apply -
apply (rule disjl1)
apply (rule-tac x=sop' in exl)
apply (rule-tac x=a"' in exl)
apply (rule-tac x=v' in exl)
apply (rule-tac x=(x#ys) in exl)
apply (rule-tac x=zs in exl)
apply (simp add: Readsb)
done
    next
    assume ?ghst
    then obtain ys zs A' L' R' W' where
        sb: sb = ys @ Ghostsb A' L' R' W'# zs and
        props: a }\in\mp@subsup{A}{}{\prime}\textrm{a}\not\in\mathrm{ outstanding-refs is-Writesb ys
        by auto
    show ?thesis
    using props sb
    apply -
    apply (rule disjl2)
    apply (rule-tac x=A' in exl)
    apply (rule-tac x=L' in exl)
    apply (rule-tac x=R' in exl)
    apply (rule-tac x=W' in exl)
    apply (rule-tac x=(x#ys) in exl)
    apply (rule-tac x=zs in exl)
    apply (simp add: Readsb )
    done
qed
```

```
next
    case Progsb
    from Cons.prems
    obtain
        consis': sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb and
        a-nro': a & read-only S and
        a-unowned: a }\not\in\mathcal{O}\mathrm{ and
        nvo': non-volatile-owned-or-read-only True S\mathcal{O}}\mathrm{ sb and
        a-unforw: a \in unforwarded-non-volatile-reads sb X
        by (auto simp add: Progsb)
    from Cons.hyps [OF nvo' consis' a-nro' a-unowned a-unforw]
    have (\exists sop a' v ys zs A L R W
                sb = ys @ Writesb True a' sop v A LRW # zs ^
                a\inA ^a\not\in outstanding-refs is-Writesb ys }\wedge\mp@subsup{a}{}{\prime}\not=a)
            (\existsA LR W ys zs. sb = ys @ Ghostsb A LR W# zs ^a\inA ^a & outstanding-refs is-Writesb ys)
        (is ?write \vee ?ghst)
        by simp
    then show ?thesis
    proof
        assume ?write
    then obtain sop' a" v}\mp@subsup{v}{}{\prime}\mathrm{ ys zs A' L' R' R' W' where
        sb: sb = ys @ Writesb True a" sop' v' A' L' R' W' # zs and
        props: a }\in\mp@subsup{A}{}{\prime}a\not\in\mathrm{ outstanding-refs is-Write sb ys }\wedge\mp@subsup{a}{}{\prime\prime}\not=
        by auto
    show ?thesis
    using props sb
        apply -
apply (rule disjl1)
apply (rule-tac x=sop' in exl)
apply (rule-tac x=a"' in exl)
apply (rule-tac x=v' in exl)
apply (rule-tac x=(x#ys) in exl)
apply (rule-tac x=zs in exl)
apply (simp add: Prog}\mp@subsup{g}{\textrm{s}}{}\mathrm{ )
done
    next
        assume ?ghst
        then obtain ys zs A' L' R' W' where
        sb: sb = ys @ Ghostsb A'L'R' W'# zs and
        props: a }\in\mp@subsup{A}{}{\prime}\textrm{a}\not\in\mathrm{ outstanding-refs is-Write sb ys
        by auto
    show ?thesis
    using props sb
    apply -
    apply (rule disjl2)
    apply (rule-tac x=A' in exl)
    apply (rule-tac }x=\mp@subsup{L}{}{\prime}\mathrm{ in exl)
    apply (rule-tac x=R' in exl)
    apply (rule-tac x=W' in exl)
    apply (rule-tac x=(x#ys) in exl)
    apply (rule-tac x=zs in exl)
    apply (simp add: Progsb )
    done
    qed
next
    case (Ghost sb A L R W)
```


## from Cons.prems obtain

nvo': non-volatile-owned-or-read-only $\operatorname{True}\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and a-nro: a $\notin$ read-only $\mathcal{S}$ and
a-unowned: $a \notin \mathcal{O}$ and
A-shared-owns: $\mathrm{A} \subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}$ and $\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and
R-owns: $\mathrm{R} \subseteq \mathcal{O}$ and
consis': sharing-consistent $\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and
a-unforw: a $\in$ unforwarded-non-volatile-reads sb $X$
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
show ?thesis
proof (cases a $\in A$ )
case True
then show ?thesis
apply -
apply (rule disjı2)
apply (rule-tac $x=A$ in exl)
apply (rule-tac $x=L$ in exl)
apply (rule-tac $x=R$ in exl)
apply (rule-tac $x=W$ in exl)
apply (rule-tac $x=[]$ in exl)
apply (rule-tac $x=s b$ in exl)
apply (simp add: Ghost ${ }_{\text {sb }}$ True)
done
next
case False
with a-unowned a-nro L-A R-owns a-nro L-A A-R
obtain a-nro': a $\notin$ read-only $\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ and a-unowned': a $\notin \mathcal{O} \cup \mathrm{A}-\mathrm{R}$
by (force simp add: in-read-only-convs)
from Cons.hyps [OF nvo' consis' a-nro' a-unowned' a-unforw]
have ( $\exists$ sop a' v ys zs A L R W.
sb = ys @ Writesb True a' sop v A LRW \# zs $\wedge$
$a \in A \wedge a \notin$ outstanding-refs is-Write sb ys $\left.\wedge a^{\prime} \neq a\right) \vee$
( $\exists \mathrm{ALRW}$ ys zs. sb $=$ ys @ Ghostsb $\mathrm{A} L \mathrm{RW} \mathrm{W} \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A} \wedge \mathrm{a} \notin$ outstanding-refs is-Writesb ys ) (is ?write $\vee$ ?ghst)
by simp
then show ?thesis
proof
assume ?write
then obtain sop' $a^{\prime \prime} v^{\prime}$ ys zs $A^{\prime} L^{\prime} R^{\prime} W^{\prime}$ where sb: sb = ys @ Write ${ }_{\text {sb }}$ True $a^{\prime \prime}$ sop $^{\prime} v^{\prime} A^{\prime} L^{\prime} R^{\prime} W^{\prime} \#$ zs and props: $a \in A^{\prime} a \notin$ outstanding-refs is-Write ${ }_{\text {sb }}$ ys $\wedge a^{\prime \prime} \neq a$
by auto

```
show ?thesis
using props sb
    apply -
    apply (rule disjl1)
    apply (rule-tac x=sop' in exl)
    apply (rule-tac x=a" in exl)
    apply (rule-tac x=v' in exl)
    apply (rule-tac x=(x#ys) in exl)
    apply (rule-tac x=zs in exl)
    apply (simp add: Ghostsb False )
    done
        next
assume ?ghst
then obtain ys zs A' L' R' W' where
```

```
sb: sb = ys @ Ghost sb A' L' R' W'# zs and
props: a }\in\mp@subsup{A}{}{\prime}a\not\in\mathrm{ outstanding-refs is-Write sb ys
```

by auto

```
show ?thesis
using props sb
    apply -
    apply (rule disjl2)
    apply (rule-tac \(x=A^{\prime}\) in exl)
    apply (rule-tac \(x=L^{\prime}\) in exl)
    apply (rule-tac \(x=R^{\prime}\) in exl)
    apply (rule-tac \(x=W^{\prime}\) in exl)
    apply (rule-tac \(x=(x \# y s)\) in exl)
    apply (rule-tac \(x=z s\) in exl)
    apply (simp add: Ghost \({ }_{\text {sb }}\) False )
    done
qed
        qed
    qed
qed
```

lemma release-shared-exchange-weak:
assumes shared-eq: $\forall \mathrm{a} \in \mathcal{O} \cup$ all-acquired sb . ( $\mathcal{S}^{\prime}::$ shared) $\mathrm{a}=\mathcal{S}$ a
assumes consis: weak-sharing-consistent $\mathcal{O}$ sb
shows release sb $\left(\operatorname{dom} \mathcal{S}^{\prime}\right) \mathcal{R}=$ release $\mathrm{sb}(\operatorname{dom} \mathcal{S}) \mathcal{R}$
using shared-eq consis
proof (induct sb arbitrary: $\mathcal{S} \mathcal{S}^{\prime} \mathcal{O} \mathcal{R}$ )
case Nil thus ?case by auto
next
case (Cons x sb)
show ?case
proof (cases x)
case ( Writesb $_{\text {sb }}$ volatile a sop v A L R W)
show ?thesis
proof (cases volatile)
case True
from Cons.prems obtain
$\mathrm{L}-\mathrm{A}: \mathrm{L} \subseteq \mathrm{A}$ and $\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}$ and R -owns: $\mathrm{R} \subseteq \mathcal{O}$ and consis': weak-sharing-consistent $(\mathcal{O} \cup A-R)$ sb and
shared-eq: $\forall a \in \mathcal{O} \cup \mathrm{~A} \cup$ all-acquired $\mathrm{sb} . \mathcal{S}^{\prime} \mathrm{a}=\mathcal{S}$ a
by (clarsimp simp add: Write sb $^{\text {True })}$
from shared-eq
have shared-eq': $\forall \mathrm{a} \in \mathcal{O} \cup \mathrm{A}-\mathrm{R} \cup$ all-acquired sb. $\left(\mathcal{S}^{\prime} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{a}=\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ a
by (auto simp add: augment-shared-def restrict-shared-def)
from Cons.hyps [OF shared-eq' consis']
have release sb $\left(\operatorname{dom}\left(\mathcal{S}^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$ Map.empty $=$ release $\mathrm{sb}\left(\operatorname{dom}\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)$ Map.empty .
then show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ True domIff)
next
case False with Cons show ?thesis
by (auto simp add: Write ${ }_{\text {sb }}$ )
qed
next
case Read ${ }_{\text {sb }}$ with Cons show ?thesis by auto
next
case Prog $_{\text {sb }}$ with Cons show ?thesis

```
        by auto
    next
    case (Ghost sb A L R W)
    from Cons.prems obtain
        L-A:L\subseteqA and A-R:A \cap R = {} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
        consis': weak-sharing-consistent (\mathcal{O}\cupA-R) sb and
        shared-eq: }\forall\textrm{a}\in\mathcal{O}\cup\textrm{A}\cup\mathrm{ all-acquired sb. S'S
        by (clarsimp simp add: Ghostsb )
    from shared-eq
    have shared-eq': }\forall\textrm{a}\in\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb. (S'S
        by (auto simp add: augment-shared-def restrict-shared-def)
    from shared-eq R-owns have }\forall\textrm{a}\in\textrm{R}.(\textrm{a}\in\operatorname{dom}\mathcal{S})=(a\in\operatorname{dom}\mp@subsup{\mathcal{S}}{}{\prime}
        by (auto simp add: domlff)
    from augment-rels-shared-exchange [OF this]
    have (augment-rels (dom S'S
    with Cons.hyps [OF shared-eq' consis']
    have release sb (dom (S'S
                release sb (dom (\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})) (augment-rels (dom S) R \mathcal{R}) by simp
    then show ?thesis
        by (clarsimp simp add: Ghostsb domlff)
    qed
qed
lemma read-only-share-all-shared: }\\mathcal{S}.\llbracketa|\mathrm{ read-only (share sb S S)】
a }\in\mathrm{ read-only }\mathcal{S}\cup\mathrm{ all-shared sb
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Writesb volatile a sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            with Write sb Cons.hyps [of (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})]}\mathrm{ ] Cons.prems
            show ?thesis
            by (auto simp add: read-only-def augment-shared-def restrict-shared-def
                split: if-split-asm option.splits)
        next
            case False with Write sb Cons show ?thesis by auto
        qed
    next
        case Read
    next
        case Progsb with Cons show ?thesis by auto
    next
        case (Ghost sb A L R W)
        with Cons.hyps [of (\mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})]\mathrm{ Cons.prems
        show ?thesis
            by (auto simp add: read-only-def augment-shared-def restrict-shared-def
                split: if-split-asm option.splits)
    qed
qed
lemma read-only-shared-all-until-volatile-write-subset':
S
read-only (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ) }
    read-only \mathcal{S}\cup(\bigcup((\lambda(-, -, -, sb, -, -,-). all-shared (takeWhile (Not ○ is-volatile-Write sb) sb)) ' set ts))
```

```
proof (induct ts)
    case Nil thus ?case by simp
next
    case (Cons t ts)
    obtain p is \mathcal{O R D }0\mathrm{ sb where}
        t: t = (p,is, , sb,\mathcal{D},\mathcal{O},\mathcal{R})
        by (cases t)
    have aargh: (Not \circ is-volatile-Write sb
    by (rule ext) auto
    let ?take-sb = (takeWhile (Not \circ is-volatile-Write sb) sb)
    let ?drop-sb = (dropWhile (Not \circ is-volatile-Writesb) sb)
    {
    fix a
    assume a-in: a \in read-only
                (share-all-until-volatile-write ts
                (share ?take-sb S)) and
    a-notin-shared: a }\not\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
    a-notin-rest: a }\not\in(\bigcup((\lambda(-,-,-, sb, -, -,-). all-shared (takeWhile (Not \circ is-volatile-Write sb) sb)) ' set ts)
    have a }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write sb}) sb
    proof -
        from Cons.hyps [of (share ?take-sb \mathcal{S})] a-in a-notin-rest
        have a }\in\mathrm{ read-only (share ?take-sb S )
            by (auto simp add: aargh)
        from read-only-share-all-shared [OF this] a-notin-shared
        show ?thesis by auto
    qed
    }
    then show ?case
        by (auto simp add: t aargh)
qed
lemma read-only-share-acquired-all-shared:
\(\wedge \mathcal{O} \mathcal{S}\). weak-sharing-consistent \(\mathcal{O}\) sb \(\Longrightarrow \mathcal{O} \cap\) read-only \(\mathcal{S}=\{ \} \Longrightarrow\)
    a}\in\mathrm{ read-only (share sb S) }\Longrightarrow\textrm{a}\in\mathcal{O}\cup\mathrm{ all-acquired sb }\Longrightarrow\textrm{a}\in\mathrm{ all-shared sb
proof (induct sb)
    case Nil thus ?case by auto
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a' sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True
            note volatile=this
            from Cons.prems obtain
owns-ro: \mathcal{O}\cap read-only S}={}\mathrm{ and L-A: L }\subseteq\textrm{A}\mathrm{ and A-R: A }\cap\textrm{R}={}\mathrm{ and
R-owns: R\subseteq\mathcal{O}\mathrm{ and consis': weak-sharing-consistent (O }\cup\textrm{O}-\textrm{R})\mathrm{ sb and}
            a-share: a }\in\mathrm{ read-only (share sb (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mathrm{ ) and
            a-A-all: a }\in\mathcal{O}\cupA\cup\mathrm{ all-acquired sb
by (clarsimp simp add: Writesb True)
```

```
    from owns-ro A-R R-owns have owns-ro': (\mathcal{O}\cup\textrm{A}-\textrm{R})\cap\mathrm{ read-only (S ( }\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})={}
        by (auto simp add: in-read-only-convs)
    from Cons.hyps [OF consis' owns-ro' a-share]
    show ?thesis
    using L-A A-R R-owns owns-ro a-A-all
        by (auto simp add: Write sb volatile augment-shared-def restrict-shared-def read-only-def domlff
        split: if-split-asm)
    next
    case False
    with Cons Write sb show ?thesis by (auto)
    qed
next
    case Read
next
    case Progsb with Cons show ?thesis by auto
    next
    case (Ghostsb A L R W)
    from Cons.prems obtain
        owns-ro: \mathcal{O}\cap read-only S = {} and L-A: L \subseteqA and A-R: A \cap R={} and
        R-owns: R\subseteq\mathcal{O}\mathrm{ and consis': weak-sharing-consistent (O)}\mathcal{O}|A-R) sb and
        a-share: a }\in\mathrm{ read-only (share sb (S }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\textrm{L}}\textrm{L})\mathrm{ ) and
        a-A-all: a }\in\mathcal{O}\cup\textrm{A}\cup\mathrm{ all-acquired sb
        by (clarsimp simp add: Ghostsb)
    from owns-ro A-R R-owns have owns-ro': (\mathcal{O}\cup\textrm{A}-\textrm{R})\cap\mathrm{ read-only (S }\mp@subsup{\mathcal{W}}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})={}
        by (auto simp add: in-read-only-convs)
    from Cons.hyps [OF consis' owns-ro' a-share]
    show ?thesis
    using L-A A-R R-owns owns-ro a-A-all
        by (auto simp add: Ghost sb augment-shared-def restrict-shared-def read-only-def domlff
            split: if-split-asm)
    qed
qed
lemma read-only-share-unowned':}\\mathcal{O}\mathcal{S}\mathrm{ .
    \llbracketweak-sharing-consistent }\mathcal{O}\mathrm{ sb; }\mathcal{O}\cap\mathrm{ read-only }\mathcal{S}={};a\not\in\mathcal{O}\cup\mathrm{ all-acquired sb; a }\in\mathrm{ read-only }\mathcal{S}
    "a read-only (share sb S
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Write sb volatile a' sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case False
            with Cons Write sb show ?thesis by auto
        next
            case True
            from Cons.prems obtain
    owns-ro: \mathcal{O}\cap read-only S = {} and L-A: L\subseteq A and A-R: A \cap R={} and
    R-owns: R\subseteq\mathcal{O}\mathrm{ and consis': weak-sharing-consistent (O }\cup\textrm{A}-\textrm{R})\mathrm{ sb and}
        a-share: a }\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
        a-notin: a \not\in\mathcal{O}\mathrm{ a }\not\in\textrm{A}\mathrm{ a # all-acquired sb}
    by (clarsimp simp add: Write sb True)
        from owns-ro A-R R-owns have owns-ro': (\mathcal{O}\cup\textrm{A}-\textrm{R})\cap\mathrm{ read-only (S (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})={}
        by (auto simp add: in-read-only-convs)
        from a-notin have a-notin': a }\not=\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb
            by auto
        from a-share a-notin L-A A-R R-owns have a-ro': a \in read-only (\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
```

```
            by (auto simp add: read-only-def restrict-shared-def augment-shared-def)
            from Cons.hyps [OF consis' owns-ro' a-notin' a-ro']
            have a }\in\mathrm{ read-only (share sb (S 
            by auto
            then show ?thesis
            by (auto simp add: Write sb True)
    qed
    next
    case Read
    next
    case Progsb with Cons show ?thesis by auto
    next
    case (Ghost sb A L R W)
    from Cons.prems obtain
        owns-ro:\mathcal{O}\cap read-only \mathcal{S}={} and L-A: L\subseteqA and A-R: A \cap R = {} and
        R-owns: R\subseteq\mathcal{O}\mathrm{ and consis': weak-sharing-consistent (O}\cup\textrm{O}
        a-share: a }\in\mathrm{ read-only }\mathcal{S}\mathrm{ and
        a-notin: a }\not\in\mathcal{O}\mathrm{ a }\not\in\textrm{A}\mathrm{ a }\not\in\mathrm{ all-acquired sb
        by (clarsimp simp add: Ghostsb
    from owns-ro A-R R-owns have owns-ro':(\mathcal{O}\cup\textrm{A}-\textrm{R})\cap\mathrm{ read-only ( }\mathcal{S}\oplus\textrm{w}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})={}
        by (auto simp add: in-read-only-convs)
    from a-notin have a-notin': a }\not=\mathcal{O}\cup\textrm{A}-\textrm{R}\cup\mathrm{ all-acquired sb
        by auto
    from a-share a-notin L-A A-R R-owns have a-ro': a }\in\mathrm{ read-only (S }\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
        by (auto simp add: read-only-def restrict-shared-def augment-shared-def)
    from Cons.hyps [OF consis' owns-ro' a-notin' a-ro']
    have a }\in\mathrm{ read-only (share sb (S }\mp@subsup{\mathcal{W}}{\textrm{w}}{\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
        by auto
    then show ?thesis
        by (auto simp add: Ghost sb
    qed
qed
lemma release-False-mono:
    \S \mathcal{R. R a = Some False }\Longrightarrow\mathrm{ outstanding-refs is-volatile-Write sb sb ={} }\Longrightarrow
    release sb S \mathcal{R a = Some False}
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Ghost sb A L R W)
        have rels-a: \mathcal{R a = Some False by fact}
        then have (augment-rels S R \mathcal{R}) a = Some False
            by (auto simp add: augment-rels-def)
        from Cons.hyps [where \mathcal{R}=(\mathrm{ augment-rels S R R ), OF this] Cons.prems}
        show ?thesis
            by (clarsimp simp add: Ghostsb)
    next
        case Writesb with Cons show ?thesis by auto
    next
        case Read
    next
        case Progsb with Cons show ?thesis by auto
    qed
qed
```

```
lemma release-False-mono-take:
```



```
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
        case (Ghostsb A L R W)
        have rels-a: \mathcal{R a = Some False by fact}
        then have (augment-rels S R \mathcal{R}) a = Some False
            by (auto simp add: augment-rels-def)
        from Cons.hyps [where \mathcal{R}=(\mathrm{ augment-rels S R R}\mathrm{ ), OF this]}],\mp@code{l}
        show ?thesis
            by (clarsimp simp add: Ghostsb)
    next
        case Writesb with Cons show ?thesis by auto
    next
        case Read sb with Cons show ?thesis by auto
    next
        case Progsb with Cons show ?thesis by auto
    qed
qed
lemma shared-switch:
    \\mathcal{S}\mathcal{O}.\llbracketweak-sharing-consistent \mathcal{O}\mathrm{ sb; read-only }\mathcal{S}\cap\mathcal{O}={};
        S a # Some False; share sb S a = Some False\rrbracket
    \Longrightarrowa\in\mathcal{O}\cup\mathrm{ all-acquired sb}
proof (induct sb)
    case Nil thus ?case by (auto simp add: read-only-def)
next
    case (Cons x sb)
    have aargh: (Not \circ is-volatile-Writesb
        by (rule ext) auto
    show ?case
    proof (cases x)
        case (Ghost sb A L R W)
        from Cons.prems obtain
            dist: read-only }\mathcal{S}\cap\mathcal{O}={}\mathrm{ and
            share: }\mathcal{S}\mathrm{ a }\not=\mathrm{ Some False and
            share': share sb (\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{\prime}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}) a = Some False and
            L-A:L\subseteqA and A-R:A \capR={} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
            consis': weak-sharing-consistent (\mathcal{O}\cup\textrm{A}-\textrm{R}) sb by (clarsimp simp add: Ghost sb aargh)
            from dist L-A A-R R-owns have dist': read-only (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\cap(\mathcal{O}\cup\textrm{A}-\textrm{R})={}
                by (auto simp add: in-read-only-convs)
    show ?thesis
    proof (cases (S ©w R ӨA L) a = Some False)
            case False
            from Cons.hyps [OF consis' dist' this share']
            show ?thesis by (auto simp add: Ghostsb)
        next
            case True
            with share L-A A-R R-owns dist
            have a }\in\mathcal{O}\cup
                by (cases S a)
                    (auto simp add: augment-shared-def restrict-shared-def read-only-def split: if-split-asm )
            thus ?thesis by (auto simp add: Ghostsb
qed
```

```
next
    case ( Write \(_{\text {sb }}\) volatile a' sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        note volatile=this
        from Cons.prems obtain
        dist: read-only \(\mathcal{S} \cap \mathcal{O}=\{ \}\) and
        share: \(\mathcal{S}\) a \(\neq\) Some False and
        share': share sb \(\left(\mathcal{S} \oplus_{W} R \ominus_{A} L\right) a=\) Some False and
        \(L-A: L \subseteq A\) and \(A-R: A \cap R=\{ \}\) and \(R\)-owns: \(R \subseteq \mathcal{O}\) and
        consis': weak-sharing-consistent \((\mathcal{O} \cup \mathrm{A}-\mathrm{R})\) sb by (clarsimp simp add: Write \({ }_{\text {sb }}\) True aargh)
    from dist L-A A-R R-owns have dist': read-only \(\left(\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right) \cap(\mathcal{O} \cup \mathrm{A}-\mathrm{R})=\{ \}\)
        by (auto simp add: in-read-only-convs)
    show ?thesis
    proof (cases ( \(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) ) a \(=\) Some False)
        case False
        from Cons.hyps [OF consis' dist' this share']
        show ?thesis by (auto simp add: Write \({ }_{\text {sb }}\) True)
    next
        case True
        with share L-A A-R R-owns dist
        have \(a \in \mathcal{O} \cup A\)
            by (cases \(\mathcal{S}\) a)
                (auto simp add: augment-shared-def restrict-shared-def read-only-def split: if-split-asm )
        thus ?thesis by (auto simp add: Write sb volatile)
    qed
    next
        case False
        with Cons show ?thesis by (auto simp add: Write \({ }_{\text {sb }}\) )
    qed
next
    case Read \({ }_{\text {sb }}\) with Cons show ?thesis by (auto)
next
    case Prog \({ }_{\text {sb }}\) with Cons show ?thesis by (auto)
qed
qed
lemma shared-switch-release-False:
    \(\wedge \mathcal{S} \mathcal{R} . \llbracket\)
        outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb \(=\{ \}\);
        a \(\notin \operatorname{dom} \mathcal{S}\);
        a \(\in \operatorname{dom}\) (share sb \(\mathcal{S}\) )】
    \(\Longrightarrow\)
        release sb (dom \(\mathcal{S}) \mathcal{R} \mathrm{a}=\) Some False
proof (induct sb)
    case Nil thus ?case by (auto simp add: read-only-def)
next
    case (Cons x sb)
    have aargh: (Not \(\circ\) is-volatile-Write \(\left.{ }_{\text {sb }}\right)=\left(\lambda a . \neg\right.\) is-volatile-Write \(_{\text {sb }}\) a)
    by (rule ext) auto
show ?case
proof (cases x)
    case (Ghost \({ }_{\text {sb }}\) A L R W)
    from Cons. prems obtain
        a-notin: a \(\notin \operatorname{dom} \mathcal{S}\) and
        share: \(a \in \operatorname{dom}\left(\right.\) share sb \(\left.\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)\) and
        out' \(^{\prime}\) : outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb \(=\{ \}\)
        by (clarsimp simp add: Ghost \({ }_{\text {sb }}\) aargh)
```

```
    show ?thesis
    proof (cases a }\in\textrm{R}\mathrm{ )
        case False
        with a-notin have a }\not=\operatorname{dom}(\mathcal{S}\oplus\textrm{w}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
        by auto
    from Cons.hyps [OF out' this share]
    show ?thesis
        by (auto simp add: Ghost sb)
    next
    case True
    with a-notin have augment-rels (dom S ) R \mathcal{R a = Some False}
        by (auto simp add: augment-rels-def split: option.splits)
    from release-False-mono [of augment-rels (dom S)R \mathcal{R},\textrm{OF}\mathrm{ this out']}
    show ?thesis
        by (auto simp add: Ghostsb)
    qed
next
    case Write sb with Cons show ?thesis by (clarsimp split: if-split-asm)
    next
    case Readsb with Cons show ?thesis by auto
    next
    case Progsb with Cons show ?thesis by auto
    qed
qed
lemma release-not-unshared-no-write:
    \\mathcal{S R}.\llbracket
        outstanding-refs is-volatile-Write }\mp@subsup{\mathrm{ sb }}{\mathrm{ sb }}{\mathrm{ sb }}={}
    non-volatile-writes-unshared S sb;
    release sb (dom S ) \mathcal{R a }=\mathrm{ Some False;}
    a \in dom (share sb S )\rrbracket
        \Longrightarrow
        a }\not\in\mathrm{ outstanding-refs is-non-volatile-Write sb sb
proof (induct sb)
    case Nil thus ?case by (auto simp add: read-only-def)
next
    case (Cons x sb)
    have aargh:(Not ० is-volatile-Write sb})=(\lambdaa.\neg\mp@subsup{\mathrm{ is-volatile-Write }}{\mathrm{ sb }}{}\mathrm{ a)
        by (rule ext) auto
    show ?case
    proof (cases x)
        case (Ghost sb A L R W)
        from Cons.prems obtain
            share: a }\in\operatorname{dom}(\mathrm{ share sb (S }\mp@subsup{\mathcal{S}}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}))\mathrm{ and
            rel: release sb
                    (dom (\mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})) (augment-rels (dom S ) R R ) a \not= Some False and
        out': outstanding-refs is-volatile-Write sb sb = {} and
        nvu: non-volatile-writes-unshared (S 
        by (clarsimp simp add: Ghost sb )
        from Cons.hyps [OF out' nvu rel share]
        show ?thesis by (auto simp add: Ghost sb)
    next
        case (Write sb volatile a' sop v A L R W)
        show ?thesis
        proof (cases volatile)
            case True with Writesb Cons.prems have False by auto
            thus ?thesis ..
```

```
    next
        case False
        note not-vol = this
        from Cons.prems obtain
            rel: release sb (dom S ) \mathcal{R a }=\mathrm{ Some False and}
            out': outstanding-refs is-volatile-Write sb sb ={} and
            nvo: non-volatile-writes-unshared S sb and
            a'-not-dom: a' }\not\in\operatorname{dom}\mathcal{S}\mathrm{ and
            a-dom: a }\in\operatorname{dom}(share sb S )
            by (auto simp add: Writesb False)
    from Cons.hyps [OF out' nvo rel a-dom]
    have a-notin-rest: a & outstanding-refs is-non-volatile-Writesb sb.
    show ?thesis
    proof (cases a'=a)
        case False with a-notin-rest
        show ?thesis by (clarsimp simp add: Writesb not-vol )
    next
        case True
        from shared-switch-release-False [OF out' a'-not-dom [simplified True] a-dom]
        have release sb (dom S ) \mathcal{R a = Some False.}
        with rel have False by simp
        thus ?thesis ..
        qed
        qed
    next
        case Read
    next
    case Prog}\mp@subsup{\mp@code{sb}}{\mathrm{ with Cons show ?thesis by auto}}{
    qed
qed
corollary release-not-unshared-no-write-take:
assumes nvw: non-volatile-writes-unshared \(\mathcal{S}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb)
assumes rel: release (takeWhile (Not o is-volatile-Writesb) sb) (dom S ) \mathcal{R}}\mathrm{ a }\not=\mathrm{ Some False
assumes a-in: a \in dom (share (takeWhile (Not o is-volatile-Write sb) sb) S S)
shows
    a \not\in outstanding-refs is-non-volatile-Write sb (takeWhile (Not ○ is-volatile-Write sb) sb)
using release-not-unshared-no-write[OF takeWhile-not-vol-write-outstanding-refs [of sb] nvw rel a-in]
by simp
lemma read-only-unacquired-share':
    \S O.\llbracket\mathcal{O}\cap read-only S = {}; weak-sharing-consistent \mathcal{O sb; a \in read-only S;}
    a & all-shared sb; a & acquired True sb \mathcal{O \}
a }\in\mathrm{ read-only (share sb S)
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
    proof (cases x)
    case (Write sb volatile a' sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        note volatile=this
        from Cons.prems
        obtain a-ro: a }\in\mathrm{ read-only S and a-R: a }\not=\textrm{R}\mathrm{ and a-unsh: a & all-shared sb and
    owns-ro: }\mathcal{O}\cap\mathrm{ read-only }S={}\mathrm{ and
    L-A:L\subseteqA and A-R: A }\cap\textrm{R}={}\mathrm{ and R-owns: R}\subseteq\mathcal{O}\mathrm{ and
```

consis': weak-sharing-consistent $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ sb and a-notin: a $\notin$ acquired True sb $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
show ?thesis
proof (cases a $\in A$ )
case True
with a-R have $a \in \mathcal{O} \cup A-R$ by auto
from all-shared-acquired-in [OF this a-unsh]
have $a \in$ acquired $\operatorname{True} s b(\mathcal{O} \cup A-R)$ by auto
with a-notin have False by auto
thus ?thesis ..
next
case False
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup A-R) \cap$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)=\{ \}$
by (auto simp add: in-read-only-convs)
from a-ro False owns-ro R-owns L-A have a-ro': a $\in$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-ro' a-unsh a-notin]
show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ True)
qed
next
case False
with Cons show ?thesis
by (clarsimp simp add: Write ${ }_{\text {sb }}$ False)
qed
next
case Read $_{\text {sb }}$ with Cons show ?thesis by (clarsimp)
next
case $\mathrm{Prog}_{\mathrm{sb}}$ with Cons show ?thesis by (clarsimp)
next
case (Ghost ${ }_{\text {sb }}$ A L R W)
from Cons.prems
obtain a-ro: a $\in$ read-only $S$ and $a-R$ : $a \notin R$ and a-unsh: $a \notin$ all-shared sb and owns-ro: $\mathcal{O} \cap$ read-only $S=\{ \}$ and $L-A: L \subseteq A$ and $A-R: A \cap R=\{ \}$ and $R$-owns: $R \subseteq \mathcal{O}$ and consis': weak-sharing-consistent $(\mathcal{O} \cup A-R)$ sb and a-notin: a $\notin$ acquired $\operatorname{True} s b(\mathcal{O} \cup \mathrm{~A}-\mathrm{R})$
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
show ?thesis
proof (cases $a \in A$ )
case True
with a-R have $a \in \mathcal{O} \cup A-R$ by auto
from all-shared-acquired-in [OF this a-unsh]
have $a \in$ acquired True sb $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ by auto
with a-notin have False by auto
thus ?thesis ..
next
case False
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup A-R) \cap$ read-only $\left(S \oplus_{w} R \ominus_{A} L\right)=\{ \}$ by (auto simp add: in-read-only-convs)
from a-ro False owns-ro R-owns L-A have a-ro': a $\in$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-ro' a-unsh a-notin]
show ?thesis
by (clarsimp simp add: Ghost ${ }_{\text {sb }}$ )
qed
qed
qed
lemma read-only-share-all-until-volatile-write-unacquired':
$\wedge \mathcal{S}$. 【ownership-distinct ts; read-only-unowned $\mathcal{S}$ ts; weak-sharing-consis ts;
$\forall \mathrm{i}<$ length ts. (let $(-,-,-$, sb,-, $\mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i}$ in a $\notin$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{O} \wedge$ a $\notin$ all-shared (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb ));
a $\in$ read-only $\mathcal{S} \rrbracket$
$\Longrightarrow \mathrm{a} \in$ read-only (share-all-until-volatile-write ts $\mathcal{S}$ )
proof (induct ts)
case Nil thus ?case by simp
next
case (Const ts)
obtain p is $\mathcal{O} \mathcal{R} \mathcal{D} \theta$ sb where
$\mathrm{t}: \mathrm{t}=(\mathrm{p}, \mathrm{is}, \theta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
by (cases t )
have dist: ownership-distinct ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret ownership-distinct $\mathrm{t} \# \mathrm{ts}$.
from ownership-distinct-tl [OF dist]
have dist': ownership-distinct ts.
have aargh: $\left(\right.$ Not $\circ$ is-volatile-Write $\left._{\text {sb }}\right)=\left(\lambda a . \neg\right.$ is-volatile-Write $_{\text {sb }}$ a)
by (rule ext) auto
have a-ro: a $\in$ read-only $\mathcal{S}$ by fact
have ro-unowned: read-only-unowned $\mathcal{S}$ ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret read-only-unowned $\mathcal{S} \mathrm{t} \# \mathrm{ts}$.
have consis: weak-sharing-consis ( $\mathrm{t} \# \mathrm{ts}$ ) by fact
then interpret weak-sharing-consis $\mathrm{t} \# \mathrm{ts}$.
note consis ${ }^{\prime}=$ weak-sharing-consis-tl [OF consis]
let ? take-sb $=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {bb }}\right) \mathrm{sb}\right)$
let ?drop-sb $=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}\right)$
from weak-sharing-consis [of 0] t
have consis-sb: weak-sharing-consistent $\mathcal{O}$ sb by force
with weak-sharing-consistent-append [of $\mathcal{O}$ ?take-sb ?drop-sb]
have consis-take: weak-sharing-consistent $\mathcal{O}$ ?take-sb
by auto
have ro-unowned': read-only-unowned (share ?take-sb $\mathcal{S}$ ) ts
proof
fix j
fix $p_{j}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \theta_{\mathrm{j}} \mathrm{sb} \mathrm{b}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length ts
assume jth: ts! $=\left(p_{j}, \mathrm{is}_{\mathrm{j}}, \mathrm{\theta}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\mathcal{O}_{\mathrm{j}} \cap$ read-only (share ?take-sb $\mathcal{S}$ ) $=\{ \}$
proof -
\{
fix a
assume a-owns: $a \in \mathcal{O}_{j}$
assume a-ro: a $\in$ read-only (share ?take-sb $\mathcal{S}$ )
have False
proof -

```
            from ownership-distinct [of 0 Suc j] j-bound jth t
            have dist: (\mathcal{O}\cup\mathrm{ all-acquired sb) }\cap(\mp@subsup{\mathcal{O}}{j}{}\cup\mathrm{ all-acquired sb}}\mp@subsup{\textrm{j}}{\textrm{j}}{})={
                by fastforce
            from read-only-unowned [of Suc j] j-bound jth
            have dist-ro: }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only }\mathcal{S}={}\mathrm{ by force
            show ?thesis
            proof (cases a }\in(\mathcal{O}\cup\mathrm{ all-acquired sb))
            case True
            with dist a-owns show False by auto
            next
            case False
            hence a & (\mathcal{O \cup all-acquired ?take-sb)}\\mp@code{*})
            using all-acquired-append [of ?take-sb ?drop-sb]
                by auto
            from read-only-share-unowned [OF consis-take this a-ro]
            have a }\in\mathrm{ read-only }\mathcal{S}\mathrm{ .
            with dist-ro a-owns show False by auto
                qed
    qed
    }
    thus ?thesis by auto
    qed
qed
from Cons.prems
obtain unacq-ts: }\forall\textrm{i}<<length ts. (let (-,-,-,sb,-,\mathcal{O},-)= ts!i in
    a & acquired True (takeWhile (Not o is-volatile-Write sb) sb) \mathcal{O ^}
    a & all-shared (takeWhile (Not o is-volatile-Writesb) sb)) and
    unacq-sb: a & acquired True (takeWhile (Not o is-volatile-Write sb) sb)\mathcal{O}\mathrm{ and}
    unsh-sb: a }\not=\mathrm{ all-shared (takeWhile (Not ० is-volatile-Write sb) sb)
    apply clarsimp
    apply (rule that)
    apply (auto simp add: t aargh)
    done
    from read-only-unowned [of 0] t
    have owns-ro: }\mathcal{O}\cap\mathrm{ read-only }\mathcal{S}={
    by force
    from read-only-unacquired-share' [OF owns-ro consis-take a-ro unsh-sb unacq-sb]
    have a }\in\mathrm{ read-only (share (takeWhile (Not o is-volatile-Write esb) sb) S).
    from Cons.hyps [OF dist' ro-unowned' consis' unacq-ts this]
    show ?case
    by (simp add: t)
qed
lemma not-shared-not-acquired-switch:
    \LambdaXY. \llbracketa & all-shared sb; a & X; a & acquired True sb X; a & Y\rrbracket\Longrightarrowa& acquired True sb Y
proof (induct sb)
    case Nil thus ?case by simp
next
    case (Cons x sb)
    show ?case
```

```
proof (cases x)
    case (Write sb volatile a' sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        from Cons.prems obtain
        a-X: a \not\inX and a-acq: a \not\in acquired True sb ( }X\cupA-R) an
        a-Y: a }\not\in\textrm{Y}\mathrm{ and a-R: a & R and
        a-shared: a }\not\in\mathrm{ all-shared sb
        by (clarsimp simp add: Writesb True)
    show ?thesis
    proof (cases a }\inA\mathrm{ )
        case True
        with a-X a-R
        have a }\inX\cupA-R by auto
        from all-shared-acquired-in [OF this a-shared]
        have a }\in\mathrm{ acquired True sb ( }X\cupA-R)
        with a-acq have False by simp
        thus ?thesis ..
    next
        case False
        with a-X a-Y obtain a-X': a }\not=X\cupA - R and a-Y': a \not\inY\cupA - 
            by auto
        from Cons.hyps [OF a-shared a-X' a-acq a-Y']
        show ?thesis
            by (auto simp add: Writesb True)
    qed
    next
    case False with Cons.hyps [of X Y] Cons.prems show ?thesis by (auto simp add: Write sb)
    qed
next
    case Readsb with Cons.hyps [of X Y] Cons.prems show ?thesis by (auto)
next
    case Progsb with Cons.hyps [of X Y] Cons.prems show ?thesis by (auto)
next
    case (Ghost sb A L R W)
    from Cons.prems obtain
        a-X: a }\not\inX\mathrm{ and a-acq: a }\not\in\mathrm{ acquired True sb ( }X\cupA-R) and
        a-Y: a \not\in Y and a-R: a &R and
        a-shared: a }\not\in\mathrm{ all-shared sb
        by (clarsimp simp add: Ghost sb)
    show ?thesis
    proof (cases a \in A)
        case True
        with a-X a-R
        have a }\inX\cupA-R by aut
        from all-shared-acquired-in [OF this a-shared]
        have a }\in\mathrm{ acquired True sb ( }X\cupA-R)
        with a-acq have False by simp
        thus ?thesis ..
    next
        case False
        with a-X a-Y obtain a-X': a }\not=X\cupA - R and a- ' ': a & Y UA - R
            by auto
        from Cons.hyps [OF a-shared a-X' a-acq a-Y']
        show ?thesis
            by (auto simp add: Ghostsb)
    qed
qed
qed
```

lemma read-only-share-all-acquired-in':
$\wedge S \mathcal{O} . \llbracket \mathcal{O} \cap$ read-only $\mathrm{S}=\{ \}$; weak-sharing-consistent $\mathcal{O}$ sb; a $\in$ read-only (share sb S ) $\rrbracket$
$\Longrightarrow a \in$ read-only (share sb Map.empty) $\vee(a \in$ read-only $S \wedge a \notin$ acquired True sb $\mathcal{O} \wedge a \notin$ all-shared sb )
proof (induct sb)
case Nil thus ?case by auto next
case (Cons $\times \mathrm{sb}$ )
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile a' sop v A L R W)
show ?thesis
proof (cases volatile)
case True
note volatile=this
from Cons.prems
obtain a-in: a $\in$ read-only (share sb $\left(S \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ ) and
owns-ro: $\mathcal{O} \cap$ read-only $S=\{ \}$ and
$L-A: L \subseteq A$ and $A-R: A \cap R=\{ \}$ and $R$-owns: $R \subseteq \mathcal{O}$ and
consis': weak-sharing-consistent $(\mathcal{O} \cup A-R)$ sb
by (clarsimp simp add: Write sb True)
from owns-ro A-R R-owns have owns-ro': $(\mathcal{O} \cup A-R) \cap$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)=\{ \}$
by (auto simp add: in-read-only-convs)
from Cons.hyps [OF owns-ro' consis' a-in]
have hyp: a $\in$ read-only (share sb Map.empty) $\vee$
( $a \in \operatorname{read}-$ only $\left(S \oplus_{W} R \ominus_{A} L\right) \wedge a \notin$ acquired $\operatorname{True} s b(\mathcal{O} \cup A-R) \wedge a \notin$ all-shared sb).
have $a \in$ read-only (share sb (Map.empty $\oplus_{W} R \ominus_{A} L$ )) $V$
( $a \in$ read-only $S \wedge a \notin R \wedge a \notin$ acquired True sb $(\mathcal{O} \cup A-R) \wedge a \notin$ all-shared sb)
proof -
\{
assume a-emp: a $\in$ read-only (share sb Map.empty)
have read-only Map.empty $\subseteq$ read-only (Map.empty $\oplus_{\mathrm{w}} \mathrm{R} \ominus_{A} \mathrm{~L}$ )
by (auto simp add: in-read-only-convs)
from share-read-only-mono-in [OF a-emp this]
have $a \in$ read-only (share sb (Map.empty $\oplus_{W} R \ominus_{A} L$ )).
\}
moreover
\{
assume a-ro: a $\in$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right)$ and a-not-acq: a $\notin$ acquired $\operatorname{True}$ sb $(\mathcal{O} \cup \mathrm{A}-\mathrm{R})$ and a-unsh: a $\notin$ all-shared sb
have ?thesis
proof (cases a $\in$ read-only S)
case True
with a-ro obtain a-A: a $\notin A$
by (auto simp add: in-read-only-convs)
with True a-not-acq a-unsh R-owns owns-ro
show ?thesis
by auto
next
case False
with a-ro have a-ro-empty: $a \in$ read-only (Map.empty $\oplus_{w} R \ominus_{A} L$ )
by (auto simp add: in-read-only-convs split: if-split-asm)
have read-only (Map.empty $\oplus_{W} R \ominus_{A} L$ ) $\subseteq$ read-only $\left(S \oplus_{W} R \ominus_{A} L\right.$ )
by (auto simp add: in-read-only-convs)
with owns-ro'

```
    have owns-ro-empty: (\mathcal{O}\cupA-R) \cap read-only (Map.empty }\mp@subsup{\oplus}{W}{}R\textrm{R}\mp@subsup{\ominus}{A}{}\textrm{L})={
        by blast
    from read-only-unacquired-share' [OF owns-ro-empty consis' a-ro-empty a-unsh a-not-acq]
    have a \in read-only (share sb (Map.empty }\mp@subsup{\oplus}{W}{}R\mp@subsup{R}{A}{\prime}L\mathrm{ )).
    thus ?thesis
        by simp
    qed
}
moreover note hyp
ultimately show ?thesis by blast
    qed
    then show ?thesis
by (clarsimp simp add: Writesb True)
    next
        case False with Cons show ?thesis
by (auto simp add: Write sb)
    qed
next
    case Read
next
    case Progsb with Cons show ?thesis by auto
next
    case (Ghost sb A L R W)
    from Cons.prems
    obtain a-in: a }\in\mathrm{ read-only (share sb (S }\mp@subsup{\oplus}{W}{}R|\mp@subsup{\ominus}{A}{}L))\mathrm{ and
        owns-ro: \mathcal{O}\cap read-only S = {} and
        L-A:L\subseteqA and A-R: A \capR={} and R-owns: R\subseteq\mathcal{O}\mathrm{ and}
        consis': weak-sharing-consistent (\mathcal{O}\cupA-R) sb
        by (clarsimp simp add: Ghostsb)
```



```
        by (auto simp add: in-read-only-convs)
    from Cons.hyps [OF owns-ro' consis' a-in]
    have hyp: a }\in\mathrm{ read-only (share sb Map.empty) V
                            (a \in read-only (S }\mp@subsup{\oplus}{W}{}R\mp@subsup{\ominus}{A}{}L)\wedge a & acquired True sb (\mathcal{O}\cupA - R) ^a\not\in all-shared sb).
    have a }\in\mathrm{ read-only (share sb (Map.empty }\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}))
            (a \in read-only S ^a\not\inR\wedgea\not\in acquired True sb (\mathcal{O}\cupA - R) ^a\not\in all-shared sb)
    proof -
        {
assume a-emp: a G read-only (share sb Map.empty)
have read-only Map.empty \subseteq read-only (Map.empty }\mp@subsup{\oplus}{W}{}R\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}\mathrm{ )
    by (auto simp add: in-read-only-convs)
from share-read-only-mono-in [OF a-emp this]
have a \in read-only (share sb (Map.empty }\mp@subsup{\oplus}{W}{}R\textrm{R}\mp@subsup{\ominus}{A}{}L\mathrm{ L).
    }
    moreover
    {
assume a-ro: a \in read-only (S }\mp@subsup{\oplus}{w}{}R\mp@subsup{\ominus}{A}{}L)\mathrm{ and
        a-not-acq: a & acquired True sb (\mathcal{O}\cup\textrm{A}-\textrm{R})\mathrm{ and}
        a-unsh: a & all-shared sb
        have ?thesis
        proof (cases a \in read-only S)
case True
with a-ro obtain a-A: a \not\in A
    by (auto simp add: in-read-only-convs)
```

```
        with True a-not-acq a-unsh R-owns owns-ro
        show ?thesis
            by auto
        next
        case False
    with a-ro have a-ro-empty: a \in read-only (Map.empty }\mp@subsup{\oplus}{w}{}
    by (auto simp add: in-read-only-convs split: if-split-asm)
    have read-only (Map.empty }\mp@subsup{\oplus}{W}{}\textrm{R}\mp@subsup{\ominus}{A}{}\textrm{L})\subseteq\mathrm{ read-only (S }\mp@subsup{\oplus}{W}{}\textrm{R}\mp@subsup{\ominus}{A}{}\textrm{L}
        by (auto simp add: in-read-only-convs)
    with owns-ro'
    have owns-ro-empty: (\mathcal{O}\cupA-R) \cap read-only (Map.empty }\oplus\textrm{w}R|A\textrm{L})={
        by blast
    from read-only-unacquired-share' [OF owns-ro-empty consis' a-ro-empty a-unsh a-not-acq]
    have a \in read-only (share sb (Map.empty }\mp@subsup{\oplus}{W}{}R|\mp@subsup{\ominus}{A}{}L\mathrm{ )).
    thus ?thesis
        by simp
qed
        }
        moreover note hyp
        ultimately show ?thesis by blast
    qed
    then show ?thesis
        by (clarsimp simp add: Ghost }\mp@subsup{}{\mathrm{ bb }}{}\mathrm{ )
    qed
qed
lemma in-read-only-share-all-until-volatile-write':
    assumes dist: ownership-distinct ts
    assumes consis: sharing-consis }\mathcal{S}\mathrm{ ts
    assumes ro-unowned: read-only-unowned }\mathcal{S}\mathrm{ ts
    assumes i-bound: i < length ts
    assumes ts-i: ts!i = (p,is,0,sb,\mathcal{D},\mathcal{O},\mathcal{R})
    assumes a-unacquired-others: }\forall\textrm{j}<l\mathrm{ length ts. }\textrm{i}\not=\textrm{j}
        (let (-,-,,-s\mp@subsup{b}{j}{\prime},-,\mathcal{O},-)= ts!j in
        a & acquired True (takeWhile (Not o is-volatile-Write sb ) sb j) }\mathcal{O}
        a & all-shared (takeWhile (Not ० is-volatile-Write sb) sb j ))
    assumes a-ro-share: a }\in\mathrm{ read-only (share sb }\mathcal{S}\mathrm{ )
    shows a }\in\mathrm{ read-only (share (dropWhile (Not O is-volatile-Write sb) sb)
            (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ))
proof -
    from consis
    interpret sharing-consis }\mathcal{S}\mathrm{ ts .
    interpret read-only-unowned S S ts by fact
    from sharing-consis [OF i-bound ts-i]
    have consis-sb: sharing-consistent }\mathcal{S}\mathcal{O}\mathrm{ sb.
    from sharing-consistent-weak-sharing-consistent [OF this]
    have weak-consis: weak-sharing-consistent }\mathcal{O}\mathrm{ sb.
    from read-only-unowned [OF i-bound ts-i]
    have owns-ro: }\mathcal{O}\cap\mathrm{ read-only }\mathcal{S}={}\mathrm{ .
    from read-only-share-all-acquired-in' [OF owns-ro weak-consis a-ro-share]
    have a \in read-only (share sb Map.empty) \vee a \in read-only \mathcal{S}\wedge a # acquired True sb }\mathcal{O}\wedge a \not\exists all-shared sb
    moreover
    let ?take-sb = (takeWhile (Not ○ is-volatile-Write sb) sb)
    let ?drop-sb = (dropWhile (Not \circ is-volatile-Writesb) sb)
```

from weak-consis weak-sharing-consistent-append [of $\mathcal{O}$ ?take-sb ?drop-sb]
obtain weak-consis': weak-sharing-consistent (acquired True ?take-sb $\mathcal{O}$ ) ?drop-sb and weak-consis-take: weak-sharing-consistent $\mathcal{O}$ ?take-sb
by auto
\{
assume a $\in$ read-only (share sb Map.empty)
with share-append [of ?take-sb ?drop-sb]
have a-in': a $\in$ read-only (share ?drop-sb (share ?take-sb Map.empty))
by auto
have owns-empty: $\mathcal{O} \cap$ read-only Map.empty $=\{ \}$
by auto
from weak-sharing-consistent-preserves-distinct [OF weak-consis-take owns-empty]
have acquired True ?take-sb $\mathcal{O} \cap$ read-only (share ?take-sb Map.empty) $=\{ \}$.
from read-only-share-all-acquired-in [OF this weak-consis' a-in']
have a $\in$ read-only (share ?drop-sb Map.empty) $\vee a \in$ read-only (share ?take-sb Map.empty) $\wedge a \notin$
all-acquired ?drop-sb.
moreover
\{
assume a-ro-drop: $a \in$ read-only (share ?drop-sb Map.empty)
have read-only Map.empty $\subseteq$ read-only (share-all-until-volatile-write ts $\mathcal{S}$ )
by auto
from share-read-only-mono-in [OF a-ro-drop this]
have ?thesis .
\}
moreover
\{
assume a-ro-take: a $\in$ read-only (share ?take-sb Map.empty)
assume a-unacq-drop: a $\notin$ all-acquired ?drop-sb
from read-only-share-unowned-in [OF weak-consis-take a-ro-take]
have $\mathrm{a} \in \mathcal{O} \cup$ all-acquired ?take-sb by auto
hence $a \in \mathcal{O} \cup$ all-acquired sb using all-acquired-append [of ?take-sb ?drop-sb]
by auto
from share-all-until-volatile-write-thread-local' [OF dist consis i-bound ts-i this] a-ro-share
have ?thesis by (auto simp add: read-only-def)
\}
ultimately have ?thesis by blast
\}

## moreover

\{
assume a-ro: a $\in$ read-only $\mathcal{S}$
assume a-unacq: a $\notin$ acquired True sb $\mathcal{O}$
assume a-unsh: a $\notin$ all-shared sb
with all-shared-append [of ?take-sb ?drop-sb]
obtain a-notin-take: a $\notin$ all-shared ?take-sb and a-notin-drop: a $\notin$ all-shared ?drop-sb by auto
have ?thesis
proof (cases a $\in$ acquired True ?take-sb $\mathcal{O}$ )
case True
from all-shared-acquired-in [OF this a-notin-drop] acquired-append [of True ?take-sb ?drop-sb $\mathcal{O}$ ] a-unacq
have False by auto
thus ?thesis ..
next
case False

```
    with a-unacquired-others i-bound ts-i a-notin-take
    have a-unacq': }\forall\textrm{j}<llength ts
        (let (-,-,-,s\mp@subsup{b}{j}{\prime},-,\mathcal{O},-) = ts!j in
        a & acquired True (takeWhile (Not o is-volatile-Writesb
            a & all-shared (takeWhile (Not o is-volatile-Write sb) sb b ))
        by (auto simp add: Let-def)
    from local.weak-sharing-consis-axioms have weak-sharing-consis ts .
    from read-only-share-all-until-volatile-write-unacquired' [OF dist ro-unowned
        (weak-sharing-consis ts) a-unacq' a-ro]
    have a-ro-all: a }\in\mathrm{ read-only (share-all-until-volatile-write ts }\mathcal{S}\mathrm{ ).
    from weak-consis weak-sharing-consistent-append [of \mathcal{O ?take-sb ?drop-sb]}]
    have weak-consis-drop: weak-sharing-consistent (acquired True ?take-sb \mathcal{O) ?drop-sb}
        by auto
    from weak-sharing-consistent-preserves-distinct-share-all-until-volatile-write [OF dist
        ro-unowned (weak-sharing-consis ts) i-bound ts-i]
    have acquired True ?take-sb }\mathcal{O}
        read-only (share-all-until-volatile-write ts S})={}
            from read-only-unacquired-share' [OF this weak-consis-drop a-ro-all a-notin-drop]
            acquired-append [of True ?take-sb ?drop-sb }\mathcal{O}\mathrm{ ] a-unacq
        show ?thesis by auto
    qed
}
ultimately show ?thesis by blast
qed
lemma all-acquired-unshared-acquired:
\(\wedge \mathcal{O}\). a \(\in\) all-acquired \(\mathrm{sb}==>\mathrm{a} \notin\) all-shared \(\mathrm{sb}==>\mathrm{a} \in\) acquired True \(\mathrm{sb} \mathcal{O}\)
apply (induct sb)
apply (auto split: memref.split intro: all-shared-acquired-in)
done
lemma safe-RMW-common:
assumes safe: \(\mathcal{O}\) s, \(\mathcal{R s}, \mathrm{i} \vdash(\mathrm{RMW}\) a t \((\mathrm{D}, \mathrm{f})\) cond ret \(\mathrm{A} L \mathrm{R} \mathrm{W} \#\) is, \(\theta, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \sqrt{ }\) shows \((a \in \mathcal{O} \vee a \in \operatorname{dom} \mathcal{S}) \wedge(\forall j<\) length \(\mathcal{O} s . i \neq j \longrightarrow(\mathcal{R s}!j) a \neq\) Some False \()\)
using safe
apply (cases)
apply (auto simp add: domlff)
done
lemma acquired-reads-all-acquired': \(\Lambda \mathcal{O}\).
acquired-reads True sb \(\mathcal{O} \subseteq\) acquired True sb \(\mathcal{O} \cup\) all-shared sb
apply (induct sb)
apply clarsimp
apply (auto split: memref.splits dest: all-shared-acquired-in)
done
lemma release-all-shared-exchange:
\(\wedge \mathcal{R} \mathrm{S}^{\prime} \mathrm{S} . \forall \mathrm{a} \in\) all-shared sb. \(\left(\mathrm{a} \in \mathrm{S}^{\prime}\right)=(\mathrm{a} \in \mathrm{S}) \Longrightarrow\) release sb \(\mathrm{S}^{\prime} \mathcal{R}=\) release \(\mathrm{sb} \mathrm{S} \mathcal{R}\) proof (induct sb)
case Nil thus ?case by auto
next
case (Cons \(\times \mathrm{sb}\) )
```

```
show ?case
proof (cases x)
    case (Writesb volatile a' sop v A L R W)
    show ?thesis
    proof (cases volatile)
        case True
        note volatile=this
        from Cons.hyps [of (S'\cupR-L) (S\cupR-L) Map.empty] Cons.prems
        show ?thesis
            by (auto simp add: Write sb volatile)
    next
        case False with Cons Write sb show ?thesis by auto
    qed
next
    case Readsb with Cons show ?thesis by auto
next
    case Prog}\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ with Cons show ?thesis by auto
next
    case (Ghost sb A L R W)
    from augment-rels-shared-exchange [of R S S' }\mathcal{R}\mathrm{ ] Cons.prems
    have augment-rels S'R\mathcal{R}=\mathrm{ augment-rels S R R}
        by (auto simp add: Ghostsb)
    with Cons.hyps [of (S'\cupR-L) (S\cupR-L) augment-rels S R R}]\mathrm{ Cons.prems
    show ?thesis
        by (auto simp add: Ghostsb
    qed
qed
lemma release-append-Progss:
\S\mathcal{R}.(release (takeWhile (Not o is-volatile-Writesb) (sb @ [Prog
    (release (takeWhile (Not ○ is-volatile-Write sb) sb) S R )
by (induct sb) (auto split: memref.splits)
```


## A. 5 Simulation of Store Buffer Machine with History by Virtual Machine with Delayed Releases

theorem (in xvalid-program) concurrent-direct-steps-simulates-store-buffer-history-step:
assumes step-sb: $\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{t}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$
assumes valid-own: valid-ownership $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}$
assumes valid-sb-reads: valid-reads $\mathrm{m}_{\text {sb }} \mathrm{ts}_{\text {sb }}$
assumes valid-hist: valid-history program-step $\mathrm{ts}_{\mathrm{sb}}$
assumes valid-sharing: valid-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$
assumes tmps-distinct: tmps-distinct $\mathrm{ts}_{\mathrm{sb}}$
assumes valid-sops: valid-sops $\mathrm{ts}_{\mathrm{sb}}$
assumes valid-dd: valid-data-dependency $\mathrm{ts}_{\mathrm{sb}}$
assumes load-tmps-fresh: load-tmps-fresh $\mathrm{ts}_{\mathrm{sb}}$
assumes enough-flushs: enough-flushs $\mathrm{ts}_{\mathrm{sb}}$
assumes valid-program-history: valid-program-history $\mathrm{ts}_{\mathrm{sb}}$
assumes valid: valid $\mathrm{ts}_{\mathrm{sb}}$
assumes sim: $\left(\mathrm{t}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})$
assumes safe-reach: safe-reach-direct safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ )
shows valid-ownership $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ valid-reads $\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ valid-history program-step $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$
valid-sharing $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ valid-data-dependency $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$
valid-sops $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ load-tmps-fresh $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ enough-flushs $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$
valid-program-history $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$ valid $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge$

$$
\begin{gathered}
\left(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime} .(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow_{\mathrm{d}^{*}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge\right. \\
\left.\left(\mathrm{tss}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\right)
\end{gathered}
$$

## proof -

interpret direct-computation:
computation direct-memop-step empty-storebuffer-step program-step $\lambda$ p $p^{\prime}$ is sb. sb .
interpret sbh-computation:
computation sbh-memop-step flush-step program-step
$\lambda \mathrm{p}^{\prime}$ is $\mathrm{sb} . \mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime}\right.$ is].
interpret valid-ownership $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}$ by fact
interpret valid-reads $\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ by fact
interpret valid-history program-step $\mathrm{ts}_{\mathrm{sb}}$ by fact
interpret valid-sharing $\mathcal{S}_{\text {sb }}$ ts $_{\text {sb }}$ by fact
interpret tmps-distinct $\mathrm{ts}_{\mathrm{sb}}$ by fact
interpret valid-sops $\mathrm{ts}_{\mathrm{sb}}$ by fact
interpret valid-data-dependency $\mathrm{ts}_{\text {sb }}$ by fact
interpret load-tmps-fresh $\mathrm{ts}_{\text {sb }}$ by fact
interpret enough-flushs $\mathrm{ts}_{\text {sb }}$ by fact
interpret valid-program-history $\mathrm{ts}_{\mathrm{sb}}$ by fact
from valid-own valid-sharing
have valid-own-sharing: valid-ownership-and-sharing $\mathcal{S}_{\text {sb }}$ ts sb
by (simp add: valid-sharing-def valid-ownership-and-sharing-def)
then
interpret valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$.
from safe-reach-safe-refl [OF safe-reach]
have safe: safe-delayed ( $\mathrm{ts}, \mathrm{m}, \mathcal{S}$ ).
from step-sb
show ?thesis
proof (cases)
case (Memop i p $\mathrm{p}_{\mathrm{sb}}$ is $\mathrm{s}_{\mathrm{sb}} \vartheta_{\mathrm{sb}} \mathrm{sb} \mathcal{D}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathcal{R}_{\mathrm{sb}} \mathrm{is}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}$ )
then obtain
$\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}=\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}{ }^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}{ }^{\prime}, \mathcal{O}_{\mathrm{sb}}{ }^{\prime}, \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\right)\right]$ and
i-bound: $\mathrm{i}<$ length ts stb and
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)$ and
sbh-step: (is $\left.{ }_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}, \mathrm{m}_{\mathrm{sb}}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \rightarrow_{\mathrm{sbh}}$
$\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}{ }^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{D}_{\mathrm{sb}}{ }^{\prime}, \mathcal{O}_{\mathrm{sb}}{ }^{\prime}, \mathcal{R}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$
by auto

## from sim obtain

$\mathrm{m}: \mathrm{m}=$ flush-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}$ and
$\mathcal{S}: \mathcal{S}=$ share-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}$ and
leq: length $\mathrm{ts}_{\mathrm{sb}}=$ length ts and
ts-sim: $\forall \mathrm{i}<$ length ts $\mathrm{ts}_{\mathrm{sb}}$.
let $\left(\mathrm{p}, \mathrm{is}_{\mathrm{sb}}, \vartheta, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}$;
suspends $=$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb
in $\exists$ is $\mathcal{D}$. instrs suspends @ is st $=$ is @ prog-instrs suspends $\wedge$

```
\mathcal{D}
ts!i=
(hd-prog p suspends,
is,
\vartheta |}\mathrm{ (dom ७ - read-tmps suspends), (),
D}\mathrm{ ,
acquired True (takeWhile (Not o is-volatile-Write sb) sb) }\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mathrm{ ,
release (takeWhile (Not 0 is-volatile-Write sb) sb) (dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mathcal{R}
```

by cases blast
from i-bound leq have i-bound': $\mathrm{i}<$ length ts by auto
have split-sb: sb $=$ takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb @ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb
(is sb = ?take-sb@?drop-sb)
by simp
from ts-sim [rule-format, OF i-bound] ts $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ obtain suspends is $\mathcal{D}$ where
suspends: suspends $=$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb and
is-sim: instrs suspends @ $\mathrm{is}_{\mathrm{sb}}=$ is @ prog-instrs suspends and
$\mathcal{D}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}\right)$ and
ts-i: ts ! i =
(hd-prog $\mathrm{p}_{\mathrm{sb}}$ suspends, is, $\left.\vartheta_{\mathrm{sb}}\right|^{\circ}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}\right.$ - read-tmps suspends), ( $), \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
by (auto simp add: Let-def)
from sbh-step-preserves-valid [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i sbh-step valid]
have valid': valid $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
from $\mathcal{D}$ have $\mathcal{D}_{\mathrm{sb}}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}}$ ?drop-sb $\left.\neq\{ \}\right)$
apply -
apply (case-tac outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
apply (fastforce simp add: outstanding-refs-conv dest: set-dropWhileD)
apply (clarsimp)
apply (drule outstanding-refs-non-empty-dropWhile)
apply blast
done
let ? $\mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{i} \mathrm{s}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},(), \mathcal{D}_{\mathrm{sb}}\right.\right.$, acquired True $\mathrm{sb} \mathcal{O}_{\mathrm{sb}}$,
release $\left.\left.\mathrm{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right]$
have i-bound-ts': i < length ?ts ${ }^{\prime}$
using i-bound ${ }^{\prime}$
by auto
hence ts' i : ?ts' $\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.$,
$\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$, release sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$ )
by simp
from local.sharing-consis-axioms
have sharing-consis-ts $\mathrm{s}_{\mathrm{sb}}$ : sharing-consis $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$.
from sharing-consis [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}$ - ]
have sharing-consis-sb: sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}$ sb.
from sharing-consistent-weak-sharing-consistent [OF this]
have weak-consis-sb: weak-sharing-consistent $\mathcal{O}_{\text {sb }} \mathrm{sb}$.
from this weak-sharing-consistent-append [of $\mathcal{O}_{\text {sb }}$ ? take-sb ?drop-sb]
have weak-consis-drop:weak-sharing-consistent (acquired True ?take-sb $\mathcal{O}_{\text {sb }}$ ) ?drop-sb by auto
from local.ownership-distinct-axioms
have ownership-distinct- $\mathrm{ts}_{\mathrm{sb}}$ : ownership-distinct $\mathrm{ts}_{\mathrm{sb}}$ -
have steps-flush-sb: $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}\left(? \mathrm{ts}^{\prime}\right.$, flush ?drop-sb m, share ?drop-sb $\left.\mathcal{S}\right)$
proof -
from valid-reads [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have reads-consis: reads-consistent False $\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}$.
from reads-consistent-drop-volatile-writes-no-volatile-reads [OF this]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{\text {sb }}$ ? drop-sb $=\{ \}$.
from valid-program-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - i ]
have causal-program-history is $_{\mathrm{sb}} \mathrm{sb}$.
then have cph: causal-program-history is stb ?drop-sb
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb] )
apply (simp)
done
from valid-last-prog [OF i-bound $\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}} \mathrm{i}}$ ] have last-prog: last-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}$.
then
have lp : last-prog $\mathrm{p}_{\mathrm{sb}}$ ? drop-sb $=\mathrm{p}_{\mathrm{sb}}$
apply -
apply (rule last-prog-same-append [where sb=?take-sb])
apply simp
done
from reads-consistent-flush-all-until-volatile-write [OF valid-own-sharing i-bound $\mathrm{ts}_{\mathrm{sb}}$-i reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$ ) m ?drop-sb by (simp add: m)
from valid-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have h-consis: history-consistent $\vartheta_{\mathrm{sb}}$ (hd-prog $\mathrm{p}_{\mathrm{sb}} \quad$ (?take-sb@?drop-sb)) (?take-sb@?drop-sb)
by ( $\operatorname{simp}$ )
have last-prog-hd-prog: last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}$ ) ?take-sb $=$ (hd-prog $\mathrm{p}_{\mathrm{sb}}$ ?drop-sb) proof -
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] last-prog
have last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}}$ ?drop-sb) ?take-sb $=$ hd-prog $\mathrm{p}_{\mathrm{sb}}$ ?drop-sb
by (simp)
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}}(?$ take-sb @ ?drop-sb)) ?take-sb =
last-prog (hd-prog $\mathrm{p}_{\mathrm{sb}}$ ?drop-sb) ?take-sb
by (rule last-prog-hd-prog-append)
ultimately show ? thesis
by (simp)
qed
from valid-write-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}$ ] $]$
have $\forall$ sop $\in$ write-sops (?take-sb@?drop-sb). valid-sop sop
by (simp)
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops ?drop-sb. valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have distinct-read-tmps (?take-sb@?drop-sb)
by (simp)
then obtain
read-tmps-take-drop: read-tmps ?take-sb $\cap$ read-tmps ?drop-sb $=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps ?drop-sb
by (simp only: distinct-read-tmps-append)
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis]
have hist-consis': history-consistent $\vartheta_{\mathrm{sb}}$ (hd-prog $\mathrm{p}_{\mathrm{sb}}$ ?drop-sb) ?drop-sb
by (simp add: last-prog-hd-prog)
have rel-eq: release ?drop-sb $(\operatorname{dom} \mathcal{S})$ (release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)=$ release $\mathrm{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$
proof -
from release-append [of ?take-sb ?drop-sb]
have release $\mathrm{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}=$
release ?drop-sb (dom (share ?take-sb $\left.\mathcal{S}_{\mathrm{sb}}\right)$ ) (release ?take-sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$ )
by $\operatorname{simp}$
also
have dist: ownership-distinct $\mathrm{ts}_{\mathrm{sb}}$ by fact
have consis: sharing-consis $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}$ by fact
have release ?drop-sb (dom (share ?take-sb $\left.\mathcal{S}_{\mathrm{sb}}\right)$ ) (release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$ $=$
release ?drop-sb (dom $\mathcal{S}$ ) (release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
apply (simp only: $\mathcal{S}$ )
apply (rule release-shared-exchange-weak [rule-format, OF - weak-consis-drop])
apply (rule share-all-until-volatile-write-thread-local [OF dist consis i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$, symmetric])
using acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\text {sb }}$ ] all-acquired-append [of ?take-sb ?drop-sb]
by auto
finally
show ?thesis by simp
qed
from flush-store-buffer [OF i-bound ${ }^{\prime}$ is-sim [simplified suspends]
cph ts-i [simplified suspends] refl lp reads-consis-m hist-consis ${ }^{\prime}$
valid-sops-drop distinct-read-tmps-drop no-volatile-Read sb $_{\text {-volatile-reads-consistent }}$ [OF no-vol-read], of $\mathcal{S}$ ]
show ?thesis by (simp add: acquired-take-drop [where pending-write=True, simplified] $\mathcal{D}_{\text {sb }}$ rel-eq)
qed
from safe-reach-safe-rtrancl [OF safe-reach steps-flush-sb]
have safe-ts': safe-delayed (?ts', flush ?drop-sb m, share ?drop-sb $\mathcal{S}$ ).
from safe-delayedE [OF safe-ts' i-bound-ts' ts ${ }^{\prime}$-i]
have safe-memop-flush-sb: map owned ?ts' ${ }^{\prime}$ map released ? $\mathrm{ts}^{\prime}$, $\mathrm{i} \vdash$
(is $\mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}$, flush ?drop-sb m, $\mathcal{D}_{\mathrm{sb}}$, acquired True $\operatorname{sb} \mathcal{O}_{\mathrm{sb}}$,
share ?drop-sb $\mathcal{S}) \sqrt{ }$.
from acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$
have acquired-take-sb: acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}} \subseteq \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired ?take-sb .

```
    from sbh-step
    show ?thesis
    proof (cases)
        case (SBHReadBuffered a v volatile t)
        then obtain
```

$\mathrm{is}_{\mathrm{sb}}$ : is $\mathrm{s}_{\mathrm{sb}}=$ Read volatile a $\mathrm{t} \# \mathrm{is}_{\mathrm{sb}}{ }^{\prime}$ and
$\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}$ and
$\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \mathcal{D}_{\mathrm{sb}}{ }^{\prime}=\mathcal{D}_{\mathrm{sb}}$ and
$\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v})$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\mathrm{t} v\right]$ and
$\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}$ and
$\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}$ and
$\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}$ and
buf-v: buffered-val sb $\mathrm{a}=$ Some v
by auto
from safe-memop-flush-sb [simplified is ${ }_{\text {sb }}$ ]
obtain access-cond': a $\in$ acquired True sb $\mathcal{O}_{\text {sb }} \vee$
a $\in$ read-only (share ?drop-sb $\mathcal{S}$ ) $\vee$
(volatile $\wedge \mathrm{a} \in \operatorname{dom}$ (share ?drop-sb $\mathcal{S}$ )) and
volatile-clean: volatile $\longrightarrow \neg \mathcal{D}_{\text {sb }}$ and
rels-cond: $\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$ released (ts! j$)$ a $\neq$ Some False and
rels-nv-cond: $\neg$ volatile $\longrightarrow(\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin$ dom (released (ts! j$))$ )
by cases auto
from clean-no-outstanding-volatile-Write ${ }_{\text {sb }}\left[\mathrm{OF}\right.$ i-bound $\left.\mathrm{ts}_{s b}-\mathrm{i}\right]$ volatile-clean have volatile-cond: volatile $\longrightarrow$ outstanding-refs is-volatile-Write $_{\text {sb }} \mathrm{sb}=\{ \}$ by auto
from buffered-val-witness [OF buf-v] obtain volatile' $\operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
witness: Write $_{\text {sb }}$ volatile ${ }^{\prime}$ a sop' $\mathrm{v}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \in \operatorname{set} \mathrm{sb}$
by auto

## \{

fix j $\mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}$
assume j-bound: $\mathrm{j}<$ length $^{\mathrm{ts}} \mathrm{s}_{\mathrm{sb}}$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume jth: $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume non-vol: $\neg$ volatile
have a $\notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof
assume $\mathrm{a}-\mathrm{j}: \mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
let ? take-sb ${ }_{j}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ? drop-sb ${ }_{j}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)$
from ts-sim [rule-format, OF j-bound] jth
obtain suspends $\mathrm{j}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}}$ where
suspends $_{\mathrm{j}}$ : suspends ${ }_{\mathrm{j}}=$ ? $\mathrm{ddrop}^{2}-\mathrm{sb}_{\mathrm{j}}$ and
$\mathrm{is}_{\mathrm{j}}$ : instrs suspends ${ }_{\mathrm{j}}$ @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{is}_{\mathrm{j}} @$ prog-instrs suspends $\mathrm{j}_{\mathrm{j}}$ and
$\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)$ and
$\mathrm{ts}_{\mathrm{j}}: \mathrm{ts}!\mathrm{j}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.$ suspends $\mathrm{s}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.$ dom $\vartheta_{\text {sbj }}$ - read-tmps suspends $\left.{ }\right),()$,
$\mathcal{D}_{\mathrm{j}}$, acquired True ?take-sb $\mathrm{O}_{\mathrm{j}}$, release ? take-sb $\left._{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
by (auto simp add: Let-def)
from a-j ownership-distinct [ OF i-bound j-bound neq-i-j $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{jth}$ ]
have a-notin-sb: a $\notin \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired sb by auto
with acquired-all-acquired [of True sb $\mathcal{O}_{\text {sb }}$ ]
have a-not-acq: a $\notin$ acquired True sb $\mathcal{O}_{\text {sb }}$ by blast
with access-cond' non-vol
have a-ro: a $\in$ read-only (share ?drop-sb $\mathcal{S}$ )
by auto
from read-only-share-unowned-in [OF weak-consis-drop a-ro] a-notin-sb acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\text {sb }}$ ] all-acquired-append [of ?take-sb ?drop-sb]
have a-ro-shared: a $\in$ read-only $\mathcal{S}$ by auto
from rels-nv-cond [rule-format, OF non-vol j-bound [simplified leq] neq-i-j] ts ${ }_{j}$

```
have a \(\notin \operatorname{dom}\) (release ?take-sb \(\left._{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}\right)\)
by auto
with dom-release-takeWhile \(\left[\right.\) of \(\operatorname{sb}_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}\) ]
obtain
a-rels \(\mathrm{j}_{\mathrm{j}}\) : \(\notin \operatorname{dom} \mathcal{R}_{\mathrm{j}}\) and
a-shared \({ }_{j}\) : a \(\notin\) all-shared ?take-sb \(_{j}\)
by auto
```

have a $\notin \bigcup\left(\left(\lambda(-,-,-\right.\right.$, sb,,,---$)$. all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ )
sb)) ${ }^{\prime}$

```
            set \(\mathrm{ts}_{\mathrm{sb}}\) )
proof -
    \{
        fix k \(\mathrm{p}_{\mathrm{k}}\) is \(_{\mathrm{k}} \vartheta_{\mathrm{k}} \operatorname{sb}_{\mathrm{k}} \mathcal{D}_{\mathrm{k}} \mathcal{O}_{\mathrm{k}} \mathcal{R}_{\mathrm{k}}\)
        assume k -bound: \(\mathrm{k}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
        assume ts-k: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{k}=\left(\mathrm{p}_{\mathrm{k}}, \mathrm{is}_{\mathrm{k}}, \vartheta_{\mathrm{k}}, \mathrm{sb}_{\mathrm{k}}, \mathcal{D}_{\mathrm{k}}, \mathcal{O}_{\mathrm{k}}, \mathcal{R}_{\mathrm{k}}\right)\)
        assume a-in: \(a \in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{k}}\) )
        have False
        proof (cases \(\mathrm{k}=\mathrm{j}\) )
            case True with a-shared \(j_{j}\) jth ts-k a-in show False by auto
        next
            case False
            from ownership-distinct [OF j-bound k-bound False [symmetric] jth ts-k] a-j
            have a \(\notin\left(\mathcal{O}_{\mathrm{k}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{k}}\right)\) by auto
        with all-shared-acquired-or-owned [OF sharing-consis [OF k-bound ts-k]] a-in
            show False
            using all-acquired-append [of takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{k}}\)
                dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \(\mathrm{sb}_{\mathrm{k}}\) ]
                all-shared-append [of takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(b_{k}\)
                dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \({ }_{\mathrm{k}}\) ] by auto
        qed
    \}
    thus ?thesis by (fastforce simp add: in-set-conv-nth)
qed
with a-ro-shared
    read-only-shared-all-until-volatile-write-subset \({ }^{\prime}\) [of \(\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}\) ]
have a-ro-shared \({ }_{\mathrm{sb}}: \mathrm{a} \in\) read-only \(\mathcal{S}_{\mathrm{sb}}\)
    by (auto simp add: \(\mathcal{S}\) )
with read-only-unowned [OF j-bound jth]
have a-notin-owns-j: a \(\notin \mathcal{O}_{j}\)
    by auto
```

have own-dist: ownership-distinct $\mathrm{ts}_{\text {sb }}$ by fact
have share-consis: sharing-consis $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ by fact
from sharing-consistent-share-all-until-volatile-write [OF own-dist share-consis i-bound $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]$
have consis': sharing-consistent $\mathcal{S}$ (acquired True ?take-sb $\mathcal{O}_{\text {sb }}$ ) ?drop-sb
by ( $\operatorname{simp}$ add: $\mathcal{S}$ )
from share-all-until-volatile-write-thread-local [OF own-dist share-consis j-bound jth a-j] a-ro-shared
have a-ro-take: a $\in$ read-only (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )
by (auto simp add: domIff $\mathcal{S}$ read-only-def)
from sharing-consis [OF j-bound jth]
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
from sharing-consistent-weak-sharing-consistent [ $\begin{array}{ll}\mathrm{OF} & \text { this] }\end{array}$
weak-sharing-consistent-append [of $\mathcal{O}_{\mathrm{j}}$ ?take-sb $\mathrm{T}_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have weak-consis-drop:weak-sharing-consistent $\mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ by auto
from read-only-share-acquired-all-shared [OF this read-only-unowned [OF j-bound jth] a-ro-take ] a-notin-owns-j a-shared ${ }_{j}$
have a $\notin$ all-acquired ?take-sb $_{j}$
by auto
with a-j a-notin-owns-j
have a-drop: $\mathrm{a} \in$ all-acquired ?drop-sb ${ }_{j}$
using all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
by simp
from i-bound j-bound leq have j-bound-ts': $\mathrm{j}<$ length ?ts ${ }^{\prime}$
by auto
note conflict-drop $=$ a-drop [simplified suspends ${ }_{j}$ [symmetric]]
from split-all-acquired-in [OF conflict-drop]
show False
proof
assume $\exists$ sop $\mathrm{a}^{\prime}$ v ys zs A L R W.
(suspends ${ }_{j}=$ ys @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v A L R W\# zs) $\wedge a \in A$
then
obtain $\mathrm{a}^{\prime}$ sop' $\mathrm{v}^{\prime}$ ys zs $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
split-suspends $\mathrm{j}_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Write ${ }_{\mathrm{sb}}$ True $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$
(is suspends $\mathrm{s}_{\mathrm{j}}=$ ?suspends) and
$a-A^{\prime}: a \in A^{\prime}$
by blast
from sharing-consis [OF j-bound jth]
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
then have $A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}$
by (simp add: sharing-consistent-append [of - ? ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$, simplified]
suspends ${ }_{j}$ [symmetric] split-suspends ${ }_{j}$ sharing-consistent-append)
from valid-program-history [OF j-bound jth]
have causal-program-history is sbj $\mathrm{sb}_{\mathrm{j}}$.
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb] ] )
apply (simp only: split-suspends [symmetric] suspendsj)
apply (simp add: split-suspends ${ }_{\mathrm{j}}$ )

## done

from $t_{\mathrm{j}}$ neq-i-j j-bound
have ts' ${ }^{\prime} \mathrm{j}$ : ? $\mathrm{ts}^{\prime}!\mathrm{j}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$, is ${ }_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.$ read-tmps suspends $\left.{ }_{j}\right),()$,
$\mathcal{D}_{\mathrm{j}}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$, release ? take-sb $\mathrm{t}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}$ )
by auto
from valid-last-prog [OF j-bound jth] have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$.
then
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$
apply -
apply (rule last-prog-same-append [where sb=?take-sb ${ }_{\mathrm{j}}$ ])
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
from valid-reads [OF j-bound jth]
have reads-consis-j: reads-consistent False $\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ • j-bound jth reads-consis-j]
have reads-consis- $\mathrm{m}-\mathrm{j}$ : reads-consistent True (acquired True ?take-sb $\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends $\mathrm{m}_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $\mathrm{m}_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j ts stb $_{\text {- }}$ i jth]
have outstanding-refs is-Write ${ }_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by ( $\operatorname{simp}$ add: suspends ${ }_{\mathrm{j}}$ )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) (flush ?drop-sb m) suspends ${ }_{\mathrm{j}}$.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) (ys@[Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)$
by (simp add: split-suspends ${ }_{j}$ reads-consistent-append)
from valid-write-sops [OF j-bound jth]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ? take-sb ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops (ys@[Write ${ }_{\text {sb }}$ True $\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound jth]
have distinct-read-tmps (?take-sb $@_{j}$ @uspends ${ }_{j}$ )
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb ${ }_{j} \cap$ read-tmps suspends ${ }_{j}=\{ \}$ and distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound jth]
have h-consis:

apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ?take-sb ${ }_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$
proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $_{\mathrm{j}} @$ ? drop-sb $\left.\mathrm{b}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by $\operatorname{simp}$
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb ${ }_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )

## moreover

have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{t}_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by ( simp add: split-suspends $\mathrm{s}_{\mathrm{j}}[$ symmetric $]$ suspends $_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) suspends $\mathrm{s}_{\mathrm{j}}$ by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile- $\operatorname{Read}_{\text {sb }}$ $\left(y s @\left[W_{r i t e}^{s b}\right.\right.$ True $a^{\prime}$ sop $\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}$
by (auto simp add: outstanding-refs-append suspends ${ }_{j}$ [symmetric]
split-suspends ${ }_{j}$ )

```
have acq-simp:
    acquired True (ys @ \(\left[W^{2} i t e_{\text {sb }}\right.\) True \(\mathrm{a}^{\prime}\) sop \(\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) )
            (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) \(=\)
            acquired True ys (acquired True ?take-sb \(\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
    by (simp add: acquired-append)
```

from flush-store-buffer-append [where $s b=y s @\left[W_{r i t e}^{s b}\right.$ True $a^{\prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ and $\mathrm{sb}^{\prime}=\mathrm{zs}$, simplified,

OF j-bound-ts ${ }^{\prime}$ is $_{\mathrm{j}}$ [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts' ${ }^{\prime}$ j [simplified split-suspends ${ }_{j}$ ] refl lp [simplified split-suspends ${ }_{j}$ ] reads-consis-ys
hist-consis ${ }^{\prime}$ [simplified split-suspends ${ }_{j}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends ${ }_{j}$ ]
no-volatile-Read sb -volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}]$
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$ : instrs $\mathrm{zs} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts' , flush ?drop-sb m, share ?drop-sb $\mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}$
(?ts' ${ }^{\prime} \mathrm{j}:=$ (last-prog
(hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\left(\mathrm{ys} @\left[W_{r i t e}\right.\right.$ sb True $\mathrm{a}^{\prime}$ $\left.\operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \mathrm{J}\right)$,
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\operatorname{dom} \vartheta_{\text {sbj }}\right.$ - read-tmps zs),
(), True, acquired True ys (acquired True ?take-sb $\left.\left.\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]$,
flush (ys@[Writesb True $\mathrm{a}^{\prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ) (flush ?drop-sb m),
share (ys@[Write ${ }_{\text {sb }}$ True $\left.\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)($ share ?drop-sb $\mathcal{S})$ )
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}$, ?shared-ys) $)$
by (auto simp add: acquired-append outstanding-refs-append)
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys! $i=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.$,
$\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$, release sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
by simp
note conflict-computation $=$ rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is ${ }_{\text {sb }}$ ] non-vol a-not-acq
have $a \in$ read-only (share (ys@[Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}$ ) (share ? drop-sb S))
apply cases
apply (auto simp add: Let-def is sbb )
done
with a-A ${ }^{\prime}$
show False
by (simp add: share-append in-read-only-convs)
next
assume $\exists \mathrm{ALR} \mathrm{W}$ ys zs. suspends $\mathrm{f}_{\mathrm{j}}=$ ys @ Ghost $_{\text {sb }} \mathrm{A} L \mathrm{R} W \#$ zs $\wedge \mathrm{a} \in \mathrm{A}$
then
obtain $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ ys zs where
split-suspends $\mathrm{j}_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Ghost $\mathrm{sb}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$
(is suspends ${ }_{\mathrm{j}}=$ ?suspends) and
$a-A^{\prime}: a \in A^{\prime}$
by blast

```
from valid-program-history [OF j-bound jth]
have causal-program-history is sbj \(\mathrm{sb}_{j}\).
then have cph: causal-program-history is \(_{\text {sbj }}\) ?suspends
    apply -
    apply (rule causal-program-history-suffix [where sb=?take-sb \({ }_{j}\) ] )
    apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
    apply (simp add: split-suspends \({ }_{\mathrm{j}}\) )
    done
from \(t s_{j}\) neq-i-j \(j\)-bound
have ts' \(\mathrm{j}: ~ ?\) ?ts \(!\mathrm{j}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{j}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
    \(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps suspends \(\left.{ }_{\mathrm{j}}\right),()\),
    \(\mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\left.\mathrm{D}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
    by auto
from valid-last-prog [OF j-bound jth] have last-prog: last-prog \(p_{j} s b_{j}=p_{j}\).
then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\)
    apply -
    apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \(_{\mathrm{j}}\) ])
    apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
    apply simp
    done
from valid-reads [OF j-bound jth]
have reads-consis-j: reads-consistent False \(\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}}\) sb \({ }_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}} /\) j-bound jth reads-consis-j]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{s}_{\mathrm{j}}\) by (simp add: m suspends \({ }_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j \(\mathrm{ts}_{\mathbf{s b}^{\boldsymbol{s}}}\)-i
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \(_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by (simp add: suspends \({ }_{\mathrm{j}}\) )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
(flush ?drop-sb m) suspendsj.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) )
(flush ?drop-sb m) (ys@[Ghostsb \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) )
by (simp add: split-suspends \({ }_{j}\) reads-consistent-append)
from valid-write-sops [OF j-bound jth]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
```

 jth]

```
    apply (simp only: write-sops-append)
    apply auto
    done
```

    from read-tmps-distinct [OF j-bound jth]
    have distinct-read-tmps (?take-sb $@_{j}$ suspends $_{\mathrm{j}}$ )
by ( simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb ${ }_{\mathrm{j}} \cap$ read-tmps suspends ${ }_{\mathrm{j}}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspends ${ }_{j}$
apply (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound jth]
have $h$-consis:
history-consistent $\vartheta_{s b j}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$ ) ?take-sb $\mathrm{H}_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\left._{\mathrm{j}}\right)$
proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{b}_{\mathrm{j}} @$ ?drop-sb $\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by $\operatorname{simp}$
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ? take-sb $\mathrm{t}_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends ${ }_{\mathrm{j}}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) suspends $\mathrm{j}_{\mathrm{j}}$
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{\text {sb }}$
$\left(y_{s} @\left[\right.\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}$
by (auto simp add: outstanding-refs-append suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric]
split-suspends ${ }_{j}$ )
have acq-simp:
acquired True (ys @ [Ghost sb $\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
(acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) $=$
acquired True ys (acquired True ?take-sb $\left.\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
by (simp add: acquired-append)
from flush-store-buffer-append [where $\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Ghost}_{s b} \mathrm{~A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]$ and $\mathrm{sb}{ }^{\prime}=\mathrm{zs}$, simplified,

OF j-bound-ts' is $_{\mathrm{j}}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] cph [simplified suspends ${ }_{\mathrm{j}}$ ]
ts'-j [simplified split-suspendsj] refl lp [simplified split-suspendsj] reads-consis-ys
hist-consis' ${ }^{\prime}$ simplified split-suspends ${ }_{\mathrm{j}}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ]
no-volatile-Read ${ }_{\text {sb }}$-volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}$ ]
obtain is ${ }_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}$ ': instrs zs @ is $\mathrm{sbj}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts ${ }^{\prime}$, flush ?drop-sb m, share ?drop-sb $\left.\mathcal{S}\right) \Rightarrow \mathrm{d}^{*}$
(?ts' $[\mathrm{j}:=($ last-prog

is ${ }_{j}{ }^{\prime}$,
$\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps zs $)$,
(),
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}\left(\right.$ ys $@\left[\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\right) \neq\{ \}$,
acquired True ys (acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$ ) $\left.\cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)$ ],
flush (ys@[Ghostsb $\left.A^{\prime} L^{\prime} R^{\prime} W^{\prime}\right]$ ) (flush ?drop-sb m),

(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}$, ?shared-ys))
by (auto simp add: acquired-append)
from i-bound' have i-bound-ys: i < length ?ts-ys by auto
from i-bound ${ }^{\prime}$ neq-i-j
have ts-ys-i: ?ts-ys!i $=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.$,
$\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$, release sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
by simp
note conflict-computation $=$ rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is sts ] non-vol a-not-acq

apply cases
apply (auto simp add: Let-def is $_{\text {sb }}$ )
done
with $a-A^{\prime}$
show False
by (simp add: share-append in-read-only-convs)
qed
qed

## \{

assume a-in: a $\in$ read-only (share (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb) $\mathcal{S}$ )
assume nv: $\neg$ volatile
have a $\in$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
proof (cases a $\in \mathcal{O}_{\text {sb }} \cup$ all-acquired sb)
case True
from share-all-until-volatile-write-thread-local' $\left[\right.$ OF ownership-distinct-ts ${ }_{\text {sb }}$ sharing-consis- $\mathrm{ts}_{\mathrm{sb}} \mathrm{i}$-bound $\mathrm{ts}_{\mathrm{sb}}$ - i True] True a -in
show ?thesis
by ( $\operatorname{simp}$ add: $\mathcal{S}$ read-only-def)
next
case False
from read-only-share-unowned [OF weak-consis-drop - a-in] False acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\text {sb }}$ ] all-acquired-append [of ?take-sb ?drop-sb]
have a-ro-shared: a $\in$ read-only $\mathcal{S}$
by auto
have a $\notin \bigcup((\lambda(-,-,-$, sb,,,---$)$.
all-shared ( takeWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \mathrm{sb}\right)$ ) ' set $\mathrm{ts}_{\text {sb }}$ )
proof -
\{
fix k $\mathrm{p}_{\mathrm{k}}$ is $_{\mathrm{k}} \vartheta_{\mathrm{k}} \mathrm{sb}_{\mathrm{k}} \mathcal{D}_{\mathrm{k}} \mathcal{O}_{\mathrm{k}} \mathcal{R}_{\mathrm{k}}$
assume k-bound: $\mathrm{k}<$ length $\mathrm{ts}_{\mathrm{sb}}$
assume ts-k: $\mathrm{ts}_{\mathrm{sb}}!\mathrm{k}=\left(\mathrm{p}_{\mathrm{k}}, \mathrm{is}_{\mathrm{k}}, \vartheta_{\mathrm{k}}, \mathrm{sb}_{\mathrm{k}}, \mathcal{D}_{\mathrm{k}}, \mathcal{O}_{\mathrm{k}}, \mathcal{R}_{\mathrm{k}}\right)$
assume a-in: $\mathrm{a} \in$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{k}}$ ) have False proof (cases $\mathrm{k}=\mathrm{i}$ )
case True with False $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ts- k a-in
all-shared-acquired-or-owned [OF sharing-consis [OF k-bound ts-k]]
all-shared-append [of takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{k}}$
dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }_{k}$ ] show False by auto
next
case False
from rels-nv-cond [rule-format, OF nv k-bound [simplified leq] False [symmetric]
]
ts-sim [rule-format, OF k-bound] ts-k
have a $\notin$ dom (release (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{k}}$ ) (dom $\left(\mathcal{S}_{\mathrm{sb}}\right)$ )
$\mathcal{R}_{\mathrm{k}}$ )
by (auto simp add: Let-def)
with dom-release-takeWhile $\left[\right.$ of $\left.\operatorname{sb}_{\mathrm{k}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{k}}\right]$
obtain
a-rels $\mathrm{j}_{\mathrm{j}}$ : $\notin \operatorname{dom} \mathcal{R}_{\mathrm{k}}$ and
a-shared ${ }_{j}$ : a $\notin$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb $_{\mathrm{k}}$ )
by auto
with False a-in show ?thesis
by auto

```
                    qed
                    }
                thus ?thesis by (fastforce simp add: in-set-conv-nth)
                qed
                with read-only-shared-all-until-volatile-write-subset' [of ts ssb
                have a }\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{\textrm{sb}}{
                by (auto simp add: S
        from read-only-share-unowned' [OF weak-consis-sb read-only-unowned [OF i-bound
ts sb-i] False this]
show ?thesis .
qed
} note non-vol-ro-reduction = this
    have valid-own': valid-ownership }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
    proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (cases volatile)
    case False
    from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts tsb-i]
    have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mathrm{ sb
    then
    have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}(\textrm{sb@}[\mp@subsup{\operatorname{Read}}{\textrm{sb}}{}\mathrm{ False a t v])
        using access-cond' False non-vol-ro-reduction
        by (auto simp add: non-volatile-owned-or-read-only-append)
    from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
    show ?thesis by (auto simp add: False ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mathrm{ )
next
    case True
    from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts tsb-i]
    have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\textrm{sb}
    then
    have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}(\textrm{sb}@[\mp@subsup{R}{ead}{sb
        using True
        by (simp add: non-volatile-owned-or-read-only-append)
    from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
    show ?thesis by (auto simp add: True ts }\mp@subsup{\textrm{tsb}}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}
qed
        next
show outstanding-volatile-writes-unowned-by-others ts }\mp@subsup{\textrm{s}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime
proof -
    have out: outstanding-refs is-volatile-Write e
                outstanding-refs is-volatile-Write sb sb
        by (auto simp add: outstanding-refs-append)
    have all-acquired (sb @ [Readsb volatile a t v]) \subseteq all-acquired sb
        by (auto simp add: all-acquired-append)
    from outstanding-volatile-writes-unowned-by-others-store-buffer
    [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ out this]
    show ?thesis by (simp add: ts sb ' sb' }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}
```

```
qed
```

    next
    show read-only-reads-unowned $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof (cases volatile)
case True
have r: read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) (sb @
$\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at v])) $\mathcal{O}_{\mathrm{sb}}$ )
(dropWhile (Not o is-volatile-Write ${ }_{\mathbf{s b}}$ ) (sb @ $\left[\operatorname{Read}_{\text {sb }}\right.$ volatile a t v]))
$\subseteq$ read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) sb)
$\mathcal{O}_{\mathrm{sb}}$ )
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (case-tac outstanding-refs (is-volatile-Write ${ }_{\text {sb }}$ ) sb $=\{ \}$ )
apply (simp-all add: outstanding-vol-write-take-drop-appends
acquired-append read-only-reads-append True)
done
have $\mathcal{O}_{\text {sb }} \cup$ all-acquired $\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.]\right) \subseteq \mathcal{O}_{\text {sb }} \cup$ all-acquired sb
by (simp add: all-acquired-append)
from read-only-reads-unowned-nth-update [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{r}$ this]
show ?thesis
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
next
case False
show ?thesis
proof (unfold-locales)
fix n m
fix $\mathrm{p}_{\mathrm{n}} \mathrm{is}_{\mathrm{n}} \mathcal{O}_{\mathrm{n}} \mathcal{R}_{\mathrm{n}} \mathcal{D}_{\mathrm{n}} \vartheta_{\mathrm{n}} \mathrm{sb}_{\mathrm{n}} \mathrm{p}_{\mathrm{m}}$ is $_{\mathrm{m}} \mathcal{O}_{\mathrm{m}} \mathcal{R}_{\mathrm{m}} \mathcal{D}_{\mathrm{m}} \vartheta_{\mathrm{m}} \mathrm{sb}_{\mathrm{m}}$
assume n-bound: $\mathrm{n}<$ length $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
and m -bound: $\mathrm{m}<$ length $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
and neq- $\mathrm{n}-\mathrm{m}: \mathrm{n} \neq \mathrm{m}$
and nth: $\mathrm{ts}_{\mathrm{sb}}$ ! $\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)$
and mth: $\mathrm{ts}_{\mathrm{sb}}$ ! $\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)$
from n -bound have n -bound': $\mathrm{n}<$ length $\mathrm{ts}_{\mathrm{sb}}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\text {}}$ )
from m -bound have m -bound': $\mathrm{m}<$ length $\mathrm{ts}_{\mathrm{sb}}$ by (simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
have acq-eq: $\left(\mathcal{O}_{\mathrm{sb}}{ }^{\prime} \cup\right.$ all-acquired $\left.\mathrm{sb}^{\prime}\right)=\left(\mathcal{O}_{\mathrm{sb}} \cup\right.$ all-acquired sb $)$
by (simp add: all-acquired-append $\mathrm{sb}^{\prime} \mathcal{O}_{\text {sb }}{ }^{\prime}$ )
show $\left(\mathcal{O}_{\mathrm{m}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{m}}\right) \cap$
read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{n}}$ ) $\mathcal{O}_{\mathrm{n}}$ )
$\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{n}}\right)=$
\{\}
proof (cases m=i)
case True
with neq-n-m have neq-n-i: $n \neq \mathrm{i}$
by auto
with n-bound nth i-bound have nth': $\mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ )
note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth ${ }^{\prime}$ ts $_{\mathbf{s b}}-\mathrm{i}$ ]
moreover
note acq-eq
ultimately show ? thesis
using True ts $\mathrm{sb}_{\mathrm{sb}}-\mathrm{i}$ nth mth n-bound' m -bound ${ }^{\prime}$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
next
case False
note neq-m-i $=$ this
with m-bound mth i-bound have $\mathrm{mth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}$ )
show ?thesis
proof (cases $\mathrm{n}=\mathrm{i}$ )
case True
note read-only-reads-unowned [OF i-bound m-bound' neq-m-i [symmetric] $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{mth}$ ]
moreover
note acq-eq
moreover
note non-volatile-unowned-others [OF m-bound' neq-m-i [symmetric] mth ${ }^{\text {' }}$
ultimately show ?thesis
using True $\mathrm{ts}_{\mathrm{sb}}$-i nth mth n -bound' $\mathrm{m}^{\text {-bound }}{ }^{\prime}$ neq-m-i
apply (case-tac outstanding-refs (is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}=\{ \}$ )
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends acquired-append read-only-reads-append $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\right)+$
done next
case False
with n-bound nth i-bound have $\mathrm{nth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \operatorname{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)$ by (auto simp add: $\mathrm{ts}_{\text {sb }}$ ')
from read-only-reads-unowned [OF n-bound'm-bound' neq-n-m nth' mth'] False neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
qed
next
show ownership-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
have all-acquired $\left(\mathrm{sb} @\left[\operatorname{Read}_{\text {sb }}\right.\right.$ volatile a t v $]$ ) $\subseteq$ all-acquired sb by (auto simp add: all-acquired-append)
from ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ this]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}$ )
qed
qed
have valid-hist ${ }^{\prime}$ : valid-history program-step $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from valid-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ ]
have hcons: history-consistent $\vartheta_{\mathrm{sb}}$ (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}$ ) sb.
from load-tmps-read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have t-notin-reads: $\mathrm{t} \notin$ read-tmps sb by (auto simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from load-tmps-write-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}$ - ]
have t-notin-writes: $\mathrm{t} \notin \bigcup$ (fst ' write-sops sb) by (auto simp add: $\mathrm{is}_{\mathrm{sb}}$ )
from valid-write-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i} \text { ] }] ~}^{\text {[ }}$
have valid-sops: $\forall$ sop $\in$ write-sops sb. valid-sop sop by auto
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}} \text { - }}$ ]
have t-fresh: $\mathrm{t} \notin \operatorname{dom} \vartheta_{\text {sb }}$
using is ${ }_{\text {sb }}$
by simp
have history-consistent $\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v})\right)$
$\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile atv $\left.\left.\mathbf{t}\right]\right)\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.]\right)$
using t-notin-writes valid-sops t-fresh hcons
valid-implies-valid-prog-hd [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ valid]
apply -
apply (rule history-consistent-appendI)
apply (auto simp add: hd-prog-append-Read ${ }_{\text {sb }}$ )
done
from valid-history-nth-update [OF i-bound this]
show ?thesis
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}$ )
qed

volatile-cond]
have reads-consis': reads-consistent False $\mathcal{O}_{\text {sb }} \mathrm{m}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.]\right)$
by (simp split: if-split-asm)
from valid-reads-nth-update [OF i-bound this]
have valid-reads': valid-reads $\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ )
have valid-sharing': valid-sharing $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ ]
have non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v])
by (auto simp add: non-volatile-writes-unshared-append)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}{ }^{\prime}$ ts $_{\text {sb }}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$
next
from sharing-consis [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}$.
then
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v])
by (simp add: sharing-consistent-append)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ )
next
note read-only-unowned [ OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
from read-only-unowned-nth-update [OF i-bound this]
show read-only-unowned $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by (simp add: $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ ) next
from unowned-shared-nth-update [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i subset-refl]
show unowned-shared $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}$ ) next
from no-outstanding-write-to-read-only-memory [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - ]
have no-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}} \mathrm{sb}$.
hence no-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}}\left(\mathrm{sb} @_{\mathrm{B}}\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v] $)$
by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
have tmps-distinct': tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof (intro-locales)
from load-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have distinct-load-tmps is ${ }_{\text {sb }}{ }^{\prime}$
by (auto split: instr.splits simp add: is $\mathrm{isb}_{\text {sb }}$ )
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ ) next
from read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have distinct-read-tmps sb.
moreover
from load-tmps-read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have $\mathrm{t} \notin$ read-tmps sb
by (auto simp add: is $\mathrm{s}_{\text {sb }}$ )
ultimately have distinct-read-tmps (sb @ $\left[\operatorname{Read}_{s b}\right.$ volatile a $\left.\mathrm{t} v\right]$ ) by (auto simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ ) next
from load-tmps-read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}$ ] load-tmps-distinct [OF i-bound ts $\mathrm{ts}_{\text {sb }}$ i]
have load-tmps is $\mathbf{s b}^{\prime} \cap$ read-tmps $\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v$\left.]\right)=\{ \}$ by (clarsimp simp add: read-tmps-append is sb )
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{sb}^{\prime}$ ) qed
have valid-sops': valid-sops $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof -
from valid-store-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have valid-store-sops ${ }^{\prime}: \forall$ sop $\in$ store-sops $\mathrm{is}_{\mathrm{sb}}{ }^{\prime} \cdot$ valid-sop sop by (auto simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from valid-write-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text { - }}$ ]
have valid-write-sops': $\forall$ sop $\in$ write-sops (sb@ $\left[\operatorname{Read}_{\text {sb }}\right.$ volatile a t v]). valid-sop sop by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound valid-write-sops' valid-store-sops ']
show ?thesis by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ sb )
qed
have valid-dd': valid-data-dependency $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof -
from data-dependency-consistent-instrs [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have dd-is: data-dependency-consistent-instrs (dom $\vartheta_{\mathrm{sb}}{ }^{\prime}$ ) is $\mathrm{is}_{\mathrm{sb}}{ }^{\prime}$ by (auto simp add: is sb $\vartheta_{\text {sb }}$ )
from load-tmps-write-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}$ ]
have load-tmps is sb $^{\prime} \cap \bigcup\left(\right.$ fst ' write-sops $\left(\operatorname{sb@}\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.\left.]\right)\right)=\{ \}$
by (auto simp add: write-sops-append is sb )
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
have load-tmps-fresh ${ }^{\prime}$ : load-tmps-fresh $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i} \text { ] }] ~}^{\text {[ }}$
have load-tmps (Read volatile at $\#$ is $\left._{\text {sb }}{ }^{\prime}\right) \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}$ by (simp add: is $_{\text {sb }}$ )
moreover
from load-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i] have $\mathrm{t} \notin$ load-tmps is $\mathrm{s}_{\mathrm{sb}}{ }^{\prime}$ by (auto simp add: is sb $_{\text {sb }}$ )
ultimately have load-tmps $\mathrm{is}_{\mathrm{sb}}{ }^{\prime} \cap \operatorname{dom}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v})\right)=\{ \}$
by auto
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}$ )
qed
have enough-flushs': enough-flushs $\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}$
proof -
from clean-no-outstanding-volatile-Write ${ }_{s b}$ [OF i-bound ts $\mathrm{s}_{\mathrm{sb}}-\mathrm{i}$ ]
have $\neg \mathcal{D}_{\mathrm{sb}} \longrightarrow$ outstanding-refs is-volatile-Write $\mathrm{s}_{\mathrm{sb}}\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.]\right)=\{ \}$ by (auto simp add: outstanding-refs-append )
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}$ )
qed
have valid-program-history ${ }^{\prime}$ : valid-program-history ts $_{\text {sb }}{ }^{\prime}$
proof -
from valid-program-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have causal-program-history is $\mathrm{s}_{\mathrm{sb}} \mathrm{sb}$.
then have causal': causal-program-history is sb $^{\prime}$ ( $\operatorname{sb@}\left[\operatorname{Read}_{s b}\right.$ volatile at v$\left.]\right)$ by (auto simp: causal-program-history-Read is $_{\text {sb }}$ )
from valid-last-prog [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have last-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}$.
hence last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v]$)=\mathrm{p}_{\mathrm{sb}}$
by (simp add: last-prog-append-Read ${ }_{\text {sb }}$ )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}^{\prime}}{ }^{\prime} \mathrm{sb}$ )
qed
show ?thesis
proof (cases outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
case True
from True have flush-all: takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}=\mathrm{sb}$ by (auto simp add: outstanding-refs-conv )
from True have suspend-nothing: dropWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}=[]$ by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends $=[]$
by (simp add: suspends)
from suspends-empty is-sim have is: is $=$ Read volatile a t \# is $\mathbf{s b}_{\mathbf{s}}{ }^{\prime}$ by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )
with suspends-empty ts-i
have ts-i: ts!i $=\left(\mathrm{p}_{\mathrm{sb}}\right.$, Read volatile a $\mathrm{t} \# \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$ )
by simp
from direct-memop-step.Read
have (Read volatile a t \# is $\mathrm{sb}^{\prime}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow$
$\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),(), \mathrm{m}, \mathcal{D}\right.$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb (dom
$\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right)$.
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \quad \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),()\right.\right.\right.$,
$\mathcal{D}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{sb}}$, release ? take-sb (dom $\left.\left.\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)$.

## moreover

from flush-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{\mathbf{s b}-\mathrm{i}}$ [simplified $\mathrm{is}_{\mathbf{s b}}$ ] ]
have flush-commute: flush-all-until-volatile-write
$\left(\mathrm{ts}_{\mathbf{s b}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathbf{s b}}{ }^{\prime}\right.\right.\right.$,
$\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v})$, sb @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a t v$\left.\left.\left.], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\right) \mathrm{m}_{\mathrm{sb}}=$ flush-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}$.
from True witness have not-volatile ${ }^{\prime}$ : volatile ${ }^{\prime}=$ False
by (auto simp add: outstanding-refs-conv)
from witness not-volatile ${ }^{\prime}$ have a-out-sb: a $\in$ outstanding-refs (Not $\circ$ is-volatile) sb apply (cases sop)
apply (fastforce simp add: outstanding-refs-conv is-volatile-def split: memref.splits) done
with non-volatile-owned-or-read-only-outstanding-refs
[OF outstanding-non-volatile-refs-owned-or-read-only [OF i-bound $\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}}}$ - $]$ ]
have a-owned: $a \in \mathcal{O}_{\text {sb }} \cup$ all-acquired sb $\cup$ read-only-reads $\mathcal{O}_{\text {sb }}$ sb by auto
have flush-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{a}=\mathrm{v}$
proof -
have $\forall \mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile- Write $_{s b}$ ) $\mathrm{sb}_{\mathrm{j}}$ ))
proof \{
fix $\mathrm{j}_{\mathrm{p}}$ is $\mathrm{s}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume $\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ ( takeWhile (Not o is-volatile-Write $_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ )

## proof

let ? take-sb ${ }_{\mathrm{j}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ${ }^{\text {d drop- }} \mathrm{sb}_{\mathrm{j}}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
assume a-in: a $\in$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ ? take-sb ${ }_{j}$
with outstanding-refs-takeWhile [where $\mathrm{P}^{\prime}=$ Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ]
have a-in': a $\in$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ sb $_{\mathrm{j}}$
by auto
with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]]
have $j$-owns: $a \in \mathcal{O}_{j} \cup$ all-acquired $\operatorname{sb}_{j}$
by auto
with ownership-distinct [OF i-bound j-bound neq-i-j ts $\mathrm{s}_{\mathrm{sb}}-\mathrm{i} j \mathrm{jth}$ ]
have a-not-owns: $\mathrm{a} \notin \mathcal{O}_{\text {sb }} \cup$ all-acquired sb
by blast
from non-volatile-owned-or-read-only-append [of False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb $\mathrm{T}_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ] outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]
have non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ by simp
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF this] a-in
have j-owns-drop: $\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired ?take-sb $\mathrm{j}_{\mathrm{j}}$
by auto
from rels-cond [rule-format, OF j-bound [simplified leq] neq-i-j] ts-sim [rule-format, OF j-bound] jth
have no-unsharing:release ?take-sb $\mathrm{j}_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}$ a $\neq$ Some False
by (auto simp add: Let-def)

## \{

assume a $\in$ acquired True sb $\mathcal{O}_{\text {sb }}$
with acquired-all-acquired-in [OF this] ownership-distinct [OF i-bound j-bound neq-i-j $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} j \mathrm{jth}\right]$
j-owns
have False
by auto
\}
moreover
\{
assume a-ro: a $\in$ read-only (share ?drop-sb $\mathcal{S}$ )
from read-only-share-unowned-in [OF weak-consis-drop a-ro] a-not-owns acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\mathrm{sb}}$ ]
all-acquired-append [of ?take-sb ?drop-sb]
have a $\in$ read-only $\mathcal{S}$
by auto
with share-all-until-volatile-write-thread-local [OF ownership-distinct-ts ${ }_{\text {sb }}$
sharing-consis-ts $\mathrm{st}_{\mathrm{sb}} \mathrm{j}$-bound jth j-owns]
have $\mathrm{a} \in$ read-only (share ? take-sb ${ }_{\mathrm{j}} \mathcal{S}_{\mathrm{sb}}$ )
by (auto simp add: read-only-def $\mathcal{S}$ )
hence a-dom: a $\in$ dom (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )
by (auto simp add: read-only-def domIff)
from outstanding-non-volatile-writes-unshared [OF j-bound jth]
non-volatile-writes-unshared-append [of $\mathcal{S}_{\text {sb }}$ ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have nvw: non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}$ ?take-sb ${ }_{j}$ by auto
from release-not-unshared-no-write-take [OF this no-unsharing a-dom] a-in
have False by auto

## \} <br> moreover <br> \{

assume a-share: volatile $\wedge \mathrm{a} \in$ dom (share ?drop-sb $\mathcal{S}$ )
from outstanding-non-volatile-writes-unshared [OF j-bound jth]
have non-volatile-writes-unshared $\mathcal{S}_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$.
with non-volatile-writes-unshared-append [of $\mathcal{S}_{\text {sb }}$ ?take-sb ${ }_{j}$
?drop-sb ${ }_{\mathrm{j}}$ ]
have unshared-take: non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}$ (takeWhile (Not o is-volatile-Write $\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{j}}$ )
by clarsimp
from valid-own have own-dist: ownership-distinct $\mathrm{ts}_{\mathrm{sb}}$
by (simp add: valid-ownership-def)
from valid-sharing have sharing-consis $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}$
by (simp add: valid-sharing-def)
from sharing-consistent-share-all-until-volatile-write [OF own-dist this i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ] have sc: sharing-consistent $\mathcal{S}$ (acquired True ?take-sb $\mathcal{O}_{\text {sb }}$ ) ?drop-sb by (simp add: $\mathcal{S}$ )
from sharing-consistent-share-all-shared have dom (share ?drop-sb $\mathcal{S}$ ) $\subseteq \operatorname{dom} \mathcal{S} \cup$ all-shared ?drop-sb by auto
also from sharing-consistent-all-shared [OF sc]
have $\ldots \subseteq \operatorname{dom} \mathcal{S} \cup$ acquired True ?take-sb $\mathcal{O}_{\text {sb }}$ by auto
also from acquired-all-acquired all-acquired-takeWhile
have $\ldots \subseteq \operatorname{dom} \mathcal{S} \cup\left(\mathcal{O}_{\text {sb }} \cup\right.$ all-acquired sb) by force
finally
have a-shared: $\mathrm{a} \in \operatorname{dom} \mathcal{S}$
using a-share a-not-owns
by auto
with share-all-until-volatile-write-thread-local [OF ownership-distinct-ts $\mathrm{sb}_{\mathrm{sb}}$ sharing-consis-ts ${ }_{\text {sb }} \mathrm{j}$-bound jth j-owns]
have a-dom: $\mathrm{a} \in$ dom (share ?take-sb $\mathcal{S}_{\mathrm{jb}}$ )
by (auto simp add: $\mathcal{S}$ domIff)
from release-not-unshared-no-write-take [OF unshared-take no-unsharing
a-dom] a-in
have False by auto

## \}

ultimately show False
using access-cond'
by auto
qed
\}
thus ?thesis
by (fastforce simp add: Let-def)
qed
from flush-all-until-volatile-write-buffered-val-conv
[OF True i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ this]
show ?thesis
by (simp add: buf-v)
qed

```
hence \(\mathrm{m}-\mathrm{a}-\mathrm{v}\) : \(\mathrm{ma}=\mathrm{v}\)
    by (simp add: m)
have tmps-commute: \(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v})=\left(\left.\vartheta_{\mathrm{sb}}\right|^{\cdot}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\{\mathrm{t}\}\right)\right)(\mathrm{t} \mapsto \mathrm{v})\)
    apply (rule ext)
    apply (auto simp add: restrict-map-def domIff)
    done
```

from suspend-nothing
have suspend-nothing': (dropWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb $\left.{ }^{\prime}\right)=[]$
by (simp add: sb ${ }^{\prime}$ )
from $\mathcal{D}$
have $\mathcal{D}^{\prime}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}}\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at v$\left.]\right) \neq$ \{\})
by (auto simp: outstanding-refs-append)
have $\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.$,
$\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}\right.$ a) $,(), \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$,
release ?take-sb $\left.\left.\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)$
apply (rule sim-config.intros)
apply ( $\operatorname{simp}$ add: m flush-commute $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}$ )
using share-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{s_{s b}}$-i [simplified is $\mathrm{s}_{\mathrm{sb}}$ ]]
apply ( $\operatorname{simp}$ add: $\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}$ )
using leq
apply ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
using i-bound i-bound ${ }^{\prime}$ ts-sim ts-i True $\mathcal{D}^{\prime}$
apply (clarsimp simp add: Let-def nth-list-update
outstanding-refs-conv m-a-v $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}$ suspend-nothing ${ }^{\prime}$
$\mathcal{D}_{\text {sb }}$ ' flush-all acquired-append release-append
split: if-split-asm )
apply (rule tmps-commute)
done
ultimately show ?thesis
using valid-own' valid-hist ' valid-reads' valid-sharing' tmps-distinct ${ }^{\prime}$ valid-sops ${ }^{\prime}$ valid-dd' load-tmps-fresh' enough-flushs ${ }^{\prime}$ valid-program-history' ${ }^{\prime}$ valid $^{\prime}$
$\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$
by (auto simp del: fun-upd-apply )
next
case False
then obtain $r$ where $r$-in: $r \in$ set sb and volatile-r: is-volatile-Write ${ }_{s b} r$
by (auto simp add: outstanding-refs-conv)
from takeWhile-dropWhile-real-prefix
[OF r-in, of (Not $\circ$ is-volatile-Write ${ }_{s b}$ ), simplified, OF volatile-r]
obtain $a^{\prime} v^{\prime} \operatorname{sb}^{\prime \prime}$ sop ${ }^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
sb-split: sb $=$ takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime}$ $\mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{sb}^{\prime \prime}$
and
drop: dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $s b=$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ sb"
apply (auto)
subgoal for y ys
apply (case-tac y)
apply auto
done
done
from drop suspends have suspends: suspends $=$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ $\mathrm{sb}^{\prime \prime}$ by simp
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})$ by auto
moreover
from flush-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}$ [simplified is $\left._{\text {sb }}\right]$ ]
have flush-commute: flush-all-until-volatile-write $\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.\right.\right.$ volatile a t v$\left.\left.\left.], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\right)$
$\mathrm{m}_{\mathrm{sb}}=$ flush-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}$.
have $W_{r i t e}$ sb True $a^{\prime}$ sop $^{\prime} v^{\prime} A^{\prime} L^{\prime} R^{\prime} W^{\prime} \in$ set sb by (subst sb-split) auto
from dropWhile-append 1 [OF this, of (Not $\circ$ is-volatile-Write $\left.{ }_{\mathbf{s b}}\right)$ ]
have drop-app-comm:
$\left(\operatorname{dropWhile}\left(\right.\right.$ Not is-volatile-Write $\left.{ }_{\mathrm{sb}}\right)\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a t v$\left.\left.]\right)\right)=$
dropWhile (Not o is-volatile-Write $\mathbf{s b}$ ) sb @ $\left[\operatorname{Read}_{s b}\right.$ volatile a t v $]$
by $\operatorname{simp}$
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathbf{s b}}$ - ]
have $\mathrm{t} \notin \operatorname{dom} \vartheta_{\text {sb }}$
by (auto simp add: is sb $_{\text {b }}$ )
then have tmps-commute:

```
    \varthetasb}\mp@subsup{|}{}{\prime}(\operatorname{dom}\mp@subsup{\vartheta}{\mathrm{ sb }}{}-\mathrm{ read-tmps sb}\mp@subsup{}{}{\prime\prime})
```



```
    apply -
    apply (rule ext)
    apply auto
    done
```

from $\mathcal{D}$
have $\mathcal{D}^{\prime}: \mathcal{D}_{\text {sb }}=\left(\mathcal{D} \vee\right.$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}\left(\operatorname{sb@}\left[\operatorname{Read}_{\text {sb }}\right.\right.$ volatile a t v$\left.]\right) \neq$ \{\}) by (auto simp: outstanding-refs-append)
have $\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})$
apply (rule sim-config.intros)
apply ( $\operatorname{simp}$ add: m flush-commute $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}$ )
using share-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ [simplified $\mathrm{is}_{\mathrm{sb}}$ ]]
apply (simp add: $\left.\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
using leq
apply ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
using i-bound i-bound ${ }^{\prime}$ ts-sim ts-i is-sim $\mathcal{D}^{\prime}$

```
apply (clarsimp simp add: Let-def nth-list-update is-sim drop-app-comm
    read-tmps-append suspends prog-instrs-append-Read \({ }_{s b}\) instrs-append-Read \({ }_{s b}\)
    hd-prog-append-Read \({ }_{\text {sb }}\)
    drop is \(\mathrm{s}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) acquired-append takeWhile-append1 [OF r-in]
volatile-r
    split: if-split-asm)
    apply (simp add: drop tmps-commute) +
    done
ultimately show ?thesis
    using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-dd'
        valid-sops' load-tmps-fresh' enough-flushs'
        valid-program-history' valid \(^{\prime} \mathrm{m}_{\text {sb }}{ }^{\prime} \mathcal{S}_{\text {sb }}{ }^{\prime}\)
    by (auto simp del: fun-upd-apply )
        qed
    next
        case (SBHReadUnbuffered a volatile t)
        then obtain
\(\mathrm{is}_{\mathrm{sb}}\) : \(\mathrm{is}_{\mathrm{sb}}=\) Read volatile at \# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and
        \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}\) and
\(\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\) and
\(s b^{\prime}: s^{\prime}=s b @\left[\operatorname{Read}_{s b}\right.\) volatile a \(\left.t\left(m_{s b} a\right)\right]\) and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\) and
\(\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \mathcal{D}_{\mathrm{sb}}{ }^{\prime}=\mathcal{D}_{\mathrm{sb}}\) and
buf-None: buffered-val sb a \(=\) None
```

by auto
from safe-memop-flush-sb [simplified is s $_{\text {sb }}$ ]
obtain access-cond': a $\in$ acquired True sb $\mathcal{O}_{\text {sb }} \vee$
$\mathrm{a} \in$ read-only (share ?drop-sb $\mathcal{S}) \vee($ volatile $\wedge \mathrm{a} \in \operatorname{dom}$ (share ?drop-sb $\mathcal{S}$ )) and
volatile-clean: volatile $\longrightarrow \neg \mathcal{D}_{\mathrm{sb}}$ and
rels-cond: $\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow$ released (ts $!\mathrm{j}) \mathrm{a} \neq$ Some False and rels-nv-cond: $\neg$ volatile $\longrightarrow(\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin$ dom (released (ts j j$))$ )
by cases auto
from clean-no-outstanding-volatile-Write ${ }_{\text {sb }}\left[\mathrm{OF}\right.$ i-bound $\left.\mathrm{ts}_{\text {sb }} \mathrm{i}\right]$ volatile-clean
have volatile-cond: volatile $\longrightarrow$ outstanding-refs is-volatile-Write $_{\text {sb }} \mathrm{sb}=\{ \}$
by auto

$$
\{
$$

fix $\mathrm{jp}_{\mathrm{j}} \mathrm{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\text {sbj }} \vartheta_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}$
assume j-bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume jth : $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume non-vol: $\neg$ volatile
have a $\notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$

## proof

assume $a-j: a \in \mathcal{O}_{j} \cup$ all-acquired $\operatorname{sb}_{j}$
let ?take-sb $_{\mathrm{j}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ?drop-sb ${ }_{\mathrm{j}}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
from ts-sim [rule-format, OF j-bound] jth
obtain suspends $\mathrm{j}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}}$ where
suspends $_{\mathrm{j}}$ : suspends $_{\mathrm{j}}=$ ?drop-sb $\mathrm{b}_{\mathrm{j}}$ and $\mathrm{is}_{\mathrm{j}}$ : instrs suspends $\mathrm{j}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{s}_{\mathrm{j}} @$ prog-instrs suspends $\mathrm{s}_{\mathrm{j}}$ and $\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)$ and $\mathrm{ts}_{\mathrm{j}}:$ ts $!\mathrm{j}=$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$, $\mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{\mid}\left(\right.$dom $\vartheta_{\text {sbj }}-$ read-tmps suspends j$),()$, $\mathcal{D}_{\mathrm{j}}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$, release ? take-sb $\mathrm{T}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}$ ) by (auto simp add: Let-def)
from a-j ownership-distinct [OF i-bound j-bound neq-i-j ts $\mathbf{s b}_{\mathbf{s}}-\mathrm{i} j$ jth]
have a-notin-sb: a $\notin \mathcal{O}_{\text {sb }} \cup$ all-acquired sb by auto
with acquired-all-acquired [of True sb $\mathcal{O}_{\text {sb }}$ ]
have a-not-acq: a $\notin$ acquired True sb $\mathcal{O}_{\text {sb }}$ by blast
with access-cond' non-vol
have a-ro: a $\in$ read-only (share ?drop-sb $\mathcal{S}$ )
by auto
from read-only-share-unowned-in [OF weak-consis-drop a-ro] a-notin-sb acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\text {sb }}$ ] all-acquired-append [of ?take-sb ?drop-sb]
have a-ro-shared: a $\in$ read-only $\mathcal{S}$ by auto
from rels-nv-cond [rule-format, OF non-vol j-bound [simplified leq] neq-i-j] ts $\mathrm{s}_{\mathrm{j}}$
have a $\notin \operatorname{dom}$ (release ?take-sb $\left.{ }_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}\right)$
by auto
with dom-release-takeWhile [of $\operatorname{sb}_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}$ ]
obtain
a-rels j : a $\notin \operatorname{dom} \mathcal{R}_{\mathrm{j}}$ and
a-shared ${ }_{j}$ : a $\notin$ all-shared ?take-sb ${ }_{j}$
by auto
have a $\notin \bigcup\left(\left(\lambda(-,-,-\right.\right.$, sb,,,---$)$. all-shared (takeWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$
sb)) '

```
            set \(\mathrm{ts}_{\mathrm{sb}}\) )
proof -
        \{
            fix k \(\mathrm{p}_{\mathrm{k}}\) is \(_{\mathrm{k}} \vartheta_{\mathrm{k}} \operatorname{sb}_{\mathrm{k}} \mathcal{D}_{\mathrm{k}} \mathcal{O}_{\mathrm{k}} \mathcal{R}_{\mathrm{k}}\)
            assume k -bound: \(\mathrm{k}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
            assume ts-k: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{k}=\left(\mathrm{p}_{\mathrm{k}}, \mathrm{is}_{\mathrm{k}}, \vartheta_{\mathrm{k}}, \mathrm{sb}_{\mathrm{k}}, \mathcal{D}_{\mathrm{k}}, \mathcal{O}_{\mathrm{k}}, \mathcal{R}_{\mathrm{k}}\right)\)
            assume a-in: a \(\in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{k}}\) )
```

```
            have False
            proof (cases k=j)
                case True with a-shared j jth ts-k a-in show False by auto
                next
                case False
                from ownership-distinct [OF j-bound k-bound False [symmetric] jth ts-k] a-j
                have a }\not\in(\mp@subsup{\mathcal{O}}{\textrm{k}}{}\cup\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{\textrm{k}}{})\mathrm{ by auto
                with all-shared-acquired-or-owned [OF sharing-consis [OF k-bound ts-k]] a-in
                show False
                using all-acquired-append [of takeWhile (Not ○ is-volatile-Write sb
                dropWhile (Not o is-volatile-Write sb
                all-shared-append [of takeWhile (Not ○ is-volatile-Write sb
                dropWhile (Not o is-volatile-Write sb
            qed
    }
    thus ?thesis by (fastforce simp add: in-set-conv-nth)
qed
with a-ro-shared
    read-only-shared-all-until-volatile-write-subset' [of tssb }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mathrm{ ]
have a-ro-shared cb: a }\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{
    by (auto simp add: S
    with read-only-unowned [OF j-bound jth]
    have a-notin-owns-j: a }\not\in\mp@subsup{\mathcal{O}}{\textrm{j}}{
    by auto
```

    have own-dist: ownership-distinct \(\mathrm{ts}_{\text {sb }}\) by fact
    have share-consis: sharing-consis \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\mathrm{sb}}\) by fact
    from sharing-consistent-share-all-until-volatile-write [OF own-dist share-consis i-bound
    $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]$
have consis': sharing-consistent $\mathcal{S}$ (acquired True ?take-sb $\mathcal{O}_{\text {sb }}$ ) ?drop-sb
by (simp add: $\mathcal{S}$ )
from share-all-until-volatile-write-thread-local [OF own-dist share-consis j-bound
jth a-j] a-ro-shared
have a-ro-take: a $\in$ read-only (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )
by (auto simp add: domIff $\mathcal{S}$ read-only-def)
from sharing-consis [OF j-bound jth]
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}$.
from sharing-consistent-weak-sharing-consistent [OF this]
weak-sharing-consistent-append [of $\mathcal{O}_{\mathrm{j}}$ ? take-sb $\mathrm{t}_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have weak-consis-drop:weak-sharing-consistent $\mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$
by auto
from read-only-share-acquired-all-shared [OF this read-only-unowned [OF j-bound
jth] a-ro-take ] a-notin-owns-j a-shared ${ }_{j}$
have a $\notin$ all-acquired ?take-sb ${ }_{j}$
by auto
with a-j a-notin-owns-j
have a-drop: $a \in$ all-acquired ?drop-sb ${ }_{j}$
using all-acquired-append [of ?take-sb $\mathrm{b}_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
by simp

```
from i-bound j-bound leq have j-bound-ts': \(\mathrm{j}<\) length ? ts \({ }^{\prime}\)
    by auto
note conflict-drop \(=\) a-drop [simplified suspends \({ }_{j}\) [symmetric]]
from split-all-acquired-in [OF conflict-drop]
show False
proof
    assume \(\exists\) sop \(\mathrm{a}^{\prime}\) v ys zs A L R W.
                (suspends \({ }_{j}=y s @ W_{r i t e}\) sb True \(\left.a^{\prime} \operatorname{sop} v A L R W \# z s\right) \wedge a \in A\)
    then
    obtain \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime}\) ys zs \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
        split-suspends \(\mathrm{j}_{\mathrm{j}}\) : suspends \(\mathrm{j}_{\mathrm{j}}=\mathrm{ys} @\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) zs
        (is suspends \({ }_{\mathrm{j}}=\) ?suspends) and
\(a-A^{\prime}: a \in A^{\prime}\)
        by blast
    from sharing-consis [OF j-bound jth]
    have sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
    then have \(\mathrm{A}^{\prime}-\mathrm{R}^{\prime}: \mathrm{A}^{\prime} \cap \mathrm{R}^{\prime}=\{ \}\)
        by (simp add: sharing-consistent-append [of - - ?take-sb \({ }_{\mathrm{j}}\) ? drop-sb \(_{\mathrm{j}}\), simplified]
suspends \({ }_{j}\) [symmetric] split-suspends \(\mathrm{s}_{\mathrm{j}}\) sharing-consistent-append)
    from valid-program-history [OF j-bound jth]
    have causal-program-history is \(_{s_{b j}} \mathrm{sb}_{\mathrm{j}}\).
    then have cph: causal-program-history is sbj ?suspends
        apply -
        apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? \({ }^{\text {take-sb }}{ }_{\mathrm{j}}\) ] )
        apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
        apply (simp add: split-suspends \({ }_{\mathrm{j}}\) )
        done
    from \(t_{\mathrm{j}}\) neq-i-j j-bound
    have ts' \({ }^{\mathrm{j}}\) : ? \(\mathrm{ts}^{\prime}!\mathrm{j}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\), is \(_{\mathrm{j}}\),
        \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps suspends \(\left.{ }_{j}\right),()\),
        \(\mathcal{D}_{\mathrm{j}}\), acquired True ? take-sb \(\mathcal{O}_{\mathrm{j}}\), release ? take-sb \(\left.\mathrm{T}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
        by auto
    from valid-last-prog [OF j-bound jth] have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
    then
    have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\)
        apply -
        apply (rule last-prog-same-append [where sb=?take-sb \({ }_{\mathrm{j}}\) ])
        apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
        apply simp
        done
    from valid-reads [OF j-bound jth]
    have reads-consis-j: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
```

from reads-consistent-flush-all-until-volatile-write [OF 〔valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ > j-bound jth reads-consis-j]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb $\mathrm{b}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends $\mathrm{m}_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j ts $_{\text {sb }}$-i jth]
have outstanding-refs is-Write ${ }_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$ by ( $\operatorname{simp}$ add: suspends ${ }_{\mathrm{j}}$ )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) (flush ?drop-sb m) suspends ${ }_{j}$.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) (ys@[Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
by (simp add: split-suspends ${ }_{j}$ reads-consistent-append)
from valid-write-sops [OF j-bound jth]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops (ys@ $\left[W^{2}\right.$ rite $_{\text {sb }}$ True $\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \eta$ ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound jth]
have distinct-read-tmps (?take-sb $@_{j}$ suspends $_{\mathrm{j}}$ )
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb ${ }_{\mathrm{j}} \cap$ read-tmps suspends ${ }_{\mathrm{j}}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound jth]
have h-consis:
history-consistent $\vartheta_{\text {sbj }}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ?take-sb $\mathrm{j}_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$
proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{b}_{\mathrm{j}} @$ ?drop-sb $\left.\mathrm{b}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by $\operatorname{simp}$
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$ ) ?take-sb $\mathrm{s}_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )

## moreover

have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{b}_{\mathrm{j}} @$ suspends $\left.\left.\mathrm{s}_{\mathrm{j}}\right)\right)$ ?take-sb $\mathrm{j}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) suspends $\mathrm{s}_{\mathrm{j}}$ by (simp add: split-suspends ${ }_{j}\left[\right.$ symmetric] suspends ${ }_{j}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{\text {sb }}$ $\left(y^{( } @\left[W_{r i t e}^{s b}\right.\right.$ True $\left.\left.a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\right)=\{ \}$
by (auto simp add: outstanding-refs-append suspends ${ }_{j}$ [symmetric]
split-suspends $_{j}$ )
have acq-simp:
acquired True (ys @ $\left[\right.$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{l}^{\prime}$ )
(acquired True ?take-sb $\mathcal{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) $=$
acquired True ys (acquired True ? take-sb $\left.\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
by (simp add: acquired-append)
from flush-store-buffer-append [where $s b=y s @\left[W_{r i t e}^{s b}\right.$ True $\left.a^{\prime} \operatorname{sop}^{\prime} v^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ and $\mathrm{sb}^{\prime}=\mathrm{zs}$, simplified,

OF j-bound-ts' is $_{\mathrm{j}}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] cph [simplified suspends ${ }_{\mathrm{j}}$ ]
ts'-j [simplified split-suspends ${ }_{j}$ ] refl lp [simplified split-suspends ${ }_{j}$ ] reads-consis-ys
hist-consis' ${ }^{\prime}$ simplified split-suspends ${ }_{j}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ]
no-volatile-Read ${ }_{\text {sb }}$-volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}$ ]
obtain is $_{j}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}$ ': instrs zs @ is $\mathrm{sbj}=\mathrm{is}{ }_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts', flush ?drop-sb m, share ?drop-sb $\mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}$
(? $\mathrm{ts}^{\prime}$ ' j :=(last-prog
 $\operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{l}^{\prime}$ ),
is ${ }_{j}$,
$\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\right.$ dom $\vartheta_{\text {sbj }}$ - read-tmps zs $)$,
(), True, acquired True ys (acquired True ?take-sb $\mathcal{O}_{j}$ ) $\left.\cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)$ ],
flush (ys@[Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}{ }^{\prime}$ ) (flush ?drop-sb m),
share (ys@[Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]$ ) (share ?drop-sb $\left.\mathcal{S}\right)$ )
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}$, ?shared-ys))
by (auto simp add: acquired-append outstanding-refs-append)

```
    from i-bound' have i-bound-ys: i < length ?ts-ys
        by auto
    from i-bound' neq-i-j
    have ts-ys-i: ?ts-ys!i=( }\mp@subsup{\textrm{p}}{\textrm{sb}}{},\mp@subsup{\textrm{is}}{\textrm{sb}}{},\mp@subsup{\vartheta}{\textrm{sb}}{},()
        \mathcal{D}
        by simp
    note conflict-computation = rtranclp-trans [OF steps-flush-sb steps-ys]
    from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
    have safe-delayed (?ts-ys,?m-ys,?shared-ys).
    from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is sb
    have a \in read-only (share (ys@[Write sb True a' sop' v' A' L' R'W | ) (share ?drop-sb
        apply cases
        apply (auto simp add: Let-def is sb)
        done
    with a-A'
    show False
    by (simp add: share-append in-read-only-convs)
next
    assume \existsALR W ys zs. suspendsj}=\mp@code{ys@ Ghostsb ALR W # zs }\wedge a \in 
    then
    obtain (A' L' R' W' ys zs where
        split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{\prime}\mathrm{ : suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{}= ys @ Ghost mb A' L' R'W'# zs
        (is suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}=\mathrm{ ?suspends) and
a-A':a}\in\mp@code{A'
        by blast
    from valid-program-history [OF j-bound jth]
    have causal-program-history is isbj }\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ .
    then have cph: causal-program-history is sbj ?suspends
        apply -
    apply (rule causal-program-history-suffix [where sb=?take-sb}\mp@subsup{}{\textrm{j}}{}]\mathrm{ ])
    apply (simp only: split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{[symmetric] suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
    apply (simp add: split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
    done
from \(t_{\mathrm{j}}\) neq-i-j j-bound
have ts \({ }_{-} \mathrm{j}: ~ ? \mathrm{ts}^{\prime}!\mathrm{j}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{s}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps suspends \(\left.{ }_{j}\right),()\),
        \mathcal{D}}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ , acquired True ?take-sb
        by auto
    from valid-last-prog [OF j-bound jth] have last-prog: last-prog pj sbj
    then
    have lp: last-prog }\mp@subsup{\textrm{p}}{\textrm{j}}{}\mp@subsup{\mathrm{ suspends}}{\textrm{j}}{}=\mp@subsup{\textrm{p}}{\textrm{j}}{
```

S))

```
apply -
apply (rule last-prog-same-append [where sb=?take-sb j}]\mathrm{ ])
apply (simp only: split-suspendsj [symmetric] suspends }\mp@subsup{}{j}{}\mathrm{ )
apply simp
done
```

from valid-reads [OF j-bound jth]
have reads-consis-j: reads-consistent False $\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ ’ j-bound jth reads-consis-j]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends $\mathrm{m}_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $\mathrm{s}_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j $\mathrm{ts}_{\mathrm{sb}}$-i jth]
have outstanding-refs is-Write ${ }_{s b}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by ( $\operatorname{simp}$ add: suspends ${ }_{j}$ )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) suspends ${ }_{\mathrm{j}}$.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) (ys@[Ghost $\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ reads-consistent-append)
from valid-write-sops [OF j-bound jth]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb ${ }_{j}$. valid-sop sop and valid-sops-drop: $\forall$ sop $\in$ write-sops ( $\mathrm{ys}^{( }$@ $\left.\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound jth]
have distinct-read-tmps (?take-sb $\mathrm{@}_{\mathrm{j}}$ suspends ${ }_{\mathrm{j}}$ )
by (simp add: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb b $_{\mathrm{j}} \cap$ read-tmps suspends ${ }_{j}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspends ${ }_{j}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound jth]
have h-consis:
history-consistent $\vartheta_{\text {sbj }}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left.{ }_{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends ${ }_{\mathrm{j}}$ ) apply (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )

## apply simp <br> done

have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ? take-sb $\mathrm{j}_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\left._{\mathrm{j}}\right)$ proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ? take-sb $\mathrm{j}_{\mathrm{j}} @$ ? drop-sb $\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by $\operatorname{simp}$
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb ${ }_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends ${ }_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{b}_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends ${ }_{j}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) suspends $\mathrm{j}_{\mathrm{j}}$
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{\text {sb }}$ $\left.\left({\text { ys } @\left[\operatorname{Ghost}_{\text {sb }}\right.} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}$
by (auto simp add: outstanding-refs-append suspends ${ }_{j}$ [symmetric]
split-suspends ${ }_{j}$ )
have acq-simp:
acquired True (ys @ $\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
(acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) $=$
acquired True ys (acquired True ?take-sb $\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
by (simp add: acquired-append)
from flush-store-buffer-append [where $\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Ghost}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ and $\mathrm{sb}{ }^{\prime}=\mathrm{zs}$, simplified,

OF j-bound-ts ${ }^{\prime}$ is $_{\mathrm{j}}$ [simplified split-suspends $\mathrm{j}_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts ${ }^{\prime}-\mathrm{j}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] refl lp [simplified split-suspends ${ }_{\mathrm{j}}$ ] reads-consis-ys
hist-consis' [simplified split-suspends ${ }_{\mathrm{j}}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends ${ }_{\mathrm{j}}$ ]
no-volatile-Read sb $^{\mathbf{b}}$-volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}]$
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}^{\prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts', flush ?drop-sb m, share ?drop-sb $\mathcal{S}$ ) $\Rightarrow_{\mathrm{d}}{ }^{*}$
(?ts' $[\mathrm{j}:=$ (last-prog
(hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\left(\mathrm{ys}^{(\mathrm{O}}\left[\right.\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)$,
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$,

```
    \vartheta\mp@code{sbj }}\mp@subsup{|}{}{`}(\operatorname{dom}\mp@subsup{\vartheta}{\mathrm{ sbj }}{}-\mathrm{ read-tmps zs ),
    (),
```



```
acquired True ys (acquired True ?take-sb;}\mp@subsup{\mathcal{O}}{\textrm{j}}{})\cup\mp@subsup{\textrm{A}}{}{\prime}-\mp@subsup{\textrm{R}}{}{\prime},\mp@subsup{\mathcal{R}}{\textrm{j}}{\prime})]
            flush (ys@[Ghostsb A' L' R'' W']) (flush ?drop-sb m),
            share (ys@[Ghost sb A A L' }\mp@subsup{\textrm{R}}{}{\prime}\mp@subsup{\textrm{W}}{}{\prime}])\mathrm{ ) (share ?drop-sb S S)
        (is (-,-,-) 看**(?ts-ys,?m-ys,?shared-ys))
            by (auto simp add: acquired-append)
    from i-bound' have i-bound-ys: i < length ?ts-ys
        by auto
    from i-bound' neq-i-j
    have ts-ys-i: ?ts-ys!i = ( }\mp@subsup{p}{\textrm{sb}}{},\mp@subsup{\textrm{is}}{\mathbf{sb}}{},\mp@subsup{\vartheta}{\textrm{sb}}{},()
        \mathcal{D}
        by simp
    note conflict-computation = rtranclp-trans [OF steps-flush-sb steps-ys]
    from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
    have safe-delayed (?ts-ys,?m-ys,?shared-ys).
    from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is isb] non-vol a-not-acq
    have a }\in\mathrm{ read-only (share (ys@[Ghost sb A' }\mp@subsup{\textrm{A}}{}{\prime}\mp@subsup{\textrm{R}}{}{\prime}\textrm{W}])\mathrm{ ) (share ?drop-sb S S )
        apply cases
        apply (auto simp add: Let-def is⿱sm
        done
    with a-A'
    show False
        by (simp add: share-append in-read-only-convs)
    qed
qed
    }
    note non-volatile-unowned-others = this
        {
        assume a-in: a }\in\mathrm{ read-only (share (dropWhile (Not o is-volatile-Write sb) sb) S S)
        assume nv: ᄀ volatile
        have a }\in\mathrm{ read-only (share sb }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mathrm{ )
        proof (cases a }\in\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb)
        case True
        from share-all-until-volatile-write-thread-local' [OF ownership-distinct-ts sb
            sharing-consis-ts sb i-bound tssb
            show ?thesis
            by (simp add: S read-only-def)
        next
        case False
        from read-only-share-unowned [OF weak-consis-drop - a-in] False
            acquired-all-acquired [of True ?take-sb }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mathrm{ ] all-acquired-append [of ?take-sb
?drop-sb]
```

have a-ro-shared: a $\in$ read-only $\mathcal{S}$
by auto
have a $\notin \bigcup((\lambda(-,-,-, s b,-,-,-)$.

proof \{ fix k $\mathrm{p}_{\mathrm{k}}$ is $_{\mathrm{k}} \vartheta_{\mathrm{k}} \mathrm{sb}_{\mathrm{k}} \mathcal{D}_{\mathrm{k}} \mathcal{O}_{\mathrm{k}} \mathcal{R}_{\mathrm{k}}$ assume k-bound: $\mathrm{k}<$ length $\mathrm{ts}_{\mathrm{sb}}$ assume ts-k: $\mathrm{ts}_{\mathrm{sb}}!\mathrm{k}=\left(\mathrm{p}_{\mathrm{k}}, \mathrm{is}_{\mathrm{k}}, \vartheta_{\mathrm{k}}, \mathrm{sb}_{\mathrm{k}}, \mathcal{D}_{\mathrm{k}}, \mathcal{O}_{\mathrm{k}}, \mathcal{R}_{\mathrm{k}}\right)$ assume a-in: $\mathrm{a} \in$ all-shared (takeWhile (Not $\circ$ is-volatile-Write $\mathrm{sb}_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{k}}$ ) have False proof (cases $\mathrm{k}=\mathrm{i}$ )
case True with False $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ts-k a-in all-shared-acquired-or-owned [OF sharing-consis [OF k-bound ts-k]] all-shared-append [of takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{k}}$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $_{k}$ ] show False by auto next
case False
from rels-nv-cond [rule-format, OF nv k-bound [simplified leq] False [symmetric]
]
ts-sim [rule-format, OF k-bound] ts-k
have a $\notin \operatorname{dom}$ (release ( takeWhile $\left(\operatorname{Not} \circ\right.$ is-volatile-Write $\left.{ }_{s b}\right)$ sb $\left._{\mathrm{k}}\right)\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right)$
$\mathcal{R}_{\mathrm{k}}$ )
by (auto simp add: Let-def)
with dom-release-takeWhile $\left[\begin{array}{l} \\ \operatorname{sb}_{\mathrm{k}} \\ \left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \\ \mathcal{R}_{\mathrm{k}}\end{array}\right]$
obtain
a-rels $\mathrm{s}_{\mathrm{j}}$ : $\notin \operatorname{dom} \mathcal{R}_{\mathrm{k}}$ and
a-shared ${ }_{j}$ : a $\notin$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb $_{k}$ )
by auto
with False a-in show ?thesis
by auto
qed
\}
thus ?thesis
by (auto simp add: in-set-conv-nth)
qed
with read-only-shared-all-until-volatile-write-subset' $\left[\mathrm{of} \mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\text {sb }}\right.$ ] a-ro-shared
have $\mathrm{a} \in$ read-only $\mathcal{S}_{\mathrm{sb}}$
by (auto simp add: $\mathcal{S}$ )
from read-only-share-unowned' [OF weak-consis-sb read-only-unowned [OF i-bound
$\mathrm{ts}_{\mathrm{sb} \text { - }}$ ] False this]
show ?thesis .
qed
$\}$ note non-vol-ro-reduction $=$ this
have valid-own': valid-ownership $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (cases volatile)
case False
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}$.
then
have non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}\left({\left.\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}} \text { False att }\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)}\right.$ ) using access-cond' False non-vol-ro-reduction by (auto simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by (auto simp add: False $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ )
next
case True
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}$.
then
have non-volatile-owned-or-read-only False $\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\left({\left.\operatorname{sb} @\left[\operatorname{Read}_{\mathrm{sb}} \operatorname{True} \operatorname{tat}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)}\right.$ using True by (simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by (auto simp add: True $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ )
qed next
show outstanding-volatile-writes-unowned-by-others $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
have out: outstanding-refs is-volatile- Write $_{\text {sb }}\left(\right.$ sb $@\left[\operatorname{Read}_{\text {sb }}\right.$ volatile a t $\left.\left.\left(\mathrm{m}_{\text {sb }} \mathrm{a}\right)\right]\right) \subseteq$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}$ by (auto simp add: outstanding-refs-append)
have all-acquired $\left(\mathrm{sb}\right.$ @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\left.\mathrm{t}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right) \subseteq$ all-acquired sb by (auto simp add: all-acquired-append)
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ out this]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ )
qed next
show read-only-reads-unowned ts sb ${ }^{\prime}$
proof (cases volatile)
case True
have r: read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ )
$\left(\mathrm{sb}\right.$ @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at $\left.\left.\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)\right) \mathcal{O}_{\mathrm{sb}}\right)$
(dropWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\mathbf{s b}}\right)\left(\operatorname{sb} @\left[\operatorname{Read}_{\mathbf{s b}}\right.\right.$ volatile att $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$ )
$\subseteq$ read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb)
$\mathcal{O}_{\mathrm{sb}}$ )
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (case-tac outstanding-refs (is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}=\{ \}$ ) apply (simp-all add: outstanding-vol-write-take-drop-appends acquired-append read-only-reads-append True)
done
have $\mathcal{O}_{\mathrm{sb}} \cup$ all-acquired $\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right) \subseteq \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired sb
by (simp add: all-acquired-append)

```
    from read-only-reads-unowned-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i r this]
    show ?thesis
    by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
next
    case False
    show ?thesis
    proof (unfold-locales)
        fix n m
    fix \(\mathrm{p}_{\mathrm{n}}\) is \(_{\mathrm{n}} \mathcal{O}_{\mathrm{n}} \mathcal{R}_{\mathrm{n}} \mathcal{D}_{\mathrm{n}} \vartheta_{\mathrm{n}} \mathrm{sb}_{\mathrm{n}} \mathrm{p}_{\mathrm{m}}\) is \(_{\mathrm{m}} \mathcal{O}_{\mathrm{m}} \mathcal{R}_{\mathrm{m}} \mathcal{D}_{\mathrm{m}} \vartheta_{\mathrm{m}} \mathrm{sb}_{\mathrm{m}}\)
    assume n -bound: \(\mathrm{n}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    and m -bound: \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    and neq- \(\mathrm{n}-\mathrm{m}: \mathrm{n} \neq \mathrm{m}\)
    and nth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
    and mth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\)
    from \(n\)-bound have \(n\)-bound': \(\mathrm{n}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}\) )
    from \(m\)-bound have \(m\)-bound': \(m<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
    have acq-eq: \(\left(\mathcal{O}_{\mathrm{sb}}{ }^{\prime} \cup\right.\) all-acquired \(\left.\mathrm{sb}{ }^{\prime}\right)=\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb \()\)
        by ( \(\operatorname{simp}\) add: all-acquired-append \(\operatorname{sb}^{\prime} \mathcal{O}_{\text {sb }}\) )
    show \(\left(\mathcal{O}_{m} \cup\right.\) all-acquired \(\left.\operatorname{sb}_{m}\right) \cap\)
                read-only-reads (acquired True (takeWhile (Not o is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{n}}\) ) \(\mathcal{O}_{\mathrm{n}}\) )
                \(\left(\right.\) dropWhile \(\left.\left.^{(\text {Not } \circ \text { is-volatile-Write }}{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{n}}\right)=\)
                \{\}
    proof (cases \(\mathrm{m}=\mathrm{i}\) )
        case True
        with neq-n-m have neq-n-i: \(\mathrm{n} \neq \mathrm{i}\)
```

by auto
with n-bound nth i-bound have $n t h ': \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ )
note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth ${ }^{\prime} \mathrm{ts}_{\mathrm{s}_{\mathrm{sb}}}$-i]
moreover
note acq-eq
ultimately show? thesis
using True ts $_{\text {sb }}$-i nth mth n-bound' m -bound ${ }^{\prime}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
next
case False
note neq-m-i $=$ this
with m-bound mth i-bound have $\mathrm{mth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ )
show ?thesis
proof (cases $\mathrm{n}=\mathrm{i}$ )
case True
note read-only-reads-unowned [OF i-bound m-bound' neq-m-i [symmetric] $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ mth']
moreover

```
note acq-eq
moreover
note non-volatile-unowned-others [OF m-bound' neq-m-i [symmetric] mth]
ultimately show ?thesis
    using True ts sb-i nth mth n-bound' m-bound' neq-m-i
    apply (case-tac outstanding-refs (is-volatile-Write sb})\textrm{sb}={}
    apply (clarsimp simp add: outstanding-vol-write-take-drop-appends
        acquired-append read-only-reads-append ts }\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime})
    done
        next
case False
with n-bound nth i-bound have nth': ts sm!n =( }\mp@subsup{\textrm{p}}{\textrm{n}}{},\mp@subsup{\textrm{is}}{\textrm{n}}{},\mp@subsup{\vartheta}{\textrm{n}}{},\mp@subsup{\textrm{sb}}{\textrm{n}}{},\mp@subsup{\mathcal{D}}{\textrm{n}}{},\mp@subsup{\mathcal{O}}{\textrm{n}}{},\mp@subsup{\mathcal{R}}{\textrm{n}}{}
    by (auto simp add: ts sb
from read-only-reads-unowned [OF n-bound'm-bound' neq-n-m nth' mth] False neq-m-i
show ?thesis
    by (clarsimp)
        qed
        qed
    qed
qed
show ownership-distinct ts sb }\mp@subsup{}{}{\prime
proof -
    have all-acquired (sb @ [Read sb volatile a t (m
        by (auto simp add: all-acquired-append)
    from ownership-distinct-instructions-read-value-store-buffer-independent
    [OF i-bound ts sb-i this]
    show ?thesis by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}
qed
        qed
        have valid-hist': valid-history program-step ts sbb
        proof -
from valid-history [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}
have hcons: history-consistent }\mp@subsup{\vartheta}{\textrm{sb}}{}\mathrm{ (hd-prog p psb sb) sb.
from load-tmps-read-tmps-distinct [OF i-bound ts tsb-i]
have t-notin-reads: t # read-tmps sb
    by (auto simp add: is ib )
from load-tmps-write-tmps-distinct [OF i-bound ts sb-i]
have t-notin-writes: t \not\exists\bigcup(fst ' write-sops sb )
    by (auto simp add: is isb
from valid-write-sops [OF i-bound ts sb-i]
have valid-sops: }\forall\mathrm{ sop }\in\mathrm{ write-sops sb. valid-sop sop
    by auto
from load-tmps-fresh [OF i-bound ts sb-i]
have t-fresh: t & dom }\mp@subsup{\vartheta}{\mathrm{ sb }}{
    using is sb
    by simp
```

from valid-implies-valid-prog-hd [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i valid]
have history-consistent $\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)$
(hd-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$ )
(sb@ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile a $\left.\mathrm{t}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]$ )
using t-notin-writes valid-sops $t$-fresh hcons
apply -
apply (rule history-consistent-appendI)
apply (auto simp add: hd-prog-append-Read ${ }_{\text {sb }}$ )
done
from valid-history-nth-update [OF i-bound this]
show ?thesis
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \boldsymbol{\vartheta}_{\mathrm{sb}}{ }^{\prime}$ )
qed

## from

reads-consistent-unbuffered-snoc [OF buf-None refl valid-reads [OF i -bound $\mathrm{ts}_{\mathrm{sb}}$ - i ] volatile-cond ]
have reads-consis': reads-consistent False $\mathcal{O}_{\text {sb }} \mathrm{m}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$
by (simp split: if-split-asm)
from valid-reads-nth-update [OF i-bound this]
have valid-reads': valid-reads $\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ )
have valid-sharing': valid-sharing $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}$ ]
have non-volatile-writes-unshared $\mathcal{S}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$ by (auto simp add: non-volatile-writes-unshared-append)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}$ )
next
from sharing-consis [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}$ ]
have sharing-consistent $\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}$ sb.
then
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$
by (simp add: sharing-consistent-append)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ )
next
note read-only-unowned [ OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ ]
from read-only-unowned-nth-update [OF i-bound this]
show read-only-unowned $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}$
by ( simp add: $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ )
next
from unowned-shared-nth-update [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{s}}}$-i subset-refl]
show unowned-shared $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ ) next
from no-outstanding-write-to-read-only-memory [OF i-bound ts $\mathrm{ts}_{\mathrm{sb}}$ - ]
have no-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}} \mathrm{sb}$.
 by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
have tmps-distinct': tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
from load-tmps-distinct [OF i-bound $\mathrm{ts}_{\text {sb }-\mathrm{i} \text { ] }] ~}^{\text {] }}$
have distinct-load-tmps is sb $^{\prime}{ }^{\prime}$
by (auto split: instr.splits simp add: is sb $_{\text {b }}$ )
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ ) next
from read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have distinct-read-tmps sb.
moreover
from load-tmps-read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathbf{s b}}$ - ]
have $\mathrm{t} \notin$ read-tmps sb
by (auto simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
ultimately have distinct-read-tmps ( sb @ $\left[\operatorname{Read}_{\mathrm{sb}}\right.$ volatile at $\left.\left(\mathrm{m}_{\mathbf{s b}} \mathrm{a}\right)\right]$ ) by (auto simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{sb}^{\prime}$ ) next
from load-tmps-read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
load-tmps-distinct [OF i-bound $\mathrm{ts}_{\text {sb }}$ - ]
have load-tmps is ${ }_{s b}{ }^{\prime} \cap$ read-tmps $\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)=\{ \}$ by (clarsimp simp add: read-tmps-append is sb )
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}^{\prime}}$ by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{sb}^{\prime}$ ) qed have valid-sops': valid-sops $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof -

have valid-store-sops': $\forall$ sop $\in$ store-sops is $_{\text {sb }}$ '. valid-sop sop by (auto simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from valid-write-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}$ ]
have valid-write-sops': $\forall$ sop $\in$ write-sops ( $s$ @ $\left[\operatorname{Read}_{\text {sb }}\right.$ volatile at $\left.\left(m_{s b} a\right)\right]$. valid-sop sop
by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound valid-write-sops' valid-store-sops']
show ?thesis by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed have valid-dd': valid-data-dependency $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from data-dependency-consistent-instrs [OF i-bound $\mathrm{ts}_{\mathbf{s b}_{\mathrm{s}}-\mathrm{i}}$ ]
have dd-is: data-dependency-consistent-instrs (dom $\left.\vartheta_{\mathbf{s b}}{ }^{\prime}\right) \mathrm{is}_{\mathbf{s b}}{ }^{\prime}$ by (auto simp add: is sb $\vartheta_{\text {sb }}$ )
from load-tmps-write-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}$ ]
have load-tmps is sb $^{\prime} \cap \bigcup\left(\right.$ fst ' write-sops $\left(\operatorname{sb@}\left[\operatorname{Read}_{s b}\right.\right.$ volatile at $\left.\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)\right)=\{ \}$ by (auto simp add: write-sops-append is sb $_{\text {sb }}$ )
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by (simp add: $\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
have load-tmps-fresh': load-tmps-fresh $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof -
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i} \text { ] }] ~}^{\text {[ }}$
have load-tmps (Read volatile at $\#$ is $\left._{\text {sb }}{ }^{\prime}\right) \cap \operatorname{dom} \vartheta_{\mathrm{sb}}=\{ \}$ by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )
moreover
 by (auto simp add: is sb $_{\text {sb }}$ )
ultimately have load-tmps is $\mathrm{s}_{\mathrm{sb}}{ }^{\prime} \cap \operatorname{dom}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)=\{ \}$ by auto
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}$ )
qed
have enough-flushs': enough-flushs $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof -
from clean-no-outstanding-volatile-Write ${ }_{s b}$ [OF i-bound $\mathrm{ts}_{s b}-\mathrm{i}$ ]
have $\neg \mathcal{D}_{\text {sb }} \longrightarrow$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}\left(\operatorname{sb@}\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile a $\left.\left.\mathrm{t}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)=$
\{\}
by (auto simp add: outstanding-refs-append )
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}$ )
qed
have valid-program-history ${ }^{\prime}$ : valid-program-history ts $_{\text {sb }}{ }^{\prime}$ proof -
from valid-program-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}$ ]
have causal-program-history is $\mathrm{s}_{\mathrm{sb}} \mathrm{sb}$.
then have causal': causal-program-history is $\mathbf{s b}_{\mathbf{s b}}{ }^{\prime}\left(\operatorname{sb@} @ \operatorname{Read}_{\mathbf{s b}}\right.$ volatile a t $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)$
by (auto simp: causal-program-history-Read is $_{\text {sb }}$ )
from valid-last-prog [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - i ]
have last-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}$.
hence last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)=\mathrm{p}_{\mathrm{sb}}$ by (simp add: last-prog-append-Read ${ }_{\text {sb }}$ )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ sb )
qed
show ?thesis
proof (cases outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}=\{ \}$ )
case True
from True have flush-all: takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}=\mathrm{sb}$ by (auto simp add: outstanding-refs-conv )
from True have suspend-nothing: dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}=[]$ by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends $=[]$
by (simp add: suspends)
from suspends-empty is-sim have is: is $=$ Read volatile a $\mathrm{t} \# \mathrm{is}_{\text {sb }}{ }^{\prime}$ by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )
with suspends-empty ts-i
have ts-i: $\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \operatorname{Read}\right.$ volatile a $\mathrm{t} \# \mathrm{is}{ }_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()$,
$\mathcal{D}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$ )
by $\operatorname{simp}$
from direct-memop-step.Read
have (Read volatile a t \# is ${ }_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}$,
$\mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow$ $\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),(), \mathrm{m}, \mathcal{D}\right.$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right)$.
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),()\right.\right.\right.$, $\mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$, release ?take-sb (dom $\left.\left.\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)$.

## moreover

from flush-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}$ [simplified is $\mathrm{s}_{\mathrm{sb}}$ ] ]
have flush-commute: flush-all-until-volatile-write

$$
\begin{aligned}
& \left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{~m}_{\mathrm{sb}} \mathrm{a}\right), \mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}} \text { volatile at }\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\right) \\
& \quad \mathrm{m}_{\mathrm{sb}}= \\
& \quad \text { flush-all-until-volatile-write } \mathrm{ts}_{\mathrm{sb}} \mathrm{~m}_{\mathrm{sb}} .
\end{aligned}
$$

have flush-all-until-volatile-write $\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{a}=\mathrm{m}_{\mathrm{sb}} \mathrm{a}$
proof -
have $\forall \mathrm{j}<$ length $\mathrm{ts}_{\text {sb }} . \mathrm{i} \neq \mathrm{j} \longrightarrow$
(let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}$
in a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o
is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ))

```
proof -
    \{
        fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}}\) is \(\mathrm{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
        assume j -bound: \(\mathrm{j}<\mathrm{leng}\) th \(\mathrm{ts}_{\mathrm{sb}}\)
        assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
        assume jth: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
```

have a $\notin$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) proof
let ? take-sb $_{\mathrm{j}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ? drop-sb ${ }_{j}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\right)$
assume a-in: a $\in$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }}$ ?take-sb $_{\mathrm{j}}$
with outstanding-refs-takeWhile [where $\mathrm{P}^{\prime}=$ Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ]
have a-in': $\mathrm{a} \in$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$
by auto
with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]]
have j -owns: $\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
by auto
with ownership-distinct [OF i-bound j-bound neq-i-j ts $\mathrm{s}_{\mathrm{sb}}-\mathrm{i} j$ jth]
have a-not-owns: $\mathrm{a} \notin \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired sb
by blast
from non-volatile-owned-or-read-only-append [of False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ] outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]
have non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ by simp
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF this] a-in have j-owns-drop: $\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired ?take-sb ${ }_{j}$
by auto
from rels-cond [rule-format, OF j-bound [simplified leq] neq-i-j] ts-sim [rule-format, OF j-bound] jth
have no-unsharing:release ?take-sb ${ }_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}} \mathrm{a} \neq$ Some False by (auto simp add: Let-def)

## \{

assume a $\in$ acquired True sb $\mathcal{O}_{\text {sb }}$
with acquired-all-acquired-in [OF this] ownership-distinct [OF i-bound j-bound neq-i-j

```
ts sb-i jth]
```

j-owns
have False
by auto
\}
moreover
\{
assume a-ro: a $\in$ read-only (share ?drop-sb $\mathcal{S}$ )
from read-only-share-unowned-in [OF weak-consis-drop a-ro] a-not-owns acquired-all-acquired [of True ?take-sb $\mathcal{O}_{\text {sb }}$ ]
all-acquired-append [of ?take-sb ?drop-sb]
have a $\in$ read-only $\mathcal{S}$
by auto
with share-all-until-volatile-write-thread-local [OF ownership-distinct-ts ${ }_{\text {sb }}$ sharing-consis-ts ${ }_{\text {sb }} \mathrm{j}$-bound jth j-owns]
have a $\in$ read-only (share ?take-sb ${ }_{j} \mathcal{S}_{\text {sb }}$ )
by (auto simp add: read-only-def $\mathcal{S}$ )
hence a-dom: $\mathrm{a} \in \operatorname{dom}$ (share ?take-sb ${ }_{\mathrm{j}} \mathcal{S}_{\mathrm{sb}}$ )

```
                    by (auto simp add: read-only-def domIff)
                            from outstanding-non-volatile-writes-unshared [OF j-bound jth]
                                    non-volatile-writes-unshared-append [of }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mathrm{ ?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ ?drop-sb}\mp@subsup{\textrm{j}}{\textrm{j}}{}
                                    have nvw: non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mathrm{ ?take-sbj by auto
                                    from release-not-unshared-no-write-take [OF this no-unsharing a-dom] a-in
                                    have False by auto
}
moreover
{
    assume a-share: volatile }\wedge a \in dom (share ?drop-sb \mathcal{S}
    from outstanding-non-volatile-writes-unshared [OF j-bound jth]
    have non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ .
    with non-volatile-writes-unshared-append [of S S sb (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ )
sbj)
    (dropWhile (Not o is-volatile-Write sb) sb 
            have unshared-take: non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mathrm{ (takeWhile (Not o
is-volatile-Write sb
    by clarsimp
    from valid-own have own-dist: ownership-distinct ts tsb
        by (simp add: valid-ownership-def)
    from valid-sharing have sharing-consis }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\textrm{ts}}{\textrm{sb}}{
        by (simp add: valid-sharing-def)
    from sharing-consistent-share-all-until-volatile-write [OF own-dist this i-bound ts sb-i]
    have sc: sharing-consistent }\mathcal{S}\mathrm{ (acquired True ?take-sb }\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{}\mathrm{ ) ?drop-sb
        by (simp add: S 
    from sharing-consistent-share-all-shared
    have dom (share ?drop-sb S\mathcal{S}}\subseteq\operatorname{dom}\mathcal{S}\cup\mathrm{ all-shared ?drop-sb
        by auto
    also from sharing-consistent-all-shared [OF sc]
    have ...\subseteq dom S U acquired True ?take-sb }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mathrm{ by auto
    also from acquired-all-acquired all-acquired-takeWhile
    have \ldots\subseteq dom S \cup (\mathcal{O}
    finally
    have a-shared: a }\in\operatorname{dom}\mathcal{S
        using a-share a-not-owns
        by auto
            with share-all-until-volatile-write-thread-local [OF ownership-distinct-ts sb
sharing-consis-ts sb j-bound jth j-owns]
            have a-dom: a }\in\mathrm{ dom (share ?take-sb j}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}
                            by (auto simp add: }\mathcal{S}\mathrm{ domIff)
                            from release-not-unshared-no-write-take [OF unshared-take no-unsharing
a-dom] a-in
            have False by auto
    }
ultimately show False
    using access-cond'
    by auto
        qed
```

```
    }
    thus ?thesis
        by (fastforce simp add: Let-def)
    qed
    from flush-all-until-volatile-write-buffered-val-conv
    [OF True i-bound ts ssb-i this]
    show ?thesis
    by (simp add: buf-None)
qed
hence m-a:m a = msb a
    by (simp add: m)
have tmps-commute: }\mp@subsup{\vartheta}{\textrm{sb}}{}(\textrm{t}\mapsto(\mp@subsup{\textrm{m}}{\textrm{sb}}{}\textrm{a}))
    (\mp@subsup{\vartheta}{\mathrm{ sb }}{\prime}\mp@subsup{|}{}{\prime}(\operatorname{dom}\mp@subsup{\vartheta}{\mathrm{ sb }}{}-{t}))(t\mapsto( (msb a))
    apply (rule ext)
    apply (auto simp add: restrict-map-def domIff)
    done
from suspend-nothing
have suspend-nothing':(dropWhile (Not \circ is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{})\mp@subsup{\textrm{sb}}{}{\prime})=[
    by (simp add: sb)
from \(\mathcal{D}\)
have \(\mathcal{D}^{\prime}: \mathcal{D}_{\text {sb }}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \(\mathrm{e}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.\) volatile at \(\left(\mathrm{m}_{\mathrm{sb}}\right.\) a)]) \(\neq\{ \}\) )
by (auto simp: outstanding-refs-append)
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),(), \mathcal{D}\right.\right.\right.\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),release ?take-sb (dom \(\left.\left.\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)\)
apply (rule sim-config.intros)
apply (simp add: m flush-commute \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) )
using share-all-until-volatile-write-Read-commute [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) [simplified is \(\mathrm{s}_{\mathrm{sb}}\) ]]
apply (simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
using leq
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
using i-bound i-bound' ts-sim ts-i True \(\mathcal{D}^{\prime}\)
apply (clarsimp simp add: Let-def nth-list-update
outstanding-refs-conv m-a \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) suspend-nothing \({ }^{\prime}\)
flush-all acquired-append release-append
split: if-split-asm )
apply (rule tmps-commute)
done
ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' \({ }^{\prime}\) valid-sops' valid-dd' load-tmps-fresh' enough-flushs' valid-program-history' valid \({ }^{\prime}\)
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\text {sb }}{ }^{\prime}\)
```

by (auto simp del: fun-upd-apply )
next
case False
then obtain $r$ where $r$-in: $r \in$ set sb and volatile-r: is-volatile-Write ${ }_{s b} r$
by (auto simp add: outstanding-refs-conv)
from takeWhile-dropWhile-real-prefix
[OF r-in, of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ), simplified, OF volatile-r]
obtain $\mathrm{a}^{\prime} \mathrm{v}^{\prime} \mathrm{sb}^{\prime \prime}$ sop ${ }^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
sb-split: $\mathrm{sb}=$ takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime}$
$\mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{sb}^{\prime \prime}$
and
drop: dropWhile (Not o is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}=\mathrm{Write}_{\mathrm{sb}}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ sb"
apply (auto)
subgoal for y ys
apply (case-tac y)
apply auto
done
done
from drop suspends have suspends: suspends $=$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ $\mathrm{sb}^{\prime \prime}$
by $\operatorname{simp}$
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})$ by auto

## moreover

note flush-commute $=$ flush-all-until-volatile-write-Read-commute [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - i [simplified is sbb ] ]
have Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \in$ set sb by (subst sb-split) auto
from dropWhile-append 1 [OF this, of (Not $\circ$ is-volatile-Write $\left.{ }_{\mathbf{s b}}\right)$ ]
have drop-app-comm:
$\left(\operatorname{dropWhile}\left(\right.\right.$ Not is-volatile-Write $\left.{ }_{s b}\right)\left(\operatorname{sb} @\left[\operatorname{Read}_{s b}\right.\right.$ volatile a t $\left.\left.\left.\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]\right)\right)=$ dropWhile (Not o is-volatile-Write ${ }_{s b}$ ) sb @ $\left[\operatorname{Read}_{s b}\right.$ volatile a t ( $\left.\left.\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right]$
by $\operatorname{simp}$
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have $\mathrm{t} \notin \operatorname{dom} \vartheta_{\text {sb }}$
by (auto simp add: is sb )
then have tmps-commute:
$\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\right.$ read-tmps sb$\left.{ }^{\prime \prime}\right)=$
$\left.\vartheta_{\text {sb }}\right|^{\prime}\left(\right.$ dom $\vartheta_{\text {sb }}-$ insert t $\left(\right.$ read-tmps sb $\left.\left.{ }^{\prime \prime}\right)\right)$
apply -
apply (rule ext)
apply auto

## done

from $\mathcal{D}$
have $\mathcal{D}^{\prime}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.$ outstanding-refs is-volatile-Write $\mathrm{e}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.$ volatile at $\left(\mathrm{m}_{\mathrm{sb}}\right.$ a)]) $\neq\{ \}$ )
by (auto simp: outstanding-refs-append)

```
have ( }\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime},\mp@subsup{\textrm{m}}{\textrm{sb}}{},\mathcal{S}\mp@subsup{\mathcal{S}}{\textrm{sb}}{})~(\textrm{ts},\textrm{m},\mathcal{S}
    apply (rule sim-config.intros)
    apply (simp add: m flush-commute ts sb
    using share-all-until-volatile-write-Read-commute [OF i-bound ts sbb-i [simplified is isb
```



```
    using leq
    apply (simp add: ts sb
    using i-bound i-bound' ts-sim ts-i is-sim (\mathcal{D}
    apply (clarsimp simp add: Let-def nth-list-update is-sim drop-app-comm
        read-tmps-append suspends prog-instrs-append-Read
        hd-prog-append-Read sb
        drop is }\mp@subsup{\textrm{sb}}{\textrm{sb}}{}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{\vartheta}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{D}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mathrm{ acquired-append takeWhile-append1 [OF r-in]
volatile-r split: if-split-asm)
    apply (simp add: drop tmps-commute)+
    done
```

ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-dd'
valid-sops ${ }^{\prime}$ load-tmps-fresh' enough-flushs ${ }^{\prime}$
valid-program-history ${ }^{\prime}$ valid ${ }^{\prime}$
$\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$
by (auto simp del: fun-upd-apply )
qed
next
case (SBHWriteNonVolatile a D f A L R W)
then obtain
$\mathrm{is}_{\mathrm{sb}}$ : $\mathrm{is}_{\mathrm{sb}}=$ Write False a (D, f) A L R W\# $\mathrm{is}_{\mathrm{sb}}{ }^{\prime}$ and
$\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}$ and
$\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}$ and
$\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}$ and
$\mathcal{D}_{\mathrm{sb}}: \mathcal{D}_{\mathrm{sb}}{ }^{\prime}=\mathcal{D}_{\mathrm{sb}}$ and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\right.$ Write $_{\text {sb }}$ False a (D, f) (f $\vartheta_{\mathrm{sb}}$ ) A L R W] and
$\mathrm{m}_{\mathrm{sb}}^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}$ and
$\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}$
by auto
from data-dependency-consistent-instrs [OF i-bound $\mathrm{ts}_{\mathrm{sb}} \mathrm{-}$ ]
have D-tmps: $\mathrm{D} \subseteq \operatorname{dom} \vartheta_{\mathrm{sb}}$
by (simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from safe-memop-flush-sb [simplified is sb ]
obtain a-owned': a $\in$ acquired True sb $\mathcal{O}_{\text {sb }}$ and a-unshared': a $\notin$ dom (share ?drop-sb $\mathcal{S})$ and rels-cond: $\forall \mathrm{j}<$ length ts. $\mathrm{i} \neq \mathrm{j} \longrightarrow \mathrm{a} \notin \operatorname{dom}($ released (ts! j$))$
by cases auto
from a-owned ${ }^{\prime}$ acquired-all-acquired
have a-owned ${ }^{\prime \prime}: a \in \mathcal{O}_{\text {sb }} \cup$ all-acquired $s b$ by auto

```
    {
```

fix j
fix $\mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}$
assume $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
have $\mathrm{a} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof -
from ownership-distinct [OF i-bound j-bound neq-i-j ts sbb $-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ] a-owned ${ }^{\prime \prime}$
show ?thesis
by auto
qed
$\}$ note a-unowned-others $=$ this

```
have a-unshared: a \(\notin \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
proof
assume a-share: a \(\in \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
from valid-sharing have sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\)
by (simp add: valid-sharing-def)
from in-shared-sb-share-all-until-volatile-write [OF this i-bound \(\mathrm{ts}_{\mathbf{s b}^{\mathrm{b}} \text {-i }}\) a-owned \({ }^{\prime \prime}\) a-share]
have a \(\in \operatorname{dom}\) (share ?drop-sb \(\mathcal{S}\) )
    by (simp add: \(\mathcal{S}\) )
with a-unshared'
show False
by auto
    qed
```

        have valid-own': valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
        proof (intro-locales)
    show outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have non-volatile-owned-or-read-only False $\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }} \mathrm{sb}$.
with a-owned ${ }^{\prime}$
have non-volatile-owned-or-read-only False $\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ False a $(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right)$ A L R W]) by (simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}$ )

## qed

next
show outstanding-volatile-writes-unowned-by-others $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof -
have outstanding-refs is-volatile-Write ${ }_{\text {sb }}\left(\mathrm{sb} @\left[\mathrm{Write}_{\mathrm{sb}}\right.\right.$ False a (D,f) (f $\left.\left.\left.\vartheta_{\mathrm{sb}}\right) \mathrm{A} L R \mathrm{R}\right]\right)$ by (auto simp add: outstanding-refs-append)
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ this]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ all-acquired-append)
qed next
show read-only-reads-unowned $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof -
have $r$ : read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\left(\mathrm{sb} @\left[\mathrm{Write}_{\text {sb }}\right.\right.$ False a $\left.\left.\left.\left.(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{ALR} \mathrm{W}\right]\right)\right) \mathcal{O}_{\mathrm{sb}}\right)$ (dropWhile (Not o is-volatile-Writesb) (sb @ [Write ${ }_{s b}$ False a (D,f) (f $\vartheta_{s b}$ ) A L R
W])) $\subseteq$ read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\mathcal{O}_{\text {sb }}$ ) (dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb)
apply (case-tac outstanding-refs (is-volatile-Write ${ }_{\text {sb }}$ ) sb $=\{ \}$ ) apply (simp-all add: outstanding-vol-write-take-drop-appends acquired-append read-only-reads-append )
done
have $\mathcal{O}_{\text {sb }} \cup$ all-acquired $\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ False a $\left.\left.(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{A} \mathrm{L} R \mathrm{~W}\right]\right) \subseteq \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired sb by (simp add: all-acquired-append)
from read-only-reads-unowned-nth-update [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{r}$ this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
next
show ownership-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - ]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}$ ' all-acquired-append)
qed
qed
have valid-hist ${ }^{\prime}$ : valid-history program-step $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from valid-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ ]
have history-consistent $\vartheta_{\mathrm{sb}}$ (hd-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}$ ) sb.
with valid-write-sops [OF i-bound $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]$ D-tmps
valid-implies-valid-prog-hd [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ valid]
have history-consistent $\vartheta_{s b}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\mathrm{sb}}$ False a (D,f) (f $\left.\vartheta_{\mathrm{sb}}\right)$ A L R W]) ) (sb@ [Write ${ }_{\text {sb }}$ False a $\left.\left.(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{ALRW}\right]\right)$
apply -
apply (rule history-consistent-appendI)
apply (auto simp add: hd-prog-append-Write ${ }_{\text {sb }}$ )
done
from valid-history-nth-update [OF i-bound this]
show ?thesis by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}$ )
qed
have valid-reads': valid-reads $\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof -
from valid-reads [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have reads-consistent False $\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}$.
from reads-consistent-snoc-Write ${ }_{\text {sb }}$ [OF this]
have reads-consistent False $\mathcal{O}_{s b} \mathrm{~m}_{\mathrm{sb}}$ ( sb @ [Write sb $_{\text {b }}$ False a (D,f) (f $\vartheta_{\mathrm{sb}}$ ) A L R W]).
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
qed
have valid-sharing': valid-sharing $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound $\mathrm{ts}_{\mathrm{sb}-\mathrm{i} \text { ] a-unshared }}$
have non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}$
(sb @ [Write ${ }_{\text {sb }}$ False a (D,f) (f $\vartheta_{\text {sb }}$ ) A L R W])
by (auto simp add: non-volatile-writes-unshared-append)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared $\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$ next
from sharing-consis [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}$.
then
have sharing-consistent $\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\left(\mathrm{sb} @\left[\mathrm{Write}_{\text {sb }}\right.\right.$ False a (D,f) (f $\left.\left.\left.\vartheta_{\mathrm{sb}}\right) \mathrm{A} L \mathrm{R} W\right]\right)$
by (simp add: sharing-consistent-append)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$ next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
]
show read-only-unowned $\mathcal{S}_{\text {sb }}{ }^{\prime}$ ts $_{\text {sb }}{ }^{\prime}$
by (simp add: $\left.\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\right)$
next
from unowned-shared-nth-update [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}$-i subset-refl]
show unowned-shared $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)$
next
from a-unshared
have a $\notin$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
by (auto simp add: read-only-def dom-def)
with no-outstanding-write-to-read-only-memory [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$-i]
have no-write-to-read-only-memory $\mathcal{S}_{\text {sb }}$ ( sb @ [Write ${ }_{\text {sb }}$ False a (D,f) (f $\vartheta_{\mathrm{sb}}$ ) A L R W]) by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
have tmps-distinct': tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
from load-tmps-distinct [OF i-bound $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]$
have distinct-load-tmps is sb $^{\prime}{ }^{\prime}$
by (auto split: instr.splits simp add: is sb )
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct ts sb $^{\prime}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
next
from read-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have distinct-read-tmps sb.
hence distinct-read-tmps (sb @ [Write ${ }_{\text {sb }}$ False a (D,f) (f $\vartheta_{\text {sb }}$ ) A L R W])
by (simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
next

load-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have load-tmps is sb $^{\prime}{ }^{\prime} \cap$ read-tmps (sb @ $\left[\right.$ Write $_{\text {sb }}$ False a $\left.\left.(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{A} \mathrm{L} R \mathrm{R}\right]\right)=\{ \}$
by (clarsimp simp add: read-tmps-append is sb )
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}$ )
qed
have valid-sops': valid-sops $\mathrm{ts}_{\text {sb }}{ }^{\prime}$ proof -
from valid-store-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}} \mathrm{i}$ ]
obtain valid-Df: valid-sop ( $\mathrm{D}, \mathrm{f}$ ) and
valid-store-sops ${ }^{\prime}: \forall$ sop $\in$ store-sops is $_{\text {sb }}{ }^{\prime}$. valid-sop sop by (auto simp add: is sb )
from valid-Df valid-write-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have valid-write-sops ${ }^{\prime}: \forall \mathrm{sop} \in$ write-sops $\left(\mathrm{sb} @\left[W_{r i t e}{ }_{\text {sb }}\right.\right.$ False a (D, f) (f $\left.\vartheta_{\text {sb }}\right)$ A L R W]).
valid-sop sop
by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound valid-write-sops' valid-store-sops']
show ?thesis
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
qed
have valid-dd': valid-data-dependency $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from data-dependency-consistent-instrs [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
obtain D-indep: $\mathrm{D} \cap$ load-tmps is $_{\text {sb }}{ }^{\prime}=\{ \}$ and
dd-is: data-dependency-consistent-instrs $\left(\operatorname{dom} \vartheta_{\mathrm{sb}}\right)$ is $\mathrm{sb}^{\prime}{ }^{\prime}$
by (auto simp add: is sb $\vartheta_{\text {sb }}$ )
from load-tmps-write-tmps-distinct [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text {-i }}$ ] D-indep
have load-tmps is sb $^{\prime} \cap$
$\bigcup\left(\right.$ fst ${ }^{\prime}$ write-sops $\left(\operatorname{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ False a $(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\text {sb }}\right)$ A L R W]) $)=\{ \}$
by (auto simp add: write-sops-append is sb $_{\text {sb }}$ )
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}$ )
qed
have load-tmps-fresh': load-tmps-fresh $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof -
from load-tmps-fresh [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}} \mathrm{i}$ ]
have load-tmps is sb $^{\prime} \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}$
by (auto simp add: is s $_{\text {sb }}$ )
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis
by ( $\operatorname{simp}$ add: $\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\right)$
qed
have enough-flushs': enough-flushs $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ proof -
from clean-no-outstanding-volatile-Write ${ }_{s b}\left[\mathrm{OF}\right.$ i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}$ ]
have $\neg \mathcal{D}_{\text {sb }} \longrightarrow$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}\left(\mathrm{sb} @\left[\right.\right.$ Write $_{\text {sb }}$ False a $(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\text {sb }}\right) \mathrm{A}$
$\mathrm{L} R \mathrm{~W}])=\{ \}$
by (auto simp add: outstanding-refs-append )
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\text {sb }}$ )
qed
have valid-program-history ${ }^{\prime}$ : valid-program-history $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof -
from valid-program-history [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}$ ]
have causal-program-history is $\mathrm{s}_{\mathrm{sb}} \mathrm{sb}$.
then have causal': causal-program-history is sb $^{\prime}$ ( $\mathrm{sb} @\left[\mathrm{Write}_{\text {sb }}\right.$ False a (D,f) (f $\vartheta_{\text {sb }}$ ) A L R W])

```
    by (auto simp: causal-program-history-Write \(\mathrm{is}_{\mathrm{sb}}\) )
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb}\right.\) @ \(\left[\mathrm{Write}_{\mathrm{sb}}\right.\) False a \(\left.\left.(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{ALR} W\right]\right)=\mathrm{p}_{\mathrm{sb}}\)
    by (simp add: last-prog-append-Write \({ }_{\text {sb }}\) )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
    by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
        qed
    from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) - i , rule-format]
    have valid-sop (D,f) by (auto simp add: is stb )
    then interpret valid-sop ( \(\mathrm{D}, \mathrm{f}\) ) .
    show ?thesis
    proof (cases outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}=\{ \}\) )
case True
from True have flush-all: takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}=\mathrm{sb}\)
    by (auto simp add: outstanding-refs-conv)
from True have suspend-nothing: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}=[]\)
    by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends \(=[]\)
    by (simp add: suspends)
from suspends-empty is-sim have is: is = Write False a (D,f) A L R W\# is st \(^{\prime}{ }^{\prime}\)
    by (simp add: is sb )
with suspends-empty ts-i
have ts-i: \(\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}\right.\), Write False a (D,f) A L R W\# is \({ }_{\mathrm{sb}}{ }^{\prime}\),
                        \(\vartheta_{\mathrm{sb}},()\),
                        \(\mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb \(\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}\right)\)
    by simp
from direct-memop-step. WriteNonVolatile [OF ]
have (Write False a (D, f) A L R W\# is \(\mathrm{sb}^{\prime}\),
    \(\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}\),acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),release ?take-sb \(\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\)
        (is \({ }_{\text {sb }}{ }^{\prime}\),
            \(\vartheta_{\mathrm{sb}},(), \mathrm{m}\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right), \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),
            release ?take-sb \(\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\)
            \(\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}\right.\right.\right.\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),
            release ?take-sb \(\left.\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}\right)\right]\),
    \(\left.\mathrm{m}\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right), \mathcal{S}\right)\).
```

moreover

```
have \(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
                (let \(\left(-,-,-, \mathrm{sb}_{\mathrm{j},-,-,-)}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\)
            in \(\mathrm{a} \notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )
\(\left.\mathrm{sb}_{\mathrm{j}}\right)\) )
    proof -
    \{
        fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
        assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
        assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
        assume \(j\) th: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
    have a \(\notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )
\(\mathrm{sb}_{\mathrm{j}}\) )
    proof
            assume a-in: a \(\in\) outstanding-refs \(^{\text {is-non-volatile-Write }}\) sb (takeWhile (Not o
is-volatile-Write \(\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\) )
            hence \(a \in\) outstanding-refs is-non-volatile-Write \({ }_{s b} \operatorname{sb}_{j}\)
            using outstanding-refs-append [of is-non-volatile-Write sb (takeWhile (Not o
is-volatile-Write \(\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\) )
    (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{\mathrm{j}}\) )]
    by auto
            with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
            [OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]]
            have j-owns: \(a \in \mathcal{O}_{j} \cup\) all-acquired sb \(_{j}\)
    by auto
            from j-owns a-owned \({ }^{\prime \prime}\) ownership-distinct [OF i-bound j-bound neq-i-j ts \(\mathrm{sb}_{\mathrm{sb}}-\mathrm{i} j\) th]
            show False
    by auto
        qed
    \}
    thus ?thesis by (fastforce simp add: Let-def)
qed
note flush-commute \(=\) flush-all-until-volatile-write-append-non-volatile-write-commute
        [OF True i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) this]
from suspend-nothing
have suspend-nothing': (dropWhile (Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{s b}\right) \operatorname{sb}^{\prime}\right)=[]\)
    by ( \(\operatorname{simp}\) add: \(\mathrm{sb}^{\prime}\) )
```


## from $\mathcal{D}$

```
have \(\mathcal{D}^{\prime}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) \(\left(\operatorname{sb} @\left[\right.\right.\) Write \(_{\text {sb }}\) False a (D,f) (f \(\left.\vartheta_{\text {sb }}\right)\) A L R W] \(\left.) \neq\{ \}\right)\) by (auto simp: outstanding-refs-append)
```

```
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\)
```

have $\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim$
$\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}\right.\right.\right.$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$,
$\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}\right.\right.\right.$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$,
release ?take-sb $\left.\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}\right)\right]$,
release ?take-sb $\left.\left.\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{sb}}\right)\right]$,
$\left.\mathrm{m}\left(\mathrm{a}:=\mathrm{f} \boldsymbol{\vartheta}_{\mathrm{sb}}\right), \mathcal{S}\right)$

```
        \(\left.\mathrm{m}\left(\mathrm{a}:=\mathrm{f} \boldsymbol{\vartheta}_{\mathrm{sb}}\right), \mathcal{S}\right)\)
```

```
apply (rule sim-config.intros)
apply (simp add: m flush-commute \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \boldsymbol{\vartheta}_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) )
using share-all-until-volatile-write-Write-commute
    [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\left[\right.\) simplified \(\mathrm{is}_{\mathrm{sbb}}\) ].
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \boldsymbol{v}_{\mathrm{sb}}{ }^{\prime}\) )
using leq
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
using i-bound i-bound \({ }^{\prime}\) ts-sim ts-i True \(\mathcal{D}^{\prime}\)
apply (clarsimp simp add: Let-def nth-list-update
outstanding-refs-conv \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) suspend-nothing \({ }^{\prime}\) flush-all
acquired-append release-append split: if-split-asm)
done
ultimately
show ?thesis
    using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-sops'
    valid-dd' load-tmps-fresh' enough-flushs \({ }^{\prime}\)
    valid-program-history' valid \(^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
    next
case False
then obtain \(r\) where \(r\)-in: \(r \in\) set sb and volatile-r: is-volatile-Write \({ }_{s b} r\)
    by (auto simp add: outstanding-refs-conv)
from takeWhile-dropWhile-real-prefix
[OF r-in, of (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ), simplified, OF volatile-r]
obtain \(\mathrm{a}^{\prime} \mathrm{v}^{\prime} \mathrm{sb}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
    sb-split: sb \(=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb @ Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime}\)
R' W'\# sb"
    and
    drop: dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \(=\) Write \(_{\text {sb }}\) True \(^{\prime} \mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\)
\(\mathrm{sb}^{\prime \prime}\)
    apply (auto)
    subgoal for y ys
    apply (case-tac y)
    apply auto
    done
    done
    from drop suspends have suspends: suspends \(=\) Write \({ }_{\text {sb }}\) True \(a^{\prime} \operatorname{sop}^{\prime} v^{\prime} A^{\prime} L^{\prime} R^{\prime} W^{\prime} \#\)
sb"
    by simp
```

    have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by auto
    moreover
    note flush-commute $=$
flush-all-until-volatile-write-append-unflushed [OF False i-bound $\mathrm{ts}_{\mathrm{sb}}$ - ]
have Write ${ }_{\text {sb }}$ True $a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \in$ set sb
by (subst sb-split) auto
note drop-app $=$ dropWhile-append1 [OF this, of (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ), simplified]
from $\mathcal{D}$
 $\left.\vartheta_{\mathrm{sb}}\right)$ A L R W] $) \neq\{ \}$ ) by (auto simp: outstanding-refs-append)

```
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
    apply (rule sim-config.intros)
    apply (simp add: m flush-commute \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \boldsymbol{\vartheta}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
    using share-all-until-volatile-write-Write-commute
            [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) [simplified is \(\mathrm{s}_{\mathrm{sb}}\) ]]
    apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \boldsymbol{\vartheta}_{\mathrm{sb}}{ }^{\prime}\) )
    using leq
    apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
    using i-bound i-bound \({ }^{\prime}\) ts-sim ts-i is-sim \(\mathcal{D}^{\prime}\)
    apply (clarsimp simp add: Let-def nth-list-update is-sim drop-app
        read-tmps-append suspends
        prog-instrs-append-Write \({ }_{s b}\) instrs-append-Write \(_{s b}\) hd-prog-append-Write \({ }_{s b}\)
        drop is \(\mathrm{sbb} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
            \(\vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) acquired-append takeWhile-append 1 [OF r-in]
        volatile-r
        split: if-split-asm)
    done
ultimately show ?thesis
    using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-dd'
        valid-sops' load-tmps-fresh' enough-flushs'
        valid-program-history' valid' \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
    by (auto simp del: fun-upd-apply )
        qed
    next
        case (SBHWriteVolatile a D f A L R W)
        then obtain
\(\mathrm{is}_{\mathrm{sb}}: \mathrm{is}_{\mathrm{sb}}=\) Write True a (D, f) A L R W\# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and
        \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}\) and
\(\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}\) and
\(\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \mathcal{D}_{\mathrm{sb}}{ }^{\prime}=\) True and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\right.\) Write \(_{\mathrm{sb}}\) True a (D, f) (f \(\vartheta_{\mathrm{sb}}\) ) A L R W] and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\)
by auto
```

    from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
    have D-subset: \(\mathrm{D} \subseteq \operatorname{dom} \vartheta_{\text {sb }}\)
    by (simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from safe-memop-flush-sb [simplified is s sb ] obtain
a-unowned-others-ts:
$\forall \mathrm{j}<$ length $($ map owned ts$) . \mathrm{i} \neq \mathrm{j} \longrightarrow(\mathrm{a} \notin$ owned $(\mathrm{ts}!\mathrm{j}) \cup$ dom (released $(\mathrm{ts}!\mathrm{j})))$ and

L-subset: $\mathrm{L} \subseteq \mathrm{A}$ and
A-shared-owned: A $\subseteq \operatorname{dom}$ (share ? drop-sb $\mathcal{S}$ ) $\cup$ acquired True sb $\mathcal{O}_{\text {sb }}$ and
R -acq: $\mathrm{R} \subseteq$ acquired True sb $\mathcal{O}_{\text {sb }}$ and
$A-R: A \cap R=\{ \}$ and
A-unowned-by-others-ts:
$\forall \mathrm{j}<$ length $($ map owned ts$) . \mathrm{i} \neq \mathrm{j} \longrightarrow(\mathrm{A} \cap($ owned $(\mathrm{ts}!\mathrm{j}) \cup$ dom $($ released $(\mathrm{ts}!\mathrm{j})))=\{ \})$ and
a-not-ro': a $\notin$ read-only (share ?drop-sb $\mathcal{S}$ )
by cases auto
from a-unowned-others-ts ts-sim leq
have a-unowned-others:

$$
\begin{aligned}
& \forall \mathrm{j}<{\text { length } \mathrm{ts}_{\mathrm{sb}} \cdot \mathrm{i}}_{\mathrm{i} \neq \mathrm{j} \longrightarrow}^{\left(\operatorname{let}\left(-,-,-,-\mathrm{bb}_{\mathrm{j}}^{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right. \text { in }}
\end{aligned}
$$

$\mathrm{a} \notin$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}} \wedge$
$\mathrm{a} \notin$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ))
apply (clarsimp simp add: Let-def)
subgoal for j
apply (drule-tac $\mathrm{x}=\mathrm{j}$ in spec)
apply (auto simp add: dom-release-takeWhile)
done
done
have a-not-ro: a $\notin$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
proof
assume a: a $\in$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
from local.read-only-unowned-axioms have read-only-unowned $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$.
from in-read-only-share-all-until-volatile-write ${ }^{\prime}$ [OF ownership-distinct-ts ${ }_{\text {sb }}$ shar-
ing-consis-ts ${ }_{\text {sb }}$
〔read-only-unowned $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\mathrm{sb}}$ 〉 i-bound $\mathrm{ts}_{\mathrm{sb}}$-i a-unowned-others a]
have a $\in$ read-only (share ?drop-sb $\mathcal{S}$ )
by (simp add: $\mathcal{S}$ )
with a-not-ro' show False by simp
qed
from A-unowned-by-others-ts ts-sim leq
have A-unowned-by-others:
$\forall \mathrm{j}<$ length $\mathrm{ts}_{\text {sb }} . \mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.$ let $\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{s}}!\mathrm{j}$
in $\mathrm{A} \cap$ (acquired True (takeWhile (Not $\circ$ is-volatile-Write $\mathrm{e}_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}} \cup$
all-shared $\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)\right)=\{ \}\right)$
apply (clarsimp simp add: Let-def)
subgoal for j
apply (drule-tac $x=j$ in spec)
apply (force simp add: dom-release-takeWhile)

```
done
    done
        have a-not-acquired-others: }\forall\textrm{j}<l=\mp@code{length (map \mathcal{O}
        (let (\mathcal{O}
    proof -
{
    fix j O}\mp@subsup{\mathcal{j}}{\textrm{jb}}{\textrm{j}
    assume j-bound: j < length (map owned ts sb)
    assume neq-i-j: i\not=j
    assume ts }\mp@subsup{\textrm{sb}}{\textrm{sb}}{\textrm{j}}\mathbf{j}:(\operatorname{map}\mathcal{O}-\textrm{sb ts
    assume conflict: a }\in\mathrm{ all-acquired sbj
    have False
    proof -
    from j-bound leq
    have j-bound': j < length (map owned ts)
        by auto
    from j-bound have j-bound '/: j < length ts sb
        by auto
    from j-bound' have j-bound ''': j < length ts
        by simp
    let ?take-sb }\mp@subsup{j}{j}{}=(\mp@subsup{\mathrm{ takeWhile (Not o is-volatile-Write }}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{}
    let ?drop-sb}\mp@subsup{j}{j}{}=(\mathrm{ dropWhile (Not }\circ\mathrm{ is-volatile-Write }\mp@subsup{\textrm{Sb}}{\mathbf{b}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{}
    from ts-sim [rule-format, OF j-bound'] ts tsb-j j-bound '/
    obtain pj suspendsj is sbj }\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\mathrm{ sbj }}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\vartheta}{\mathrm{ sbj }}{}\mathrm{ is is where
    ts sb-j: ts stb !j = (pj, is sbj
```



```
    isj: instrs suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{@}\mp@subsup{\textrm{is}}{\textrm{sbj}}{}= i\mp@subsup{\textrm{i}}{\textrm{j}}{}@ prog-instrs suspendsj and
            \mathcal{D}}\textrm{j}:\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}=(\mp@subsup{\mathcal{D}}{\textrm{j}}{}\vee\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathrm{ sb }}{\textrm{sb}
    tsj: ts!j = (hd-prog p puspendsj, is 
                            \varthetasbj |
                \mathcal{D}}\mp@subsup{\textrm{j}}{}{\prime
                    acquired True ?take-sb
                        release ?take-sb
        apply (cases ts sb!j)
        apply (force simp add: Let-def)
        done
```

    from a-unowned-others [rule-format,OF - neq-i-j] ts \(\mathrm{sb}_{\mathrm{sb}}-\mathrm{j}\) j-bound
        obtain a-unacq: a \(\notin\) acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) and a-not-shared: a \(\notin\) all-shared
    ?take-sb
by auto
have conflict-drop: $\mathrm{a} \in$ all-acquired suspends $_{\mathrm{j}}$
proof (rule ccontr)
assume a $\notin$ all-acquired suspends ${ }_{j}$
with all-acquired-append [of ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ] conflict
have a $\in$ all-acquired ?take-sb ${ }_{j}$

```
    by (auto simp add: suspendsj)
    from all-acquired-unshared-acquired [OF this a-not-shared] a-unacq
    show False by auto
        qed
```

    from j-bound \({ }^{\prime \prime \prime}\) i-bound \({ }^{\prime}\) have j -bound-ts': \(\mathrm{j}<\) length ?ts \({ }^{\prime}\)
        by simp
    from split-all-acquired-in [OF conflict-drop]
    show ?thesis
    proof
        assume \(\exists\) sop \(a^{\prime}\) v ys zs A L R W.
        suspends \(_{\mathrm{j}}=\) ys @ Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v A L R W\# zs \(\wedge a \in A\)
    then
    obtain \(a^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime}\) ys zs \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
    split-suspendsj: suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Write ${ }_{\mathrm{sb}}$ True $\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$
(is suspends ${ }_{j}=$ ?suspends) and
$a-A^{\prime}: a \in A^{\prime}$
by blast
from sharing-consis [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have sharing-consis-j: sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
then have $A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}$
by (simp add: sharing-consistent-append [of - ? ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$, simplified]
suspends ${ }_{j}$ [symmetric] split-suspends ${ }_{j}$ sharing-consistent-append)
from valid-program-history [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\text {sb }}-\mathrm{j}$ ]
have causal-program-history $\mathrm{is}_{\text {sbj }} \mathrm{sb}_{\mathrm{j}}$.
then have cph: causal-program-history is $\mathrm{s}_{\mathrm{sbj}}$ ?suspends
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb ${ }_{j}$ ] )
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp add: split-suspends ${ }_{\mathrm{j}}$ )
done
from $t_{s j}$ neq-i-j j-bound
have ts' -j : ?ts $!\mathrm{j}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps suspends $\left.{ }_{j}\right),()$,
$\mathcal{D}_{\mathrm{j}}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$, release ? take-sb $\left.\mathrm{s}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
by auto
from valid-last-prog [OF j-bound ${ }^{\prime \prime}$ ts $_{\text {sb- }}$ j] have last-prog: last-prog $p_{j} s b l_{j}=p_{j}$.
then
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$
apply -
apply (rule last-prog-same-append [where $\mathrm{sb}=$ ? take-sb $\mathrm{s}_{\mathrm{j}}$ ])
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric suspends $_{\mathrm{j}}$ )
apply simp
done
from valid-reads [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have reads-consis-j: reads-consistent False $\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ )
j-bound ${ }^{\prime \prime}$ ts $_{\text {sb }}$-j this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb $\mathrm{b}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends ${ }_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $\mathrm{s}_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound ${ }^{\prime \prime}$ neq-i-j $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have outstanding-refs is-Write ${ }_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by ( $\operatorname{simp}$ add: suspends ${ }_{j}$ )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) (flush ?drop-sb m) suspends ${ }_{j}$.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) (ys@[Write sb True $\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}$ )
by ( $\operatorname{simp}$ add: split-suspends $\mathrm{s}_{\mathrm{j}}$ reads-consistent-append)
from valid-write-sops [OF j-bound ${ }^{\prime \prime}$ ts $_{\text {sb-j }}$ ]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops ( $y$ s $@\left[W_{r i t e}^{s b}\right.$ True $a^{\prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have distinct-read-tmps (?take-sb $@_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ )
by ( simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb $_{\mathrm{j}} \cap$ read-tmps suspends $\mathrm{t}_{\mathrm{j}}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have h-consis:
history-consistent $\vartheta_{\text {sbj }}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @_{\mathrm{s}}$ suspends $_{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ?take-sb ${ }_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\left._{\mathrm{j}}\right)$ proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ? take-sb $\mathrm{b}_{\mathrm{j}} @$ ? drop-sb $\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by simp
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb ${ }_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{p}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{t}_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends ${ }_{\mathrm{j}}$ ) suspends $\mathrm{j}_{\mathrm{j}}$
by (simp add: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{s b}$
$\left(y s @\left[W_{r i t e}^{s b}\right.\right.$ True $\mathrm{a}^{\prime}$ sop $\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}$
by (auto simp add: outstanding-refs-append suspends ${ }_{\mathrm{j}}$ [symmetric]
split-suspends $\mathrm{j}_{\mathrm{j}}$ )
have acq-simp:
acquired True (ys @ $\left[W^{2} i t e_{\text {sb }}\right.$ True $\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
(acquired True ? take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) $=$
acquired True ys (acquired True ? take-sb $\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
by (simp add: acquired-append)
from flush-store-buffer-append [where $s b=y s @\left[W r i t e e_{s b}\right.$ True $a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}$ ]
and $\mathrm{sb}^{\prime}=\mathrm{zs}$, simplified,
OF j-bound-ts ${ }^{\prime}$ is $_{\mathrm{j}}$ [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts'-j [simplified split-suspends ${ }_{\mathrm{j}}$ ] refl lp [simplified split-suspends ${ }_{\mathrm{j}}$ ] reads-consis-ys
hist-consis ${ }^{\prime}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends ${ }_{j}$ ]
no-volatile-Read ${ }_{\text {sb }}$-volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}]$
obtain is ${ }_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}^{\prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts', flush ?drop-sb m, share ?drop-sb $\mathcal{S}$ ) $\Rightarrow_{\mathrm{d}}{ }^{*}$
(?ts' ${ }^{[j}:=$ (last-prog
(hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop $\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\left(\mathrm{ys} @\left[\right.\right.$ Write $_{\text {sb }}$
True $\left.a^{\prime} \operatorname{sop}^{\prime} v^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ),

$$
\begin{aligned}
& \mathrm{is}_{\mathrm{j}}{ }^{\prime} \\
& \left.\vartheta_{\mathrm{sbj}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sbj}}-\text { read-tmps zs }\right),
\end{aligned}
$$

from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys! $i=\left(p_{s b}, i i_{s b}, \vartheta_{\mathrm{sb}},()\right.$,
$\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$, release sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
by simp
note conflict-computation $=$ rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is ${ }_{\text {sb }}$ ]
have a-unowned:
$\forall \mathrm{j}<$ length ?ts-ys. $\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.$ let $\left(\mathcal{O}_{\mathrm{j}}\right)=$ map owned ?ts-ys! j in a $\left.\notin \mathcal{O}_{\mathrm{j}}\right)$
apply cases
apply (auto simp add: Let-def is $_{\text {sb }}$ )
done
from a- $\mathrm{A}^{\prime}$ a-unowned [rule-format, of j ] neq-i-j j-bound ${ }^{\prime} \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
show False
by (auto simp add: Let-def)
next
assume $\exists \mathrm{A} L \mathrm{R} \mathrm{W}$ ys zs. suspends $\mathrm{s}_{\mathrm{j}}=\mathrm{ys} @$ Ghost $_{\mathrm{sb}} \mathrm{ALR} \mathrm{R} \# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A}$ then
obtain $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ ys zs where
split-suspends $\mathrm{j}_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Ghost ${ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$
(is suspends ${ }_{\mathrm{j}}=$ ?suspends) and

$$
a-A^{\prime}: a \in A^{\prime}
$$

by blast
from sharing-consis [OF j-bound ${ }^{\prime \prime}$ ts $_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}$ ]
have sharing-consis-j: sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
then have $A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}$
by (simp add: sharing-consistent-append [of - - ?take-sb ${ }_{\mathrm{j}}$ ? drop-sb $_{\mathrm{j}}$, simplified]
suspends ${ }_{j}$ [symmetric] split-suspends ${ }_{j}$ sharing-consistent-append)
from valid-program-history [OF j-bound ${ }^{\prime \prime}$ ts $_{\text {sb }}-\mathrm{j}$ ]
have causal-program-history is $_{s_{b j}} \mathrm{sb}_{\mathrm{j}}$.
then have cph: causal-program-history is $_{\text {sbj }}$ ?suspends
apply -
apply (rule causal-program-history-suffix [where $\mathrm{sb}=$ ? take-sb $_{\mathrm{j}}$ ])
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp add: split-suspends ${ }_{j}$ )
done
from $t s_{j}$ neq-i-j j-bound
have ts' ${ }^{\prime} \mathrm{j}$ : ?ts $!\mathrm{j}=\left(\mathrm{hd}-\right.$ prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}$, $\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps suspends $\left.{ }_{\mathrm{j}}\right),()$, $\mathcal{D}_{\mathrm{j}}$, acquired True ? ${ }^{\text {take-sb }} \mathrm{O}_{\mathrm{j}}$, release ? take-sb $\left._{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
by auto
from valid-last-prog [ OF j -bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathbf{s b}^{\mathrm{s}}} \mathrm{j}\right]$ have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$. then
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$
apply -
apply (rule last-prog-same-append [where sb=?take-sb ${ }_{\mathrm{j}}$ )
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric $^{2}$ suspends ${ }_{j}$ )
apply simp
done
from valid-reads [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have reads-consis-j: reads-consistent False $\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{j}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}$ ’ j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}$-j this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb $_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends $\mathrm{j}_{\mathrm{j}}$ by (simp add: m suspends $_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound" neq-i-j $\mathrm{ts}_{\mathrm{sb} \text { - }} \mathrm{ts}_{\mathrm{sb} \text { - }} \mathrm{j}$ ]
have outstanding-refs is-Write ${ }_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by (simp add: suspends ${ }_{\mathrm{j}}$ )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) (flush ?drop-sb m) suspendsj.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ )
(flush ? drop-sb m) (ys@[Ghost $\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ )
by (simp add: split-suspends ${ }_{j}$ reads-consistent-append)
from valid-write-sops [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j}$ @?suspends). valid-sop sop
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ? take-sb ${ }_{j}$. valid-sop sop and valid-sops-drop: $\forall$ sop $\in$ write-sops ( ${\mathrm{ys} @\left[\text { Ghost }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right] \text { ). valid-sop sop }}^{\text {a }}$
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have distinct-read-tmps (?take-sb ${ }_{j} @_{\text {suspends }}^{j}$ )
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb ${ }_{j} \cap$ read-tmps suspends ${ }_{j}=\{ \}$ and distinct-read-tmps-drop: distinct-read-tmps suspends ${ }_{j}$
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric] suspends $s_{j}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history $\left[\mathrm{OF} \mathrm{j}\right.$-bound ${ }^{\prime \prime}$ ts $_{\text {sb-j }}$ ]
have h -consis:

apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$ ) ?take-sb $\mathrm{j}_{\mathrm{j}}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.$ suspends $\mathrm{j}_{\mathrm{j}}$ ) proof -
from last-prog have last-prog $p_{j}\left(\right.$ ?take-sb ${ }_{j} @$ ? drop-sb $\left.{ }_{j}\right)=p_{j}$
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$ ) ?take-sb ${ }_{\mathrm{j}}=\mathrm{hd}$-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{j}_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{s}_{\mathrm{j}}=$ last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{s}_{\mathrm{j}}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{j}_{\mathrm{j}}$ )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) suspends $\mathrm{j}_{\mathrm{j}}$
by (simp add: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read ${ }_{\text {sb }}$

by (auto simp add: outstanding-refs-append suspends ${ }_{j}$ [symmetric]
split-suspendsj
have acq-simp:
acquired True (ys @ $\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \boldsymbol{\jmath}\right]$ )
(acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) $=$
acquired True ys (acquired True ? take-sb $\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
by (simp add: acquired-append)
from flush-store-buffer-append [where $\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Ghost}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ and $\mathrm{sb}{ }^{\prime}=\mathrm{zs}$, simplified,

OF j-bound-ts' is $_{\mathrm{j}}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts'-j [simplified split-suspendsj] reff lp [simplified split-suspendsj] reads-consis-ys
hist-consis ${ }^{\prime}$ [simplified split-suspends ${ }_{j}$ ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends ${ }_{j}$ ]
no-volatile-Read sb -volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ share ?drop-sb $\mathcal{S}]$
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @$ prog-instrs zs and
steps-ys: (?ts ${ }^{\prime}$, flush ?drop-sb m, share ?drop-sb $\left.\mathcal{S}\right) \Rightarrow_{\mathrm{d}}{ }^{*}$
(? $\mathrm{ts}^{\prime}[\mathrm{j}:=$ (last-prog
$\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.$ Ghost $\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\left(\mathrm{ys}^{@}\left[\operatorname{Ghost}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)$,
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$,
$\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\operatorname{dom} \vartheta_{\text {sbj }}\right.$ - read-tmps zs $)$,
(),
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write $\mathrm{sb}_{\mathrm{sb}}$ (ys @ [Ghost $\mathrm{sb}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime}$
$\left.\left.\mathrm{W}^{\prime}\right]\right) \neq\{ \}$, acquired True ys (acquired True ? take-sb $\mathrm{O}_{\mathrm{j}}$ ) $\left.\left.\cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]$,
flush (ys@[Ghost $\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]$ ) (flush ?drop-sb m),
share (ys@[Ghost $\left.\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)$ (share ?drop-sb $\left.\mathcal{S}\right)$ )
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}$, ?shared-ys) $)$
by (auto simp add: acquired-append)
from i-bound ${ }^{\prime}$ have i-bound-ys: $\mathrm{i}<$ length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys!i $=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.$,
$\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$, release sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)$
by simp
note conflict-computation $=$ rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this i-bound-ys ts-ys-i, simplified is ${ }_{\text {sb }}$ ]
have a-unowned:
$\forall \mathrm{j}<$ length ?ts-ys. $\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.$ let $\left(\mathcal{O}_{\mathrm{j}}\right)=$ map owned ?ts-ys! j in a $\left.\notin \mathcal{O}_{\mathrm{j}}\right)$
apply cases
apply (auto simp add: Let-def is sb )
done
from a-A' a-unowned [rule-format, of j] neq-i-j j-bound ${ }^{\prime} \mathrm{A}^{\prime}-\mathrm{R}^{\prime}$
show False
by (auto simp add: Let-def)
qed
qed
\}
thus ? thesis
by (auto simp add: Let-def)
qed
have A-unused-by-others:
$\forall \mathrm{j}<$ length $\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right) . \mathrm{i} \neq \mathrm{j} \longrightarrow$

```
            (let (\mathcal{O}}\mp@subsup{\textrm{j}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{})=\operatorname{map}\mathcal{O}-\textrm{sb}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}!\textrm{j
            in A \cap outstanding-refs is-volatile-Write sb sb
        proof -
{
    fix j ()
    assume j-bound: j < length (map owned tssb)
    assume neq-i-j: i\not=j
    assume ts sbb-j:(map \mathcal{O-sb ts sb})!j=(\mathcal{O}
    assume conflict: A \cap outstanding-refs is-volatile-Write sb sb }\mp@subsup{\textrm{sb}}{\textrm{j}}{}\not={
    have False
    proof -
        from j-bound leq
        have j-bound': j < length (map owned ts)
        by auto
    from j-bound have j-bound '/: j < length tssb
        by auto
    from j-bound' have j-bound ''\prime: j < length ts
        by simp
    from conflict obtain a' where
    a'-in: a'}\inA\mathrm{ and
            a}\mp@subsup{}{}{\prime}-\textrm{in}-\textrm{j}: \mp@subsup{a}{}{\prime}\in\mathrm{ outstanding-refs is-volatile-Write sb }\mp@subsup{\textrm{sb}}{\textrm{j}}{
        by auto
    let ?take-sb
    let ?drop-sb
    from ts-sim [rule-format, OF j-bound 'л ts tsb
    obtain p puspendsj is isbj }\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\vartheta}{\mathrm{ sbj }}{}\mathrm{ isj where
    ts sb-j: ts sbb}!j=(\mp@subsup{p}{j}{},\mp@subsup{\textrm{is}}{\textrm{sbj}}{},\mp@subsup{\vartheta}{\textrm{sbj}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{sbj}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{})\mathrm{ and
    suspendsj: suspendsj}=\mathrm{ ? drop-sb
    is}\mp@subsup{j}{j}{}: instrs suspendsj @ is csbj = is j @ prog-instrs suspends j and
    \mathcal{D}}\textrm{j}:\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}=(\mp@subsup{\mathcal{D}}{\textrm{j}}{}\vee\mathrm{ outstanding-refs is-volatile-Write}\mp@subsup{\textrm{sb}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\not={})\mathrm{ and
    tsj: ts!j = (hd-prog p puspends}\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{
                    \vartheta sbj |' (dom }\mp@subsup{\vartheta}{\mathrm{ sbj }}{}-\mathrm{ read-tmps suspendsj}),(),\mathcal{D
                acquired True ?take-sb }\mp@subsup{\mp@code{O}}{\textrm{j}}{\textrm{j}
                release ?take-sb 
    apply (cases ts sb!j)
    apply (force simp add: Let-def)
    done
    have a}\mp@subsup{a}{}{\prime}\in\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\mp@code{sb}}{}{\prime}\mp@subsup{\mathrm{ suspends }}{\textrm{j}}{
    proof -
        from a'-in-j
        have a' }\in\mathrm{ outstanding-refs is-volatile-Write 
by simp
    thus ?thesis
apply (simp only: outstanding-refs-append suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
apply (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
done
```


## qed

```
    from split-volatile-Write \({ }_{\text {sb }}\)-in-outstanding-refs [OF this]
    obtain sop v ys \(\mathrm{zs} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
    split-suspends \({ }_{j}\) : suspends \(j_{j}=y s @ W r i t e ~_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) zs (is suspends \({ }_{j}\)
\(=\) ?suspends)
by blast
```

    from direct-memop-step.WriteVolatile [where \(\vartheta=\vartheta_{\text {sb }}\) and \(\mathrm{m}=\) flush ?drop-sb m]
    have (Write True a (D, f) A L R W\# is \({ }_{\text {sb }}{ }^{\prime}\),
        \(\vartheta_{\mathrm{sb}},()\), flush ?drop-sb \(\mathrm{m}, \mathcal{D}_{\mathrm{sb}}\),acquired True \(\mathrm{sb} \mathcal{O}_{\mathrm{sb}}\),
        release \(\mathrm{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\),
        share ?drop-sb \(\mathcal{S}) \rightarrow\)
            \(\left(\mathrm{is}_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()\right.\), (flush ?drop-sb m)(a:=f \(\left.\vartheta_{\mathrm{sb}}\right)\), True, acquired True sb \(\mathcal{O}_{\mathrm{sb}} \cup\)
    A - R, Map.empty,
share ?drop-sb $\left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
from direct-computation.concurrent-step.Memop [OF

have store-step: (?ts ${ }^{\prime}$, flush ?drop-sb m,share ?drop-sb $\left.\mathcal{S}\right) \Rightarrow_{\mathrm{d}}$
$\left(? \mathrm{ts}^{\prime}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()\right.\right.\right.$,
True, acquired True sb $\mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}$,Map.empty)],
(flush ?drop-sb m) $\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right)$, share ?drop-sb $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
$\left(\right.$ is $-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}, ?$ share-A $\left.)\right)$
by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )
from i-bound' have i-bound ${ }^{\prime \prime}$ : i < length ?ts-A
by simp
from valid-program-history [OF j-bound ${ }^{\prime \prime}$ ts $_{\text {sbb }-\mathrm{j}}$ ]
have causal-program-history $\mathrm{is}_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}$.
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where $\mathrm{sb}=$ ? take-sb $_{\mathrm{j}}$ ])
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp add: split-suspends ${ }_{\mathrm{j}}$ )
done
from $t s_{j}$ neq-i-j j-bound
have ts-A-j: ?ts-A! $j=\left(h d-p r o g p_{j}\left(y s @ W_{i t e}\right.\right.$ sb True $\left.a^{\prime} \operatorname{sop} v A^{\prime} L^{\prime} R^{\prime} W^{\prime} \# z s\right)$, is ${ }_{j}$, $\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-r e a d-t m p s\left(y s @ W_{i t e}\right.\right.$ sb True $\left.\left.\mathrm{a}^{\prime} \operatorname{sop} v \mathrm{~A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),(), \mathcal{D}_{\mathrm{j}}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$, release ?take-sb ${ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}$ )
by ( $\operatorname{simp}$ add: split-suspends ${ }_{\mathrm{j}}$ )
from j-bound ${ }^{\prime \prime \prime}$ i-bound ${ }^{\prime}$ neq-i-j have j-bound ${ }^{\prime \prime \prime \prime}:$ j < length ?ts-A
by simp
from valid-last-prog [OF j-bound $\left.{ }^{\prime \prime} \operatorname{ts}_{\mathrm{sb}} \mathrm{j}\right]$ have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$.

```
then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ? suspends \(=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \({ }_{\mathrm{j}}\) ])
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
```

from valid-reads [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have reads-consis: reads-consistent False $\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ > j-bound ${ }^{\prime \prime}$
$\mathrm{ts}_{\mathrm{sb}}$-j reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) m suspends $\mathrm{m}_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $\mathrm{j}_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound ${ }^{\prime \prime}$ neq-i-j ts $\mathrm{s}_{\mathrm{sb}}-\mathrm{i}$ $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have outstanding-refs is-Write ${ }_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read sb $_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by ( $\operatorname{simp}$ add: suspends ${ }_{j}$ )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) suspends ${ }_{j}$.
from a-unowned-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathrm{sb}}$-j
obtain a-notin-owns-j: a $\notin$ acquired True ?take-sb $_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ and a-unshared: a $\notin$ all-shared ?take-sb ${ }_{j}$ by auto
from a-not-acquired-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$
have a-not-acquired-j: a $\notin$ all-acquired sb $_{\mathrm{j}}$
by auto
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}$ ]
have nvo-j: non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
have a-no-non-vol-read: a $\notin$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ ? drop-sb ${ }_{j}$ proof
assume a-in-nvr:a $\in$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ ? drop-sb $_{j}$
from reads-consistent-drop [OF reads-consis]
have rc: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) (flush ?take-sb $\mathrm{m}_{\mathrm{j}}$ ) ?drop-sb ${ }^{\text {. }}$
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-j-drop: non-volatile-owned-or-read-only True (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )
(acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
?drop-sb ${ }_{j}$
by $\operatorname{simp}$
from outstanding-refs-non-volatile-Read sb -all-acquired [OF rc this a-in-nvr]
have a-owns-acq-ror:
$a \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}} \cup$ read-only-reads (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by (auto dest!: acquired-all-acquired-in all-acquired-takeWhile-dropWhile-in
simp add: acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ )
have a-unowned-j: a $\notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired sb $_{\mathrm{j}}$
proof (cases $a \in \mathcal{O}_{\mathrm{j}}$ )
case False with a-not-acquired-j show ?thesis by auto
next
case True
from all-shared-acquired-in [OF True a-unshared] a-notin-owns-j
have False by auto thus ?thesis ..
qed
with a-owns-acq-ror
have a-ror: $\mathrm{a} \in$ read-only-reads (acquired True ? take-sb $\mathrm{J}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by auto
with read-only-reads-unowned [OF j-bound ${ }^{\prime \prime}$ i-bound neq-i-j [symmetric] $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ts $\mathrm{s}_{\mathrm{sb}}-\mathrm{i}$ ]
have a-unowned-sb: a $\notin \mathcal{O}_{\text {sb }} \cup$ all-acquired sb
by auto
from sharing-consis [ OF j -bound ${ }^{\prime \prime}$ ts $_{\mathrm{sb}} \mathrm{j} \mathrm{j}$ ] sharing-consistent-append [of $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ? take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{j}$ ]
have consis-j-drop: sharing-consistent (share ?take-sb $\mathcal{S}_{\mathrm{j}}$ ) (acquired True ?take-sb ${ }_{\mathrm{j}}$ $\mathcal{O}_{\mathrm{j}}$ ) ?drop-sb $\mathrm{j}_{\mathrm{j}}$
by auto
from read-only-reads-read-only [OF nvo-j-drop consis-j-drop] a-ror a-unowned-j
all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ] acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ [of $\mathrm{sb}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ]
have a $\in$ read-only (share ?take-sb ${ }_{j} \mathcal{S}_{\text {sb }}$ )
by (auto simp add: )
from read-only-share-all-shared [OF this] a-unshared
have a $\in$ read-only $\mathcal{S}_{\text {sb }}$
by fastforce
from read-only-unacquired-share [OF read-only-unowned [OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - ] weak-sharing-consis [ OF i-bound $\mathrm{ts}_{\mathrm{sb}}$ - i ] this] a-unowned-sb
have a $\in$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
by auto
with a-not-ro show False
by simp
qed reads-consis-flush-m]
have reads-consistent True (acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$ ) ?m-A suspends $\mathrm{j}_{\mathrm{j}}$ by (auto simp add: suspends ${ }_{\mathrm{j}}$ )
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?m-A ys
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ reads-consistent-append)
from valid-history [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}-\mathrm{j}}$ ]
have h-consis:
history-consistent $\vartheta_{\text {sbj }}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{j} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends ${ }_{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends ${ }_{\mathrm{j}}$ )
proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{j}_{\mathrm{j}} @$ ?drop-sb $\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by simp
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb ${ }_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{j}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{sb}_{\mathrm{j}}=$
last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{j}_{\mathrm{j}}$ apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ ) by (rule last-prog-hd-prog-append)
ultimately show ?thesis

qed
from valid-write-sops [OF j-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb. ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound ${ }^{\prime \prime}$ ts $_{\text {sb }}$-j]
have distinct-read-tmps (?take-sb ${ }_{j} @$ suspends ${ }_{j}$ )
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb $\cap$ read-tmps suspends ${ }_{j}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends ${ }_{j}$ )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) suspends ${ }_{\mathrm{j}}$
by (simp add: split-suspends ${ }_{j}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}} \mathrm{ys}=\{ \}$
by (auto simp add: outstanding-refs-append suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric]
split-suspends $\mathrm{j}_{\mathrm{j}}$ )

## from flush-store-buffer-append [

OF j-bound ${ }^{\prime \prime \prime \prime} \mathrm{is}_{\mathrm{j}}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts-A-j [simplified split-suspends ${ }_{j}$ ] refl lp [simplified split-suspends ${ }_{\mathrm{j}}$ ] reads-consis-m-A-ys
hist-consis ${ }^{\prime}$ [simplified split-suspendsj] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends ${ }_{\mathrm{j}}$ ]
no-volatile-Read sb -volatile-reads-consistent [OF no-vol-read], where $\mathcal{S}=$ ?share-A]
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}^{\prime}$ : instrs (Write ${ }_{\text {sb }}$ True $\left.\mathrm{a}^{\prime} \operatorname{sop} \mathrm{v} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right) @ \mathrm{is}_{\mathrm{sbj}}=$
is $_{j}^{\prime} @$ prog-instrs (Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$ ) and steps-ys: (?ts-A, ?m-A, ?share-A) $\Rightarrow_{d}{ }^{*}$
$\left(? \mathrm{ts}-\mathrm{A}\left[\mathrm{j}:=\left(\operatorname{last-prog}\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\right.\right.$ Write $_{\mathrm{sb}}$ True $\left.\left.\mathrm{a}^{\prime} \operatorname{sop} v \mathrm{~A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)$ ys, $\mathrm{is}_{\mathrm{j}}{ }^{\prime}$, $\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.$ read-tmps $\left(\right.$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime}$ sop v $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$
$\mathrm{zs})),()$,
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write $\mathrm{s}_{\mathrm{sb}}$ ys $\neq\{ \}$, acquired True ys
(acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ), $\mathcal{R}_{\mathrm{j}}{ }^{\prime}$ )],
flush ys ?m-A,
share ys ?share-A)
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?$ shared-ys $\left.)\right)$
by (auto)
note conflict-computation $=$ rtranclp-trans [OF rtranclp-r-rtranclp [OF steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj $\left(\left(y s @\left[W_{r i t e}^{s b}\right.\right.\right.$ True $\mathrm{a}^{\prime}$ sop v $\left.\left.\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \boldsymbol{\prime}\right]\right) @ \mathrm{zs}\right)$
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is sbj $^{\text {zs. }}$
interpret causal ${ }_{j}$ : causal-program-history is sbj zs by (rule cph ${ }^{\text {' }}$ )
from causal ${ }_{j}$.causal-program-history [of [], simplified, OF refl] $\mathrm{is}_{\mathrm{j}}{ }^{\prime}$
obtain is ${ }_{j}{ }^{\prime \prime}$
where $\mathrm{is}_{\mathrm{j}}^{\prime}: \mathrm{is}_{\mathrm{j}}^{\prime}=$ Write True $\mathrm{a}^{\prime}$ sop $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ and
$\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime} @$ prog-instrs zs
by clarsimp
from j-bound ${ }^{\prime \prime \prime}$

```
    have j-bound-ys: j < length ?ts-ys
        by auto
    from j-bound-ys neq-i-j
    have ts-ys-j: ?ts-ys!j=(last-prog (hd-prog p
ys, isj ',
```



```
            \mathcal{D}}\mp@subsup{\textrm{j}}{}{\vee
            acquired True ys (acquired True ?take-sb⿱丶万⿱⿰㇒一乂⿴⿱冂一⿰丨丨丁心
        by auto
    from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
    have safe-delayed (?ts-ys,?m-ys,?shared-ys).
    from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is j}\mp@subsup{}{j}{}
    have a-unowned:
    \foralli< length ?ts-ys. j }\not=\textrm{i}\longrightarrow(let (\mathcal{O
        apply cases
        apply (auto simp add: Let-def is isb
        done
    from a'-in a-unowned [rule-format, of i] neq-i-j i-bound' A-R
    show False
        by (auto simp add: Let-def)
    qed
}
thus ?thesis
    by (auto simp add: Let-def)
        qed
        have A-unaquired-by-others:
    |j<length (map \mathcal{O-sb ts sb}). i 
        (let (\mathcal{O}
            in A \cap all-acquired sb
        proof -
{
    fix j ( )
    assume j-bound: j < length (map owned tssb)
    assume neq-i-j: i\not=j
    assume ts sb-j:(map \mathcal{O-sb ts }
    assume conflict: A \cap all-acquired sb
    have False
    proof -
        from j-bound leq
        have j-bound': j < length (map owned ts)
        by auto
    from j-bound have j-bound '/: j < length tssb
        by auto
    from j-bound' have j-bound '/': j < length ts
        by simp
```


## from conflict obtain $a^{\prime}$ where

$a^{\prime}-\mathrm{in}: \mathrm{a}^{\prime} \in \mathrm{A}$ and
$a^{\prime}$-in-j: $a^{\prime} \in$ all-acquired $\mathrm{sb}_{j}$
by auto
let ? take-sb $_{\mathrm{j}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ? drop-sb ${ }_{j}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)$
from ts-sim [rule-format, OF j-bound ${ }^{\prime \prime}$ ] ts $\mathrm{sb}_{\mathrm{sb}}-\mathrm{j}$ j-bound ${ }^{\prime \prime}$
obtain $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{ji}_{\mathrm{sbj}} \mathcal{D}_{\text {sbj }} \mathcal{D}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \vartheta_{\mathrm{sbj}}$ is $\mathrm{is}_{\mathrm{j}}$ where
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$ and
suspends $_{\mathrm{j}}$ : suspends ${ }_{\mathrm{j}}=$ ? drop-sb $\mathrm{b}_{\mathrm{j}}$ and
$\mathrm{is}_{\mathrm{j}}$ : instrs suspends $\mathrm{j}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{is}_{\mathrm{j}} @$ prog-instrs suspends $\mathrm{j}_{\mathrm{j}}$ and
$\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.$ outstanding-refs is-volatile-Write $\left.\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)$ and
$\mathrm{ts}_{\mathrm{j}}:$ ts $!\mathrm{j}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.$ suspends $\mathrm{j}, \mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps suspends $\left.{ }_{j}\right),()$,
$\mathcal{D}_{\mathrm{j}}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$, release ? take-sb $\left._{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
apply ( (ases $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}$ )
apply (force simp add: Let-def)
done
from a'-in-j all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have $\mathrm{a}^{\prime} \in$ all-acquired ?take-sb $\mathrm{sb}_{\mathrm{j}} \vee \mathrm{a}^{\prime} \in$ all-acquired suspends ${ }_{\mathrm{j}}$
by (auto simp add: suspends ${ }_{\mathrm{j}}$ )
thus False
proof
assume $\mathrm{a}^{\prime} \in$ all-acquired ?take-sb ${ }_{j}$
with A-unowned-by-others [rule-format, OF - neq-i-j] ts sbb-j j-bound $\mathrm{a}^{\prime}$-in
show False
by (auto dest: all-acquired-unshared-acquired)
next
assume conflict-drop: $\mathrm{a}^{\prime} \in$ all-acquired suspends ${ }_{j}$
from split-all-acquired-in [OF conflict-drop]
show False
proof
assume $\exists$ sop a" v ys zs A L R W.
suspends $_{j}=$ ys @ Writesb True $a^{\prime \prime}$ sop v A L R W\# zs $\wedge a^{\prime} \in A$

## then

obtain $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime}$ ys zs $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
split-suspendsj: suspends $s_{j}=y s$ @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime \prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$
(is suspends ${ }_{j}=$ ?suspends) and
$a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}$
by auto
from direct-memop-step.WriteVolatile [where $\vartheta=\vartheta_{\text {sb }}$ and $\mathrm{m}=$ flush ?drop-sb m]
have (Write True a (D, f) A L R W \# is ${ }_{\text {sb }}{ }^{\prime}$,
$\vartheta_{\mathrm{sb}},()$, flush ?drop-sb m, $\mathcal{D}_{\mathrm{sb}}$, acquired True sb $\mathcal{O}_{\mathrm{sb}}$,
release sb ( $\left.\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$,
share ?drop-sb $\mathcal{S}) \rightarrow$
$\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(),(\right.$ flush $?$ drop-sb m$)\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right)$, True, acquired True sb $\mathcal{O}_{\mathrm{sb}} \cup$ A - R,Map.empty,

$$
\text { share ?drop-sb } \left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
$$

from direct-computation.concurrent-step.Memop [OF
i-bound-ts ${ }^{\prime}\left[\right.$ simplified is $s_{s b}$ ] ts ${ }^{\prime}$-i [simplified is stb ] this [simplified is stb ]]
have store-step: $\left(? \mathrm{ts}^{\prime}\right.$, flush ?drop-sb m , share ?drop-sb $\left.\mathcal{S}\right) \Rightarrow_{\mathrm{d}}$ $\left(? \mathrm{ts}{ }^{\prime}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.$,
$\vartheta_{\mathrm{sb}},()$,True, acquired True sb $\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}$, Map.empty $\left.)\right]$,
(flush ?drop-sb m) $\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right)$,share ? drop-sb $\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
(is $-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}, ?$ share-A) $)$
by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )
from i-bound' have i-bound ${ }^{\prime \prime}$ : i < length ?ts-A
by $\operatorname{simp}$
from valid-program-history [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have causal-program-history is $_{\text {sbj }} \mathrm{sb}_{j}$.
then have cph: causal-program-history is ${ }_{\text {sbj }}$ ?suspends
apply -
apply (rule causal-program-history-suffix [where $\mathrm{sb}=$ ? take-sb $_{\mathrm{j}}$ ])
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp add: split-suspendsj)
done
from $t_{s j}$ neq-i-j j-bound
have ts-A-j: ?ts-A!j = (hd-prog $p_{j}\left(y s @ W r i t e ~ s b r u e ~ a^{\prime \prime} \operatorname{sop}^{\prime} v^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)$, $\mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\right.$ dom $\vartheta_{\text {sbj }}$ - read-tmps (ys @ Write sb True a ${ }^{\prime \prime}$ sop $\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)$ ), (), $\mathcal{D}_{\mathrm{j}}$, acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$, release ? take-sb $\mathrm{f}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}$ )
by ( $\operatorname{simp}$ add: split-suspends ${ }_{\mathrm{j}}$ )
from j-bound ${ }^{\prime \prime \prime}$ i-bound ${ }^{\prime}$ neq-i-j have j-bound ${ }^{\prime \prime \prime \prime}:$ j $<$ length ?ts-A
by $\operatorname{simp}$
from valid-last-prog [ OF j -bound $\left.{ }^{\prime \prime} \operatorname{ts}_{\mathrm{sb}} \mathrm{j}\right]$ have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$.
then
have lp: last-prog $p_{j}$ ?suspends $=p_{j}$
apply -
apply (rule last-prog-same-append [where $\mathrm{sb}=$ ? take-sb ${ }_{\mathrm{j}}$ ])
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
from valid-reads [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have reads-consis: reads-consistent False $\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}$ >
j-bound ${ }^{\prime \prime}$
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) m suspends ${ }_{\mathrm{j}}$ by ( $\operatorname{simp}$ add: m suspends $\mathrm{j}_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound ${ }^{\prime \prime}$ neq-i-j $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have outstanding-refs is-Write sb ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ suspends $_{\mathrm{j}}=\{ \}$
by ( $\operatorname{simp}$ add: suspends ${ }_{j}$ )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) suspends ${ }^{\mathrm{j}}$.
from a-unowned-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$
obtain a-notin-owns-j: a $\notin$ acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ and a-unshared: a $\notin$ all-shared ?take-sb ${ }_{j}$
by auto
from a-not-acquired-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}}-\mathrm{j}}$
have a-not-acquired-j: a $\notin$ all-acquired sb $_{j}$
by auto
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}$ ]
have nvo-j: non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
have a-no-non-vol-read: a $\notin$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ ? ${ }^{\text {drop-sb }}{ }_{\mathrm{j}}$ proof
assume a-in-nvr:a $\in$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ ? drop-sb $_{\mathrm{j}}$
from reads-consistent-drop [OF reads-consis]
have rc: reads-consistent True (acquired True ?take-sb $\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) (flush ?take-sb $\mathrm{m}_{\mathrm{sb}}$ ) ?drop-sb ${ }_{\mathrm{j}}$.

```
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-j-drop: non-volatile-owned-or-read-only True (share ?take-sb \(\mathcal{S}_{\mathrm{jb}}\) )
            (acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) )
    ?drop-sb \({ }_{j}\)
    by \(\operatorname{simp}\)
```

from outstanding-refs-non-volatile-Read sb -all-acquired [OF rc this a-in-nvr]
have a-owns-acq-ror:
$\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}} \cup$ read-only-reads (acquired True ?take-sb $\mathrm{B}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by (auto dest!: acquired-all-acquired-in all-acquired-takeWhile-dropWhile-in simp add: acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ )
have a-unowned-j: a $\notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof (cases $a \in \mathcal{O}_{j}$ )
case False with a-not-acquired-j show ?thesis by auto

```
next
    case True
    from all-shared-acquired-in [OF True a-unshared] a-notin-owns-j
    have False by auto thus ?thesis ..
qed
```

with a-owns-acq-ror
have a-ror: $a \in$ read-only-reads (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$ by auto
with read-only-reads-unowned [OF j-bound ${ }^{\prime \prime}$ i-bound neq-i-j [symmetric] $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ] have a-unowned-sb: $\mathrm{a} \notin \mathcal{O}_{\text {sb }} \cup$ all-acquired sb by auto
from sharing-consis $\left[\mathrm{OF} \mathrm{j}\right.$-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}$ - j ] sharing-consistent-append [of $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ? take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have consis-j-drop: sharing-consistent (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ ) (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{j}$ by auto
from read-only-reads-read-only [OF nvo-j-drop consis-j-drop] a-ror a-unowned-j
all-acquired-append [of ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ] acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ [of $\operatorname{sb}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ]
have a $\in$ read-only (share ?take-sb ${ }_{\mathrm{j}} \mathcal{S}_{\mathrm{sb}}$ )
by (auto)
from read-only-share-all-shared [OF this] a-unshared
have a $\in$ read-only $\mathcal{S}_{\text {sb }}$
by fastforce
from read-only-unacquired-share [OF read-only-unowned [OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ] weak-sharing-consis [OF i-bound $\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}} \mathrm{i}$ ] this] a-unowned-sb
have a $\in$ read-only (share sb $\mathcal{S}_{\text {sb }}$ )
by auto
with a-not-ro show False
by simp
qed
with reads-consistent-mem-eq-on-non-volatile-reads [OF - subset-refl reads-consis-flush-m]
have reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) ?m-A suspends ${ }_{\mathrm{j}}$
by (auto simp add: suspends $\mathrm{j}_{\mathrm{j}}$ )
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?m-A ys
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ reads-consistent-append)
from valid-history [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}$-j ]
have $h$-consis:

apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ?take-sb ${ }_{\mathrm{j}}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.$ suspends $\left._{\mathrm{j}}\right)$ proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ? take-sb $\mathrm{j}_{\mathrm{j}}$ @?drop-sb $\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$ ) ?take-sb $\mathrm{j}_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $\mathrm{j}_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{j}_{\mathrm{j}}=$ last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$ apply (simp only: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ ) by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
qed
from valid-write-sops [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}$-j]
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ? suspends). valid-sop sop
by (simp add: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have distinct-read-tmps (?take-sb @suspends $_{\mathrm{j}}$ )
by (simp add: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb ${ }_{j} \cap$ read-tmps suspends ${ }_{j}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspends ${ }_{j}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis]
last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$ ) suspends $\mathrm{s}_{\mathrm{j}}$
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- $\operatorname{Read}_{\mathrm{sb}}$ ys $=\{ \}$
by (auto simp add: outstanding-refs-append suspends ${ }_{\mathrm{j}}$ [symmetric]
split-suspends ${ }_{j}$ )
from flush-store-buffer-append [
OF j-bound ${ }^{\prime \prime \prime \prime}$ is $\mathrm{s}_{\mathrm{j}}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts-A-j [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ] refl lp [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ] reads-consis-m-A-ys
hist-consis' ${ }^{\prime}$ [simplified split-suspends ${ }_{\mathrm{j}}$ ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends ${ }_{\mathrm{j}}$ ]
no-volatile-Read ${ }_{\text {sb }}$-volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ ?share-A]
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
$\mathrm{is}_{\mathrm{j}}^{\prime}$ : instrs (Write $\mathrm{sb}_{\text {b }}$ True $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$ ) @ is $\mathrm{sbj}_{\text {sb }}=$
$\mathrm{is}_{\mathrm{j}}{ }^{\prime}$ @ prog-instrs (Write ${ }_{\mathrm{sb}}$ True $\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$ ) and
steps-ys: (?ts-A, ?m-A, ?share-A) $\Rightarrow{ }_{\mathrm{d}}{ }^{*}$
(?ts-A $\left[\mathrm{j}:=\left(\operatorname{last}-\operatorname{prog}\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\right.$ Write $_{\mathrm{sb}}$ True $\left.\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)$ ys, is $_{j}{ }^{\prime}$, $\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.$ read-tmps $\left(\right.$ Write $_{\text {sb }}$ True $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ zs) ),(),
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ ys $\neq\{ \}$, acquired True ys (acquired True ?take-sb $\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) $\mathcal{R}_{\mathrm{j}}{ }^{\prime}$ ) ],
flush ys ?m-A, share ys ?share-A)
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?$ shared-ys))
by (auto)
note conflict-computation $=$ rtranclp-trans $[\mathrm{OF}$ rtranclp-r-rtranclp $[\mathrm{OF}$ steps-flush-sb, OF store-step] steps-ys]
from cph
 zs)
by simp
from causal-program-history-suffix [OF this]
have $\mathrm{cph}^{\prime}$ : causal-program-history is $\mathrm{s}_{\text {sbj }} \mathrm{zs}$.
interpret causal ${ }_{j}$ : causal-program-history $\mathrm{is}_{\text {sbj }} \mathrm{zs}$ by (rule cph')
from causal ${ }_{j}$.causal-program-history [of [], simplified, OF refl] $\mathrm{is}_{\mathrm{j}}{ }^{\prime}$
obtain is ${ }^{\prime \prime}{ }^{\prime \prime}$
where is ${ }_{j}^{\prime}$ : is $_{j}{ }^{\prime}=$ Write True $\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ and
$\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ @ prog-instrs zs
by clarsimp
from j-bound ${ }^{\prime \prime \prime}$
have j-bound-ys: j < length ?ts-ys
by auto
from j -bound-ys neq-i-j
have ts-ys-j: ?ts-ys! $j=\left(\operatorname{last-prog}\left(h d-p r o g p_{j}\left(W_{r i t e}^{s b}\right.\right.\right.$ True $a^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ zs)) ys, isj ${ }^{\prime}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.$ read-tmps $\left(\right.$ Write $_{\text {sb }}$ True $\left.\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),(), \mathcal{D}_{\mathrm{j}}$ $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ ys $\neq\{ \}$, acquired True ys (acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$ ), $\mathcal{R}_{\mathrm{j}}{ }^{\text {}}$ )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is ${ }_{j}$ ]
have $\mathrm{A}^{\prime}$-unowned:
$\forall \mathrm{i}<$ length ?ts-ys. $\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.$ let $\left(\mathcal{O}_{\mathrm{i}}\right)=$ map owned ?ts-ys!i in $\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)$
apply cases
apply (fastforce simp add: Let-def is stb ) +
done
from $\mathrm{a}^{\prime}$-in $\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}$-unowned [rule-format, of i$]$ neq-i-j i -bound ${ }^{\prime} \mathrm{A}-\mathrm{R}$
show False
by (auto simp add: Let-def)
next
assume $\exists \mathrm{A}$ L R W ys zs.
suspends $_{\mathrm{j}}=$ ys @ Ghost $_{\text {sb }}$ A L R W \# zs $\wedge \mathrm{a}^{\prime} \in \mathrm{A}$
then
obtain ys zs A ${ }^{\prime} L^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
split-suspends $\mathrm{j}_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=\mathrm{ys}$ @ Ghost $_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$ (is suspends $\mathrm{s}_{\mathrm{j}}=$ ?suspends)
and
$a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}$
by auto
from direct-memop-step.WriteVolatile [where $\vartheta=\vartheta_{\mathrm{sb}}$ and $\mathrm{m}=$ flush ?drop-sb m] have (Write True a (D, f) A L R W\# is ${ }_{\text {sb }}{ }^{\prime}$, $\vartheta_{\mathrm{sb}},()$, flush ?drop-sb $\mathrm{m}, \mathcal{D}_{\mathrm{sb}}$,acquired True $\mathrm{sb} \mathcal{O}_{\mathrm{sb}}$, release sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$, share ?drop-sb $\mathcal{S}) \rightarrow$
$\left(\mathrm{is}_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(),\left(\right.\right.$ flush $?$ drop-sb m) $\left(\mathrm{a}:=\mathrm{f} \vartheta_{\mathrm{sb}}\right)$, True, acquired True sb $\mathcal{O}_{\mathrm{sb}} \cup$
A - R, Map.empty,

$$
\text { share ?drop-sb } \left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right) \text {. }
$$

from direct-computation.concurrent-step.Memop [OF i-bound-ts ${ }^{\prime}$ [simplified is sb ] ts'-i $\left[\right.$ simplified is $_{\text {sb }}$ ] this [simplified is $\mathrm{s}_{\mathrm{sb}}$ ]]
have store-step: (?ts', flush ?drop-sb m, share ?drop-sb $\mathcal{S}) \Rightarrow_{\mathrm{d}}$
(?ts' $1 \mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}\right.$,
$\vartheta_{\text {sb }},()$, True, acquired True sb $\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}$, Map.empty $\left.)\right]$,
(flush ?drop-sb m) (a $\left.:=\mathrm{f} \vartheta_{\mathrm{sb}}\right)$,share ? drop-sb $\left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
(is $-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}$, ?share- A$)$ )
by (simp add: is $\mathrm{s}_{\mathrm{sb}}$ )
from i-bound' have i-bound": i < length ?ts-A by simp
from valid-program-history [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have causal-program-history $\mathrm{is}_{\text {sbj }} \mathrm{sb}_{j}$.
then have cph: causal-program-history is sbj ?suspends apply -
apply (rule causal-program-history-suffix [where sb=?take-sbj] )
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric] suspends ${ }_{j}$ )
apply (simp add: split-suspends ${ }_{\mathrm{j}}$ )
done
from $t s_{j}$ neq-i-j $j$-bound

 acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$, release ? take-sb $\mathrm{b}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}$ ) by (simp add: split-suspends ${ }_{\mathrm{j}}$ )
from j-bound ${ }^{\prime \prime \prime}$ i-bound' neq-i-j have j-bound ${ }^{\prime \prime \prime \prime}:$ j < length ?ts-A by simp
from valid-last-prog [ OF j -bound ${ }^{\prime \prime}$ ts $\left.\mathrm{s}_{\mathrm{sb}}-\mathrm{j}\right]$ have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$. then
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ ?suspends $=\mathrm{p}_{\mathrm{j}}$
apply -
apply (rule last-prog-same-append [where $\mathrm{sb}=$ ? $\mathrm{take}^{2}-\mathrm{sb}_{\mathrm{j}}$ ])
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}\left[\right.$ symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
from valid-reads [ OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\text {sb }}$-j]
have reads-consis: reads-consistent False $\mathcal{O}_{j} m_{s b} s b b_{j}$.
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing $\mathcal{S}_{\text {sb }} \mathrm{ts}_{\mathrm{sb}}$ )
j-bound ${ }^{\prime \prime}$
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathbf{j}}$ ) m suspends $\mathrm{s}_{\mathrm{j}}$ by (simp add: $m$ suspends $_{\mathrm{j}}$ )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound" neq-i-j $\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}-\mathrm{j}} \mathrm{j}\right]$
have outstanding-refs is- Write $_{\text {sb }}$ ?drop-sb $\cap$ outstanding-refs is-non-volatile-Read ${ }_{\mathbf{s b}}$ suspends $_{\mathrm{j}}=\{ \}$
by (simp add: suspends $\mathrm{j}_{\mathrm{j}}$ )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ )
(flush ?drop-sb m) suspends .
from a-unowned-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$
obtain a-notin-owns-j: a $\notin$ acquired True ${ }^{\text {ttake-sb }}{ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ and a-unshared: a $\notin$ all-shared ?take-sb ${ }_{j}$ by auto
from a-not-acquired-others [rule-format, OF - neq-i-j] j-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$
have a-not-acquired-j: a $\notin$ all-acquired sb $_{\mathrm{j}}$
by auto
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound ${ }^{\prime \prime}$ ts ssb-j ]
have nvo-j: non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
have a-no-non-vol-read: a $\notin$ outstanding-refs is-non-volatile-Read $\mathrm{Rb}_{\mathrm{sb}}$ ?drop-sb $\mathrm{sb}_{\mathrm{j}}$ proof
assume a-in-nvr:a $\in$ outstanding-refs is-non-volatile-Read ${ }_{\text {sb }}$ ?drop-sb ${ }_{j}$
from reads-consistent-drop [OF reads-consis]
have rc: reads-consistent True (acquired True ?take-sb $\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) (flush ?take-sb $\mathrm{m}_{\mathrm{sb}}$ ) ?drop-sbj .
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-j-drop: non-volatile-owned-or-read-only True (share ?take-sb ${ }_{\mathrm{j}} \mathcal{S}_{\mathrm{sb}}$ )
(acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
?drop-sb ${ }_{j}$
by simp
from outstanding-refs-non-volatile-Read ${ }_{\text {sb }}$-all-acquired [OF rc this a-in-nvr]
have a-owns-acq-ror:
$\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}} \cup$ read-only-reads (acquired True ?take-sb $\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb $\mathrm{b}_{\mathrm{j}}$ by (auto dest!: acquired-all-acquired-in all-acquired-takeWhile-dropWhile-in simp add: acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ )
have a-unowned-j: $\mathrm{a} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof (cases a $\in \mathcal{O}_{\mathrm{j}}$ )
case False with a-not-acquired-j show ?thesis by auto
next
case True
from all-shared-acquired-in [OF True a-unshared] a-notin-owns-j
have False by auto thus ?thesis ..
qed
with a-owns-acq-ror
have a-ror: $\mathrm{a} \in$ read-only-reads (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by auto
with read-only-reads-unowned [OF j-bound ${ }^{\prime \prime}$ i-bound neq-i-j [symmetric] ts $\mathrm{tsb}_{\mathrm{sb}}-\mathrm{j} \mathrm{ts}_{\mathrm{sb}} \mathrm{i}$ ]
have a-unowned-sb: $\mathrm{a} \notin \mathcal{O}_{\mathrm{sb}} \cup$ all-acquired sb
by auto
from sharing-consis $\left[\mathrm{OF} \mathrm{j}\right.$-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$ sharing-consistent-append [of $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ? take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{j}$ ]
have consis-j-drop: sharing-consistent (share ?take-sb ${ }_{j} \mathcal{S}_{\mathrm{sb}}$ ) (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{j}$
by auto
from read-only-reads-read-only [OF nvo-j-drop consis-j-drop] a-ror a-unowned-j
all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ] acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}$ [of $\mathrm{sb}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ]
have a $\in$ read-only (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )

```
    by (auto)
    from read-only-share-all-shared [OF this] a-unshared
    have a }\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{
        by fastforce
    from read-only-unacquired-share [OF read-only-unowned [OF i-bound ts tsb-i]
    weak-sharing-consis [OF i-bound ts ssb-i] this] a-unowned-sb
    have a }\in\mathrm{ read-only (share sb }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mathrm{ )
    by auto
    with a-not-ro show False
    by simp
        qed
```

with reads-consistent-mem-eq-on-non-volatile-reads [OF - subset-refl reads-consis-flush-m]
have reads-consistent True (acquired True ?take-sb $\mathrm{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?m-A suspends ${ }_{\mathrm{j}}$
by (auto simp add: suspends $\mathrm{s}_{\mathrm{j}}$ )
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?m-A ys
by (simp add: split-suspends ${ }_{\mathrm{j}}$ reads-consistent-append)
from valid-history [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have h -consis:
history-consistent $\vartheta_{\mathrm{sbj}}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb ${ }_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends $_{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog $\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)$ ?take-sb ${ }_{\mathrm{j}}=\left(\right.$ hd- $\operatorname{prog} \mathrm{p}_{\mathrm{j}}$ suspends ${ }_{\mathrm{j}}$ ) proof -
from last-prog have last-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb $_{\mathrm{j}} @$ ? drop-sb $\left.\mathrm{b}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}$
by simp
from last-prog-hd-prog-append ${ }^{\prime}$ [OF h-consis] this
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$
by (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
moreover
have last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(?\right.$ take-sb $_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}=$ last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{j}_{\mathrm{j}}$ ) ?take-sb $\mathrm{b}_{\mathrm{j}}$
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
qed
from valid-write-sops [OF j-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have $\forall$ sop $\in$ write-sops (?take-sb ${ }_{j} @$ ?suspends). valid-sop sop
by ( $\operatorname{simp}$ add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain valid-sops-take: $\forall$ sop $\in$ write-sops ?take-sb ${ }_{j}$. valid-sop sop and
valid-sops-drop: $\forall$ sop $\in$ write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}$ ]
have distinct-read-tmps (?take-sb ${ }_{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}$ )
by (simp add: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ? take-sb ${ }_{\mathrm{j}} \cap$ read-tmps suspends ${ }_{\mathrm{j}}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis]
last-prog-hd-prog
have hist-consis': history-consistent $\vartheta_{\text {sbj }}$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $_{\mathrm{j}}$ ) suspends $\mathrm{j}_{\mathrm{j}}$
by ( $\operatorname{simp}$ add: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- $\operatorname{Read}_{\text {sb }}$ ys $=\{ \}$
by (auto simp add: outstanding-refs-append suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric]
split-suspends ${ }_{j}$ )
from flush-store-buffer-append [
OF j-bound ${ }^{\prime \prime \prime \prime} \mathrm{is}_{\mathrm{j}}$ [simplified split-suspends $\mathrm{s}_{\mathrm{j}}$ ] cph [simplified suspends $\mathrm{s}_{\mathrm{j}}$ ]
ts-A-j [simplified split-suspends ${ }_{j}$ ] refl lp [simplified split-suspends ${ }_{j}$ ] reads-consis-m-A-ys
hist-consis' [simplified split-suspends ${ }^{\text {}}$ ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends ${ }_{j}$ ]
no-volatile-Read sb -volatile-reads-consistent [OF no-vol-read], where
$\mathcal{S}=$ ? share- A ]
obtain is ${ }_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ where
is $_{j}^{\prime}$ : instrs (Ghost $\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right) @$ is $_{\text {sbj }}=$
$\mathrm{is}_{\mathrm{j}}^{\prime}$ @ prog-instrs (Ghost $\mathrm{sb}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}$ ) and
steps-ys: (?ts-A, ?m-A, ?share-A) $\Rightarrow_{d}{ }^{*}$
$\left(? \mathrm{ts}-\mathrm{A}\left[\mathrm{j}:=\left(\operatorname{last}-\mathrm{prog}\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\right.\right.$ Ghost $\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)$ ys, $\mathrm{is}_{\mathrm{j}}{ }^{\prime}$,
$\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.$ read-tmps $\left(\right.$ Ghost $\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),()$,
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ ys $\neq\{ \}$, acquired True ys (acquired True ?take-sb $\left.\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}$ ) ],
flush ys ?m-A,
share ys ?share-A)
(is $(-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?$ shared-ys $\left.)\right)$
by (auto)
note conflict-computation $=$ rtranclp-trans $[\mathrm{OF}$ rtranclp-r-rtranclp $[\mathrm{OF}$ steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is $\mathrm{sbj}_{\text {sj }}\left(\left(\mathrm{ys} @\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right) @ \mathrm{zs}\right)$
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is sbj zs .
interpret causal ${ }_{j}$ : causal-program-history $\mathrm{is}_{\mathrm{sbj}} \mathrm{zs}$ by (rule cph ${ }^{\prime}$ )
from causal ${ }_{j}$.causal-program-history [of [], simplified, OF refl] is $_{\mathrm{j}}{ }^{\prime}$
obtain $\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$
where is $_{\mathrm{j}}{ }^{\prime}: \mathrm{is}_{\mathrm{j}}{ }^{\prime}=$ Ghost $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ and
$\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}$ : instrs zs @ $\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime} @$ prog-instrs zs
by clarsimp
from j-bound ${ }^{\prime \prime \prime}$
have j-bound-ys: j < length ?ts-ys
by auto
from $j$-bound-ys neq-i-j
have ts-ys-j: ?ts-ys! $j=\left(\operatorname{last}-\mathrm{prog}\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.$ Ghost $\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)$ ys, is $_{\mathrm{j}}{ }^{\prime}$, $\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps $\left(W_{r i t e}\right.$ sb True $\left.\left.\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),(), \mathcal{D}_{\mathrm{j}}$ $\vee$ outstanding-refs is-volatile-Write ${ }_{\text {sb }}$ ys $\neq\{ \}$, acquired True ys (acquired True ? $\operatorname{take}^{-s b}{ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ), $\mathcal{R}_{\mathrm{j}}{ }^{\prime}$ )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is ${ }_{j}$ ]
have A'-unowned:
$\forall \mathrm{i}<$ length ?ts-ys. $\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.$ let $\left(\mathcal{O}_{\mathrm{i}}\right)=$ map owned ?ts-ys!i in $\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)$
apply cases
apply (fastforce simp add: Let-def is stb ) +
done
from $\mathrm{a}^{\prime}$-in $\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}$-unowned [rule-format, of i] neq-i-j i-bound ${ }^{\prime} \mathrm{A}-\mathrm{R}$
show False
by (auto simp add: Let-def)
qed
qed
qed
\}
thus ?thesis
by (auto simp add: Let-def)
qed
have A-no-read-only-reads-by-others:
$\forall \mathrm{j}<$ length $\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right) . \mathrm{i} \neq \mathrm{j} \longrightarrow$ $\left(\operatorname{let}\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)=\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right.$
in $\mathrm{A} \cap$ read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write $\mathrm{sb}_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\left.\mathcal{O}_{\mathrm{j}}\right)$
$\left(\right.$ dropWhile $^{(N o t} \circ$ is-volatile-Write $\left.\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)=\{ \}\right)$
proof -
\{
fix $\mathrm{j} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j-bound: $\mathrm{j}<$ length (map $\mathcal{O}$-sb ts $\mathrm{st}_{\mathrm{sb}}$ )
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}:\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right)!j=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)$
let ? take-sb ${ }_{\mathrm{j}}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
let ? drop- $^{-\mathrm{sb}_{\mathrm{j}}}=\left(\operatorname{dropWhile}\left(\right.\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)$
assume conflict: $\mathrm{A} \cap$ read-only-reads (acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ ) ? drop-sb $\mathrm{T}_{\mathrm{j}} \neq\{ \}$
have False
proof -
from j-bound leq
have j -bound': $\mathrm{j}<$ length (map owned ts)
by auto
from j-bound have $j$-bound ${ }^{\prime \prime}$ : $\mathrm{j}<$ length ts $_{\text {sb }}$
by auto
from j-bound ${ }^{\prime}$ have $j$-bound ${ }^{\prime \prime \prime}$ : $\mathrm{j}<$ length ts
by $\operatorname{simp}$

## from conflict obtain $\mathrm{a}^{\prime}$ where

 $a^{\prime}$-in: $a^{\prime} \in A$ and$\mathrm{a}^{\prime}$-in-j: $\mathrm{a}^{\prime} \in$ read-only-reads (acquired True ?take-sb $\mathrm{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by auto
from ts-sim [rule-format, OF j-bound ${ }^{\prime \prime}$ ts $\mathrm{ts}_{\mathrm{sb}}$-j j-bound ${ }^{\prime \prime}$
obtain $p_{j}$ suspends is $_{\text {sbj }} \mathcal{D}_{\text {sbj }} \mathcal{D}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \vartheta_{\mathrm{sbj}}$ is $\mathrm{s}_{\mathrm{j}}$ where
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$ and
suspends $_{\mathrm{j}}$ : suspends ${ }_{\mathrm{j}}=$ ? drop-sb $\mathrm{sb}_{\mathrm{j}}$ and
$i_{j}$ : instrs suspends $\mathrm{s}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{s}_{\mathrm{j}} @$ prog-instrs suspends $\mathrm{j}_{\mathrm{j}}$ and
$\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)$ and
$\mathrm{ts}_{\mathrm{j}}:$ ts! $\mathrm{j}=$ (hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}$, $\mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}\right.$ - read-tmps suspends $\left.{ }_{j}\right),(), \mathcal{D}_{\mathrm{j}}$, acquired True ? take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$, release
?take-sb $\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
apply (cases $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}$ )
apply (force simp add: Let-def)
done
from split-in-read-only-reads [OF a'-in-j [simplified suspends ${ }_{j}$ [symmetric]]]
obtain t v ys zs where
split-suspends $_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Read $_{\mathrm{sb}}$ False a't v\# zs (is suspends ${ }_{\mathrm{j}}=$ ?suspends)
and
$\mathrm{a}^{\prime}$-unacq: $\mathrm{a}^{\prime} \notin$ acquired True ys (acquired True ? take-sb $\mathrm{O}_{\mathrm{j}}$ )
by blast

```
    from direct-memop-step.WriteVolatile [where }\vartheta=\mp@subsup{\vartheta}{\textrm{sb}}{}\mathrm{ and m=flush ?drop-sb m]
    have (Write True a (D, f) A L R W# is sb
        \varthetasb
        release sb (dom }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{})\mp@subsup{\mathcal{R}}{\mathrm{ sb }}{}\mathrm{ , share ?drop-sb }\mathcal{S})
            (is sb}\mp@subsup{}{}{\prime},\mp@subsup{\vartheta}{\textrm{sb}}{},(),(flush ?drop-sb m)(a := f \mp@subsup{\vartheta}{\textrm{sb}}{}), True, acquired True sb \mathcal{O
A - R,Map.empty,
                    share ?drop-sb }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
from direct-computation.concurrent-step.Memop [OF
    i-bound-ts' [simplified is sb] ts'-1 [simplified is isb
have store-step: (?ts', flush ?drop-sb m, share ?drop-sb S) }\mp@subsup{=>}{\textrm{d}}{
    (?ts'{i := ( }\mp@subsup{\textrm{p}}{\mathbf{sb}}{},\mp@subsup{\textrm{is}}{\mathbf{sb}}{\prime},\mp@subsup{\vartheta}{\mathbf{sb}}{},()
                            True, acquired True sb }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\textrm{A}-\textrm{R},Map.empty)]
                            (flush ?drop-sb m)(a := f \vartheta \varthetasb),share ?drop-sb S }\mp@subsup{\mathcal{S}}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
(is - = }\mp@subsup{\textrm{d}}{\textrm{d}}{(?ts-A, ?m-A, ?share-A))
    by (simp add: is sb
from i-bound' have i-bound": i < length ?ts-A
    by simp
from valid-program-history [OF j-bound " ts tsb-j]
have causal-program-history is isbj sbj
then have cph: causal-program-history is isbj ?suspends
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb;}]\mathrm{ ])
apply (simp only: split-suspendsj [symmetric] suspendsj)
apply (simp add: split-suspendsj)
done
from tsj neq-i-j j-bound
have ts-A-j: ?ts-A!j = (hd-prog p (ys @ Read cb False a't v v zs), isj,
    \vartheta vbj |
    acquired True ?take-sb }\mp@subsup{\boldsymbol{\mathcal{O}}}{\textrm{j}}{\mathrm{ ,release ?take-sb}
    by (simp add: split-suspendsj
from j-bound'/\prime i-bound' neq-i-j have j-bound '/\prime\prime: j < length ?ts-A
    by simp
```

```
from valid-last-prog [OF j-bound \({ }^{\prime \prime}\) ts s \(\left._{\mathbf{s b}}-\mathrm{j}\right]\) have last-prog: \(\operatorname{last-prog~} \mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
```

from valid-last-prog [OF j-bound ${ }^{\prime \prime}$ ts s $\left._{\mathbf{s b}}-\mathrm{j}\right]$ have last-prog: $\operatorname{last-prog~} \mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$.
then
then
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ ?suspends $=\mathrm{p}_{\mathrm{j}}$
have lp: last-prog $\mathrm{p}_{\mathrm{j}}$ ?suspends $=\mathrm{p}_{\mathrm{j}}$
apply -
apply -
apply (rule last-prog-same-append [where sb=?take-sb ${ }_{\mathrm{j}}$ ])
apply (rule last-prog-same-append [where sb=?take-sb ${ }_{\mathrm{j}}$ ])
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric $^{2}$ suspends ${ }_{j}$ )
apply (simp only: split-suspends ${ }_{j}\left[\right.$ symmetric $^{2}$ suspends ${ }_{j}$ )
apply simp
apply simp
done
done
from valid-reads [OF j-bound" ts sb-j]

```
have reads-consis: reads-consistent False \(\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ) j -bound \({ }^{\prime \prime}\) \(\mathrm{ts}_{\mathrm{sb}}\)-j reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{j}\) ) m suspends \({ }_{j}\) by (simp add: m suspends \(\mathrm{j}_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound \({ }^{\prime \prime}\) neq-i-j \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by (simp add: suspends \({ }_{\mathrm{j}}\) )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb \(\mathcal{O}_{j}\) ) (flush ?drop-sb m) suspendsj.
from a-unowned-others [rule-format, OF j-bound" neq-i-j ] j-bound \(\mathrm{ts}_{\text {sb- }} \mathrm{j}\)
obtain a-notin-owns-j: a \(\notin\) acquired True ?take-sb \({ }_{j} \mathcal{O}_{j}\) and a-unshared: a \(\notin\) all-shared ?take-sb \({ }_{j}\) by auto
from a-not-acquired-others [rule-format, OF j-bound neq-i-j] j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\)
have a-not-acquired-j: a \(\notin\) all-acquired sb \(_{j}\)
by auto
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \({ }^{\prime \prime}\) ts \(\mathrm{s}_{\mathrm{sb}}\)-j] have nvo-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
have a-no-non-vol-read: a \(\notin\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) ?drop-sb \(_{\mathrm{j}}\) proof
assume a-in-nvr:a \(\in\) outstanding-refs is-non-volatile-Read \({ }_{\mathbf{s b}}\) ?drop-sb \({ }_{j}\)
from reads-consistent-drop [OF reads-consis]
have rc: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{j}\) ) (flush ?take-sb \(\mathrm{m}_{\mathrm{sb}}\) ) ?drop-sb \({ }_{j}\).
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-j-drop: non-volatile-owned-or-read-only True (share ?take-sb \({ }_{j} \mathcal{S}_{\text {sb }}\) )
(acquired True ? take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) )
?drop-sb \({ }_{j}\)
by simp
from outstanding-refs-non-volatile-Read \({ }_{\text {sb }}\)-all-acquired \([\) OF rc this a-in-nvr]
have a-owns-acq-ror:
\(\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup\) read-only-reads (acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}}\) ) ?drop-sb \({ }_{\mathrm{j}}\) by (auto dest!: acquired-all-acquired-in all-acquired-takeWhile-dropWhile-in simp add: acquired-takeWhile-non-volatile-Write \({ }_{\mathbf{s b}}\) )
have a-unowned-j: a \(\notin \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
```

proof (cases a }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{\mathbf{j}
case False with a-not-acquired-j show ?thesis by auto
next
case True
from all-shared-acquired-in [OF True a-unshared] a-notin-owns-j
have False by auto thus ?thesis ..
qed

```
with a-owns-acq-ror
have a-ror: \(\mathrm{a} \in\) read-only-reads (acquired True \({ }^{\text {?take-sb }} \mathrm{J}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \(\mathrm{b}_{\mathrm{j}}\) by auto
with read-only-reads-unowned [OF j-bound \({ }^{\prime \prime}\) i-bound neq-i-j [symmetric] \(\left.\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]\) have a-unowned-sb: a \(\notin \mathcal{O}_{\text {sb }} \cup\) all-acquired sb by auto
from sharing-consis \(\left[\mathrm{OF} \mathrm{j}\right.\)-bound \(\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\) sharing-consistent-append \(\left[\mathrm{of} \mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\right.\) ?take-sb \({ }_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
have consis-j-drop: sharing-consistent (share ?take-sb \({ }_{j} \mathcal{S}_{\mathrm{sb}}\) ) (acquired True ?take-sb \({ }_{\mathrm{j}}\) \(\mathcal{O}_{\mathrm{j}}\) ) ? drop-sb \({ }_{\mathrm{j}}\)
by auto
from read-only-reads-read-only [OF nvo-j-drop consis-j-drop] a-ror a-unowned-j
all-acquired-append \(\left[\right.\) of ? take-sb \({ }_{\mathrm{j}}\) ? drop-sb \(\mathrm{b}_{\mathrm{j}}\) ] acquired-takeWhile-non-volatile-Write \({ }_{\mathrm{sb}}\) [of \(\operatorname{sb}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ]
have \(\mathrm{a} \in\) read-only (share ? take-sb \({ }_{\mathrm{j}} \mathcal{S}_{\mathrm{sb}}\) )
by (auto)
from read-only-share-all-shared [OF this] a-unshared
have a \(\in\) read-only \(\mathcal{S}_{\mathrm{sb}}\)
by fastforce
from read-only-unacquired-share [OF read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) - ]
weak-sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) - \(]\) this] a-unowned-sb
have a \(\in\) read-only (share sb \(\mathcal{S}_{\text {sb }}\) )
by auto
with a-not-ro show False by simp
qed
with reads-consistent-mem-eq-on-non-volatile-reads [OF - subset-refl reads-consis-flush-m]
have reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?m-A suspends \(\mathrm{m}_{\mathrm{j}}\)
by (auto simp add: suspends \(\mathrm{j}_{\mathrm{j}}\) )
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?m-A ys
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) reads-consistent-append)
from valid-history [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have h -consis:
history-consistent \(\vartheta_{\mathrm{sbj}}\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\left(?\right.\right.\) take-sb \(\left.\left._{\mathrm{j}} @_{\mathrm{suspen}} \mathrm{s}_{\mathrm{j}}\right)\right)\) (?take-sb \(\mathrm{a}_{\mathrm{j}} @_{\mathrm{suspends}}^{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left.\mathrm{j}_{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) @?drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\) by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}[\) symmetric \(] ~ s u s p e n d s ~_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{b}_{\mathrm{j}}\) @ suspends j\()\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\) apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) ) by (rule last-prog-hd-prog-append)
ultimately show?thesis by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
qed
from valid-write-sops [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have distinct-read-tmps (?take-sb \(@_{j}\) asuspends \({ }_{j}\) )
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{s}_{\mathrm{j}}\)
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile-Read \({ }_{\text {sb }}\) ys \(=\{ \}\)
by (auto simp add: outstanding-refs-append suspendsj [symmetric]
split-suspends \({ }_{j}\) )
from flush-store-buffer-append [
OF j-bound \({ }^{\prime \prime \prime \prime} \mathrm{is}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ]
ts-A-j [simplified split-suspends \({ }_{j}\) ] refl lp [simplified split-suspends \({ }_{j}\) ] reads-consis-m-A-ys
hist-consis \({ }^{\prime}\) [simplified split-suspends \({ }_{j}\) ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends \({ }_{j}\) ]
no-volatile-Read sb-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?share-A]
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : instrs \(\left(\operatorname{Read}_{\mathrm{sb}}\right.\) False \(\mathrm{a}^{\prime} \mathrm{t}\) v\# zs) @ \(\mathrm{is}_{\mathrm{sbj}}=\)
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime} @ \operatorname{prog}-\mathrm{instrs}\left(\operatorname{Read}_{\mathrm{sb}}\right.\) False \(\left.\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\right)\) and
steps-ys: (?ts-A, ?m-A, ?share-A) \(\Rightarrow_{d}{ }^{*}\)
\(\left(? \mathrm{ts}-\mathrm{A}\left[\mathrm{j}:=\left(\operatorname{last}-\mathrm{prog}\left(h d-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\right.\right.\) Read \(_{\mathrm{sb}}\) False \(\left.\left.\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\right)\right)\) ys, \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\), \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\operatorname{Read}_{\text {sb }}\right.\) False a't v\# zs \(\left.)\right),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys
(acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )],
flush ys ?m-A,
share ys ?share-A)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(?\) ts-ys,?m-ys,?shared-ys) \()\)
by (auto)
note conflict-computation \(=\) rtranclp-trans [OF rtranclp-r-rtranclp [OF steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj \(\left(\left(y s @\left[\operatorname{Read}_{s b}\right.\right.\right.\) False a't v\(\left.\left.]\right) @ z s\right)\)
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is sbj \(^{\text {zs }}\).
interpret causal \({ }_{\mathrm{j}}\) : causal-program-history is \(_{\text {sbj }}\) zs by (rule cph \({ }^{\prime}\) )
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }_{j}{ }^{\prime}\)
obtain is \({ }_{j}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}: \mathrm{is}_{\mathrm{j}}{ }^{\prime}=\operatorname{Read}\) False \(\mathrm{a}^{\prime} \mathrm{t} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ \(\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime} @\) prog-instrs zs
by clarsimp
from j-bound \({ }^{\prime \prime \prime}\)
have j-bound-ys: \(\mathrm{j}<\) length ?ts-ys
by auto
from j-bound-ys neq-i-j
have ts-ys-j: ?ts-ys! \(\mathrm{j}=\left(\operatorname{last-prog}\left(\right.\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\operatorname{Read}_{\mathrm{sb}}\right.\) False \(\left.\left.\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\right)\right)\) ys, \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\), \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\operatorname{read}-\mathrm{tmps}\left(\operatorname{Read}_{\mathrm{sb}}\right.\right.\) False a't v\# zs \(\left.)\right),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\),
acquired True ys (acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
```

have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is ${ }_{j}$ ']
have $\mathrm{a}^{\prime} \in$ acquired True ys (acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$ ) $\vee$
$\mathrm{a}^{\prime} \in$ read-only (share ys (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ))
apply cases
apply (auto simp add: Let-def is s $_{\text {sb }}$ )
done
with a'-unacq
have $\mathrm{a}^{\prime}$-ro: $\mathrm{a}^{\prime} \in$ read-only (share ys (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ))
by auto
from $a^{\prime}$-in
have a'-not-ro: $\mathrm{a}^{\prime} \notin$ read-only (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
by (auto simp add: in-read-only-convs)
have $\mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof -
\{
assume a-notin: $\mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
from weak-sharing-consis [ OF j -bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have weak-sharing-consistent $\mathcal{O}_{j} \mathrm{sb}_{\mathrm{j}}$.
with weak-sharing-consistent-append [of $\mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have weak-sharing-consistent (acquired True ? take-sb $_{j} \mathcal{O}_{j}$ ) suspends ${ }_{j}$
by (auto simp add: suspends $\mathrm{j}_{\mathrm{j}}$ )
with split-suspends ${ }_{\mathrm{j}}$
have weak-consis: weak-sharing-consistent (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) ys
by (simp add: weak-sharing-consistent-append)
from all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have all-acquired ys $\subseteq$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
apply (clarsimp)
apply (clarsimp simp add: suspends ${ }_{j}\left[\right.$ symmetric] $^{\text {split-suspends }}{ }_{j}$ all-acquired-append)
done
with a-notin acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}\left[\right.$ of $\left.^{\operatorname{sb}}{ }_{j} \mathcal{O}_{j}\right]$
all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have $\mathrm{a}^{\prime} \notin$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}} \cup$ all-acquired ys
by auto
from read-only-share-unowned [OF weak-consis this a'-ro]
have a' $\in$ read-only (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ).
with $a^{\prime}$-not-ro have False
by auto
\}
thus ?thesis by blast
qed
moreover
from A-unaquired-by-others [rule-format, OF j-bound neq-i-j] $\mathrm{ts}_{\mathrm{sb}}$-j j-bound

```
```

    have A \cap all-acquired sb
        by (auto simp add: Let-def)
    moreover
    from A-unowned-by-others [rule-format, OF j-bound " neq-i-j] ts tsb-j j-bound
    have }\textrm{A}\cap\mp@subsup{\mathcal{O}}{\textrm{j}}{}={
        by (auto simp add: Let-def dest: all-shared-acquired-in)
    moreover note a'-in
    ultimately
    show False
        by auto
    qed
    }
thus ?thesis
by (auto simp add: Let-def)
qed
have valid-own': valid-ownership }\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ ]
have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mathrm{ (sb @ [Writesb True a (D,f) (f 诸b
A L R W])
by (auto simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by (simp add: ts scb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}
qed
next
show outstanding-volatile-writes-unowned-by-others tssab
proof (unfold-locales)
fix i
assume i }\mp@subsup{i}{1}{}\mathrm{ -bound: }\mp@subsup{\textrm{i}}{1}{}<\mathrm{ length tssmb
assume j-bound: j < length ts sbb
assume i }\mp@subsup{\textrm{i}}{1}{}-\textrm{j}:\mp@subsup{\textrm{i}}{1}{}\not=\textrm{j
assume ts-\mp@subsup{\textrm{i}}{1}{}:\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{!}{1}{}
assume ts-j: ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
show (}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{\textrm{j}}{})\cap\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mp@subsup{\textrm{sb}}{1}{}={
proof (cases i i =i)
case True
with \mp@subsup{i}{1}{}-\textrm{j}}\mathrm{ have i-j: i}=\textrm{j
by simp
from j-bound have j-bound': j < length ts sb
by (simp add: ts sb
hence j-bound '/: j < length (map owned ts sb
by simp
from ts-j i-j have ts-j}\mp@subsup{}{}{\prime}:\mp@subsup{\textrm{ts}}{\textrm{sb}}{}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
by (simp add: ts scb
from a-unowned-others [rule-format, OF - i-j] i-j ts-j j-bound
obtain a-notin-j: a }\not=\mathrm{ acquired True (takeWhile (Not o is-volatile-Write }\mp@subsup{\textrm{sb}}{\textrm{sb}}{}\mathrm{ ) sb }\mp@subsup{\textrm{b}}{\textrm{j}}{}\mathrm{ ) }\mp@subsup{\mathcal{O}}{\textrm{j}}{

```
```

                a-unshared: a }\not\in\mathrm{ all-shared (takeWhile (Not ○ is-volatile-Write sb
    by (auto simp add: Let-def ts sb
    from a-not-acquired-others [rule-format, OF - i-j] i-j ts-j j-bound
    have a-notin-acq: a }\not=\mathrm{ all-acquired sbj
        by (auto simp add: Let-def ts }\mp@subsup{\textrm{ts}}{\textrm{s}}{}\mathrm{ ')
    from outstanding-volatile-writes-unowned-by-others
    [OF i-bound j-bound' i-j ts scb-i ts-j}\mp@subsup{}{}{\prime}
    have}(\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
    with ts-i}\mp@subsup{i}{1}{}\mathrm{ a-notin-j a-unshared a-notin-acq True i-bound show ?thesis
        by (auto simp add: ts ssb
    acquired-takeWhile-non-volatile-Write sb dest: all-shared-acquired-in)
next
case False
note }\mp@subsup{\textrm{i}}{1}{}-\textrm{i}=\mathrm{ this
from }\mp@subsup{\textrm{i}}{1}{}\mathrm{ -bound have }\mp@subsup{\textrm{i}}{1}{}\mathrm{ -bound': }\mp@subsup{\textrm{i}}{1}{}<\mathrm{ length ts }\mp@subsup{\textrm{S}}{\mathrm{ sb}}{
by (simp add: ts scb
from ts-i}\mp@subsup{i}{1}{}\mathrm{ False have ts-i}\mp@subsup{\textrm{i}}{1}{\prime}:\mp@subsup{\textrm{ts}}{\textrm{sb}}{}!\mp@subsup{\textrm{i}}{1}{}=(\mp@subsup{\textrm{p}}{1}{},\mp@subsup{\textrm{is}}{1}{},\mp@subsup{\textrm{xs}}{1}{},\mp@subsup{\textrm{sb}}{1}{},\mp@subsup{\mathcal{D}}{1}{},\mp@subsup{\mathcal{O}}{1}{},\mp@subsup{\mathcal{R}}{1}{}
by (simp add: ts ssb
show ?thesis
proof (cases j=i)
case True
from i i-bound '
have i}\mp@subsup{i}{1}{}\mathrm{ -bound '': }\mp@subsup{i}{1}{}<\mathrm{ length (map owned ts sb )
by simp
from outstanding-volatile-writes-unowned-by-others
[OF i}\mp@subsup{1}{1}{}\mathrm{ -bound' i-bound i i
have ( }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb) }\cap\mathrm{ outstanding-refs is-volatile-Write sb }\mp@subsup{\textrm{sb}}{1}{}={}
moreover
from A-unused-by-others [rule-format, OF - False [symmetric]] False ts-i}\mp@subsup{i}{1}{}\mp@subsup{\textrm{i}}{1}{}\mathrm{ -bound
have A \cap outstanding-refs is-volatile-Write
by (auto simp add: Let-def ts }\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mathrm{ ')
ultimately
show ?thesis
using ts-j True ts sb}\mp@subsup{}{}{\prime
by (auto simp add: i-bound ts }\mp@subsup{\textrm{tsb}}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mathrm{ all-acquired-append)
next
case False
from j-bound have j-bound': j < length ts scb
by (simp add: ts sb
from ts-j False have ts-j': ts scb}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
by (simp add: ts sb
from outstanding-volatile-writes-unowned-by-others
[OF i i-bound' j-bound' }\mp@subsup{i}{1}{}-j ts-\mp@subsup{i}{1}{\prime}\mp@subsup{}{}{\prime}\mathrm{ ts-j']
show ( (\mathcal{O}}\cup\cup\mathrm{ all-acquired sb }\mp@subsup{\textrm{s}}{\textrm{j}}{})\cap\mathrm{ outstanding-refs is-volatile-Write
qed
qed
qed

```
next
show ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have \(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
\(\left(\right.\) let \(\left(p_{j}\right.\), is \(\left._{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\)
in \(\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb\() \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\right)\)
proof -
\{
fix j \(p_{j}\) is \(\mathcal{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume j-bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume \(\operatorname{ts}_{\text {sb }}-\mathrm{j}\) : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is} \mathrm{s}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
have \(\left(\mathcal{O}_{\text {sb }} \cup\right.\) all-acquired sb\() \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\)
proof -
\{
fix \(a^{\prime}\)
assume \(a^{\prime}-i n-i: a^{\prime} \in\left(\mathcal{O}_{s b} \cup\right.\) all-acquired \(\left.s b\right)\)
assume \(\mathrm{a}^{\prime}\)-in-j: \(\mathrm{a}^{\prime} \in\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)\)
have False
proof -
from \(a^{\prime}\)-in-i have \(a^{\prime} \in\left(\mathcal{O}_{s b} \cup\right.\) all-acquired \(\left.s b\right) \vee a^{\prime} \in A\)
by (simp add: \(\mathrm{sb}^{\prime}\) all-acquired-append)
then show False
proof
assume \(\mathrm{a}^{\prime} \in\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb\()\)
with ownership-distinct [OF i-bound j-bound neq-i-j \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ts \(\mathrm{s}_{\mathrm{sb}}-\mathrm{j}\) ] \({ }^{\prime}{ }^{\prime}-\mathrm{in}-\mathrm{j}\)
show ?thesis
by auto
next
assume \(a^{\prime} \in A\)
moreover
have j-bound': \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
using j -bound by auto
from A-unowned-by-others [rule-format, OF - neq-i-j] ts sbb \(^{\text {-j }} \mathrm{j}\)-bound
obtain \(\mathrm{A} \cap\) acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}}=\{ \}\) and \(\mathrm{A} \cap\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(=\{ \}\)
by (auto simp add: Let-def)
moreover
from A-unaquired-by-others [rule-format, OF - neq-i-j] ts sbb -j j-bound
have \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}}=\{ \}\)
by auto
ultimately
show ?thesis
using \(a^{\prime}-\mathrm{in}-\mathrm{j}\)
by (auto dest: all-shared-acquired-in)
qed
qed
\}
then show ?thesis by auto qed
```

    }
    then show ?thesis by (fastforce simp add: Let-def)
    qed
    from ownership-distinct-nth-update [OF i-bound ts tsb-i this]
    show ?thesis by (simp add: ts }\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}
    qed
next
show read-only-reads-unowned ts sb
proof
fix n m

```

```

    assume n-bound: n < length ts sb}\mp@subsup{}{}{\prime
        and m-bound: m < length ts sb}\mp@subsup{}{}{\prime
        and neq-n-m: n\not=m
        and nth: ts sb !!n = (p
        and mth: ts sb !m = (pm, is m, 访, sb m
    from n-bound have n-bound': n < length ts tsb by (simp add: ts scb
    from m-bound have m-bound': m < length ts sb by (simp add: tssb
    show ( }\mp@subsup{\mathcal{O}}{\textrm{m}}{}\cup\mathrm{ all-acquired sb
                read-only-reads (acquired True (takeWhile (Not o is-volatile-Write sb) sb n) (\mathcal{On}
                    (dropWhile (Not & is-volatile-Write sb
            {}
    proof (cases m=i)
        case True
        with neq-n-m have neq-n-i: n}=\textrm{i
            by auto
        with n-bound nth i-bound have nth': ts sb
            by (auto simp add: ts sb ')
        note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth' ts sb-i]
        moreover
    from A-no-read-only-reads-by-others [rule-format, OF - neq-n-i [symmetric]] n-bound '
    nth'
have A \cap read-only-reads (acquired True (takeWhile (Not \circ is-volatile-Write sb) sb }\mp@subsup{\mp@code{n}}{\textrm{n}}{}\mathrm{ )
O
(dropWhile (Not o is-volatile-Writesb})\mp@subsup{\textrm{sb}}{\textrm{n}}{})
{}
by auto
ultimately
show ?thesis
using True ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i
by (auto simp add: ts ssb}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mathrm{ 'all-acquired-append)
next
case False
note neq-m-i = this

```

```

        by (auto simp add: ts sb ')
    show ?thesis
    proof (cases n=i)
    ```
case True
note read-only-reads-unowned [OF i-bound m-bound' neq-m-i [symmetric] ts \(\mathrm{s}_{\mathrm{sb}}\)-i
mth'] then show?thesis
using True neq-m-i \(\mathrm{ts}_{\mathrm{sb}}\)-i nth mth n -bound \({ }^{\prime} \mathrm{m}\)-bound \({ }^{\prime}\)
apply (case-tac outstanding-refs (is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}=\{ \}\) )
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends
acquired-append read-only-reads-append \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\right)+\)
done

\section*{next}
case False
with n -bound nth i-bound have \(\mathrm{nth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from read-only-reads-unowned [OF n-bound' m-bound' neq-n-m nth' mth] False neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
qed
have valid-hist': valid-history program-step ts \({ }_{\text {sb }}{ }^{\prime}\) proof -
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) ]
have history-consistent \(\vartheta_{\text {sb }}\) (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\) ) sb.
with valid-write-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i] D-subset
valid-implies-valid-prog-hd [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}-\mathrm{i}\) valid]
have history-consistent \(\vartheta_{\text {sb }}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\mathrm{Write}_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right)\right.\right.\) A L R W]) ) (sb@ \(\left[\right.\) Write \(_{\text {sb }}\) True a (D,f) (f \(\vartheta_{\text {sb }}\) ) A L R W])
apply -
apply (rule history-consistent-appendI)
apply (auto simp add: hd-prog-append-Write \({ }_{\text {sb }}\) )
done
from valid-history-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-reads': valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-reads [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have reads-consistent False \(\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}\).
from reads-consistent-snoc-Write \({ }_{\text {sb }}\) [OF this]
have reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\mathrm{sb}}\) ( sb @ \(\left[\mathrm{Write}_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right)\right.\) A L R W]).
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-sharing': valid-sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\) proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}\) ]
have non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}\) (sb @ [ Write \(_{\text {sb }} \operatorname{True}\) a (D,f) (f \(\vartheta_{\text {sb }}\) ) A L R W])
by (auto simp add: non-volatile-writes-unshared-append)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have consis': sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}\).
from A-shared-owned
have \(\mathrm{A} \subseteq\) dom (share ? drop-sb \(\mathcal{S}\) ) \(\cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\)
by (simp add: sharing-consistent-append acquired-takeWhile-non-volatile-Write \({ }_{\mathbf{s b}}\) )
moreover have \(\operatorname{dom}(\) share \(? \operatorname{drop-sb} \mathcal{S}) \subseteq \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
proof
fix a'
assume \(\mathrm{a}^{\prime}\)-in: \(\mathrm{a}^{\prime} \in \operatorname{dom}\) (share ?drop-sb \(\mathcal{S}\) )
from share-unshared-in [OF a'-in]
show a' \({ }^{\prime} \in \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
proof
assume \(\mathrm{a}^{\prime} \in\) dom (share ?drop-sb Map.empty)
from share-mono-in [OF this] share-append [of ?take-sb ?drop-sb]
have \(\mathrm{a}^{\prime} \in \operatorname{dom}\) (share sb \(\mathcal{S}_{\mathrm{sb}}\) )
by auto
thus ?thesis
by simp
next
assume \(\mathrm{a}^{\prime} \in \operatorname{dom} \mathcal{S} \wedge \mathrm{a}^{\prime} \notin\) all-unshared ?drop-sb
thus ?thesis by auto
qed
qed
ultimately
have A-subset: A \(\subseteq \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) ) \(\cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\) by auto
with A-unowned-by-others
have \(\mathrm{A} \subseteq \operatorname{dom}\left(\right.\) share sb \(\left.\mathcal{S}_{\text {sb }}\right) \cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\)
proof \{
fix \(x\)
assume \(\mathrm{x}-\mathrm{A}: \mathrm{x} \in \mathrm{A}\)
have \(\mathrm{x} \in \operatorname{dom}\) (share sb \(\mathcal{S}_{\mathrm{sb}}\) ) \(\cup\) acquired True sb \(\mathcal{O}_{\mathrm{sb}}\)
proof -
\{
assume \(\mathrm{x} \in \operatorname{dom} \mathcal{S}\)
from share-all-until-volatile-write-share-acquired [OF «sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉
i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i this [simplified \(\mathcal{S}\) ]]
A-unowned-by-others \(x\)-A
```

            have ?thesis
            by (fastforce simp add: Let-def)
    }
    with A-subset show ?thesis using x-A by auto
    qed
    }
thus ?thesis by blast
qed

```
with consis' L-subset A-R R-acq
have sharing-consistent \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\left(\mathrm{sb} @\left[\mathrm{Write}_{\text {sb }} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right)\right.\right.\) A L R W])
by (simp add: sharing-consistent-append acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) )
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
]
show read-only-unowned \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\text {sb }}{ }^{\prime} \operatorname{ts}_{\text {sb }}{ }^{\prime} \mathcal{O}_{\text {sb }}{ }^{\prime}\) )
        next
from unowned-shared-nth-update [OF i-bound ts \(_{\text {sb }}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    by ( \(\operatorname{simp}\) add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
        next

have no-write-to-read-only-memory \(\mathcal{S}_{\text {sb }}\left(\right.\) sb @ \(\left[W_{r i t e}^{s b}\right.\) True a (D,f) (f \(\left.\vartheta_{\text {sb }}\right)\) A L R W])
    by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
        qed

        proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have distinct-load-tmps is s \(_{\text {sb }}{ }^{\prime}\) by (simp add: is \(\mathrm{s}_{\mathrm{sb}}\) )
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
        next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have distinct-read-tmps (sb @ [Write sb \(\left.^{\text {True }} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{ALR} \mathrm{W}\right]\) )
    by (auto simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
        next
    from load-tmps-read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have load-tmps is sb \(^{\prime} \cap\) read-tmps (sb @ [Write \({ }_{\text {sb }}\) True a (D, f) (f \(\vartheta_{\text {sb }}\) ) A L R W]) \(=\{ \}\)
    by (auto simp add: read-tmps-append is sb )
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
 qed
have valid-sops': valid-sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
obtain valid-Df: valid-sop ( \(\mathrm{D}, \mathrm{f}\) ) and
valid-store-sops \({ }^{\prime}: \forall\) sop \(\in\) store-sops is sb \({ }^{\prime}\). valid-sop sop
by (auto simp add: is sb \(_{\text {s }}\) )
from valid-Df valid-write-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have valid-write-sops': \(\forall\) sop \(\in\) write-sops ( \(\mathrm{sb} @\left[\right.\) Write \(_{\text {sb }}\) True a (D, f) (f \(\boldsymbol{\vartheta}_{\mathrm{sb}}\) ) A L R W]).
valid-sop sop
by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound valid-write-sops' valid-store-sops ']
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
obtain D-indep: \(\mathrm{D} \cap\) load-tmps is stb \(^{\prime}{ }^{\prime}=\{ \}\) and
dd-is: data-dependency-consistent-instrs ( \(\operatorname{dom} \vartheta_{\text {sb }}{ }^{\prime}\) ) is sbb \(^{\prime}\)
by (auto simp add: is sb \(\vartheta_{\text {sb }}\) )
from load-tmps-write-tmps-distinct [OF i-bound ts \(_{\text {sb }}\)-i] D-indep
have load-tmps is sb \(^{\prime} \cap \bigcup\left(\right.\) fst \(^{\prime}\) write-sops (sb@ \(\left[\right.\) Write \(_{\text {sb }} \operatorname{True}\) a (D, f) (f \(\left.\vartheta_{s b}\right)\) A L R W]) \()\)
\(=\{ \}\)
by (auto simp add: write-sops-append is \(_{\mathrm{sb}}\) )
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by (simp add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed
have load-tmps-fresh': load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have load-tmps is sb \(^{\prime} \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\)
by (auto simp add: is s \(_{\text {sb }}\) )
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from clean-no-outstanding-volatile-Write \({ }_{s b}\) [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have \(\neg\) True \(\longrightarrow\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\left(\operatorname{sb} @\left[\right.\right.\) Write \(_{\text {sb }}\) True a \((\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\text {sb }}\right) \mathrm{A}\)
\(\mathrm{L} R \mathrm{~W}])=\{ \}\) by (auto simp add: outstanding-refs-append )
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by (simp add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\right)\)
qed
have valid-program-history \({ }^{\prime}\) : valid-program-history ts \(_{\text {sb }}{ }^{\prime}\)
proof -
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have causal-program-history is \(_{\mathrm{sb}} \mathrm{sb}\).
then have causal': causal-program-history is sb \(^{\prime}\) ( \(\mathrm{sb} @\left[\mathrm{Write}_{\mathrm{sb}}\right.\) True a \((\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right)\) A L R W])
by (auto simp: causal-program-history-Write is \(_{\text {sb }}\) )
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) - ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb}\right.\) @ \(\left.\left[\mathrm{Write}_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{A} L \mathrm{R} W\right]\right)=\mathrm{p}_{\mathrm{sb}}\) by (simp add: last-prog-append-Write \({ }_{\text {sb }}\) )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) sb )
qed
show ?thesis
proof (cases outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}=\{ \}\) )
case True
from True have flush-all: takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}=\mathrm{sb}\) by (auto simp add: outstanding-refs-conv)
from True have suspend-nothing: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}=[]\) by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends \(=[]\)
by (simp add: suspends)
from suspends-empty is-sim have is: is = Write True a (D,f) A L R W\# is stb \(^{\prime}\) by ( \(\operatorname{simp}\) add: is \(_{\text {sb }}\) )
with suspends-empty ts-i
have ts-i: ts! \(=\left(\mathrm{p}_{\mathrm{sb}}\right.\), Write True a \((\mathrm{D}, \mathrm{f})\) A L R W\# is \({ }_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) )
by \(\operatorname{simp}\)
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by auto

\section*{moreover}
note flush-commute \(=\)
flush-all-until-volatile-write-append-volatile-write-commute
[OF True i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
from True
have drop-app: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )
\(\left(\mathrm{sb} @\left[\mathrm{Write}_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})\left(\mathrm{f} \vartheta_{\mathrm{sb}}\right) \mathrm{A} L \mathrm{R} \mathrm{W}\right]\right)=\)
\(\left[W^{W i t e}\right.\) sb \(\operatorname{True}\) a (D,f) (f \(\left.\vartheta_{\text {sb }}\right)\) A L R W]
by (auto simp add: outstanding-refs-conv)
```

have ( }\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime},\mp@subsup{\textrm{m}}{\textrm{sb}}{},\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime})~(\textrm{ts},\textrm{m},\mathcal{S}
apply (rule sim-config.intros)
apply (simp add: m flush-commute ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}
using share-all-until-volatile-write-Write-commute
[OF i-bound ts }\mp@subsup{\textrm{ts}}{\mathbf{sb}}{}-\textrm{i}[\mathrm{ [simplified is
apply (clarsimp simp add: S S S Sbb
using leq
apply (simp add: ts sb
using i-bound i-bound' ts-sim ts-i
apply (clarsimp simp add: Let-def nth-list-update drop-app
ts
split: if-split-asm )
done
ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct'
valid-sops'
valid-dd' load-tmps-fresh' enough-flushs'
valid-program-history' valid' m}\mp@subsup{\textrm{m}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{\prime
by auto
next
case False

```
    then obtain \(r\) where \(r\)-in: \(r \in\) set sb and volatile- \(r\) : is-volatile-Write \({ }_{s b} r\)
    by (auto simp add: outstanding-refs-conv)
    from takeWhile-dropWhile-real-prefix
    [OF r-in, of (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ), simplified, OF volatile-r]
    obtain \(a^{\prime} v^{\prime} s b^{\prime \prime} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime}\) sop' where
    sb-split: \(\mathrm{sb}=\) takeWhile (Not o is-volatile-Write \(\mathrm{sb}_{\mathrm{sb}}\) ) sb @ Write \(\mathrm{sb}_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime}\)
\(\mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \# \mathrm{sb}^{\prime \prime}\)
    and
    drop: dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \(=\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \#\)
sb"
    apply (auto)
    subgoal for y ys
    apply (case-tac y)
    apply auto
    done
    done
    from drop suspends have suspends: suspends \(=\) Write \(_{\text {sb }}\) True \(a^{\prime} \operatorname{sop}^{\prime} v^{\prime} A^{\prime \prime} L^{\prime \prime} R^{\prime \prime} W^{\prime \prime} \#\)
\(\mathrm{sb}^{\prime \prime}\)
    by \(\operatorname{simp}\)
    have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by auto
moreover
note flush-commute \(=\)
flush-all-until-volatile-write-append-unflushed [OF False i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathbf{b}}}\) - ]
have Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \in\) set sb
by (subst sb-split) auto
note drop-app \(=\) dropWhile-append1
[OF this, of (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ), simplified]
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply ( \(\operatorname{simp}\) add: m flush-commute \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
using share-all-until-volatile-write-Write-commute
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) - [simplified \(\mathrm{is}_{\mathrm{sb}}\) ]]
apply (clarsimp \(\operatorname{simp}\) add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
using leq
apply (simp add: \(\mathrm{ts}_{\text {sb }}\) )
using i-bound i-bound \({ }^{\prime}\) ts-sim ts-i is-sim
apply (clarsimp simp add: Let-def nth-list-update is-sim drop-app
read-tmps-append suspends
prog-instrs-append-Write sb instrs-append-Write \(_{s b}\) hd-prog-append-Write \({ }_{s b}\)
drop is \({ }_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) outstanding-refs-append takeWhile-tail
release-append split: if-split-asm)

\section*{done}
ultimately show ?thesis
using valid-own' valid-hist \({ }^{\prime}\) valid-reads' valid-sharing' tmps-distinct' \({ }^{\prime}\) valid-dd \({ }^{\prime}\) valid-sops \({ }^{\prime}\) load-tmps-fresh \({ }^{\prime}\) enough-flushs \({ }^{\prime}\)
valid-program-history \({ }^{\prime}\) valid \({ }^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply )
qed
next
case SBHFence
then obtain
\(\mathrm{is}_{\mathrm{sb}}\) : \(\mathrm{is}_{\mathrm{sb}}=\) Fence \(\#\) is \(_{\text {sb }}{ }^{\prime}\) and
\(\mathrm{sb}: \mathrm{sb}=[]\) and
\(\mathcal{O}_{s b}{ }^{\prime}: \mathcal{O}_{s b}{ }^{\prime}=\mathcal{O}_{s b}\) and
\(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\) Map.empty and
\(\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}\) and
\(\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \neg \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\)
by auto
have valid-own': valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
proof -
have non-volatile-owned-or-read-only False \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) [] by simp
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
qed next
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i subset-refl]
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-reads-unowned-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i, of [] \(\mathcal{O}_{\mathrm{sb}}\) ]
show read-only-reads-unowned \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) ) next
from ownership-distinct-instructions-read-value-store-buffer-independent [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
show ownership-distinct \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-hist': valid-history program-step ts sb \(^{\prime}{ }^{\prime}\) proof -
from valid-history [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
have history-consistent \(\vartheta_{\mathrm{sb}}\) (hd-prog \(\mathrm{p}_{\mathrm{sb}}[]\) )[] by simp
from valid-history-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-reads \({ }^{\prime}\) : valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
have reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\text {sb }}\) [] by simp
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) ' \(\mathrm{sb} \mathcal{O}_{\mathrm{sb}}\) )
qed
have valid-sharing': valid-sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\) proof (intro-locales)
have non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}\) []
by (simp add: sb)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
have sharing-consistent \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) [] by simp
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) sb \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - i ]
]
show read-only-unowned \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by \(\left(\operatorname{simp}\right.\) add: \(\left.\mathcal{S}_{\text {sb }}{ }^{\prime} \operatorname{ts}_{\text {sb }}{ }^{\prime} \mathcal{O}_{\text {sb }}{ }^{\prime}\right)\) next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) ) next
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound, of []]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb}\) )
qed
have tmps-distinct': tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) ]
have distinct-load-tmps is sb \(^{\prime}{ }^{\prime}\)
by (auto simp add: is sb \(_{\text {s }}\) split: instr.splits)
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}}\) ) next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have distinct-read-tmps [] by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) ) next
from load-tmps-read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathbf{s b}}\)-i] load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have load-tmps is sb \(^{\prime} \cap\) read-tmps []\(=\{ \}\) by (clarsimp)
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) by ( \(\operatorname{simp} \mathrm{add}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) ) qed
have valid-sops': valid-sops \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) proof -
from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
obtain
valid-store-sops \({ }^{\prime}: \forall\) sop \(\in\) store-sops is sb \({ }^{\prime}\). valid-sop sop
by (auto simp add: \(\mathrm{is}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from valid-sops-nth-update [OF i-bound - valid-store-sops \({ }^{\prime}\), where \(\mathrm{sb}=[]\) ]
show ?thesis by (auto \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \operatorname{sb} \mathcal{O}_{\mathrm{sb}}\) )
qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}\) ]
obtain
dd-is: data-dependency-consistent-instrs \(\left(\operatorname{dom} \vartheta_{\mathrm{sb}}\right)\) is \(_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp add: is sb \(\vartheta_{\text {sb }}\) )
from load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have load-tmps is sb \(^{\prime} \cap \bigcup\left(\mathrm{fst}^{\prime}\right.\) write-sops []\()=\{ \}\)
by (auto simp add: write-sops-append)
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ? thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have load-tmps-fresh \({ }^{\prime}\) : load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathbf{s b}}\) - ]
have load-tmps is sb \(^{\prime} \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\)
by (auto simp add: is sb \(_{\text {sb }}\) )
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{is}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \vartheta_{\mathrm{sb}}\) )
qed
from enough-flushs-nth-update [OF i-bound, where \(\mathrm{sb}=[]\) ]
have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}\) )
have valid-program-history \({ }^{\prime}\) : valid-program-history ts sb \(^{\prime}{ }^{\prime}\) proof -
have causal': causal-program-history \(\mathrm{is}_{\mathrm{sb}^{\prime}}{ }^{\prime} \mathrm{sb}^{\prime}\)
by ( \(\operatorname{simp}\) add: is \(\mathrm{s}_{\mathrm{sb}} \mathrm{sb}\) sb )
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by ( simp add: \(\mathrm{sb}^{\prime}\) sb)
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed
from is-sim have is: is \(=\) Fence \(\#\) is \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: suspends sb is sb \(_{\text {sb }}\) )
with ts-i
have ts-i: \(\mathrm{ts}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}\right.\), Fence \(\# \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release
?take-sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
by (simp add: suspends sb)
from direct-memop-step.Fence
have (Fence \(\#\) is \(_{\text {sb }}{ }^{\prime}\),
\(\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\) \(\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}\right.\), False, acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), Map.empty, \(\mathcal{S}\) ).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\)
\(\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()\right.\right.\right.\), False, acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), Map.empty \(\left.\left.)\right], \mathrm{m}, \mathcal{S}\right)\).
moreover
have \(\left(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}, \mathrm{m}_{\mathbf{s b}}, \mathcal{S}_{\mathbf{s b}}{ }^{\prime}\right) \sim\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathbf{s b}},()\right.\right.\right.\), False, acquired True ?take-sb \(\mathcal{O}_{\text {sb }}\), Map.empty)],m, \(\left.\mathcal{S}\right)\)
apply (rule sim-config.intros)
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{m}\)
flush-all-until-volatile-nth-update-unused [OF i-bound \(\mathrm{ts}_{\left.\mathrm{sb}_{\mathrm{s}}-\mathrm{i}\right]}\) )
using share-all-until-volatile-write-Fence-commute
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) [simplified is \(\left._{\mathrm{sb}} \mathrm{sb}\right]\) ]
apply (clarsimp simp add: \(\mathcal{S} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb}\) )
using leq
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
using i-bound i-bound' ts-sim
apply (clarsimp simp add: Let-def nth-list-update
\(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) ex-not \(\vartheta_{\mathrm{sb}}{ }^{\prime}\)
split: if-split-asm )
done
ultimately
show ?thesis
using valid-own' valid-hist ' valid-reads' valid-sharing' tmps-distinct' valid-sops' valid-dd \({ }^{\prime}\) load-tmps-fresh \({ }^{\prime}\) enough-flushs \({ }^{\prime}\)
valid-program-history \({ }^{\prime}\) valid \(^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
next
case (SBHRMWReadOnly cond t a D f ret A L R W)
then obtain
\(\mathrm{is}_{\mathrm{sb}}\) : \(\mathrm{is}_{\mathrm{sb}}=\) RMW at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W \# is \(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}\) and
cond: \(\neg\left(\operatorname{cond}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}}\right.\right.\right.\) a) \(\left.)\right)\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and
\(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\) Map.empty and
\(\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}}\right.\) a) and
\(\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \neg \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) and
\(\mathrm{sb}: \mathrm{sb}=[]\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=[]\) and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\)
by auto
from safe-RMW-common [OF safe-memop-flush-sb [simplified is sb \(_{\text {b }}\) ]]
obtain access-cond: \(\mathrm{a} \in \mathcal{O}_{\text {sb }} \vee \mathrm{a} \in \operatorname{dom} \mathcal{S}\) and
rels-cond: \(\forall \mathrm{j}<\) length ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\) released (ts \(!\mathrm{j})\) a \(\neq\) Some False
by (auto \(\operatorname{simp}\) add: \(\mathcal{S} \mathrm{sb}\) )
have valid-own': valid-ownership \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have non-volatile-owned-or-read-only False \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) [] by simp
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
qed
next
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) subset-refl]
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-reads-unowned-nth-update [OF i-bound ts \(\mathrm{s}_{\mathrm{sb}}\)-i, of [] \(\mathcal{O}_{\mathrm{sb}}\) ]
show read-only-reads-unowned \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound \(\mathrm{ts}_{\text {sb }}-\mathrm{i}\) ]
show ownership-distinct ts \(\mathrm{sb}^{\prime}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) ' \(\operatorname{sb} \mathcal{O}_{\mathrm{sb}}\) )
qed
have valid-hist': valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) ]
have history-consistent \(\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right.\) ) (hd-prog \(\left.\mathrm{p}_{\mathrm{sb}}[]\right)\) [] by simp
from valid-history-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-reads': valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
have reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\text {sb }}[]\) by simp
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) sb \(\mathcal{O}_{\text {sb }}\) )
qed
have valid-sharing': valid-sharing \(\mathcal{S}_{\mathrm{sb}^{\prime}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof (intro-locales)
from outstanding-non-volatile-writes-unshared [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - i ]
have non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}\) []
by (simp add: sb)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
have sharing-consistent \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) [] by simp
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}{ }^{\prime}\) sb \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - i ]
]
show read-only-unowned \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by (simp add: \(\mathcal{S}_{\mathrm{sb}^{\prime}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) i subset-refl]
show unowned-shared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) ) next
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound, of []]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\left.\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb}\right)\)
qed
have tmps-distinct': tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}-\mathrm{i}\) ]
have distinct-load-tmps is sb \(^{\prime}{ }^{\prime}\) by (auto simp add: is sb \(_{\text {s }}\) split: instr.splits)
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}}\) ) next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have distinct-read-tmps [] by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) sb \(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}\) ) next
from load-tmps-read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ] load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have load-tmps is sb \(^{\prime} \cap\) read-tmps []\(=\{ \}\) by (clarsimp)
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) by ( \(\operatorname{simp} \mathrm{add}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) ) qed
have valid-sops': valid-sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -

obtain
valid-store-sops \({ }^{\prime}: \forall \mathrm{sop} \in\) store-sops is \({ }_{\text {sb }}{ }^{\prime}\). valid-sop sop by (auto \(\operatorname{simp}\) add: is \(\mathrm{sb}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}^{\prime}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from valid-sops-nth-update [OF i-bound - valid-store-sops \({ }^{\prime}\), where \(\mathrm{sb}=[]\) ]
show ?thesis by (auto \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \operatorname{sb} \mathcal{O}_{\mathrm{sb}}\) )
qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text { - }}\) ]
obtain
dd-is: data-dependency-consistent-instrs \(\left(\operatorname{dom} \vartheta_{\mathrm{sb}}\right)\) is sbb \(^{\prime}\)
by (auto simp add: is sb \(\vartheta_{\text {sb }}\) )
from load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have load-tmps is \({ }_{\mathbf{s b}}{ }^{\prime} \cap \bigcup(\) fst ' write-sops []) \(=\{ \}\) by (auto simp add: write-sops-append)
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have load-tmps-fresh \({ }^{\prime}\) : load-tmps-fresh ts \(_{\text {sb }}{ }^{\prime}\) proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have load-tmps (RMW at (D,f) cond ret A L R W\# is sb \(^{\prime}\) ) \(\cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\) by (simp add: is sb )

\section*{moreover}
from load-tmps-distinct [OF i-bound \(\left.\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}} \mathrm{i}}\right]\) have \(\mathrm{t} \notin\) load-tmps is s \(_{\mathbf{s b}^{\prime}}{ }^{\prime}\) by (auto simp add: is sb \(_{\text {b }}\) )
ultimately have load-tmps \(\mathrm{is}_{\mathbf{s b}}{ }^{\prime} \cap \operatorname{dom}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)=\{ \}\)
by auto
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
from enough-flushs-nth-update [OF i-bound, where \(\mathrm{sb}=[]\) ] have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}\) ) have valid-program-history \({ }^{\prime}\) : valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
have causal': causal-program-history is sbb \(^{\prime}{ }^{\prime} \mathrm{sb}^{\prime}\)
by ( \(\operatorname{simp}\) add: is \(\mathrm{s}_{\mathrm{sb}} \mathrm{sb}\) sb )
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{sb}^{\prime} \mathrm{sb}\) )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed from is-sim have is: is \(=\) RMW a \(t(D, f)\) cond ret A L R W\# is sb \(^{\prime}{ }^{\prime}\)
by (simp add: suspends sb is \(\mathrm{sb}_{\mathrm{sb}}\) )
with ts-i
have ts-i: ts! \(i=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{RMW}\right.\) at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W\# is \({ }_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()\),
\(\mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) )
by (simp add: suspends sb)
```

    have flush-all-until-volatile-write ts
    proof -
    ```

```

                (let (-,-,-,sb
            in a & outstanding-refs is-non-volatile-Write sb (takeWhile (Not o is-volatile-Write sb
    proof -

```

\(\left.\mathrm{sb}_{\mathrm{j}}\right)\) )
    \{
```

    assume j-bound: j < length ts sb
    assume neq-i-j: i }=\textrm{j
    ```

```

    have a }\not\in\mathrm{ outstanding-refs is-non-volatile-Writesb
    sbj
proof
let ?take-sb
let ?drop-sb
assume a-in: a }\in\mathrm{ outstanding-refs is-non-volatile-Write
with outstanding-refs-takeWhile [where }\mp@subsup{\textrm{P}}{}{\prime}=\mathrm{ Not ० is-volatile-Write}\mp@subsup{}{sb}{}
have a-in': a }\in\mathrm{ outstanding-refs is-non-volatile-Write
by auto
with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]]
have j-owns: a }\in\mp@subsup{\mathcal{O}}{j}{}\cup\mathrm{ all-acquired sb
by auto
from rels-cond [rule-format, OF j-bound [simplified leq] neq-i-j] ts-sim [rule-format,
OF j-bound] jth
have no-unsharing:release ?take-sb
by (auto simp add: Let-def)
from access-cond
show False
proof
assume a }\in\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{
with ownership-distinct [OF i-bound j-bound neq-i-j ts ssb-i jth]
j-owns
show False
by auto
next
assume a-shared: a }\in\operatorname{dom}\mathcal{S
with share-all-until-volatile-write-thread-local [OF ownership-distinct-ts sb
sharing-consis-tssb j-bound jth j-owns]
have a-dom: a }\in\mathrm{ dom (share ?take-sb j }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mathrm{ )
by (auto simp add: }\mathcal{S}\mathrm{ domIff)
from outstanding-non-volatile-writes-unshared [OF j-bound jth]
have non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\textrm{sb}\mathrm{ j.

```

```

sbj
(dropWhile (Not ○ is-volatile-Write }\mp@subsup{\mathrm{ sb }}{\mathrm{ ( ) sb }}{\textrm{j}}\mathrm{ )]
have unshared-take: non-volatile-writes-unshared S S sb (takeWhile (Not o
is-volatile-Write sb) sb j
by clarsimp

```
from release-not-unshared-no-write-take [OF unshared-take no-unsharing
a-dom] a-in
                show False by auto
        qed
        qed
\}
thus ?thesis
```

    by (fastforce simp add: Let-def)
    qed

```
from flush-all-until-volatile-write-buffered-val-conv
[OF - i-bound ts \(_{\text {sb }}-\mathrm{i}\) this]
show ?thesis
by (simp add: sb)
qed
hence \(\mathrm{m}-\mathrm{a}: \mathrm{ma}=\mathrm{m}_{\mathrm{sb}} \mathrm{a}\)
by (simp add: m)
from cond have cond \({ }^{\prime}: \neg \operatorname{cond}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m}\right.\) a \(\left.)\right)\)
by ( \(\operatorname{simp}\) add: m-a)
from direct-memop-step.RMWReadOnly \(\left[\right.\) where cond \(=\) cond and \(\vartheta=\vartheta_{\text {sb }}\) and \(\mathrm{m}=\mathrm{m}\),
OF cond \({ }^{\prime}\)
have (RMW a t (D, f) cond ret A L R W \# is sb \(^{\prime}\),
\[
\begin{aligned}
& \left.\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow \\
& \left(\mathrm{is}_{\mathrm{sb}}^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a}),(), \mathrm{m}, \text { False, } \mathcal{O}_{\mathrm{sb}}, \text { Map.empty, } \mathcal{S}\right) .
\end{aligned}
\]
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i [simplified sb, simplified] this]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.\),
\(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),()\), False, \(\mathcal{O}_{\mathrm{sb}}\), Map.empty \(\left.\left.)\right], \mathrm{m}, \mathcal{S}\right)\).

\section*{moreover}
have tmps-commute: \(\vartheta_{\mathbf{s b}}\left(\mathrm{t} \mapsto\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)=\)
\(\left(\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\{\mathrm{t}\}\right)\right)\left(\mathrm{t} \mapsto\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\)
apply (rule ext)
apply (auto simp add: restrict-map-def domIff)
done
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),()\right.\right.\right.\), False, \(\mathcal{O}_{\mathrm{sb}}\), Map.empty \(\left.\left.)\right], \mathrm{m}, \mathcal{S}\right)\) apply (rule sim-config.intros)
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathrm{m}\)
flush-all-until-volatile-nth-update-unused [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i, simplified sb])
using share-all-until-volatile-write-RMW-commute [OF i-bound \(\mathrm{ts}_{\mathbf{s b}^{\prime}}\)-i \(\left[\right.\) simplified is \(\mathrm{s}_{\mathbf{s b}} \mathrm{sb}\) ]]
apply (clarsimp \(\operatorname{simp}\) add: \(\mathcal{S} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\mathrm{sb}}\) )
using leq
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
using i-bound i-bound' ts-sim
apply (clarsimp simp add: Let-def nth-list-update
\(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) ex-not m-a
split: if-split-asm)
apply (rule tmps-commute)
done
ultimately
show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-sops' \({ }^{\prime}\) valid-dd' load-tmps-fresh' enough-flushs \({ }^{\prime}\)
valid-program-history' \({ }^{\prime}\) valid \(^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
next
case (SBHRMWWrite cond t a D f ret A L R W)
then obtain
\(\mathrm{is}_{\mathrm{sb}}\) : is \(\mathrm{is}_{\mathrm{sb}}=\) RMW at ( \(\mathrm{D}, \mathrm{f}\) ) cond ret A L R W \# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\) and
cond: \(\left(\operatorname{cond}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\) and
\(\mathcal{R}_{\mathrm{sb}}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\) Map.empty and
\(\mathcal{D}_{\mathrm{sb}}{ }^{\prime}: \neg \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) and
\(\vartheta_{\mathrm{sb}}{ }^{\prime}: \vartheta_{\mathrm{sb}}{ }^{\prime}=\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{ret}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\right.\) ) and
sb: \(\mathrm{sb}=[]\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=[]\) and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\) and
\(\mathcal{S}_{\mathrm{sb}}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\)
by auto
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
have D-subset: \(\mathrm{D} \subseteq \operatorname{dom} \vartheta_{\text {sb }}\)
by (simp add: is \(_{\text {sb }}\) )
from is-sim have is: is = RMW at (D,f) cond ret A L R W \# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: suspends sb is \(\mathrm{s}_{\mathrm{sb}}\) )
with ts-i
have ts-i: ts!i \(=\left(\mathrm{p}_{\mathrm{sb}}\right.\), RMW at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W \# is \(\left.{ }_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\)
by (simp add: suspends sb)
from safe-RMW-common [OF safe-memop-flush-sb [simplified is \(\mathrm{s}_{\mathrm{sb}}\) ]]
obtain access-cond: \(\mathrm{a} \in \mathcal{O}_{\mathrm{sb}} \vee \mathrm{a} \in \operatorname{dom} \mathcal{S}\) and
rels-cond: \(\forall \mathrm{j}<\) length ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\) released (ts \(!\mathrm{j})\) a \(\neq\) Some False
by (auto simp add: \(\mathcal{S} \mathrm{sb}\) )
have a-unflushed:
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
\[
\left(\operatorname{let}\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right.
\]
in a \(\notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\)
is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ))
proof -
\{
fix \(\mathrm{j}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(^{\text {ts }} \mathrm{s}_{\mathrm{sb}}\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
have a \(\notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )
\(\mathrm{sb}_{\mathrm{j}}\) )
proof
```

    let ?take-sb }\mp@subsup{j}{j}{}=(\mp@subsup{\mathrm{ takeWhile (Not o is-volatile-Write }}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{}
    ```

```

    assume a-in: a }\in\mathrm{ outstanding-refs is-non-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ ?take-sb 
    with outstanding-refs-takeWhile [where }\mp@subsup{\textrm{P}}{}{\prime}=\mathrm{ Not ० is-volatile-Writesb
    have a-in': a }\in\mathrm{ outstanding-refs is-non-volatile-Write sb sb
        by auto
    with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
    [OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound jth]]
    have j-owns: a }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
        by auto
    with ownership-distinct [OF i-bound j-bound neq-i-j ts sb-i jth]
    have a-not-owns: a }\not\in\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb
        by blast
    assume a-in: a \in outstanding-refs is-non-volatile-Write sb
    (takeWhile (Not o is-volatile-Writesb) sb }\mp@subsup{}{\textrm{j}}{\mathrm{ )}
    with outstanding-refs-takeWhile [where P}\mp@subsup{\textrm{P}}{}{\prime}=\mathrm{ Not o is-volatile-Writesb
    have a-in': a }\in\mathrm{ outstanding-refs is-non-volatile-Write sb }\mp@subsup{\textrm{sb}}{j}{
        by auto
            from rels-cond [rule-format, OF j-bound [simplified leq] neq-i-j] ts-sim [rule-format,
    OF j-bound] jth
have no-unsharing:release ?take-sb
by (auto simp add: Let-def)
from access-cond
show False
proof
assume a }\in\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{
with ownership-distinct [OF i-bound j-bound neq-i-j ts sb-i jth]
j-owns
show False
by auto
next
assume a-shared: a }\in\operatorname{dom}\mathcal{S
with share-all-until-volatile-write-thread-local [OF ownership-distinct-tssb
sharing-consis-ts sb j-bound jth j-owns]
have a-dom: a }\in\mathrm{ dom (share ?take-sbj}\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mathrm{ )
by (auto simp add: }\mathcal{S}\mathrm{ domIff)
from outstanding-non-volatile-writes-unshared [OF j-bound jth]
have non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{j}{}\mathrm{ .
with non-volatile-writes-unshared-append [of S S sb (takeWhile (Not o
is-volatile-Write sb
(dropWhile (Not o is-volatile-Write sb ) sb j}\mathrm{ )]
have unshared-take: non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mathrm{ (takeWhile (Not ○
is-volatile-Write sb ) sbj
by clarsimp
from release-not-unshared-no-write-take [OF unshared-take no-unsharing
a-dom] a-in
show False by auto
qed
qed

```

\section*{\}}
thus ?thesis
by (fastforce simp add: Let-def)
qed
have flush-all-until-volatile-write \(\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{a}=\mathrm{m}_{\mathrm{sb}} \mathrm{a}\) proof -
from flush-all-until-volatile-write-buffered-val-conv
[OF - i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i a-unflushed]
show ?thesis
by (simp add: sb)
qed
hence \(\mathrm{m}-\mathrm{a}: \mathrm{ma}=\mathrm{m}_{\mathrm{sb}} \mathrm{a}\)
by ( \(\operatorname{simp}\) add: m )
from cond have cond \({ }^{\prime}\) : cond \(\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m}\right.\) a) \()\)
by (simp add: m-a)
from safe-memop-flush-sb [simplified is \({ }_{\text {sb }}\) ] cond \({ }^{\prime}\)
obtain
L-subset: \(\mathrm{L} \subseteq \mathrm{A}\) and
A-shared-owned: A \(\subseteq \operatorname{dom} \mathcal{S} \cup \mathcal{O}_{\text {sb }}\) and
R-owned: \(\mathrm{R} \subseteq \mathcal{O}_{\mathrm{sb}}\) and
\(A-R: A \cap R=\{ \}\) and
a-unowned-others-ts:
\(\forall \mathrm{j}<\) length ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow(\mathrm{a} \notin\) owned \((\mathrm{ts}!\mathrm{j}) \cup\) dom (released \((\mathrm{ts}!\mathrm{j}))\) ) and
A-unowned-by-others-ts:
\(\forall \mathrm{j}<\) length ts. \(\mathrm{i} \neq \mathrm{j} \longrightarrow(\mathrm{A} \cap(\) owned \((\mathrm{ts}!\mathrm{j}) \cup\) dom (released \((\mathrm{ts}!\mathrm{j})))=\{ \})\) and
a-not-ro: a \(\notin\) read-only \(\mathcal{S}\)
by cases (auto simp add: sb)
from a-unowned-others-ts ts-sim leq
have a-unowned-others:
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
(let \(\left(-,-,,, \mathrm{sb}_{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{sb}}!\) j in
\(\mathrm{a} \notin\) acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}} \wedge\)
\(\mathrm{a} \notin\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ))
apply (clarsimp simp add: Let-def)
subgoal for j
apply (drule-tac \(\mathrm{x}=\mathrm{j}\) in spec)
apply (auto simp add: dom-release-takeWhile)
done
done
from A-unowned-by-others-ts ts-sim leq
have A-unowned-by-others:
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\)
in \(\mathrm{A} \cap\left(\right.\) acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}} \cup\)
all-shared (takeWhile (Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)\) ) \(=\{ \}\) )
apply (clarsimp simp add: Let-def)
subgoal for \(j\)
apply (drule-tac \(x=j\) in spec)
apply (force simp add: dom-release-takeWhile)
done
done
have a-not-ro': a \(\notin\) read-only \(\mathcal{S}_{\text {sb }}\) proof
assume a: a \(\in\) read-only \(\left(\mathcal{S}_{\mathrm{sb}}\right)\)
from local.read-only-unowned-axioms have read-only-unowned \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\).
from in-read-only-share-all-until-volatile-write \({ }^{\prime}\) [OF ownership-distinct-ts \({ }_{\text {sb }}\) shar-ing-consis-ts \({ }_{\text {sb }}\)〔read-only-unowned \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i a-unowned-others, simplified sb , simplified,

OF a]
have a \(\in\) read-only \((\mathcal{S})\)
by (simp add: \(\mathcal{S}\) )
with a-not-ro show False by simp
qed
\(\{\)
fix j
fix \(p_{j}\) is \(_{\text {sbj }} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
have a \(\notin\) unforwarded-non-volatile-reads (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\}\)
proof
let ? take-sb \({ }_{\mathrm{j}}=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
let ?drop-sb \({ }_{\mathrm{j}}=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
assume a-in: a \(\in\) unforwarded-non-volatile-reads ?drop-sb \({ }_{j}\{ \}\)
from a-unowned-others [rule-format, OF - neq-i-j] ts \(\mathrm{sb}_{\mathrm{sb}}\)-j j-bound
obtain a-unacq-take: a \(\notin\) acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) and a-not-shared: a \(\notin\) all-shared ?take-sb \({ }_{j}\)
by auto
note nvo-j \(=\) outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \(\operatorname{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-drop-j: non-volatile-owned-or-read-only True (share ?take-sb \(\mathcal{S}_{\mathrm{sb}}\) )
(acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ? drop-sb \(\mathrm{S}_{\mathrm{j}}\) •
note consis- \(\mathrm{j}=\) sharing-consis [OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
with sharing-consistent-append [of \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ? take-sb \({ }_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
obtain consis-take-j: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\) and consis-drop-j: sharing-consistent (share ?take-sb \(\mathcal{S}_{\mathrm{sb}}\) )
(acquired True ? \({ }^{\text {take-sb }}{ }_{j} \mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \(_{\mathrm{j}}\)
by auto
from in-unforwarded-non-volatile-reads-non-volatile-Read \({ }_{\mathbf{s b}}\) [OF a-in] have a-in': a \(\in\) outstanding-refs is-non-volatile-Read \({ }_{s b}\) ?drop-sb \({ }_{j}\).
note reads-consis- \(\mathrm{j}=\) valid-reads \(\left[\mathrm{OF} \mathrm{j}\right.\)-bound \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
from reads-consistent-drop [OF this]
have reads-consis-drop-j:
reads-consistent True (acquired True ?take-sb \(\mathcal{O}_{j}\) ) (flush ?take-sb \(\mathrm{m}_{\mathrm{jb}}\) ) ?drop-sb \({ }_{\mathrm{j}}\).

\section*{from read-only-share-all-shared [of a ?take-sb \(\mathcal{S}_{\mathrm{sb}}\) ] a-not-ro' a-not-shared \\ have a-not-ro-j: a \(\notin\) read-only (share ?take-sb \({ }_{j} \mathcal{S}_{\text {sb }}\) ) \\ by auto}
```

from ts-sim [rule-format, OF j-bound] ts tsb-j j-bound
obtain suspendsj is }\mp@subsup{\textrm{D}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mathrm{ where
suspendsj}\mp@subsup{\textrm{j}}{\mathrm{ : suspendsj}}{\textrm{j}}=\mathrm{ = ?drop-sb
is}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ : instrs suspendsj @ is isbj = is }\mp@subsup{\textrm{i}}{\textrm{j}}{}@\mathrm{ prog-instrs suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ and
\mathcal{D}
tsj: ts!j = (hd-prog pj suspendsj, is }\mp@subsup{\textrm{s}}{\textrm{j}}{}
\vartheta
\mathcal{D}}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ , acquired True ?take-sb}\mp@subsup{}{\textrm{j}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{,}\mathrm{ ,release ?take-sb
by (auto simp: Let-def)
from tsj neq-i-j j-bound
have ts'-j: ?ts! ! = (hd-prog p pospends }\mp@subsup{\textrm{p}}{\textrm{j}}{}\mathrm{ , is }\mp@subsup{\textrm{s}}{\textrm{j}}{}\mathrm{ ,
\vartheta 滴(dom }\mp@subsup{\vartheta}{\textrm{j}}{}\mathrm{ - read-tmps suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}),()
\mathcal{D}}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ , acquired True ?take-sb}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{\mathrm{ ,release ?take-sb}
by auto

```
from valid-last-prog \(\left[\mathrm{OF} j\right.\)-bound \(\left.\mathrm{ts}_{\mathrm{sb}-\mathrm{j}}\right]\) have last-prog: \(\operatorname{last-prog~} \mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
from j-bound i-bound \({ }^{\prime}\) leq have j -bound-ts': \(\mathrm{j}<\) length ? \(\mathrm{ts}^{\prime}\)
    by simp
from read-only-read-acquired-unforwarded-acquire-witness [OF nvo-drop-j consis-drop-j
a-not-ro-j a-unacq-take a-in]
have False
proof
    assume \(\exists\) sop \(\mathrm{a}^{\prime}\) v ys zs A L R W.
?drop-sb \({ }_{j}=\) ys @ Write \(_{\text {sb }}\) True a'sop v A LRW\#zs \(\wedge \mathrm{a} \in \mathrm{A} \wedge\)
            \(a \notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) ys \(\wedge a^{\prime} \neq a\)
    with suspends \({ }_{j}\)
    obtain \(\mathrm{a}^{\prime}\) sop \({ }^{\prime} \mathrm{v}^{\prime}\) ys \(\mathrm{zs}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
```

    split-suspendsj: suspendsj = ys @ Writesb True a' sop' v' A' L' R' W'# zs' (is
    suspends}\mp@subsup{}{\textrm{j}}{=}=\mathrm{ ?suspends) and
a-A': a }\in\mp@subsup{A}{}{\prime}\mathrm{ ' and
no-write: a }\not\in\mathrm{ outstanding-refs is-Writesb (ys @ [Write sb True a' sop'v}\mp@subsup{v}{}{\prime}\mp@subsup{\textrm{A}}{}{\prime}\mp@subsup{\textrm{L}}{}{\prime}\mp@subsup{\textrm{R}}{}{\prime}\mp@subsup{\textrm{W}}{}{\prime}]
by(auto simp add: outstanding-refs-append )
from last-prog
have lp: last-prog p}\mp@subsup{\textrm{p}}{\textrm{j}}{}\mp@subsup{\mathrm{ suspends}}{\textrm{j}}{}=\mp@subsup{\textrm{p}}{\textrm{j}}{
apply -
apply (rule last-prog-same-append [where sb=?take-sb}\mp@subsup{}{j}{}]\mathrm{ )
apply (simp only: split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{[symmetric] suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{}\mathrm{ )
apply simp
done

```
    from sharing-consis [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
    have sharing-consis-j: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
    then have \(\mathrm{A}^{\prime}-\mathrm{R}^{\prime}: \mathrm{A}^{\prime} \cap \mathrm{R}^{\prime}=\{ \}\)
        by (simp add: sharing-consistent-append [of - - ?take-sb \({ }_{j}\) ? drop-sb \({ }_{j}\), simplified]
    suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] split-suspends \(\mathrm{j}_{\mathrm{j}}\) sharing-consistent-append)
    from valid-program-history [OF j-bound \(\mathrm{ts}_{\mathbf{s b}}\)-j]
    have causal-program-history is \(_{s_{b j}} \mathrm{sb}_{\mathrm{j}}\).
    then have cph: causal-program-history is sbj ?suspends
        apply -
        apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? take- \(^{\text {sb }} \mathrm{b}_{\mathrm{j}}\) ] )
        apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
        apply (simp add: split-suspends \({ }_{\mathrm{j}}\) )
        done
    from valid-reads [OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
    have reads-consis- \(j\) : reads-consistent False \(\mathcal{O}_{j} m_{s b} \mathrm{sb}_{\mathrm{j}}\).
    from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing
\(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) >
            j -bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}-\mathrm{j}\) this]
    have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ) m suspends \({ }_{\mathrm{j}}\)
        by ( \(\operatorname{simp}\) add: m suspends \(_{\mathrm{j}}\) )
    from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j \(\mathrm{ts}_{\mathbf{s b}_{\boldsymbol{b}}}\)-i
\(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
    have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read st \(_{\text {sb }}\)
suspends \(_{\mathrm{j}}=\{ \}\)
    by ( \(\operatorname{simp}\) add: suspends \(\mathrm{j}_{\mathrm{j}}\) )
    from reads-consistent-flush-independent [OF this reads-consis-m-j]
    have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) )
        (flush ?drop-sb m) suspends \({ }_{\mathrm{j}}\).
    hence reads-consis-ys: reads-consistent True (acquired True ?take-sb \(\mathrm{B}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
        (flush ?drop-sb m) (ys@[Write \({ }_{\text {sb }}\) True \(\left.\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\)
    by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) reads-consistent-append)
from valid-write-sops [OF j-bound \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops (ys@[Write \({ }_{\text {sb }}\) True \(\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \(@_{j}\) suspends \(_{j}\) )
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{\mathrm{j}} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspendsj
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \({ }_{j}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h-consis:
history-consistent \(\vartheta_{\mathrm{j}}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(@_{\mathrm{j}}\) suspends \(\left._{\mathrm{j}}\right)\) ) (?take-sb \(\mathrm{@}_{\mathrm{j}}\) Suspends \(_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\) ) \({ }^{\text {take-sb }} \mathrm{ta}_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) )
proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(_{\mathrm{j}} @\) ? drop- \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\) by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \({ }_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) by (simp only: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left.\left._{\mathrm{j}}\right)\right)\) ?take-sb \(\mathrm{t}_{\mathrm{j}}=\) last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\) apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) ) by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\mathrm{j}}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) suspends \({ }_{\mathrm{j}}\)
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read \({ }_{\text {sb }}\)
\(\left(y s @\left[W_{r i t e}^{s b}\right.\right.\) True \(a^{\prime}\) sop \(\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] split-suspends \({ }_{j}\) )
have acq-simp:
acquired True (ys @ \(\left[W^{2} i t e_{\text {sb }}\right.\) True \(\mathrm{a}^{\prime}\) sop \(\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}\right]\) )
(acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) \(=\)
acquired True ys (acquired True ? take-sb \(\left.\mathrm{H}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
by (simp add: acquired-append)
from flush-store-buffer-append [where \(s b=y s @\left[W_{r i t e}^{s b}\right.\) True \(\left.a^{\prime} \operatorname{sop}^{\prime} v^{\prime} A^{\prime} L^{\prime} R^{\prime} W^{\prime}\right]\) and \(\mathrm{sb}^{\prime}=\mathrm{zs}^{\prime}\), simplified,

OF j-bound-ts \({ }^{\prime} \mathrm{is}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{j}_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ]
ts'-j [simplified split-suspends \({ }^{\mathrm{j}}\) ] refl lp [simplified split-suspends \({ }_{\mathrm{j}}\) ] reads-consis-ys
hist-consis' \({ }^{\prime}\) [simplified split-suspendsj] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb}}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) share ?drop-sb \(\mathcal{S}]\)
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : instrs \(\mathrm{zs}^{\prime} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs \(\mathrm{zs}^{\prime}\) and
steps-ys: (?ts \({ }^{\prime}\), flush ?drop-sb m, share ?drop-sb \(\mathcal{S}\) ) \(\Rightarrow_{\mathrm{d}}{ }^{*}\)
(?ts \({ }^{\prime}[\mathrm{j}:=\) (last-prog
(hd-prog \(p_{j}\left(\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) \()\) ) (ys@[Write \({ }_{\text {sb }}\)
True \(\mathrm{a}^{\prime}\) sop \(\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \mathrm{J}\right)\),
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\mathrm{j}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{j}}-\right.\) read-tmps zs \(\left.{ }^{\prime}\right)\),
(), True, acquired True ys (acquired True ?take-sb \(\left.{ }_{j} \mathcal{O}_{\mathrm{j}}\right) \cup \mathrm{A}^{\prime}-\)
\(\left.\left.\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]\),
flush (ys@[Write sb True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ' \()\) ) (flush ?drop-sb m), share (ys@[Write \({ }_{\text {sb }}\) True \(\left.\left.\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\) (share ?drop-sb \(\left.\mathcal{S}\right)\) )
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}\), ?shared-ys) \()\)
by (auto simp add: acquired-append outstanding-refs-append)
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.\),
\(\mathcal{D}_{\mathrm{sb}}\), acquired True sb \(\mathcal{O}_{\mathrm{sb}}\), release sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
by \(\operatorname{simp}\)
note conflict-computation \(=\) rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe: safe-delayed (?ts-ys,?m-ys,?shared-ys).
from flush-unchanged-addresses [OF no-write]
have flush (ys @ \(\left[W^{\prime}\right.\) rite \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}\) ) maa=ma.
with safe-delayedE [OF safe i-bound-ys ts-ys-i, simplified is \(\mathrm{s}_{\mathrm{s}}\) ] cond \({ }^{\prime}\)
have a-unowned:
\(\forall \mathrm{j}<\) length ?ts-ys. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{j}}\right)=\) map owned ?ts-ys! j in a \(\left.\notin \mathcal{O}_{\mathrm{j}}\right)\)
apply cases
apply (auto simp add: Let-def is sbb sb) done
from a-A' a -unowned [rule-format, of j ] neq-i-j j-bound leq \(\mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
show False
by (auto simp add: Let-def)
next
 outstanding-refs is-Write \({ }_{\text {sb }}\) ys
with suspends \({ }_{j}\)
obtain ys \(\mathrm{zs}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
split-suspendsj: suspends \(\mathrm{s}_{\mathrm{j}}=\) ys @ Ghost sb \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}^{\prime}\) (is suspends \(\mathrm{j}_{\mathrm{j}}=\) ?suspends)
and
\(a-A^{\prime}: a \in A^{\prime}\) and
no-write: a \(\notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) (ys @ [Ghostsb \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\) )
by (auto simp add: outstanding-refs-append)
from last-prog
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where sb=?take-sb \({ }_{j}\) ])
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
apply simp
done
from sharing-consis [ \(\mathrm{OF} j\)-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have sharing-consis-j: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
then have \(A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}\)
by (simp add: sharing-consistent-append [of - ? ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\), simplified]
suspends \({ }_{j}\) [symmetric] split-suspends \({ }_{j}\) sharing-consistent-append)
from valid-program-history [OF j-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{j}\) ]
have causal-program-history is sbj \(\mathrm{sb}_{\mathrm{j}}\).
then have cph: causal-program-history is \(\mathrm{sbj}_{\mathrm{sbj}}\) ?suspends

\section*{apply -}
apply (rule causal-program-history-suffix [where sb=?take-sb \({ }_{j}\) ] )
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \({ }_{\mathrm{j}}\) )
done
from valid-reads [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]
have reads-consis-j: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{tt}_{\mathrm{sb}}\) )
\(j\)-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \(b_{j} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{m}_{\mathrm{j}}\) by (simp add: m suspends \(_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound neq-i-j ts \(\mathrm{s}_{\mathbf{s b}}\)-i \(\mathrm{ts}_{\text {sb-j] }}\)
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \(_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by (simp add: suspends \({ }_{\mathrm{j}}\) )
from reads-consistent-flush-independent [OF this reads-consis-m-j]
have reads-consis-flush-suspend: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{j}\) ) (flush ?drop-sb m) suspendsj.
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) )
(flush ?drop-sb m) (ys@[Ghost sb \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) ) by (simp add: split-suspends \({ }_{j}\) reads-consistent-append)
from valid-write-sops [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ? take-sb \({ }_{j}\). valid-sop sop and valid-sops-drop: \(\forall\) sop \(\in\) write-sops ( \(\mathrm{ys}^{@}\) [Ghost \(\left.\mathrm{sb}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) ). valid-sop sop apply (simp only: write-sops-append)
apply auto done
from read-tmps-distinct [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j}\) @suspends \({ }_{j}\) )
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h -consis:
history-consistent \(\vartheta_{\mathrm{j}}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(\left._{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}\right)\) ) (?take-sb \({ }_{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left.\mathrm{j}_{\mathrm{j}}\right)\)
proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) @?drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\) last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ? thesis
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\mathrm{j}}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\) by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read \({ }_{s b}\) \(\left(y_{s} @\left[\right.\right.\) Ghost \(\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \({ }_{j}\) )
have acq-simp:
acquired True (ys @ [Ghost sb \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) )
(acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) \(=\)
acquired True ys (acquired True ? take-sb \(\left.{ }_{j} \mathcal{O}_{j}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
by (simp add: acquired-append)
from flush-store-buffer-append \(\left[\right.\) where \(\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Ghost}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) and \(\mathrm{sb}^{\prime}=\mathrm{zs}^{\prime}\), simplified,

OF j-bound-ts \({ }^{\prime} \mathrm{is}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ]
ts \({ }^{\prime}-\mathrm{j}\) [simplified split-suspendsj] refl lp [simplified split-suspendsj] reads-consis-ys
hist-consis' \({ }^{\prime}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb-}}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) share ?drop-sb \(\mathcal{S}]\)
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : instrs \(\mathrm{zs}^{\prime} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs \(\mathrm{zs}^{\prime}\) and
steps-ys: (?ts \({ }^{\prime}\), flush ?drop-sb \(m\), share ?drop-sb \(\left.\mathcal{S}\right) \nRightarrow_{\mathrm{d}}{ }^{*}\)
(? \(\mathrm{ts}^{\prime} \mathrm{j} \mathrm{j}:=\) (last-prog
(hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Ghost \(_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) \()\) ) (ys@[Ghost sb \(\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\), \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\), \(\left.\vartheta_{\mathrm{j}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{j}}-\right.\) read-tmps zs \()\),
(),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys
(acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) \(\cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )],
flush (ys@[Ghost \(\left.{ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) ) (flush ?drop-sb m),
share (ys@[Ghost sb \(\left.\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\) (share ?drop-sb \(\left.\mathcal{S}\right)\) )
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}\), ?shared-ys) \()\)
by (auto simp add: acquired-append outstanding-refs-append)
from i-bound ' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound ' neq-i-j
have ts-ys- \(\mathrm{i}: ~ ? \mathrm{ts}-\mathrm{ys}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.\),
\(\mathcal{D}_{\mathrm{sb}}\), acquired True sb \(\mathcal{O}_{\mathrm{sb}}\), release \(\left.\mathrm{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
by simp
note conflict-computation \(=\) rtranclp-trans [OF steps-flush-sb steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe: safe-delayed (?ts-ys,?m-ys,?shared-ys).
from flush-unchanged-addresses [OF no-write]
have flush (ys @ [Ghost sb \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\) ) m a \(=\mathrm{m} \mathrm{a}\).
with safe-delayedE [OF safe i-bound-ys ts-ys-i, simplified is sb \(_{\text {b }}\) ] cond \({ }^{\prime}\)
have a-unowned:
\(\forall \mathrm{j}<\) length ?ts-ys. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{j}}\right)=\) map owned ?ts-ys! j in \(\left.\mathrm{a} \notin \mathcal{O}_{\mathrm{j}}\right)\)
apply cases
apply (auto simp add: Let-def is sb \(^{s b}\) )
done
from a-A' a-unowned [rule-format, of j] neq-i-j j-bound leq \(A^{\prime}-R^{\prime}\)
show False
by (auto simp add: Let-def)
qed
then show False
by simp
qed
\}
note a-notin-unforwarded-non-volatile-reads-drop \(=\) this
have A-unused-by-others:
\(\forall \mathrm{j}<\) length (map \(\mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\) ). \(\mathrm{i} \neq \mathrm{j} \longrightarrow\)
(let \(\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)=\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\) ! j
in \(\mathrm{A} \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) outstanding-refs is-volatile-Write \(\left.\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\right)=\{ \}\right)\)
proof -
\(\{\)
fix \(\mathrm{j} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
assume neq- \(\mathrm{i}-\mathrm{j}: \mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}:\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right)!j=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)\)
assume conflict: \(\mathrm{A} \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) outstanding-refs is-volatile-Write \(\left.\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\right) \neq\{ \}\)
have False
proof -
from j-bound leq
have j-bound': j < length (map owned ts)
by auto
from j-bound have \(j\)-bound \({ }^{\prime \prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
by auto
from j-bound \({ }^{\prime}\) have j -bound \({ }^{\prime \prime \prime}: \mathrm{j}\) < length ts
by simp

\section*{from conflict obtain \(\mathrm{a}^{\prime}\) where}
a-in: \(a^{\prime} \in A\) and
conflict: \(\mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{j}} \vee \mathrm{a}^{\prime} \in\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\)
by auto
from A-unowned-by-others [rule-format, OF - neq-i-j] j-bound \(\mathrm{ts}_{\mathrm{sb}}\)-j
have A -unshared-j: \(\mathrm{A} \cap\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \(\mathrm{sb}_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(=\)
by (auto simp add: Let-def)
from conflict
show ?thesis
proof
assume \(a^{\prime} \in \mathcal{O}_{j}\)
from all-shared-acquired-in [OF this] A-unshared-j a-in
have conflict: \(\mathrm{a}^{\prime} \in \operatorname{acquired} \operatorname{True}\left(t a k e W h i l e ~\left(N o t \circ\right.\right.\) is-volatile-Write \(\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}}\) by (auto)
with A-unowned-by-others [rule-format, OF - neq-i-j] j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) a-in
show False by auto
next
assume \(\mathrm{a}-\mathrm{in}-\mathrm{j}: \mathrm{a}^{\prime} \in\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}\)
let ? take-sb \(_{\mathrm{j}}=\left(\right.\) takeWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)\)
let ?drop-sb \({ }_{\mathrm{j}}=\left(\right.\) dropWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)\)
from ts-sim [rule-format, OF j-bound'] ts \(\mathrm{stb}_{\mathrm{sb}} \mathrm{j} \mathrm{j}\)-bound \({ }^{\prime \prime}\)
obtain \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) is \({ }_{\text {sbj }} \mathcal{D}_{\text {sbj }} \mathcal{D}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \vartheta_{\mathrm{sbj}}\) is \(\mathrm{is}_{\mathrm{j}}\) where
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\) and
suspends j : suspends \(\mathrm{j}=\) ? \(\mathrm{drop}^{2}-\mathrm{sb}_{\mathrm{j}}\) and
\(\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.\) outstanding-refs is-volatile-Write \(\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)\) and
\(\mathrm{is}_{\mathrm{j}}\) : instrs suspends \(\mathrm{s}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{s}_{\mathrm{j}} @\) prog-instrs suspends \(\mathrm{s}_{\mathrm{j}}\) and
\(\mathrm{ts}_{\mathrm{j}}: \mathrm{ts}!\mathrm{j}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{s}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\text { dom } \vartheta_{\text {sbj }} \text { - read-tmps suspends }\right)_{j},(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\), release
?take-sb \(\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
apply (cases \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\) )
apply (force simp add: Let-def)
done
```

have a' }\in\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mp@subsup{\mathrm{ suspendsj}}{j}{
proof -
from a-in-j
have a'\in outstanding-refs is-volatile-Write e
by simp
thus ?thesis

```
```

    apply (simp only: outstanding-refs-append suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
    apply (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
    done
    qed
    from split-volatile-Write }\mp@subsup{\mathrm{ sb}}{\mathbf{b}}{
    obtain sop' v' ys zs A' }\mp@subsup{L}{}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{W}{}{\prime}\mathrm{ where
    split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ : suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{= ys @ Write 
    = ?suspends)
by blast

```
    from valid-program-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
    have causal-program-history is \(_{\text {sbj }} \mathrm{sb}_{\mathrm{j}}\).
    then have cph: causal-program-history is sbj ?suspends
    apply -
    apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? take- \(^{\text {sb }}{ }_{\mathrm{j}}\) ] )
    apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \({ }_{j}\) )
    apply (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) )
    done
    from valid-last-prog \(\left[\mathrm{OF} \mathrm{j}\right.\)-bound \(\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}} \mathrm{j}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
    then
    have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ?suspends \(=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \(\mathrm{s}_{\mathrm{j}}\) ])
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
from valid-reads [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} m_{s b} s b_{j}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing
\(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) )
j-bound \({ }^{\prime \prime}\)
\(\mathrm{ts}_{\mathrm{sb}}\)-j this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \(\mathrm{J}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{m}_{\mathrm{j}}\) by ( \(\operatorname{simp}\) add: m suspends \({ }_{j}\) )
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \({ }^{\prime \prime} \operatorname{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
have nvo-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
with non-volatile-owned-or-read-only-append [of False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
have nvo-take-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\)
by auto
from a-unowned-others [rule-format, OF - neq-i-j] ts \(\mathrm{st}_{\mathrm{sb}}-\mathrm{j} j\)-bound
have a-not-acq: a \(\notin\) acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\)
by auto
from a-notin-unforwarded-non-volatile-reads-drop[OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}\)-j neq-i-j]
have a-notin-unforwarded-reads: a \(\notin\) unforwarded-non-volatile-reads suspends \({ }_{j}\{ \}\) by (simp add: suspends \({ }_{\mathrm{j}}\) )
let \(? \mathrm{ma}=\left(\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right)\)
from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads [where \(\mathrm{W}=\{ \}\) and \(\mathrm{m}^{\prime}=\) ? ma,simplified, OF - subset-refl reads-consis-m-j]
a-notin-unforwarded-reads
have reads-consis-ma-j:
reads-consistent True (acquired True ? take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma suspends \({ }_{\mathrm{j}}\) by auto
from reads-consis-ma-j
have reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma (ys) by (simp add: split-suspends \({ }_{j}\) reads-consistent-append)
from direct-memop-step.RMWWrite [where cond \(=\) cond and \(\vartheta=\vartheta_{\mathrm{sb}}\) and \(\mathrm{m}=\mathrm{m}, \mathrm{OF}\) cond \({ }^{\prime}\) ]
have (RMW at (D, f) cond ret A L R W\# is \(\left.{ }^{\prime}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\)
\[
\left(\mathrm{is}_{\mathrm{sb}}^{\prime}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{~m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a})\right)\right)\right),(), ? \mathrm{ma}, \text { False, } \mathcal{O}_{\mathrm{sb}} \cup \mathrm{~A}-\mathrm{R},\right.
\] Map.empty, \(\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have step-a: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\)
\[
\left(\mathrm { tss } \left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{~m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} a)\right)\right)\right),(), \text { False, } \mathcal{O}_{\mathrm{sb}} \cup\right.\right.\right.
\]

A - R,Map.empty)],
\[
\left.? \mathrm{ma}, \mathcal{S} \oplus \mathrm{w} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
\]
(is \(-\Rightarrow_{d}(? \mathrm{ts}-\mathrm{a},-\), ?shared-a)).
from \(t s_{j}\) neq-i-j \(j\)-bound
have ts-a-j: ?ts-a! \(=\left(\begin{array}{l}\text { hd-prog } \\ p_{j} \\ \text { suspends } \\ j\end{array}, \mathrm{is}_{\mathrm{j}}\right.\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps suspends \(\left.{ }_{j}\right),(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\),release ?take-sb \(\left.{ }_{\mathrm{j}}\left(\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}}\right)\right) \mathcal{R}_{\mathrm{j}}\right)\)
by auto
from valid-write-sops [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j}\) @?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ? take-sb \({ }_{j}\). valid-sop sop and valid-sops-drop: \(\forall\) sop \(\in\) write-sops (ys). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j}\) @suspends \({ }_{j}\) )
by (simp add: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \({ }_{j}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{\mathrm{j}} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and distinct-read-tmps-drop: distinct-read-tmps suspends \(\mathrm{s}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(h\)-consis:

apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \({ }_{\mathrm{j}}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left._{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{b}_{\mathrm{j}} @\) ? drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \({ }_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )

\section*{moreover}
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{t}_{\mathrm{j}} @\) suspends \(\left.\left._{\mathrm{j}}\right)\right)\) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by ( \(\operatorname{simp}\) add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\) by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}}(\mathrm{ys})=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{\mathrm{j}}\) [symmetric]
split-suspends \({ }_{j}\) )
from j-bound \({ }^{\prime}\) have j-bound-ts-a: \(\mathrm{j}<\) length ?ts-a by auto
from flush-store-buffer-append [where \(\mathrm{sb}=\mathrm{ys}\) and \(\mathrm{sb}^{\prime}=\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime}\) \(R^{\prime} W^{\prime} \# z s\), simplified,

OF j-bound-ts-a is \({ }_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ] ts-a-j [simplified split-suspends \({ }_{j}\) ] refl lp [simplified split-suspends \({ }_{j}\) ] reads-consis-ys hist-consis \({ }^{\prime}\left[\right.\) simplified split-suspends \({ }_{j}\) ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?shared-a]
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : Write True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) instrs \(\mathrm{zs} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs zs and steps-ys: (?ts-a, ?ma, ?shared-a) \(\Rightarrow_{d^{*}}{ }^{*}\)
(?ts-a \([\mathrm{j}:=\) (last-prog
(hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{zs}\) ) ys, \(i s_{j}{ }^{\prime}\),
\(\left.\vartheta_{\mathrm{sbj}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sbj}}\right.\) - read-tmps zs),
(), \(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \(\mathrm{sb}_{\mathrm{sb}}\) ys \(\neq\{ \}\), acquired True ys (acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )],
flush ys (?ma), share ys (?shared-a))
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}\), ?shared-ys) \()\) by (auto simp add: acquired-append)
from cph
 by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is sbj \(^{\text {zs }}\).
interpret causal \({ }_{\mathrm{j}}\) : causal-program-history is \(_{\text {sbj }}\) zs by (rule cph \({ }^{\text {') }}\)
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }_{j}{ }^{\prime}\)
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Write True \(\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ is \(\mathrm{sbj}_{\mathrm{sb}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime} @\) prog-instrs zs
by clarsimp
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound \({ }^{\prime}\) neq-i-j
have ts-ys-i: ?ts-ys! \(i=\left(p_{\text {sb }}\right.\), is \(_{\text {sb }}{ }^{\prime}\),
\(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{ma})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{ma})\right)\right),()\right.\), False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\), Map.empty \()\)
by simp
from j-bound-ts-a have j-bound-ys: \(\mathrm{j}<\) length ?ts-ys
by auto
then have ts-ys-j: ?ts-ys! \(\mathrm{j}=\left(\operatorname{last-prog}\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}} \mathrm{zs}\right)\right.\) ys, Write True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime}\) \(\mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime},\left.\vartheta_{\mathrm{sbj}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{sbj}}-\right.\) read-tmps zs \(),(), \mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\),
acquired True ys (acquired True ? take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )
by (clarsimp simp add: \(\mathrm{is}_{\mathrm{j}}\) )
note conflict-computation \(=\) r-rtranclp-rtranclp [OF step-a steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j]
have a-unowned:
\(\forall \mathrm{i}<\) length ts. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{i}}\right)\)
```

apply cases
apply (auto simp add: Let-def)
done
from a-in a-unowned [rule-format, of i] neq-i-j i-bound' A-R
show False
by (auto simp add: Let-def)
qed
qed
}
thus ?thesis
by (auto simp add: Let-def)
qed
have A-unacquired-by-others:
j}<l=ngth (map \mathcal{O-sb ts sb}). i f j \longrightarrow
(let (\mathcal{O}
in A \cap all-acquired sbj}={}
proof -
{
fix j ()
assume j-bound: j < length (map owned tssb)
assume neq-i-j: i\not=j
assume ts scb-j:(map \mathcal{O-sb ts sbb})!j=(\mathcal{O}
assume conflict: A \cap all-acquired sb
have False
proof -
from j-bound leq
have j-bound': j < length (map owned ts)
by auto
from j-bound have j-bound '/: j < length ts sb
by auto
from j-bound' have j-bound ''\prime: j < length ts
by simp
from conflict obtain a' where
a}\mp@subsup{}{}{\prime}-\textrm{in}:\mp@subsup{\textrm{a}}{}{\prime}\in\textrm{A}\mathrm{ and
a'-in-j: a' }\in\mathrm{ all-acquired sbj
by auto
let ?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{=(\mp@subsup{\mathrm{ takeWhile (Not O is-volatile-Write}}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{})
let ?drop-sb
from ts-sim [rule-format, OF j-bound '\eta ts tsb-j j-bound "
obtain p puspendsj is isbj }\mp@subsup{\vartheta}{\textrm{sbj}}{}\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\mathrm{ isj where}}{\textrm{j}}{
ts sb-j: ts ssb
suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ : suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}=\mathrm{ ?drop-sb
is}\mp@subsup{j}{j}{}: instrs suspendsj @ is sbbj = is j @ prog-instrs suspends j and
\mathcal{D}}\textrm{j}:\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}=(\mp@subsup{\mathcal{D}}{\textrm{j}}{}\vee\mathrm{ outstanding-refs is-volatile-Write
tsj: ts!j = (hd-prog pos suspendsj, is
\vartheta

```
```

                \mathcal{D}}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathrm{ , acquired True ?take-sb }}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathbf{j}
        apply (cases ts sb!j)
        apply (force simp add: Let-def)
        done
    ```
```

    from a'-in-j all-acquired-append [of ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\) ]
    have \(\mathrm{a}^{\prime} \in\) all-acquired ?take-sb \(\mathrm{b}_{\mathrm{j}} \vee \mathrm{a}^{\prime} \in\) all-acquired suspends \(\mathrm{j}_{\mathrm{j}}\)
    by (auto simp add: suspends \({ }_{\mathrm{j}}\) )
    thus False
    proof
        assume \(\mathrm{a}^{\prime} \in\) all-acquired ?take-sb \({ }_{j}\)
        with A-unowned-by-others [rule-format, OF - neq-i-j] ts \(\mathrm{sb}_{\mathrm{sb}}\)-j j-bound \(\mathrm{a}^{\prime}\)-in
        show False
    by (auto dest: all-acquired-unshared-acquired)
    next
        assume conflict-drop: \(\mathrm{a}^{\prime} \in\) all-acquired suspends \(_{j}\)
        from split-all-acquired-in [OF conflict-drop]
        show ?thesis
        proof
    assume \(\exists\) sop \(\mathrm{a}^{\prime \prime} \mathrm{v}\) ys zs A L R W.
                suspends \(_{j}=\) ys @ Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime \prime}\) sop v A LR W\# zs \(\wedge \mathrm{a}^{\prime} \in \mathrm{A}\)
    then
obtain $a^{\prime \prime}$ sop' $\mathrm{v}^{\prime}$ ys zs $\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ where
split-suspends $\mathrm{s}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}=$ ys @ Write ${ }_{\text {sb }}$ True $\mathrm{a}^{\prime \prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#$ zs (is suspends ${ }_{\mathrm{j}}$
$=$ ?suspends) and
$a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}$
by blast

```
from valid-program-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\text {sb }}-\mathrm{j}\) ]
have causal-program-history \(\mathrm{is}_{\text {sbj }} \mathrm{sb}_{j}\).
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? take-sb \(\mathrm{b}_{\mathrm{j}}\) ]
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) )
done
from valid-last-prog [OF j-bound \({ }^{\prime \prime}\) ts s \(\left._{\mathbf{s b}}-\mathrm{j}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ?suspends \(=\mathrm{p}_{\mathrm{j}}\)
    apply -
    apply (rule last-prog-same-append [where sb=?take-sb \({ }_{j}\) ])
    apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \({ }_{j}\) )
    apply simp
    done
from valid-reads [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}-\mathrm{j}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF
«valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 j -bound \({ }^{\prime \prime}\)
\(\mathrm{ts}_{\mathrm{sb}}\)-j this ]
have reads-consis-m-j:
reads-consistent True (acquired True ? take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \({ }_{\mathrm{j}}\)
by (simp add: m suspends \(\mathrm{s}_{\mathrm{j}}\) )
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \({ }^{\prime \prime}\) ts \(\mathrm{s}_{\mathrm{sb}}\)-j]
have nvo-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
with non-volatile-owned-or-read-only-append [of False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
have nvo-take-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\)
by auto
from a-unowned-others [rule-format, OF - neq-i-j] ts \(\mathrm{sb}_{\mathrm{sb}}-\mathrm{j}\) j-bound
have a-not-acq: a \(\notin\) acquired True ? take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\)
by auto
from a-notin-unforwarded-non-volatile-reads-drop [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}\)-j neq-i-j]
have a-notin-unforwarded-reads: a \(\notin\) unforwarded-non-volatile-reads suspendsj \(\}\) by ( simp add: suspends \(\mathrm{j}_{\mathrm{j}}\) )
let \(? \mathrm{ma}=\left(\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right)\)
from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads [where \(\mathrm{W}=\{ \}\) and \(\mathrm{m}^{\prime}=\) ? ma,simplified, OF - subset-refl reads-consis-m-j] a-notin-unforwarded-reads
have reads-consis-ma-j:
reads-consistent True (acquired True ?take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma suspends \({ }_{\mathrm{j}}\)
by auto
from reads-consis-ma-j
have reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma (ys)
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) reads-consistent-append)
from direct-memop-step.RMWWrite [where cond \(=\) cond and \(\vartheta=\vartheta_{\text {sb }}\) and \(m=m\), OF cond']
have (RMW at (D, f) cond ret A L R W\# is \({ }^{\prime}{ }^{\prime}\),
\[
\left.\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow
\]
\[
\left(\mathrm{is}_{\mathrm{sb}^{\prime}},\right.
\]
\(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}\right.\right.\) a) \(\left.\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} a)\right)\right)\right),()\), ?ma, False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\) R,Map.empty, \(\mathcal{S}\)
\(\left.\oplus_{W} R \ominus_{A} L\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i [simplified sb, simplified] this]
have step-a: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}\)
\[
\left(\mathrm { ts } \left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{~m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a})\right)\right)\right),() \text {, False, } \mathcal{O}_{\mathrm{sb}} \cup\right.\right.\right.
\]

A - R,Map.empty)],
\[
\left.? \mathrm{ma}, \mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
\]
```

(is - =}\mp@subsup{\textrm{d}}{\textrm{d}}{(?\textrm{ts}-\textrm{a}, -, ?shared-a)).

```
from \(\mathrm{ts}_{\mathrm{j}}\) neq-i-j j-bound
have ts-a-j: ?ts-a! \(=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps suspends \(\left.{ }_{j}\right),()\),
\(\mathcal{D}_{\mathrm{j}}\), acquired True ? take-sb \(\mathcal{O}_{\mathrm{j}}\), release ? take-sb \(\left.\mathrm{j}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
by auto
from valid-write-sops [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb-j }}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \(@\) @suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ? take-sb \({ }_{j}\). valid-sop sop and valid-sops-drop: \(\forall\) sop \(\in\) write-sops (ys). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have distinct-read-tmps (?take-sb \({ }_{j}\) @suspends \({ }_{j}\) )
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ? take-sb \(_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric] suspendsj \({ }_{j}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have h-consis:
history-consistent \(\vartheta_{\text {sbj }}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) ?take-sb \(_{\mathrm{j}} @\) @uspends \(\left.\left.\mathrm{j}_{\mathrm{j}}\right)\right)\) (?take-sb \(\mathrm{sb}_{\mathrm{j}} @\) suspends \(\left._{\mathrm{j}}\right)\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) ssmmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\)
proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) Q \(\left.? \mathrm{drop}-\mathrm{sb}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{j}_{\mathrm{j}}\) @ suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) ?take-sb \(\mathrm{s}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
```

    by (simp add: split-suspends }\mp@subsup{}{\textrm{j}}{[}\mp@subsup{\mathrm{ symmetric] suspends}}{\textrm{j}}{}\mathrm{ )
    qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent }\mp@subsup{\vartheta}{\textrm{sbj}}{}\mathrm{ (hd-prog }\mp@subsup{\textrm{p}}{\textrm{j}}{}\mathrm{ suspends}\mp@subsup{\textrm{g}}{\textrm{j}}{}\mathrm{ ) suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{
by (simp add: split-suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{[symmetric] suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{}\mathrm{ )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile-Read }\mp@subsup{\operatorname{Rb}}{\mathrm{ ( }}{(ys)}={
by (auto simp add: outstanding-refs-append suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ [symmetric]
split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathrm{ )}
from j-bound' have j-bound-ts-a: j < length ?ts-a by auto
from flush-store-buffer-append [where sb=ys and sb'=Write sb True a's sop'v}\mp@subsup{v}{}{\prime}\mp@subsup{A}{}{\prime}\mp@subsup{L}{}{\prime}\mp@subsup{R}{}{\prime
W'\#zs, simplified,
OF j-bound-ts-a is [ [simplified split-suspends}\mp@subsup{}{\textrm{j}}{\mathrm{ ] cph [simplified suspendsj}
ts-a-j [simplified split-suspends }\mp@subsup{}{j}{}\mathrm{ ] refl lp [simplified split-suspends j] reads-consis-ys
hist-consis' [simplified split-suspendsj] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ ]
no-volatile-Read sb
S =?shared-a]
obtain isj' }\mp@subsup{}{~}{\prime}\mp@subsup{\mathcal{j}}{}{\prime}\mathrm{ where
isj
steps-ys: (?ts-a, ?ma, ?shared-a) =\mp@subsup{d}{}{*}
(?ts-a[j:=(last-prog
(hd-prog pj zs) ys,
isj}\mp@subsup{}{}{\prime}
\varthetasbj |' (dom }\mp@subsup{\vartheta}{\mathrm{ sbj }}{}-\mathrm{ read-tmps zs),
(),
\mathcal{D}}\mp@subsup{\mathcal{j}}{}{\vee}\mathrm{ outstanding-refs is-volatile-Write sb ys }\not={}\mathrm{ , acquired True ys (acquired
True ?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{\textrm{j}}),\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{}{}{\prime})]
flush ys (?ma),
share ys (?shared-a))
(is (-,-,-) =\mp@subsup{|}{\textrm{d}}{*}
by (auto simp add: acquired-append)
from cph
have causal-program-history is sbj ((ys @ [Write sb True a'l sop'v' ' A' L' R' W'])@ zs)
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is isbj zs.
interpret causalj: causal-program-history is }\mp@subsup{\mathrm{ sbj }}{\mathrm{ zS by (rule cph')}}{
from causalj.causal-program-history [of [], simplified, OF refl] is }\mp@subsup{}{j}{\prime
obtain is }\mp@subsup{}{j}{\prime\prime
where is }\mp@subsup{}{\textrm{j}}{\prime}\mathrm{ : is }\mp@subsup{\textrm{i}}{\textrm{j}}{\prime}=\mathrm{ Write True a" sop' (' }\mp@subsup{\textrm{L}}{}{\prime}\mp@subsup{\textrm{R}}{}{\prime}\mp@subsup{\textrm{W}}{}{\prime}\#\mp@subsup{\textrm{is}}{\textrm{j}}{\prime\prime
is}\mp@subsup{}{j}{\prime\prime
by clarsimp

```
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys! \(=\left(p_{s b}\right.\), is \(_{s b}{ }^{\prime}\), \(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m}\right.\right.\right.\) a \(\left.\left.\left.)\right)\right)\right),()\), False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\) R,Map.empty \()\)
by simp
from j-bound-ts-a have j-bound-ys: j < length ?ts-ys
by auto
then have ts-ys-j: ?ts-ys!j = (last-prog (hd-prog \(\left.p_{j} z s\right) y s\), Write True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime}\)
\(\mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps zs \(),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\),
acquired True ys (acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}\) )
by (clarsimp simp add: \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) )
note conflict-computation \(=\) r-rtranclp-rtranclp [OF step-a steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j]
have \(\mathrm{A}^{\prime}\)-unowned:
\(\forall \mathrm{i}<\) length ?ts-ys. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)\)
apply cases
apply (fastforce simp add: Let-def is \(\mathbf{s}_{\mathbf{s b}}\) ) + done
from \(\mathrm{a}^{\prime}\)-in \(\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}\)-unowned [rule-format, of i] neq-i-j i-bound \({ }^{\prime} \mathrm{A}-\mathrm{R}\)
show False
by (auto simp add: Let-def)
next

then
obtain ys zs \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
split-suspends \(\mathrm{s}_{\mathrm{j}}\) : suspends \(\mathrm{s}_{\mathrm{j}}=\) ys \(@\) Ghost \(_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) zs (is suspends \({ }_{\mathrm{j}}=\) ?suspends)
and
\(a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}\)
by blast
from valid-program-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
have causal-program-history \(\mathrm{is}_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}\).
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? \({ }^{\text {take- }} \mathrm{sb}_{\mathrm{j}}\) ] )
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) )
done
from valid-last-prog [ OF j -bound \(\left.{ }^{\prime \prime} \mathrm{ts}_{\mathbf{s b}^{\prime}-\mathrm{j}}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
then
have lp : last-prog \(\mathrm{p}_{\mathrm{j}}\) ? suspends \(=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where sb=? take-sb \({ }_{\mathrm{j}}\) ])
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
from valid-reads [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb-j }}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF
«valid-ownership-and-sharing \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}\) 〉 j-bound"
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) this]
have reads-consis-m-j:
reads-consistent True (acquired True ? take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{j}_{\mathrm{j}}\)
by ( simp add: m suspends \(\mathrm{j}_{\mathrm{j}}\) )
from outstanding-non-volatile-refs-owned-or-read-only [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}\)-j]
have nvo-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
with non-volatile-owned-or-read-only-append [of False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \(\mathrm{S}_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
have nvo-take-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\)
by auto
from a-unowned-others [rule-format, OF - neq-i-j] ts \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) j-bound
have a-not-acq: a \(\notin\) acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\)
by auto
from a-notin-unforwarded-non-volatile-reads-drop [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}\)-j neq-i-j]
have a-notin-unforwarded-reads: a \(\notin\) unforwarded-non-volatile-reads suspendsj \(\}\) by (simp add: suspends \({ }_{\mathrm{j}}\) )
let \(? \mathrm{ma}=\left(\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right)\)
from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads [where \(\mathrm{W}=\{ \}\)
and \(\mathrm{m}^{\prime}=\) ?ma,simplified, OF - subset-refl reads-consis-m-j]
a-notin-unforwarded-reads
have reads-consis-ma-j: reads-consistent True (acquired True ?take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma suspends \({ }_{\mathrm{j}}\) by auto
from reads-consis-ma-j
have reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ) ?ma (ys) by (simp add: split-suspends \({ }_{j}\) reads-consistent-append)
from direct-memop-step.RMWWrite [where cond=cond and \(\vartheta=\vartheta_{\mathrm{sb}}\) and \(\mathrm{m}=\mathrm{m}\), OF cond']
have (RMW a t (D, f) cond ret A L R W\# is \(\mathrm{sb}^{\prime}\) ',
\[
\vartheta_{\mathrm{sb}},(), \underset{\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime},\right.}{\left.\mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \quad \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow}
\]
\(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{ma})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{ma})\right)\right)\right),()\), ?ma, False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\) R,Map.empty, \(\left.\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i [simplified sb, simplified] this]
have step-a: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\)
\[
\left(\mathrm { ts } \left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{~m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a})\right)\right)\right),(), \text { False, } \mathcal{O}_{\mathrm{sb}} \cup\right.\right.\right.
\]

A - R,Map.empty)],
\[
\left.? \mathrm{ma}, \mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
\]
(is \(-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{a},-\), ?shared-a)).
from \(t s_{j}\) neq-i-j j-bound
have ts-a-j: ?ts-a! \(=\left(\right.\) hd-prog \(p_{j}\) suspends \(_{j}\), is \(_{j}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}\) - read-tmps suspends \(\left.{ }_{j}\right),(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
by auto
from valid-write-sops [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{\mathrm{j}} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops (ys). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
```

from read-tmps-distinct [OF j-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$
have distinct-read-tmps (?take-sb ${ }_{j}$ @suspends ${ }_{j}$ )
by (simp add: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
then obtain
read-tmps-take-drop: read-tmps ?take-sb $\mathrm{b}_{\mathrm{j}} \cap$ read-tmps suspends ${ }_{\mathrm{j}}=\{ \}$ and
distinct-read-tmps-drop: distinct-read-tmps suspends ${ }_{j}$
apply (simp only: split-suspends ${ }_{j}$ [symmetric] suspends ${ }_{j}$ )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}-\mathrm{j}}$ ]
have h-consis:
history-consistent $\vartheta_{\mathrm{sbj}}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb ${ }_{\mathrm{j}} @$ suspends $\left._{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends $_{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{S}_{\mathrm{j}}$ )
apply simp
done

```
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \({ }_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\left.{ }_{\mathrm{j}}\right)\)
proof -
```

    from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) @?drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
        by simp
    from last-prog-hd-prog-append' [OF h-consis] this
    have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\)
        by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    moreover
    ```

```

        last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
        apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
        by (rule last-prog-hd-prog-append)
    ultimately show ?thesis
        by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
    qed

```
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}}(\mathrm{ys})=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] split-suspends \({ }_{j}\) )
from j-bound' have j-bound-ts-a: j < length ?ts-a by auto
from flush-store-buffer-append [where \(\mathrm{sb}=\mathrm{ys}\) and \(\mathrm{sb}^{\prime}=\) Ghost \(_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\), simplified,

OF j-bound-ts-a is \(\mathrm{s}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{j}_{\mathrm{j}}\) ] cph [simplified suspends \({ }_{\mathrm{j}}\) ]
ts-a-j [simplified split-suspends \({ }_{\mathrm{j}}\) ] refl lp [simplified split-suspends \(\mathrm{j}_{\mathrm{j}}\) ] reads-consis-ys hist-consis' [simplified split-suspendsj] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?shared-a]
obtain is \({ }^{\prime}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : Ghost \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) instrs zs @ \(\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @ \operatorname{prog}\)-instrs zs and
steps-ys: (?ts-a, ?ma, ?shared-a) \(\Rightarrow{ }_{\mathrm{d}}{ }^{*}\)
(?ts-a [j:=(last-prog
(hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{zs}\) ) ys,
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps zs \()\),
(),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired True ? take-sb \(\left.\left.\left.\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]\), flush ys (?ma),
share ys (?shared-a))
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}\) (?ts-ys,?m-ys,?shared-ys))
by (auto simp add: acquired-append)
from cph
have causal-program-history is sbj \(\left(\left(y s\right.\right.\) @ \(\left[\right.\) Ghost \(_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ' \(]\) ) @ zs) by simp
from causal-program-history-suffix [OF this]
have \(\mathrm{cph}^{\prime}\) : causal-program-history is \(\mathrm{sbj}_{\mathrm{sbj}} \mathrm{zs}\).
interpret causal \({ }_{j}\) : causal-program-history is sbj \(^{\text {zs }}\) by (rule cph')
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }_{j}{ }^{\prime}\)
obtain is \({ }^{\prime}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Ghost \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ is \(\mathrm{sbj}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) @ prog-instrs zs
by clarsimp
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys! \(\mathrm{i}=\left(\mathrm{p}_{\mathbf{s b}}, \mathrm{is}_{\text {sb }}{ }^{\prime}\right.\),
\(\vartheta_{\text {sb }}\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}\right.\right.\) a) \(\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m}\right.\right.\) a) \(\left.\left.)\right)\right),()\), False, \(\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}\), Map.empty \()\) by simp
from j-bound-ts-a have j-bound-ys: \(\mathrm{j}<\) length ?ts-ys
by auto
then have ts-ys-j: ?ts-ys!j \(=\left(\operatorname{last}-\mathrm{prog}\left(h d-p r o g \mathrm{p}_{\mathrm{j}} \mathrm{zs}\right)\right.\) ys, Ghost \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\), \(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\mathrm{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps zs \(),()\), \(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired True ?take-sb \(\mathrm{D}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}\) )
by (clarsimp simp add: \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) )
note conflict-computation \(=\) r-rtranclp-rtranclp [OF step-a steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j]
have A '-unowned:
\(\forall \mathrm{i}<\) length ?ts-ys. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)\)
apply cases
apply (fastforce simp add: Let-def is stb \(^{\text {}}\) )+
done
from \(\mathrm{a}^{\prime}\)-in \(\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}\)-unowned [rule-format, of i] neq-i-j i-bound \({ }^{\prime} \mathrm{A}-\mathrm{R}\)
show False
by (auto simp add: Let-def)
qed
qed
qed
\}
thus ?thesis
by (auto simp add: Let-def)
qed
```

    \{
    fix j
fix $\mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length $^{\mathrm{ts}} \mathrm{ss}_{\mathrm{sb}}$
assume $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ : $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
have $\mathrm{A} \cap$ read-only-reads (acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}}$ )
$\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{j}}\right)=\{ \}$
proof -
\{
let ?take-sb ${ }_{j}=\left(\right.$ takeWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)$
let ? drop-sb ${ }_{j}=\left(\right.$ dropWhile $\left(\right.$ Not $\circ$ is-volatile-Write $\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)$
assume conflict: $\mathrm{A} \cap$ read-only-reads (acquired True ?take-sb $_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ) ?drop-sb $_{\mathrm{j}} \neq\{ \}$
have False
proof -
from conflict obtain $\mathrm{a}^{\prime}$ where
$a^{\prime}-\mathrm{in}: \mathrm{a}^{\prime} \in \mathrm{A}$ and
$\mathrm{a}^{\prime}$-in-j $\mathrm{j}: \mathrm{a}^{\prime} \in$ read-only-reads (acquired True ?take-sb $\mathrm{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by auto
from ts-sim [rule-format, OF j -bound] $\mathrm{ts}_{\mathrm{sb}} \mathrm{j} \mathrm{j} \mathrm{j}$-bound
obtain $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}} \mathrm{is}_{\mathrm{sbj}} \mathcal{D}_{\mathrm{sbj}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{sbj}}$ is $\mathrm{s}_{\mathrm{j}}$ where
$\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{i}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$ and
suspendsj: suspendsj $=$ ?drop-sb ${ }_{j}$ and
$\mathrm{is}_{\mathrm{j}}$ : instrs suspends $\mathrm{j}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}} @$ prog-instrs suspends $\mathrm{j}_{\mathrm{j}}$ and
$\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)$ and
$\mathrm{ts}_{\mathrm{j}}:$ ts $!\mathrm{j}=\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}$ suspends $\mathrm{s}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}$,
$\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.$ dom $\vartheta_{\text {sbj }}-$ read-tmps suspends $\left.s_{j}\right),(), \mathcal{D}_{\mathrm{j}}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$, release
?take-sb $\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)$
apply (cases $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}$ )
apply (clarsimp simp add: Let-def)
done
from split-in-read-only-reads [OF a'-in-j [simplified suspends ${ }_{j}$ [symmetric]]]
obtain $t^{\prime} v^{\prime}$ ys zs where
split-suspends $\mathrm{s}_{\mathrm{j}}$ : suspends $\mathrm{s}_{\mathrm{j}}=\mathrm{ys} @ \operatorname{Read}_{\mathrm{sb}}$ False $\mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v}^{\prime} \# \mathrm{zs}$ (is suspends $\mathrm{j}_{\mathrm{j}}=$ ? suspends)
and
$\mathrm{a}^{\prime}$-unacq: $\mathrm{a}^{\prime} \notin$ acquired True ys (acquired True ?take-sb ${ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ )
by blast
from valid-program-history [ OF j -bound $\mathrm{ts}_{\mathrm{sb}}$-j]
have causal-program-history is sbj $_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$.
then have cph: causal-program-history is $_{\text {sbj }}$ ?suspends
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb ${ }_{j}$ ] )
apply (simp only: split-suspends ${ }_{\mathrm{j}}$ [symmetric] suspends ${ }_{\mathrm{j}}$ )
apply (simp add: split-suspends ${ }_{\mathrm{j}}$ )

```
done
from valid-last-prog \(\left[\mathrm{OF} \mathrm{j}\right.\)-bound \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\). then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ?suspends \(=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? \({ }^{\text {take-sb }} \mathrm{sb}_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric] \(^{\text {suspends }}{ }_{\mathrm{j}}\) )
apply simp
done
from valid-reads [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing
j-bound
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{j}_{\mathrm{j}}\) by (simp add: m suspends \(_{\mathrm{j}}\) )
from outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \(\mathrm{ts}_{\text {sb-j }}\) ]
have nvo-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
with non-volatile-owned-or-read-only-append [of False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\) ?drop-sb \({ }_{\mathrm{j}}\) ]
have nvo-take-j: non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}\) ?take-sb \({ }_{\mathrm{j}}\)
by auto
from a-unowned-others [rule-format, OF - neq-i-j] ts \(\mathrm{s}_{\mathrm{sb}}-\mathrm{j} j\)-bound
have a-not-acq: a \(\notin\) acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\)
by auto
from a-notin-unforwarded-non-volatile-reads-drop [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}\)-j neq-i-j]
have a-notin-unforwarded-reads: a \(\notin\) unforwarded-non-volatile-reads suspends \({ }_{j}\{ \}\) by (simp add: suspends \({ }_{\mathrm{j}}\) )
let \(? \mathrm{ma}=\left(\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right)\)
from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads [where \(\mathrm{W}=\{ \}\)
and \(\mathrm{m}^{\prime}=\) ? ma,simplified, OF - subset-refl reads-consis-m-j]
a-notin-unforwarded-reads
have reads-consis-ma-j:
reads-consistent True (acquired True ? \({ }^{\text {take-sb }} \mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?ma suspends \({ }_{\mathrm{j}}\) by auto
from reads-consis-ma-j
have reads-consis-ys: reads-consistent True (acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) ?ma (ys) by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) reads-consistent-append)
from direct-memop-step.RMWWrite \(\left[w h e r e\right.\) cond \(=\) cond and \(\vartheta=\vartheta_{\text {sb }}\) and \(m=m\), OF cond \({ }^{\prime}\) ]
have (RMW at (D, f) cond ret A L R W\# is \(\left.{ }_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\)
\[
\left(\mathrm{is}_{\text {sb }}^{\prime}, \vartheta_{\text {sb }}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{~m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a})\right)\right)\right),(), ? \mathrm{ma}, \text { False, } \mathcal{O}_{\text {sb }} \cup \mathrm{A}-\right.
\] R,Map.empty, \(\left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have step-a: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\)
\(\left(\operatorname{tss}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}^{\prime}}, \vartheta_{\mathbf{s b}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \operatorname{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right),()\right.\right.\right.\), False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}\)
- R,Map.empty)],
? \(\left.\mathrm{ma}, \mathcal{S} \oplus \mathrm{W} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
(is \(-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{a},-\), ?shared-a)).
from \(\mathrm{ts}_{\mathrm{j}}\) neq-i-j j-bound
have ts-a-j: ?ts-a! \(=\left(h d-p r o g ~ p_{j}\right.\) suspends \(_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps suspends \(\left.{ }_{j}\right),(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\), release \(\left.{ }^{\text {take-sb }}{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
by auto
from valid-write-sops [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops (ys). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}\) )
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{\mathrm{j}} \cap\) read-tmps suspends \({ }_{\mathrm{j}}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h -consis:

apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{s}_{\mathrm{j}}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{j}_{\mathrm{j}}\) )
proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) @ \(\mathrm{dr}_{\mathrm{drop}}\)-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) ) take-sb \(_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{j}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(_{\mathrm{j}} @\) suspends \(\left._{\mathrm{j}}\right)\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\) last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads [OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}}(\mathrm{ys})=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \(\mathrm{s}_{\mathrm{j}}\) )
from j-bound leq have j-bound-ts-a: \(\mathrm{j}<\) length ?ts-a by auto
from flush-store-buffer-append [where \(\mathrm{sb}=\mathrm{ys}\) and \(\mathrm{sb}^{\prime}=\operatorname{Read}_{\mathrm{sb}}\) False \(\mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v}^{\prime} \# \mathrm{zs}\), simplified,
OF j-bound-ts-a is \(\mathrm{s}_{\mathrm{j}}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] cph [simplified suspends \({ }_{\mathrm{j}}\) ]
ts-a-j [simplified split-suspends \({ }_{\mathrm{j}}\) ] refl lp [simplified split-suspends \({ }_{\mathrm{j}}\) ] reads-consis-ys hist-consis \({ }^{\prime}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{s b}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?shared-a]
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : Read False \(\mathrm{a}^{\prime} \mathrm{t}^{\prime} \#\) instrs zs \(@ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs zs and
steps-ys: (?ts-a, ?ma, ?shared-a) \(\Rightarrow_{\mathrm{d}}{ }^{*}\)
(?ts-a \({ }^{\mathrm{j}} \mathrm{j}:=\) (last-prog
(hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{zs}\) ) ys,
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) insert t' \((\) read-tmps zs \(\left.)\right)\),
(), \(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write sb \(^{\text {ys }} \neq\{ \}\), acquired True ys (acquired

True ? take-sb \(\left.\left.\left.\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]\),
flush ys (?ma),
share ys (?shared-a))
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}\), ?shared-ys) \()\)
by (auto simp add: acquired-append)
from cph
have causal-program-history is sbj \(\left(\left(\mathrm{ys} @\left[\operatorname{Read}_{\text {sb }}\right.\right.\right.\) False \(\left.\left.\left.\mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v} \boldsymbol{\jmath}\right]\right) @ \mathrm{zs}\right)\)
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is sbj zs .
interpret causal \({ }_{j}\) : causal-program-history \(\mathrm{is}_{\text {sbj }} \mathrm{zs}\) by (rule cph')
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }_{\mathrm{j}}{ }^{\prime}\) obtain is \({ }_{\mathrm{j}}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}: \mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Read False \(\mathrm{a}^{\prime} \mathrm{t}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and is \(_{j}{ }^{\prime \prime}\) : instrs zs @ is \(s_{\text {bj }}=\) is \(^{\prime \prime}\) @ prog-instrs zs by clarsimp
from i-bound' have i-bound-ys: i < length ?ts-ys by auto
from i-bound' neq-i-j
have ts-ys-i: ?ts-ys!i \(=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\text {sb }}{ }^{\prime}\right.\), \(\vartheta_{\mathbf{s b}}\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}\right.\right.\) a) \(\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m}\right.\right.\) a) \(\left.\left.)\right)\right),()\), False, \(\mathcal{O}_{\mathbf{s b}} \cup \mathrm{A}-\mathrm{R}\), Map.empty \()\)
by simp
from j-bound-ts-a have j-bound-ys: j < length ?ts-ys
by auto
then have ts-ys-j: ?ts-ys!j = (last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{zs}\right)\) ys, Read False a't \(\mathrm{t}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}, \vartheta_{\mathrm{sbj}}\) \(\|^{6}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\operatorname{insert} \mathrm{t}^{\prime}(\) read-tmps zs \(\left.)\right),(), \mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\) \{\},
acquired True ys (acquired True ?take-sb \(\boldsymbol{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )
by (clarsimp simp add: \(\mathrm{is}_{\mathrm{j}}\) )
note conflict-computation \(=\) r-rtranclp-rtranclp [OF step-a steps-ys]
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j]
have \(\mathrm{a}^{\prime} \in\) acquired True ys (acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}}\) ) \(\vee\)
\(\mathrm{a}^{\prime} \in\) read-only (share ys \(\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) )
apply cases
apply (auto simp add: Let-def is s \(_{\text {sb }}\) )
done
with a'-unacq
have \(\mathrm{a}^{\prime}\)-ro: \(\mathrm{a}^{\prime} \in\) read-only (share ys \(\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) )
by auto
from \(a^{\prime}\)-in
have a'-not-ro: a' \(\notin\) read-only \(\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
by (auto simp add: in-read-only-convs)
have \(\mathrm{a}^{\prime} \in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
proof -
\{
assume a-notin: \(\mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
from weak-sharing-consis [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
```

    have weak-sharing-consistent }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ .
    with weak-sharing-consistent-append [of }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ ?take-sb }\mp@subsup{\mp@code{j}}{\textrm{j}}{\mathrm{ ?drop-sb }}\textrm{j}\mathrm{ ]
    have weak-sharing-consistent (acquired True ?take-sb }\mp@subsup{}{j}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ ) suspendsj
        by (auto simp add: suspendsj
    with split-suspends}\mp@subsup{}{j}{
    have weak-consis: weak-sharing-consistent (acquired True ?take-sb}\mp@subsup{b}{j}{}\mp@subsup{\mathcal{O}}{j}{}\mathrm{ ) ys
        by (simp add: weak-sharing-consistent-append)
    from all-acquired-append [of ?take-sb [drop-sbj]
    have all-acquired ys }\subseteq\mathrm{ all-acquired sb}\mp@subsup{b}{j}{
        apply (clarsimp)
        apply (clarsimp simp add: suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{[symmetric] split-suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{}\mathrm{ all-acquired-append)
        done
            with a-notin acquired-takeWhile-non-volatile-Write sb [of sb }\mp@subsup{\textrm{j}}{\mathbf{j}}{\mathbf{j}}\mp@subsup{]}{}{\prime
                        all-acquired-append [of ?take-sb b}\mathrm{ ?drop-sb;}
    have a' }\not=\mathrm{ acquired True (takeWhile (Not o is-volatile-Write sb) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ ) }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired
    ys
by auto
from read-only-share-unowned [OF weak-consis this a'-ro]
have a' }\in\mathrm{ read-only ( }\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
with a'-not-ro have False
by auto
with a-notin read-only-share-unowned [OF weak-consis - a'-ro]
all-acquired-takeWhile [of (Not o is-volatile-Write sb ) sb j}\mathrm{ ]
have a' }\in\mathrm{ read-only (S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
by (auto simp add: acquired-takeWhile-non-volatile-Write sb)
with a'-not-ro have False
by auto
}
thus ?thesis by blast
qed
moreover
from A-unacquired-by-others [rule-format, OF - neq-i-j] ts ts-j j-bound
have A \cap all-acquired sbj}={
by (auto simp add: Let-def)
moreover
from A-unowned-by-others [rule-format, OF - neq-i-j] ts sb-j j-bound
have }\textrm{A}\cap\mp@subsup{\mathcal{O}}{j}{}={
by (auto simp add: Let-def dest: all-shared-acquired-in)
moreover note a'-in
ultimately
show False
by auto
qed
}
thus ?thesis
by (auto simp add: Let-def)

```
```

qed
\} note A-no-read-only-reads $=$ this
have valid-own': valid-ownership $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof
fix j is $\mathrm{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{p}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
assume $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show non-volatile-owned-or-read-only False $\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
proof (cases $\mathrm{j}=\mathrm{i}$ )
case True
have non-volatile-owned-or-read-only False
$\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\left(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right)[]$
by simp
then show ?thesis
using True i-bound $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}-\mathrm{j}$
by (auto $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\text {sb }}$ ' sb sb')
next
case False
from j -bound have j -bound ${ }^{\prime}$ : $\mathrm{j}<$ length $\mathrm{ts}_{\text {sb }}$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
with $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{j}$ False i-bound
have $\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by (auto simp add: $\mathrm{ts}_{\text {sb }}$ )
note nvo $=$ outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' ts $_{\text {sb-j }}$ ]
from read-only-unowned [OF i-bound $\mathrm{ts}_{\text {sb }}$ - $]$ R-owned
have $\mathrm{R} \cap$ read-only $\mathcal{S}_{\mathrm{sb}}=\{ \}$
by auto
with A-no-read-only-reads [OF j-bound' ts $_{\text {sb-j }}$ False [symmetric]] L-subset
have $\forall \mathrm{a} \in$ read-only-reads
(acquired True (takeWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}}$ )
(dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }_{\mathrm{j}}$ ).
$\mathrm{a} \in$ read-only $\mathcal{S}_{\mathrm{sb}} \longrightarrow \mathrm{a} \in$ read-only $\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
by (auto simp add: in-read-only-convs)
from non-volatile-owned-or-read-only-read-only-reads-eq' [OF nvo this]
have non-volatile-owned-or-read-only False ( $\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ) $\mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
thus ?thesis by (simp add: $\mathcal{S}_{\mathrm{sb}}$ )
qed
qed
next
show outstanding-volatile-writes-unowned-by-others $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
proof (unfold-locales)
fix $\mathrm{i}_{1} \mathrm{j}_{\mathrm{p}}^{1}$ is $\mathrm{S}_{1} \mathcal{O}_{1} \mathcal{R}_{1} \mathcal{D}_{1} \mathrm{xs}_{1} \mathrm{sb}_{1} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume $i_{1}$-bound: $i_{1}<$ length ts $_{\text {sb }}{ }^{\prime}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\text {sb }}{ }^{\prime}$

```
```

assume $\mathrm{i}_{1}-\mathrm{j}: \mathrm{i}_{1} \neq \mathrm{j}$
assume ts-i $\mathrm{i}_{1}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{l}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
assume ts-j: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile- Write $_{\text {sb }} \mathrm{sb}_{1}=\{ \}$
proof (cases $\mathrm{i}_{1}=\mathrm{i}$ )
case True
with ts- $\mathrm{i}_{1}$ i-bound show ?thesis
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ sb)
next
case False
note $i_{1}-\mathrm{i}=$ this
from $i_{1}$-bound have $i_{1}$-bound ${ }^{\prime}: i_{1}<$ length $\mathrm{ts}_{\text {sb }}$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ sb)
hence $\mathrm{i}_{1}$-bound ${ }^{\prime \prime}: \mathrm{i}_{1}<$ length (map owned $\mathrm{ts}_{\text {sb }}$ )
by auto
from ts-i $\mathrm{i}_{1}$ False have ts- $\mathrm{i}_{1}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{li}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ sb)
show ?thesis
proof (cases $\mathrm{j}=\mathrm{i}$ )
case True
from i-bound ts-j ts $\mathrm{sb}^{\prime}$ True have $\mathrm{sb}_{\mathrm{j}}$ : $\mathrm{sb}_{\mathrm{j}}=[]$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
from A-unused-by-others [rule-format, OF - False [symmetric]] ts-i $\mathrm{i}_{1} \mathrm{i}_{1}$-bound ${ }^{\prime \prime}$
False $\mathrm{i}_{1}$-bound ${ }^{\prime}$
have $\mathrm{A} \cap\left(\mathcal{O}_{1} \cup\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\text {sb }} \mathrm{sb}_{1}\right)=\{ \}$
by (auto $\operatorname{simp}$ add: Let-def $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ sb' ${ }^{\prime}$ owned-def)
moreover
from outstanding-volatile-writes-unowned-by-others
[OF $\mathrm{i}_{1}$-bound ${ }^{\prime} \mathrm{i}$-bound $\mathrm{i}_{1}-\mathrm{i}$ ts- $\mathrm{i}_{1}{ }^{\prime}$ ts $_{\text {sb }}$ - ]
have $\mathcal{O}_{\text {sb }} \cap$ outstanding-refs is-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{1}=\{ \}$ by (simp add: sb)
ultimately show ?thesis using ts-j True
by (auto simp add: i-bound $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\text {sb }^{\prime}}{ }^{\prime} \mathrm{sb}_{\mathrm{j}}$ )
next
case False
from $j$-bound have j -bound': $\mathrm{j}<$ length $\mathrm{ts}_{\text {sb }}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
from ts-j False have ts-j${ }^{\prime}$ : $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
from outstanding-volatile-writes-unowned-by-others
[OF $\mathrm{i}_{1}$-bound ${ }^{\prime} \mathrm{j}$-bound ${ }^{\prime} \mathrm{i}_{1}-\mathrm{j}$ ts- $\mathrm{i}_{1}{ }^{\prime}{ }^{\prime}$ ts-j${ }^{\prime}$ ]
show $\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right) \cap$ outstanding-refs is-volatile-Write ${ }_{\mathrm{sb}} \mathrm{sb}_{1}=\{ \}$.
qed
qed
qed
next
show read-only-reads-unowned $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof
fix nm

```
fix \(\mathrm{p}_{\mathrm{n}} \mathrm{is}_{\mathrm{n}} \mathcal{O}_{\mathrm{n}} \mathcal{R}_{\mathrm{n}} \mathcal{D}_{\mathrm{n}} \vartheta_{\mathrm{n}} \mathrm{sb}_{\mathrm{n}} \mathrm{p}_{\mathrm{m}}\) is \(\mathrm{in}_{\mathrm{m}} \mathcal{O}_{\mathrm{m}} \mathcal{R}_{\mathrm{m}} \mathcal{D}_{\mathrm{m}} \vartheta_{\mathrm{m}} \mathrm{sb}_{\mathrm{m}}\)
assume n-bound: \(\mathrm{n}<\) length \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\)
and m -bound: \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
and neq- \(\mathrm{n}-\mathrm{m}\) : \(\mathrm{n} \neq \mathrm{m}\)
and nth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
and mth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\)
from n-bound have \(n\)-bound': \(n<\) length \(^{\text {ts }}{ }_{\text {sb }}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from m -bound have m -bound': \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}\) )
show \(\left(\mathcal{O}_{\mathrm{m}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{m}}\right) \cap\)
read-only-reads (acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{n}}\) ) \(\mathcal{O}_{\mathrm{n}}\) )
\(\left(\right.\) dropWhile \(^{(N o t} \circ\) is-volatile-Write \(\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{n}}\right)=\)
\{\}
proof (cases m=i)
case True
with neq-n-m have neq-n-i: \(\mathrm{n} \neq \mathrm{i}\) by auto
with n -bound nth i-bound have \(\mathrm{nth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\) by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth \({ }^{\prime} \operatorname{ts}_{s_{s b}-i}\) ]
moreover
note A-no-read-only-reads [OF n-bound' nth']
ultimately
show ?thesis
using True \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) neq-n-i nth mth n-bound' m -bound \({ }^{\prime}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\mathrm{sb}}\) )
next
case False
note neq-m-i \(=\) this
with m-bound mth i-bound have mth': \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\) by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
show ?thesis
proof (cases \(\mathrm{n}=\mathrm{i}\) )
case True
with ts \(_{\text {sb }}\)-i nth mth neq-m-i n -bound \({ }^{\prime}\)
show ?thesis
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
next
case False
with n-bound nth i-bound have \(n t h ': \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from read-only-reads-unowned [OF n-bound' m-bound' neq-n-m nth' mth \(]\) False
neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
next
show ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
```

proof (unfold-locales)
fix $\mathrm{i}_{1} \mathrm{j}_{\mathrm{p}}^{1}$ is $\mathrm{O}_{1} \mathcal{O}_{1} \mathcal{R}_{1} \mathcal{D}_{1} \mathrm{xs}_{1} \mathrm{sb}_{1} \mathrm{p}_{\mathrm{j}}$ is $\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume $i_{1}$-bound: $i_{1}<$ length $\mathrm{ts}_{\text {sb }}{ }^{\prime}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
assume $\mathrm{i}_{1}-\mathrm{j}$ : $\mathrm{i}_{1} \neq \mathrm{j}$
assume ts-i $\mathrm{i}_{1}$ : $\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{l}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
assume ts-j: ts $\mathrm{sb}_{\mathrm{b}}{ }^{\prime} \mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{1} \cup\right.$ all-acquired $\left.\mathrm{sb}_{1}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
proof (cases $\mathrm{i}_{1}=\mathrm{i}$ )
case True
with $\mathrm{i}_{1}-\mathrm{j}$ have $\mathrm{i}-\mathrm{j}: \mathrm{i} \neq \mathrm{j}$
by simp
from i-bound $\mathrm{ts}-\mathrm{i}_{1} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ True have $\mathrm{sb}_{1}: \mathrm{sb}_{1}=[]$
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}$ ')
from j -bound have j -bound ${ }^{\prime}$ : $\mathrm{j}<$ length $\mathrm{ts}_{\text {sb }}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
hence j -bound ${ }^{\prime \prime}: \mathrm{j}$ < length (map owned $\mathrm{ts}_{\mathrm{sb}}$ )
by simp
from ts-j i-j have ts-j${ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
from A-unused-by-others [rule-format, OF - i-j] ts-j i-j j-bound ${ }^{\prime}$
have $\mathrm{A} \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
by (auto simp add: Let-def $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ owned-def)
moreover
from A-unacquired-by-others [rule-format, OF - i-j] ts-j i-j j-bound'
have $\mathrm{A} \cap$ all-acquired $\mathrm{sb}_{\mathrm{j}}=\{ \}$
by (auto simp add: Let-def $\mathrm{ts}_{\mathrm{sb}}$ )
moreover
from ownership-distinct [OF i-bound j-bound ${ }^{\prime} \mathrm{i}-\mathrm{j} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ts-j${ }^{\text {j }}$ ]
have $\mathcal{O}_{\mathrm{sb}} \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$ by (simp add: sb)
ultimately show ?thesis using ts- $\mathrm{i}_{1}$ True
by (auto simp add: i-bound $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}_{1}$ )
next
case False
note $\mathrm{i}_{1}-\mathrm{i}=$ this
from $i_{1}$-bound have $i_{1}$-bound': $i_{1}<$ length $\mathrm{ts}_{\text {sb }}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
hence $\mathrm{i}_{1}$-bound ${ }^{\prime \prime}: \mathrm{i}_{1}<$ length (map owned $\mathrm{ts}_{\text {sb }}$ )
by simp
from ts-i $\mathrm{i}_{1}$ False have ts-i ${ }_{1}$ : $\mathrm{ts}_{\mathrm{sb}}: \mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
by ( simp add: $\mathrm{ts}_{\text {sb }}$ )
show ?thesis
proof (cases $\mathrm{j}=\mathrm{i}$ )
case True
from A-unused-by-others [rule-format, OF - False [symmetric]] ts-i $\mathrm{i}_{1}$
False $\mathrm{i}_{1}$-bound ${ }^{\prime}$
have $\mathrm{A} \cap\left(\mathcal{O}_{1} \cup\right.$ outstanding-refs is-volatile-Write $\left.{ }_{\text {sb }} \mathrm{sb}_{1}\right)=\{ \}$

```

moreover
from A-unacquired-by-others [rule-format, OF - False [symmetric]] ts-i \(\mathrm{i}_{1}\) False \(i_{1}\)-bound \({ }^{\prime}\)
have \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{1}=\{ \}\)
by (auto simp add: Let-def \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) owned-def)
moreover
from ownership-distinct [OF \(i_{1}\)-bound \({ }^{\prime}\) i-bound \(i_{1}-\mathrm{i}\) ts- \(\mathrm{i}_{1}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}} \mathrm{i}\) ]
have \(\left(\mathcal{O}_{1} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{1}\right) \cap \mathcal{O}_{\text {sb }}=\{ \}\) by (simp add: sb )
ultimately show ?thesis
using ts-j True
by (auto simp add: i-bound \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
next
case False
from j -bound have j -bound \({ }^{\prime}: \mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ts-j False have \(\mathrm{ts}^{-j}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ownership-distinct [OF \(\mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{j}\)-bound \({ }^{\prime} \mathrm{i}_{1}-\mathrm{j}\) ts- \(\mathrm{i}_{1}{ }^{\prime}\) ts-j\({ }^{\prime}\) ]
show \(\left(\mathcal{O}_{1} \cup\right.\) all-acquired \(\left.\operatorname{sb}_{1}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\).
qed
qed
qed
qed
have valid-hist \({ }^{\prime}\) : valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have history-consistent \(\left(\vartheta_{s b}\left(\mathrm{t} \mapsto \mathrm{ret}\left(\mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\right)\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{sb}}[]\right)[]\right.\) by simp
from valid-history-nth-update [OF i-bound this]
show ?thesis by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}\) )
qed
from valid-reads [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}\).
have valid-reads \({ }^{\prime}\) : valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
fix \(\mathrm{j}_{\mathrm{j}}\) is \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}}\) acq \(_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume ts-j: \(\mathrm{ts}_{\mathrm{sb}!}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show reads-consistent False \(\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
from reads-consis ts-j j-bound sb show ?thesis by (clarsimp simp add: True \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}\) Write \(_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
next
case False
from j-bound have j-bound': \(j<\) length ts \(_{\text {sb }}\)
```

    by (simp add: ts sb }\mp@subsup{}{\mathrm{ ' }}{
    moreover from ts-j False have ts-j': ts sm ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{i}}{\textrm{j}}{\textrm{j}},\mp@subsup{\vartheta}{j}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
    using j-bound by (simp add: ts sb
    ultimately have consis-m: reads-consistent False }\mp@subsup{\mathcal{O}}{j}{}\mp@subsup{m}{sb}{}\mp@subsup{s}{j}{
    by (rule valid-reads)
    let ?m
    from a-unowned-others [rule-format, OF - False] j-bound' ts-j'
        obtain a-acq: a & acquired True (takeWhile (Not o is-volatile-Write sb) sb j) \mathcal{O}}\mp@subsup{\mathcal{j}}{\textrm{j}}{}\mathrm{ and
        a-unsh: a }\not\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ ) sb }\mp@subsup{}{j}{}\mathrm{ )
                by auto
        with a-notin-unforwarded-non-volatile-reads-drop [OF j-bound' ts-j' False]
    have }\forall\textrm{a}\in\mathrm{ acquired True (takeWhile (Not o is-volatile-Write sb) sb j) ()
                all-shared (takeWhile (Not o is-volatile-Write sb
                unforwarded-non-volatile-reads (dropWhile (Not o is-volatile-Write }\mp@subsup{}{\mathrm{ sb}}{}\mathrm{ ) sb }\mp@subsup{\textrm{g}}{\textrm{j}}{}\mathrm{ ) {}.
    ?m'a}=\mp@subsup{m}{\mathrm{ sb }}{}\mathrm{ a
    by auto
    from reads-consistent-mem-eq-on-unforwarded-non-volatile-reads-drop
    [where W={},simplified, OF this - consis-m]
    acquired-reads-all-acquired' [of (takeWhile (Not o is-volatile-Writesb
    have reads-consistent False 飨 ( }\mp@subsup{\textrm{m}}{\textrm{sb}}{}(\textrm{a}:=\textrm{f}(\mp@subsup{\vartheta}{\textrm{sb}}{}(\textrm{t}\mapsto\mp@subsup{\textrm{m}}{\textrm{sb}}{}\textrm{a}))))\textrm{sb
    by (auto simp del: fun-upd-apply)
    thus?thesis
    by (simp add: msb
    qed
qed
have valid-sharing': valid-sharing (\mathcal{Sb}
proof (intro-locales)
show outstanding-non-volatile-writes-unshared (\mathcal{S}
proof (unfold-locales)

```

```

    assume j-bound: j < length ts sbb
    assume jth: ts scb
    show non-volatile-writes-unshared ( (\mathcal{Sb}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}) \mp@subsup{\textrm{sb}}{\textrm{j}}{
    proof (cases i=j)
        case True
        with i-bound jth show ?thesis
            by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\textrm{sb}
    next
        case False
        from j-bound have j-bound': j < length ts sb
            by (auto simp add: ts sb}\mp@subsup{}{\mathrm{ ')}}{
        from jth False have jth': ts stb ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,}\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{x}
            by (auto simp add: ts sb )
        from outstanding-non-volatile-writes-unshared [OF j-bound' jth]
        have unshared: non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{j}{}\mathrm{ .
    ```

```

sbj
proof -
{

```
fix a
assume a-in: a \(\in \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\)
hence \(a-R: a \in R\)
by clarsimp
assume \(\mathrm{a}-\mathrm{in}-\mathrm{j}: \mathrm{a} \in\) outstanding-refs is-non-volatile-Write \({ }_{\mathbf{s b}} \mathrm{sb}_{\mathrm{j}}\)
have False
proof -
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF
outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth ']]
a-in-j
have \(\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto
moreover
with ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) jth \({ }^{\dagger}\) ] a-R R-owned
show False
by blast
qed
\}
thus ?thesis by blast
qed
from non-volatile-writes-unshared-no-outstanding-non-volatile-Write \({ }_{\text {sb }}\)
[OF unshared this]
show ?thesis .
qed
qed
next
show sharing-consis \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
fix \(\mathrm{jp}_{\mathrm{j}}\) is \(\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length \(^{\mathrm{ts}} \mathrm{sb}^{\text {' }}\)
assume jth: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show sharing-consistent \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
with i-bound jth show? ?thesis
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}\) )
next
case False
from j -bound have j -bound \({ }^{\prime}\) : \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from jth False have \(\mathrm{jth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from sharing-consis [OF j-bound' \({ }^{\text {jth }}\) ']
have consis: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
\[
\text { have acq-cond: all-acquired } \operatorname{sb}_{\mathrm{j}} \cap \operatorname{dom} \mathcal{S}_{\mathrm{sb}}-\operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)=\{ \}
\] proof -
\{
fix a
assume a-acq: \(\mathrm{a} \in\) all-acquired \(\mathrm{sb}_{j}\)
assume \(\mathrm{a} \in \operatorname{dom} \mathcal{S}_{\text {sb }}\)
assume a-L: \(\mathrm{a} \in \mathrm{L}\)
have False
proof -
from A-unacquired-by-others [rule-format, of j,OF - False] j-bound \({ }^{\prime}{ }^{j t h}{ }^{\prime}\)
have \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}}=\{ \}\)
by auto
with a-acq a-L L-subset
show False
by blast
qed
\}
thus ?thesis
by auto
qed
have uns-cond: all-unshared \(\mathrm{sb}_{\mathrm{j}} \cap \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}=\{ \}\)
proof -
\{
fix a
assume a-uns: \(\mathrm{a} \in\) all-unshared \(\mathrm{sb}_{\mathrm{j}}\)
assume a \(\notin \mathrm{L}\)
assume \(a-R: a \in R\)
have False
proof -
from unshared-acquired-or-owned [OF consis] a-uns
have \(\mathrm{a} \in\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\) by auto
with ownership-distinct [OF i-bound j-bound' False \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}} \mathrm{jth}\) ] R-owned a-R
show False
by blast
qed
\}
thus ?thesis
by auto
qed
from sharing-consistent-preservation [OF consis acq-cond uns-cond] show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
qed
next
show unowned-shared \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
show \(-\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})\right.\) 'set \(\left.\mathrm{ts}_{\text {sb }}\right) \subseteq \operatorname{dom}\left(\mathcal{S}_{\text {sb }} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
proof -
have s: \(\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})\right.\) ' set \(\mathrm{ts}_{\mathrm{sb}}\) ') =
\(\cup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})\right.\) ' set \(\left.\mathrm{ts}_{\text {sb }}\right) \cup \mathrm{A}-\mathrm{R}\)
```

        apply (unfold ts sb
        apply (rule acquire-release-ownership-nth-update [OF R-owned i-bound ts sb-i]
        apply fact
        done
    note unowned-shared L-subset A-R
    then
    show ?thesis
        apply (simp only: s)
        apply auto
        done
    qed
    qed
next
show read-only-unowned ( }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof
fix j pj isj }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{\mp@code{acq}
assume j-bound: j < length ts ssb
assume jth: ts scb
show }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only ( ( }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})={
proof (cases i=j)
case True
from read-only-unowned [OF i-bound ts tsbi] R-owned A-R
have ( (\mathcal{Osb}}\cup\cup\textrm{A}-\textrm{R})\cap\mathrm{ read-only (S S
by (auto simp add: in-read-only-convs )
with jth ts sb-i i-bound True
show ?thesis
by (auto simp add: }\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mathrm{ )
next
case False
from j-bound have j-bound': j < length ts sb
by (auto simp add: ts sb ')
with False jth have jth': ts stb ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
by (auto simp add: ts sb
from read-only-unowned [OF j-bound' jth]
have }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}={}
moreover
from A-unowned-by-others [rule-format, OF - False] j-bound' jth'
have }\textrm{A}\cap\mp@subsup{\mathcal{O}}{\textrm{j}}{}={
by (auto dest: all-shared-acquired-in )
moreover
from ownership-distinct [OF i-bound j-bound' False ts sb-i jth]
have }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cap\mp@subsup{\mathcal{O}}{j}{}={
by auto
moreover note R-owned A-R
ultimately show ?thesis
by (fastforce simp add: in-read-only-convs split: if-split-asm)
qed
qed
next

```
show no-outstanding-write-to-read-only-memory \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) ts \(_{\mathrm{sb}}{ }^{\prime}\) proof
fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}}\) is \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}}\) acq \(_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(^{\text {ts }}{ }_{\text {sb }}{ }^{\prime}\)
assume jth: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{isj}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show no-write-to-read-only-memory ( \(\mathcal{S}_{\text {sb }} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
with \(\mathrm{jth} \mathrm{ts}_{\mathrm{sb}}\)-i i -bound
show ?thesis
by (auto simp add: \(\mathrm{sb}^{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}\) )
next
case False
from j -bound have j -bound \({ }^{\prime}\) : \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
with False jth have jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from no-outstanding-write-to-read-only-memory [OF j-bound \({ }^{\prime}{ }^{j}\) th \(]\)
have nw: no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
have \(R \cap\) outstanding-refs is-Write \({ }_{s b} \mathrm{sb}_{\mathrm{j}}=\{ \}\)
proof -
note dist \(=\) ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False \(\mathrm{ts}_{\left.\mathrm{sb}_{\mathrm{b}}-\mathrm{i} j \mathrm{jth} \text { ] }\right] ~}^{\text {' }}\)
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth \({ }^{\prime}\) ]]
dist
have outstanding-refs is-non-volatile-Write \({ }_{\text {sb }} \operatorname{sb}_{j} \cap \mathcal{O}_{\text {sb }}=\{ \}\)
by auto
moreover
from outstanding-volatile-writes-unowned-by-others [OF j-bound' i-bound False [symmetric] jth' \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) ]
have outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}} \cap \mathcal{O}_{\mathrm{sb}}=\{ \}\)
by auto
ultimately have outstanding-refs is-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \cap \mathcal{O}_{\mathrm{sb}}=\{ \}\)
by (auto simp add: misc-outstanding-refs-convs)
with R-owned
show ?thesis by blast
qed
then
have \(\forall \mathrm{a} \in\) outstanding-refs is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}\).
\(\mathrm{a} \in\) read-only \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \longrightarrow \mathrm{a} \in\) read-only \(\mathcal{S}_{\mathrm{sb}}\)
by (auto simp add: in-read-only-convs)
from no-write-to-read-only-memory-read-only-reads-eq [OF nw this] show ?thesis .

\section*{qed}
qed
qed
have tmps-distinct': tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text { - }}\) ]
have distinct-load-tmps is sb \(^{\prime}{ }^{\prime}\)
by (auto simp add: is sb split: instr.splits)
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{is}_{\mathrm{sb}}\) )
next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have distinct-read-tmps [] by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb}^{\mathrm{O}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next

load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have load-tmps is sb \(^{\prime} \cap\) read-tmps []\(=\{ \}\)
by (clarsimp)
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) ) qed
have valid-sops': valid-sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}}\) ]
obtain
valid-store-sops \({ }^{\prime}: \forall\) sop \(\in\) store-sops is sb \({ }^{\prime}\). valid-sop sop
by (auto simp add: is \(\mathrm{s}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
from valid-sops-nth-update [OF i-bound - valid-store-sops' \({ }^{\prime}\), where \(\mathrm{sb}=[]\) ]
show ?thesis by (auto simp add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{sb}^{\prime}\) sb \(\mathcal{O}_{\text {sb }}\) ) qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
obtain
dd-is: data-dependency-consistent-instrs (dom \(\left.\vartheta_{\mathrm{sb}}{ }^{\prime}\right)\) is \(_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp add: is sb \(\vartheta_{\text {sb }}\) )
from load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have load-tmps is sbb \(^{\prime} \cap \bigcup\) (fst ' write-sops []) \(=\{ \}\)
by (auto simp add: write-sops-append)
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have load-tmps-fresh \({ }^{\prime}\) : load-tmps-fresh \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i} \text { ] }] ~}^{\text {[ }}\)
have load-tmps (RMW at (D,f) cond ret A L R W \# is sb ) \(\cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\) by ( \(\operatorname{simp}\) add: is \(_{\text {sb }}\) )
moreover
from load-tmps-distinct \(\left[\mathrm{OF}\right.\) i-bound \(\left.\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}} \mathrm{i}\right]\) have \(\mathrm{t} \notin\) load-tmps is s \(_{\text {sb }}{ }^{\prime}\)
by (auto simp add: is s \(_{\text {sb }}\) )
ultimately have load-tmps is sb \(^{\prime} \cap \operatorname{dom}\left(\vartheta_{s b}\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}_{\mathrm{sb}}\right.\right.\right.\) a) \(\left.\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\right)=\{ \}\) by auto
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) )
qed
from enough-flushs-nth-update [OF i-bound, where sb=[] ]
have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) sb)
have valid-program-history': valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have causal': causal-program-history is sbb \(^{\prime}{ }^{\prime}\) sb \(^{\prime}\)
by ( simp add: \(\mathrm{is}_{\mathrm{sb}} \mathrm{sb} \mathrm{sb}^{\prime}\) )
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by (simp add: sb' sb)
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed
from is-sim have is: is = RMW at (D,f) cond ret A L R W \# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: suspends sb is \(\mathrm{s}_{\mathrm{sb}}\) )
from direct-memop-step.RMWWrite [where cond \(=\) cond and \(\vartheta=\vartheta_{\mathrm{sb}}\) and \(\mathrm{m}=\mathrm{m}\), OF cond']
have (RMW at (D, f) cond ret A L R W \# is \(\left.\mathrm{sb}^{\prime}, \vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\) \(\left(\mathrm{is}_{\mathrm{sb}^{\prime}, \vartheta_{\mathrm{sb}}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \mathrm{a})\left(\mathrm{f}\left(v_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right),()\right.\), \(\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m})\right)\right)\), False, \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\), Map.empty, \(\left.\mathcal{S} \oplus_{\mathrm{W}} R \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \mathrm{a})\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{m} \mathrm{a})\right)\right)\right.\right.\right.\right.\) ), (), False, \(\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}\), Map.empty)],
\[
\left.\mathrm{m}\left(\mathrm{a}:=\mathrm{f}\left(\vartheta_{\mathrm{sb}}(\mathrm{t} \mapsto \mathrm{~m} \mathrm{a})\right)\right), \mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{~L}\right)
\]
moreover
have tmps-commute: \(\vartheta_{s b}\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}_{\mathrm{sb}}\right.\right.\) a) \(\left.\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\right)=\)
\(\left(\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\{\mathrm{t}\}\right)\right)\left(\mathrm{t} \mapsto \operatorname{ret}\left(\mathrm{m}_{\mathrm{sb}}\right.\right.\) a) \(\left.\left(\mathrm{f}\left(\vartheta_{\mathrm{sb}}\left(\mathrm{t} \mapsto \mathrm{m}_{\mathrm{sb}} \mathrm{a}\right)\right)\right)\right)\)
apply (rule ext)
apply (auto simp add: restrict-map-def domIff)
done
from a-unflushed \(\mathrm{ts}_{\text {sb }}\)-i sb
have a-unflushed':
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\).
\[
\left(\operatorname{let}\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-,\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right.
\]
in a \(\notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not o is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ))
by auto
have all-shared-L: \(\forall \mathrm{i}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D}\) acq \(\vartheta\) sb. \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow\) \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}\), is, \(\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cap \mathrm{L}=\{ \}\)
proof -
\{
fix \(\mathrm{j}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume x-shared: \(x \in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{\mathrm{j}}\) )
assume \(x-L: x \in L\)
have False
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True with x -shared \(\mathrm{ts}_{\text {sb }}\)-i jth show False by (simp add: sb)
next
case False
show False
proof -
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]] have all-shared \(\operatorname{sb}_{\mathrm{j}} \subseteq\) all-acquired \(\operatorname{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\).
moreover have all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\subseteq\) all-shared
\(\mathrm{sb}_{\mathrm{j}}\)
using all-shared-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
(dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )]
by auto
moreover
from A-unacquired-by-others [rule-format, OF - False] jth j-bound
have \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}}=\{ \}\) by auto
moreover
from A-unowned-by-others [rule-format, OF - False] jth j-bound
have \(\mathrm{A} \cap \mathcal{O}_{\mathrm{j}}=\{ \}\)
by (auto dest: all-shared-acquired-in)
ultimately
show False
using L-subset \(x\)-L \(x\)-shared
by blast
qed
qed
\}
thus ?thesis by blast
qed
have all-shared-A: \(\forall \mathrm{i}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow\) \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}\), is, \(\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
```

            all-shared (takeWhile (Not o is-volatile-Write sb) sb) \capA={}
    proof -
    {
fix j pj is j ( О
assume j-bound: j < length ts sb
assume jth: ts smb
assume x-shared: x all-shared (takeWhile (Not o is-volatile-Write sb
assume x-A: x }\in
have False
proof (cases i=j)
case True with x-shared ts sb-i jth show False by (simp add: sb)
next
case False
show False
proof -
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
have all-shared sb}\mp@subsup{\textrm{s}}{\textrm{j}}{}\subseteq\mathrm{ all-acquired sbj}\cup\mathcal{O}\mp@subsup{\mathcal{O}}{\textrm{j}}{
moreover have all-shared (takeWhile (Not ○ is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathrm{ ( ) sb }}{\textrm{j}}\mathrm{ ) }\subseteq\mathrm{ all-shared
sbj
using all-shared-append [of (takeWhile (Not ○ is-volatile-Write
(dropWhile (Not ○ is-volatile-Write sb
by auto
moreover
from A-unacquired-by-others [rule-format, OF - False] jth j-bound
have }\textrm{A}\cap\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{j}{}={}\mathrm{ by auto
moreover
from A-unowned-by-others [rule-format, OF - False] jth j-bound
have }\textrm{A}\cap\mp@subsup{\mathcal{O}}{\textrm{j}}{}={
by (auto dest: all-shared-acquired-in)
ultimately
show False
using x-A x-shared
by blast
qed
qed
}
thus ?thesis by blast
qed
hence all-shared-L: }\forall\textrm{i}\mathrm{ p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. i < length ts sb l m
ts sb !i = (p, is,\vartheta, sb,\mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow
all-shared (takeWhile (Not ○ is-volatile-Write sb})\textrm{sb})\cap\textrm{L}={
using L-subset by blast

```

```

        ts sb !i = (p, is, \vartheta, sb, \mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow
        all-unshared (takeWhile (Not ○ is-volatile-Write }\mp@subsup{\mathbf{sb}}{\mathbf{b}}{})\textrm{sb})\cap\textrm{R}={
    ```
```

    proof -
    {
fix j p jis is O
assume j-bound: j < length ts sb
assume jth: ts scb
assume x-unshared: }\textrm{x}\in\mathrm{ all-unshared (takeWhile (Not ○ is-volatile-Write }\mp@subsup{\textrm{sb}}{}{\prime}\mathrm{ ) sbj
assume x-R: x }\in
have False
proof (cases i=j)
case True with x-unshared ts }\mp@subsup{\textrm{s}}{\textrm{s}}{}-\textrm{i}\mathrm{ jth show False by (simp add: sb)
next
case False
show False
proof -
from unshared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
have all-unshared sb
moreover have all-unshared (takeWhile (Not o is-volatile-Write sb
all-unshared sbj
using all-unshared-append [of (takeWhile (Not ○ is-volatile-Write }\mp@subsup{\mathbf{sb}}{\mathbf{b}}{}\mathrm{ ) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
(dropWhile (Not O is-volatile-Write sb
by auto
moreover
note ownership-distinct [OF i-bound j-bound False ts tsb-i jth]
ultimately
show False
using R-owned x-R x-unshared
by blast
qed
qed
}
thus ?thesis by blast
qed
have all-acquired-R: }\forall\textrm{i p}\mathrm{ is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. i < length ts sb
ts sb !i = (p, is,\vartheta, sb, \mathcal{D, \mathcal{O}}\boldsymbol{\mathcal{R}})\longrightarrow
all-acquired (takeWhile (Not ○ is-volatile-Write sb})\textrm{sb})\cap\textrm{R}={
proof -
{

```

```

    assume j-bound: j < length ts sb
    assume jth: ts scb
    assume x-acq: x }\in\mathrm{ all-acquired (takeWhile (Not ○ is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb }\mp@subsup{\textrm{b}}{\textrm{j}}{}\mathrm{ )
    assume x-R: x }\in
    have False
    proof (cases i=j)
        case True with x-acq tssb-i jth show False by (simp add: sb)
    next
    ```
```

    case False
    show False
    proof -
    ```
    from x -acq have \(\mathrm{x} \in\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
    using all-acquired-append [of takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{j}\)
    dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ]
    by auto
        moreover
        note ownership-distinct [OF i-bound j-bound False \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) jth]
        ultimately
        show False
    using R-owned x -R
    by blast
        qed
    qed
\}
thus ?thesis by blast
        qed
    have all-shared-R: \(\forall\) i p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow\)
                \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
                all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cap \mathrm{R}=\{ \}\)
        proof -
\{
    fix \(\mathrm{jp}_{\mathrm{j}} \mathrm{is} \mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}\)
    assume j -bound: \(\mathrm{j}<\) length \(^{\mathrm{ts}} \mathrm{s}_{\mathrm{sb}}\)
    assume jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathcal{\vartheta}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
    assume x -shared: \(\mathrm{x} \in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
    assume \(x-R: x \in R\)
    have False
    proof (cases \(\mathrm{i}=\mathrm{j}\) )
        case True with x -shared \(\mathrm{ts}_{\mathrm{sb}^{-}} \mathrm{i}\) jth show False by (simp add: sb )
    next
        case False
        show False
        proof -
        from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
        have all-shared \(\mathrm{sb}_{\mathrm{j}} \subseteq\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\).
            moreover have all-shared (takeWhile (Not \(\circ\) is-volatile-Write \(\left.\mathrm{s}_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\) ) \(\subseteq\) all-shared
\(\mathrm{sb}_{\mathrm{j}}\)
    using all-shared-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \({ }_{j}\) )
        (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{\mathrm{j}}\) )]
    by auto
        moreover
        note ownership-distinct [OF i-bound j-bound False \(\mathrm{ts}_{\text {sb }}-\mathrm{i}\) jth]
        ultimately
        show False
    using R-owned x -R x-shared
```

by blast
qed
qed
}
thus ?thesis by blast
qed

```
from share-all-until-volatile-write-commute \(\left[\mathrm{OF}\right.\) <ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}\) 〉 «sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉
all-shared-L all-shared-A all-acquired-R all-unshared-R all-shared-R]
have share-commute: share-all-until-volatile-write ts \(\mathrm{sb}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}=\) share-all-until-volatile-write \(\mathrm{ts}_{\mathrm{sb}}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).

\section*{\{}
fix \(\mathrm{jp}_{\mathrm{j}}\) is \(\mathrm{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}\)
assume jth: \(\mathrm{ts}_{\mathrm{s} b}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume j-bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume neq: \(\mathrm{i} \neq \mathrm{j}\)
have release (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
\(\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}} \cup \mathrm{R}-\mathrm{L}\right) \mathcal{R}_{\mathrm{j}}\)
\(=\) release (takeWhile (Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)\)
(dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\)
proof -
\{
fix a
assume a-in: \(\mathrm{a} \in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
have \(\left(\mathrm{a} \in\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}} \cup \mathrm{R}-\mathrm{L}\right)\right)=\left(\mathrm{a} \in \operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right)\)
proof -
from A-unowned-by-others [rule-format, OF j-bound neq ] jth
A-unacquired-by-others [rule-format, OF - neq] j-bound
have A -dist: \(\mathrm{A} \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\)
by (auto dest: all-shared-acquired-in)
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]] a-in
all-shared-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
(dropWhile (Not o is-volatile-Write \({ }_{s b}\) ) sb \({ }_{j}\) )]
have a-in: \(\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto
with ownership-distinct [OF i-bound j-bound neq \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} j \mathrm{jth}\) ]
have a \(\notin\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb) by auto
with A-dist R-owned A-R A-shared-owned L-subset a-in
obtain a \(\notin \mathrm{R}\) and \(\mathrm{a} \notin \mathrm{L}\)
by fastforce
then show ?thesis by auto
qed
\}
then
```

        show ?thesis
    apply -
    apply (rule release-all-shared-exchange)
    apply auto
        done
    qed
    }
note release-commute = this
have }(\mp@subsup{\textrm{ts}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime},\mp@subsup{\textrm{m}}{\textrm{sb}}{}(\textrm{a}:=\textrm{f}(\mp@subsup{\vartheta}{\textrm{sb}}{}(\textrm{t}\mapsto\mp@subsup{\textrm{m}}{\textrm{sb}}{}\textrm{a}))),\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime})~(\textrm{ts}[\textrm{i}:=(\mp@subsup{\textrm{p}}{\textrm{sb}}{},\mp@subsup{\textrm{is}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime}
\vartheta sb
f (\vartheta (
apply (rule sim-config.intros)
apply (simp only: m-a )
apply (simp only: m)
apply (simp only: flush-all-until-volatile-write-update-other [OF a-unflushed',
symmetric] ts sb ')
apply (simp add: flush-all-until-volatile-nth-update-unused [OF i-bound ts smb
fied sb] sb
apply (simp add: ts sbb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\textrm{m
flush-all-until-volatile-nth-update-unused [OF i-bound ts ssb
using share-all-until-volatile-write-RMW-commute [OF i-bound ts }\mp@subsup{\textrm{sbb}}{\mathbf{s}}{}-\textrm{i}[\mathrm{ [simplified is
apply (clarsimp simp add: \mathcal{S ts sb}}\mp@subsup{}{\prime}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{is}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mathrm{ sb share-commute)
using leq
apply (simp add: ts sb
using i-bound i-bound'ts-sim
apply (clarsimp simp add: Let-def nth-list-update
ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{D}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mathrm{ ex-not m-a
split: if-split-asm)
apply (rule conjI)
apply clarsimp
apply (rule tmps-commute)
apply clarsimp
apply (frule (2) release-commute)
apply clarsimp
apply fastforce
done
ultimately
show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-sops'
valid-dd' load-tmps-fresh' enough-flushs'
valid-program-history' valid' }\mp@subsup{\textrm{m}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime
by (auto simp del: fun-upd-apply)
next
case (SBHGhost A L R W)
then obtain
is
\mathcal{O}}\mp@subsup{\textrm{sb}}{}{\prime}:\mp@subsup{\mathcal{O}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}=\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mathrm{ and
\mp@subsup{\mathcal{R}}{\textrm{sb}}{\prime}
\mp@subsup{\vartheta}{\textrm{sb}}{\prime}}:\mp@subsup{\vartheta}{\textrm{sb}}{\prime}=\mp@subsup{\vartheta}{\textrm{sb}}{}\mathrm{ and
\mathcal{D}}\mp@subsup{\textrm{sb}}{}{\prime}:\mp@subsup{\mathcal{D}}{\textrm{sb}}{\prime}=\mp@subsup{\mathcal{D}}{\textrm{sb}}{}\mathrm{ and

```
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\mathrm{Ghost}_{\mathrm{sb}}\right.\) A L R W] and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\)
by auto

\section*{from safe-memop-flush-sb [simplified is ssb ] obtain}

L-subset: \(\mathrm{L} \subset \mathrm{A}\) and
A-shared-owned: A \(\subseteq \operatorname{dom}\) (share ?drop-sb \(\mathcal{S}\) ) \(\cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\) and
R -acq: \(\mathrm{R} \subseteq\) acquired True sb \(\mathcal{O}_{\text {sb }}\) and
A-R: \(A \cap R=\{ \}\) and
A-unowned-by-others-ts:
\(\forall \mathrm{j}<\) length \((\) map owned ts\() . \mathrm{i} \neq \mathrm{j} \longrightarrow(\mathrm{A} \cap(\) owned \((\mathrm{ts}!\mathrm{j}) \cup\) dom \((\) released \((\mathrm{ts}!\mathrm{j})))=\{ \})\)
by cases auto
from A-unowned-by-others-ts ts-sim leq
have A-unowned-by-others:
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\left(\operatorname{let}\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right.\)
in \(\mathrm{A} \cap\) (acquired True (takeWhile (Not \(\circ\) is-volatile-Write \(\mathrm{e}_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}} \cup\)
all-shared \(\left(\right.\) takeWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\right)\right)=\{ \}\right)\)
apply (clarsimp simp add: Let-def)
subgoal for j
apply (drule-tac \(x=j\) in spec)
apply (force simp add: dom-release-takeWhile)
done
done
have A-unused-by-others:
\(\forall \mathrm{j}<\) length ( \(\mathrm{map} \mathcal{O}\)-sb \(\mathrm{ts}_{\mathrm{sb}}\) ). \(\mathrm{i} \neq \mathrm{j} \longrightarrow\) (let \(\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)=\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\) ! j
in \(\mathrm{A} \cap\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}=\{ \}\) )
proof -
\{
fix \({ }^{j} \mathcal{O}_{j} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\text {sb }}\) )
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{ts}_{\mathrm{sb}} \mathrm{j}:\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right)!\mathrm{j}=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)\)
assume conflict: \(\mathrm{A} \cap\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\)
have False
proof -
from j-bound leq
have j-bound': j < length (map owned ts)
by auto
from j -bound have j -bound \({ }^{\prime \prime}: \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by auto
from j-bound \({ }^{\prime}\) have j -bound \({ }^{\prime \prime \prime}: \mathrm{j}\) < length ts by simp

\section*{from conflict obtain \(\mathrm{a}^{\prime}\) where}
\(a^{\prime}-\mathrm{in}: \mathrm{a}^{\prime} \in \mathrm{A}\) and
\(\mathrm{a}^{\prime}\)-in-j: \(\mathrm{a}^{\prime} \in\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb \(_{\mathrm{j}}\)
by auto
```

    let ?take-sb
    let ?drop-sb
    from ts-sim [rule-format, OF j-bound '/] ts ssb-j j-bound "
    obtain p puspendsj is isbj }\mp@subsup{\vartheta}{\textrm{sbj}}{}\mp@subsup{\mathcal{D}}{\mathrm{ sbj }}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ is is where
        ts
        suspendsj: suspends
        \mathcal{D}}\textrm{j}:\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}=(\mp@subsup{\mathcal{D}}{\textrm{j}}{}\vee\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\not={})\mathrm{ and
        is}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathrm{ : instrs suspends}
        tsj: ts!j = (hd-prog pos suspendsj, is 
            \mp@subsup{v}{\mathrm{ sbj }}{}\mp@subsup{|}{}{\prime}(\mathrm{ dom }\mp@subsup{\vartheta}{\mathrm{ sbj }}{}- read-tmps suspendsj ),(),
            \mathcal{D}}\textrm{j}\mathrm{ , acquired True ?take-sb }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ , release ?take-sb 
        apply (cases ts sb!j)
        apply (force simp add: Let-def)
        done
    have a}\mp@subsup{a}{}{\prime}\in\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\mathrm{ sb }}{}{\prime}\mp@subsup{\mathrm{ suspends }}{\textrm{j}}{
    proof -
        from a'-in-j
        have a' }\in\mathrm{ outstanding-refs is-volatile-Write sb (?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}@\mathrm{ @ ?drop-sb
    by simp
        thus ?thesis
        apply (simp only: outstanding-refs-append suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
        apply (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
        done
        qed
    from split-volatile-Write sb-in-outstanding-refs [OF this]
    obtain sop v ys zs A' L' R' W' W' where
    split-suspends}\mp@subsup{\mp@code{j}}{\textrm{j}}{\mathrm{ : suspends }
    = ?suspends)
by blast
from direct-memop-step.Ghost [where }\vartheta=\mp@subsup{\vartheta}{\mathrm{ sb }}{}\mathrm{ and m=flush ?drop-sb m]
have (Ghost A L R W\# is cb}\mp@subsup{}{}{\prime}\mathrm{ ,
\varthetasb
acquired True sb }\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mathrm{ , release sb (dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mp@subsup{\mathcal{R}}{\textrm{sb}}{}\mathrm{ , share ?drop-sb S S )}
(is }\mp@subsup{\textrm{s}}{\textrm{s}}{}\mp@subsup{}{}{\prime},\mp@subsup{\vartheta}{\textrm{sb}}{},(),\mathrm{ flush ?drop-sb m, 埧,
acquired True sb }\mp@subsup{\mathcal{O}}{sb}{}\cup\textrm{A}-\textrm{R}\mathrm{ ,
augment-rels (dom (share ?drop-sb \mathcal{S})) R (release sb (dom S}\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mp@subsup{\mathcal{R}}{\textrm{sb}}{})\mathrm{ ),
share ?drop-sb S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
from direct-computation.concurrent-step.Memop [OF
i-bound-ts' [simplified issb
have store-step: (?ts', flush ?drop-sb m, share ?drop-sb S ) =
(?ts'[i := ( }\mp@subsup{\textrm{p}}{\textrm{sb}}{},\mp@subsup{\textrm{is}}{\textrm{sb}}{}\mp@subsup{}{}{\prime},\mp@subsup{\vartheta}{\textrm{sb}}{},(),\mp@subsup{\mathcal{D}}{\textrm{sb}}{},\mathrm{ acquired True sb }\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\cup\textrm{A}-\textrm{R},\mathrm{ augment-rels
(dom (share ?drop-sb S S) R (release sb (dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mp@subsup{\mathcal{R}}{\textrm{sb}}{})\mathrm{ )],
flush ?drop-sb m,share ?drop-sb S }\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
(is - }\mp@subsup{=>}{\textrm{d}}{\textrm{d}}(?\textrm{ts}-\textrm{A},?\textrm{m}-\textrm{A},?\mathrm{ share-A})

```
```

by ( $\operatorname{simp}$ add: is $_{\text {sb }}$ )

```
```

from i-bound' have i-bound'\prime: i < length ?ts-A
by simp
from valid-program-history [OF j-bound " ts }\mp@subsup{\textrm{sb}}{\mathbf{s}-\textrm{j}}{}\mathrm{ ]
have causal-program-history is isbj }\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ .
then have cph: causal-program-history is }\mp@subsup{\mp@code{sbj}}{}{\mathrm{ ? }
apply -
apply (rule causal-program-history-suffix [where sb=?take-sb}\mp@subsup{]}{j}{]}\mathrm{ )
apply (simp only: split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{[symmetric] suspends}\mp@subsup{\textrm{S}}{\textrm{j}}{}\mathrm{ )
apply (simp add: split-suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
done

```
from \(\mathrm{ts}_{\mathrm{j}}\) neq-i-j j -bound
have ts-A-j: ?ts-A! \(=\left(h d-p r o g p_{j}\left(y s @ W_{r i t e}^{s b}\right.\right.\) True \(\left.a^{\prime} \operatorname{sop} v A^{\prime} L^{\prime} R^{\prime} W^{\prime} \# z s\right)\), is \({ }_{j}\),
    \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\right.\) dom \(\vartheta_{\text {sbj }}\) - read-tmps (ys @ Write \({ }_{\text {sb }}\) True a'sop v \(\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\) ), (), \(\mathcal{D}_{\mathrm{j}}\),
    acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\mathrm{T}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\) )
    by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) )
from j-bound \({ }^{\prime \prime \prime}\) i-bound \({ }^{\prime}\) neq-i-j have j-bound \({ }^{\prime \prime \prime \prime}\) : \(\mathrm{j}<\) length ?ts-A
    by simp
```

from valid-last-prog $\left[\mathrm{OF} \mathrm{j}\right.$-bound $\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]$ have last-prog: last-prog $\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}$.
then
have lp : last-prog $\mathrm{p}_{\mathrm{j}}$ ? suspends $=\mathrm{p}_{\mathrm{j}}$
apply -
apply (rule last-prog-same-append [where $\mathrm{sb}=$ ? take-sb ${ }_{\mathrm{j}}$ ])
apply (simp only: split-suspends $\mathrm{s}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done

```
from valid-reads [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} m_{s b} s b_{j}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\mathrm{sb}}\) 〉 j-bound \({ }^{\prime \prime}\) \(\mathrm{ts}_{\mathrm{sb}}\)-j reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \({ }_{\mathrm{j}}\) by ( simp add: m suspends j )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound \({ }^{\prime \prime}\) neq-i-j ts sb \(_{\text {sb }}\)-i \(\left.\mathrm{ts}_{\mathrm{sb}-\mathrm{j}}\right]\)
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by (simp add: suspendsj \({ }_{\mathrm{j}}\) )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
?m-A suspends \({ }^{\text {. }}\)
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?m-A ys
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) reads-consistent-append)
from valid-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h-consis:
history-consistent \(\vartheta_{\text {sbj }}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \({ }_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) (?take-sb \({ }_{\mathrm{j}} @_{\text {suspends }}^{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{H}_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\left.\mathrm{j}_{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}} @\) ?drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\) by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left.\left._{\mathrm{j}}\right)\right)\) ?take-sb \(\mathrm{s}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from valid-write-sops [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb }}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j} @\) suspends \({ }_{j}\) )
by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by ( simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}} \mathrm{ys}=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{\mathrm{j}}\) [symmetric]
split-suspends \({ }_{j}\) )
from flush-store-buffer-append [
OF j-bound \({ }^{\prime \prime \prime \prime}\) is \(_{\mathrm{j}}\left[\right.\) simplified split-suspends \(_{\mathrm{j}}\) ] cph [simplified suspends \({ }_{\mathrm{j}}\) ]
ts-A-j [simplified split-suspends \({ }_{\mathrm{j}}\) ] refl lp [simplified split-suspends \({ }_{\mathrm{j}}\) ] reads-consis-m-A-ys
hist-consis \({ }^{\prime}\) [simplified split-suspends \({ }_{j}\) ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read sb-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?share-A]
obtain is \({ }_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : instrs (Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) @ is \(\mathrm{sbj}_{\mathrm{sbj}}=\)
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime} @ \operatorname{prog}-\) instrs (Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) and
steps-ys: (?ts-A, ?m-A, ?share-A) \(\Rightarrow_{d}{ }^{*}\)
(?ts-A[j:= (last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v A \(\left.\mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\) ) ys, \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\right.\) Write \(_{\text {sb }}\) True a'sop v A \({ }^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\)
\(\mathrm{zs})),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}}\) ys \(\neq\{ \}\), acquired True ys
(acquired True ? take-sb \(\left.\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )],
flush ys ?m-A,
share ys ?share-A)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?\) shared-ys \(\left.)\right)\) by (auto)
note conflict-computation \(=\) rtranclp-trans \([\mathrm{OF}\) rtranclp-r-rtranclp [OF steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj \(^{\text {( }}\) (ys @ \(\left[W_{r i t e}^{s b}\right.\) True a'sop v \(\left.\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\) @ zs)
by simp
from causal-program-history-suffix [OF this]
have \(\mathrm{cph}^{\prime}\) : causal-program-history is sbj \(^{\mathrm{zs}}\).
interpret causal \({ }_{j}\) : causal-program-history is \(_{\text {sbj }}\) zs by (rule cph')
from causal \({ }_{\mathrm{j}}\).causal-program-history [of [], simplified, OF refl] \(\mathrm{is}_{\mathrm{j}}{ }^{\text {' }}\)
obtain is \(_{\mathrm{j}}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}: \mathrm{is}_{\mathrm{j}}^{\prime}=\) Write True \(\mathrm{a}^{\prime}\) sop \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ \(\mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime} @\) prog-instrs zs
by clarsimp
from j-bound \({ }^{\prime \prime \prime}\)
have j-bound-ys: \(\mathrm{j}<\) length ?ts-ys
by auto
from \(j\)-bound-ys neq-i-j
have ts-ys-j: ?ts-ys!j=(last-prog \(\left(\right.\) hd-prog \(p_{j}\left(W_{\text {rite }}\right.\) sb True \(\mathrm{a}^{\prime}\) sop v \(\left.\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\) ys, \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps \(\left(\right.\) Write \(_{\text {sb }}\) True a'sop v \(\left.\left.\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),()\), \(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified \(\mathrm{is}_{\mathrm{j}}{ }^{\text {] }}\) ]
have a-unowned:
\(\forall \mathrm{i}<\) length ?ts-ys. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{i}}\right)\) apply cases apply (auto simp add: Let-def is \(_{\text {sb }}\) ) done
from a'-in a-unowned [rule-format, of i] neq-i-j i-bound \({ }^{\prime}\) A-R
show False
by (auto simp add: Let-def)
qed
\}
thus ?thesis
by (auto simp add: Let-def)

\section*{qed}
have A-unaquired-by-others:
\(\forall \mathrm{j}<\) length (map \(\mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\) ). i \(\neq \mathrm{j} \longrightarrow\) (let \(\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)=\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\) in \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}}=\{ \}\) )
proof -
\{
fix \(\mathrm{j}^{\boldsymbol{\mathcal { O }}} \mathrm{j}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}:\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right)!\mathrm{j}=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)\)
assume conflict: \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \neq\{ \}\)
have False
proof -
from j-bound leq
have j-bound': \(\mathrm{j}<\) length (map owned ts)
by auto
from j -bound have j -bound \({ }^{\prime \prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\) by auto
from j-bound \({ }^{\prime}\) have j-bound \({ }^{\prime \prime \prime}\) : \(\mathrm{j}<\) length ts by simp
from conflict obtain \(\mathrm{a}^{\prime}\) where
\(a^{\prime}-i n: a^{\prime} \in A\) and
\(a^{\prime}\)-in-j: \(a^{\prime} \in\) all-acquired \(\operatorname{sb}_{j}\)
by auto
```

    let ?take-sb
    let ?drop-sb
    from ts-sim [rule-format, OF j-bound '/] ts ssb-j j-bound "
    obtain p puspendsj is isbj }\mp@subsup{\vartheta}{\textrm{sbj}}{}\mp@subsup{\mathcal{D}}{\mathrm{ sbj }}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mathrm{ is is where
        tssb-j: ts ssb
        suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ : suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}=\mathrm{ ?drop-sb
        \mathcal{D}}\textrm{j}:\mp@subsup{\mathcal{D}}{\textrm{sbj}}{}=(\mp@subsup{\mathcal{D}}{\textrm{j}}{}\vee\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\not={})\mathrm{ and
        is}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathrm{ : instrs suspends}
        tsj: ts!j = (hd-prog pos suspendsj, is 
                            \vartheta sbj |
                            \mathcal{D}}\mp@subsup{\textrm{j}}{\textrm{j}}{\mathrm{ , acquired True ?take-sb}}\mp@subsup{\textrm{j}}{\textrm{j}}{\textrm{j}},\mathrm{ ,release ?take-sb
    apply (cases ts sb!j)
    apply (force simp add: Let-def)
    done
    from a'-in-j all-acquired-append [of ?take-sb j ?drop-sbj]
    have a' }\mp@subsup{\textrm{a}}{}{\prime}\mathrm{ all-acquired ?take-sb}\mp@subsup{\textrm{j}}{\textrm{j}}{}\vee\mp@subsup{\textrm{a}}{}{\prime}\in\mathrm{ all-acquired suspends}\mp@subsup{}{\textrm{j}}{
    by (auto simp add: suspendsj)
    thus False
    proof
    assume a' }\in\mathrm{ all-acquired ?take-sb 
    with A-unowned-by-others [rule-format, OF - neq-i-j] ts sb-j j-bound a'-in
    show False
    by (auto dest: all-acquired-unshared-acquired)
next
assume conflict-drop: a' }\in\mathrm{ all-acquired suspendsj
from split-all-acquired-in [OF conflict-drop]
show False
proof
assume }\exists\mathrm{ sop a" v ys zs A L R W.
suspends}\mp@subsup{j}{j}{}= ys @ Write sb True a'" sop v A L R W\# zs ^ a' \in A
then

```

```

    split-suspendsj: suspendsj
        (is suspends}\mp@subsup{}{j}{}=\mathrm{ ?suspends) and
    a
    by auto

```
from direct-memop-step.Ghost [where \(\vartheta=\vartheta_{s b}\) and m=flush ?drop-sb m] have (Ghost A L R W\# is \({ }_{\text {sb }}{ }^{\prime}\),
\(\vartheta_{\mathrm{sb}},()\), flush ?drop-sb \(\mathrm{m}, \mathcal{D}_{\mathrm{sb}}\),
acquired True sb \(\mathcal{O}_{\mathrm{sb}}\), release sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\),share ?drop-sb \(\left.\mathcal{S}\right) \rightarrow\) \(\left(\mathrm{is}_{\mathbf{s b}}{ }^{\prime}, \vartheta_{\mathbf{s b}},()\right.\), flush ?drop-sb m, \(\mathcal{D}_{\text {sb }}\), acquired True sb \(\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}\), augment-rels (dom (share ?drop-sb \(\mathcal{S})\) ) R (release sb (dom \(\mathcal{S}_{\mathrm{sb}}\) ) \(\mathcal{R}_{\mathrm{sb}}\) ), share ?drop-sb \(\left.\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF

have store-step: (?ts \({ }^{\prime}\), flush ?drop-sb m, share ?drop-sb \(\left.\mathcal{S}\right) \Rightarrow_{\mathrm{d}}\) \(\left(? \mathrm{ts}^{\prime}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}_{\mathrm{sb}}\right.\right.\right.\),
acquired True \(\operatorname{sb} \mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}\),
augment-rels (dom (share ?drop-sb \(\mathcal{S})\) ) \(\mathrm{R}\left(\right.\) release \(\left.\operatorname{sb}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\) )],
flush ?drop-sb m,share ?drop-sb \(\left.\mathcal{S} \oplus \mathrm{w} R \ominus_{\mathrm{A}} \mathrm{L}\right)\)
\(\left(\right.\) is \(-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}, ?\) share-A \(\left.)\right)\)
by ( \(\operatorname{simp}\) add: is \(_{\text {sb }}\) )
from i-bound' have i-bound \({ }^{\prime \prime}\) : i < length ?ts-A
by \(\operatorname{simp}\)
from valid-program-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}\) ]
have causal-program-history is \(\mathrm{sbj}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix \(\left[\right.\) where \(\mathrm{sb}=\) ? take-sb \(\mathrm{b}_{\mathrm{j}}\) ])
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \({ }_{j}\) )
done
from \(t_{s j}\) neq-i-j j-bound
have ts-A-j: ?ts-A!j \(=\left(\right.\) hd-prog \(p_{j}\left(y s @ W r i t e e_{s b}\right.\) True \(\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\operatorname{read-tmps}\left(y s @ W_{\text {lite }}\right.\right.\) True \(\left.\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),()\), \(\mathcal{D}_{\mathrm{j}}\),
acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) )
from j-bound \({ }^{\prime \prime \prime}\) i-bound \({ }^{\prime}\) neq-i-j have j-bound \({ }^{\prime \prime \prime \prime}:\) j < length ?ts-A
by \(\operatorname{simp}\)
from valid-last-prog [OF j-bound \(\left.{ }^{\prime \prime} \operatorname{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ? suspends \(=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \({ }_{\mathrm{j}}\) ])
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
from valid-reads [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ’ j-bound \({ }^{\prime \prime}\)
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ) m suspends \({ }_{\mathrm{j}}\) by ( \(\operatorname{simp}\) add: m suspends \(\mathrm{j}_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound \({ }^{\prime \prime}\) neq-i-j \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by ( \(\operatorname{simp}\) add: suspends \({ }_{\mathrm{j}}\) )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
?m-A suspends \({ }_{\mathrm{j}}\).
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}}\) ) ?m-A ys
by (simp add: split-suspendsj reads-consistent-append)
from valid-history [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(h\)-consis:
history-consistent \(\vartheta_{\text {sbj }}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \({ }_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) (?take-sb \({ }_{\mathrm{j}} @\) suspends \({ }_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \({ }_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\left._{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(_{\mathrm{j}} @\) ?drop-sb \(\left.\mathrm{d}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\) by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \({ }_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left.\left._{\mathrm{j}}\right)\right)\) ?take-sb \(\mathrm{sb}_{\mathrm{j}}=\) last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from valid-write-sops [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j} @_{\text {suspends }}^{\mathrm{j}}\) )
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ? take-sb \(_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis]
last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}}\) ys \(=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \({ }_{j}\) )
from flush-store-buffer-append [
OF j-bound \({ }^{\prime \prime \prime \prime}\) is \(_{\mathrm{j}}\left[\right.\) simplified split-suspends \({ }_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ]
ts-A-j [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] refl lp [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] reads-consis-m-A-ys
hist-consis' [simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{s b}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?share-A]
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : instrs ( Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) @ is \(\mathrm{s}_{\text {sbj }}=\)
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) @ prog-instrs (Write \({ }_{\mathrm{sb}}\) True \(\mathrm{a}^{\prime \prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) and
steps-ys: (?ts-A, ?m-A, ?share-A) \(\Rightarrow_{d}{ }^{*}\)
(?ts-A[j:= (last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Write \(_{\text {sb }}\) True \(\left.\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\) ys, is \({ }_{j}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{6}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) zs) ),(),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \(_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )],
flush ys ?m-A,share ys ?share-A)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(?\) ts-ys,?m-ys,?shared-ys))
by (auto)
note conflict-computation \(=\) rtranclp-trans \([\mathrm{OF}\) rtranclp-r-rtranclp \([\mathrm{OF}\) steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj \(^{\text {( }}\) (ys @ \(\left[\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ] \()\) @ zs)
by simp
from causal-program-history-suffix [OF this]
have \(\mathrm{cph}^{\prime}\) : causal-program-history is \(\mathrm{s}_{\text {sbj }} \mathrm{zs}\).
interpret causalj: causal-program-history is sbj \(^{\text {zs }}\) by (rule cph \({ }^{\text {) }}\)
from causalj.causal-program-history [of [], simplified, OF reff] is \({ }_{j}{ }^{\prime}\) obtain is \({ }_{\mathrm{j}}{ }^{\prime \prime}\)
where is \(\mathrm{j}_{\mathrm{j}}^{\prime}\) : \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Write True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ \(\mathrm{is}_{\text {sbj }}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) @ prog-instrs zs
by clarsimp
from j-bound \({ }^{\prime \prime \prime}\)
have j-bound-ys: j < length ?ts-ys
by auto
from j -bound-ys neq-i-j
have ts-ys-j: ?ts-ys! \(\mathrm{j}=\left(\operatorname{last-prog}\left(h d-p r o g \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) zs)) ys, isj \({ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\right.\) Write \(_{\text {sb }}\) True \(\left.\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \(_{\text {sb }}\) ys \(\neq\{ \}\),
acquired True ys (acquired True ?take-sb \(\boldsymbol{\mathcal { O }}_{\mathrm{j}}\) ), \(\mathcal{R}_{\mathrm{j}}\) )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is \({ }_{j}\) ]
have A '-unowned:
\(\forall \mathrm{i}<\) length ?ts-ys. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)\)
apply cases
apply (fastforce simp add: Let-def is stb ) +
done
from \(\mathrm{a}^{\prime}\)-in \(\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}\)-unowned [rule-format, of i\(]\) neq-i-j i-bound \({ }^{\prime} \mathrm{A}-\mathrm{R}\)
show False
by (auto simp add: Let-def)
next
assume \(\exists \mathrm{A}\) L R W ys zs.
suspends \(_{j}=\) ys @ Ghost \(_{\text {sb }}\) A L R W \# zs \(\wedge \mathrm{a}^{\prime} \in \mathrm{A}\)
then
obtain ys zs A \({ }^{\prime} L^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
split-suspends j : suspends \(\mathrm{s}_{\mathrm{j}}=\mathrm{ys} @\) Ghost \(_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) (is suspends \(\mathrm{s}_{\mathrm{j}}=\) ?suspends) and
\(a^{\prime}-A^{\prime}: a^{\prime} \in A^{\prime}\)
by auto
from direct-memop-step. Ghost [where \(\vartheta=\vartheta_{\mathrm{sb}}\) and \(\mathrm{m}=\) flush ?drop-sb m]
have (Ghost A L R W\# is \({ }^{\text {sb }}{ }^{\prime}\),
\(\vartheta_{\mathrm{sb}},()\), flush ?drop-sb m, \(\mathcal{D}_{\mathrm{sb}}\),
acquired True sb \(\mathcal{O}_{\mathrm{sb}}\), release sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\), share ?drop-sb \(\left.\mathcal{S}\right) \rightarrow\)
(is \({ }_{\text {sb }}{ }^{\prime}, \vartheta_{\text {sb }},()\), flush ?drop-sb m, \(\mathcal{D}_{\text {sb }}\),
acquired True sb \(\mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}\),
augment-rels (dom (share ?drop-sb \(\mathcal{S}\) )) R (release sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) ),
share ?drop-sb \(\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF

have store-step: (?ts \({ }^{\prime}\), flush ?drop-sb m, share ?drop-sb \(\left.\mathcal{S}\right) \Rightarrow_{\mathrm{d}}\) \(\left(? \mathrm{ts}^{\prime}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}_{\mathrm{sb}}\right.\right.\right.\), acquired True \(\operatorname{sb} \mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\),augment-rels (dom (share ?drop-sb \(\mathcal{S})\) ) R (release sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\) )],
flush ?drop-sb m,share ?drop-sb \(\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
\(\left(\right.\) is \(-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}, ?\) share- A\(\left.)\right)\)
by ( \(\operatorname{simp}\) add: is \(_{\text {sb }}\) )
from i-bound \({ }^{\prime}\) have i-bound \({ }^{\prime \prime}\) : i < length ?ts-A
by simp
from valid-program-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
have causal-program-history \(\mathrm{is}_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}\).
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? take-sb \(_{\mathrm{j}}\) ])
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \({ }_{j}\) )
done
from \(t s_{j}\) neq-i-j j-bound
have ts-A-j: ?ts-A! \(=\left(h d-p r o g ~ p_{j}\left(y s @\right.\right.\) Ghost \(\left._{\text {sb }} A^{\prime} L^{\prime} R^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\), is \(\mathrm{j}_{\mathrm{j}}\), \(\left.\vartheta_{\mathrm{sbj}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{sbj}}-\operatorname{read}-\mathrm{tmps}\left(\mathrm{ys} @\right.\right.\) Ghost \(\left.\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right),(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\mathrm{f}_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\) ) by ( \(\operatorname{simp}\) add: split-suspends \({ }_{\mathrm{j}}\) )
from j-bound \({ }^{\prime \prime \prime}\) i-bound \({ }^{\prime}\) neq-i-j have j-bound \({ }^{\prime \prime \prime \prime}:\) j \(<\) length ?ts-A by simp
from valid-last-prog [OF j-bound \(\left.{ }^{\prime \prime} \operatorname{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\). then
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ?suspends \(=\mathrm{p}_{\mathrm{j}}\) apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \({ }_{\mathrm{j}}\) ])
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
from valid-reads [ OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}-\mathrm{j}}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ) j-bound \({ }^{\prime \prime}\)
\(\mathrm{ts}_{\mathrm{sb}}\)-j reads-consis]
have reads-consis-m: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ) m suspends \({ }_{\mathrm{j}}\) by ( \(\operatorname{simp}\) add: m suspends \(\mathrm{s}_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound \({ }^{\prime \prime}\) neq-i-j \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
have outstanding-refs is-Write sb \(^{\text {? }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by ( \(\operatorname{simp}\) add: suspends \({ }_{j}\) )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}}\) )
?m-A suspendsj.
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?m-A ys
by (simp add: split-suspends \({ }_{j}\) reads-consistent-append)
from valid-history [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}\) ]
have h-consis:
history-consistent \(\vartheta_{\text {sbj }}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \({ }_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) (?take-sb \({ }_{\mathrm{j}} @\) suspends \({ }_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \(\mathrm{H}_{\mathrm{j}}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left._{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{f}_{\mathrm{j}} @\) ?drop-sb \(\left.\mathrm{d}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \({ }_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )

\section*{moreover}
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left._{\mathrm{j}}\right)\) ) ? take-sb \(\mathrm{b}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
qed
from valid-write-sops [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
by ( \(\operatorname{simp}\) add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
valid-sops-drop: \(\forall\) sop \(\in\) write-sops ys. valid-sop sop
apply (simp only: write-sops-append )
apply auto
done
from read-tmps-distinct [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{\mathrm{j}}\) @suspends \(_{\mathrm{j}}\) )
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )

\section*{then obtain}
read-tmps-take-drop: read-tmps ? take-sb \({ }_{\mathrm{j}} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis]
last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by (simp add: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \(_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile- \(\operatorname{Read}_{\mathrm{sb}}\) ys \(=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \({ }_{j}\) )
from flush-store-buffer-append [
OF j-bound \({ }^{\prime \prime \prime \prime}\) is \(\mathrm{is}_{\mathrm{j}}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] cph [simplified suspends \({ }_{\mathrm{j}}\) ]
ts-A-j [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] refl lp [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] reads-consis-m-A-ys
hist-consis \({ }^{\prime}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{s b}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?share-A]
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
is \(_{j}^{\prime}\) : instrs ( Ghost \(_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) @ \(\mathrm{is}_{\text {sbj }}=\) \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) @ prog-instrs (Ghost \({ }_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) and
steps-ys: (?ts-A, ?m-A, ?share-A) \(\Rightarrow_{\mathrm{d}}{ }^{*}\)
(?ts-A \(\left[\mathrm{j}:=\left(\operatorname{last}-\mathrm{prog}\left(h d-p r o g \mathrm{p}_{\mathrm{j}}\left(\right.\right.\right.\right.\) Ghost \(\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\) ys,
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\text {' }}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\right.\) Ghost \(\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\) ), (),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \(_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired
True ?take-sb \(\left.{ }_{j} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) ) ],
flush ys ?m-A, share ys ?share-A)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?\) shared-ys))
by (auto)
note conflict-computation \(=\) rtranclp-trans \([\mathrm{OF}\) rtranclp-r-rtranclp \([\mathrm{OF}\) steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj \(\left(\left(y s\right.\right.\) @ \(\left[\right.\) Ghost \(\left._{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}\right)\) @ zs\()\)
by simp
from causal-program-history-suffix [OF this]
have cph': causal-program-history is \(\mathrm{s}_{\text {sj }} \mathrm{zs}\).
interpret causal \({ }_{j}\) : causal-program-history \(\mathrm{is}_{\text {sbj }} \mathrm{zs}\) by (rule cph')
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }^{\prime}{ }^{\prime}\) obtain is \({ }_{\mathrm{j}}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Ghost \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{isj}_{j}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ is \(\mathrm{sbj}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) @ prog-instrs zs
by clarsimp
from j-bound \({ }^{\prime \prime \prime}\)
have j-bound-ys: j < length ?ts-ys
by auto
from j -bound-ys neq-i-j
have ts-ys-j: ?ts-ys!j=(last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Ghost \(\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\) ys, is \({ }_{\mathrm{j}}{ }^{\prime}\), \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\right.\) dom \(\vartheta_{\text {sbj }}\) - read-tmps \(\left(\right.\) Write \(_{\text {sb }}\) True \(\left.\mathrm{a}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\) ), (), \(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \(_{\mathrm{sb}}\) ys \(\neq\{ \}\), acquired True ys (acquired True ? take-sb \(\mathcal{O}_{\mathrm{j}}\) ) \(\mathcal{R}_{\mathrm{j}}{ }^{\prime}\) )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is \({ }_{j}\) ]
have A'-unowned:
\(\forall \mathrm{i}<\) length ?ts-ys. \(\mathrm{j} \neq \mathrm{i} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{i}}\right)=\) map owned ?ts-ys!i in \(\left.\mathrm{A}^{\prime} \cap \mathcal{O}_{\mathrm{i}}=\{ \}\right)\)
apply cases
apply (fastforce simp add: Let-def is ssb \(^{\text {}}\) ) +
done
from \(\mathrm{a}^{\prime}\)-in \(\mathrm{a}^{\prime}-\mathrm{A}^{\prime} \mathrm{A}^{\prime}\)-unowned [rule-format, of i] neq-i-j i-bound \({ }^{\prime} \mathrm{A}-\mathrm{R}\)
show False
by (auto simp add: Let-def)
qed
qed
qed
\}
thus ?thesis
by (auto simp add: Let-def)
qed
have A-no-read-only-reads-by-others:
\(\forall \mathrm{j}<\) length ( \(m a p \mathcal{O}\)-sb \(\mathrm{ts}_{\mathrm{sb}}\) ). \(\mathrm{i} \neq \mathrm{j} \longrightarrow\)
\[
\left(\operatorname{let}\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)=\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right.
\]
in \(\mathrm{A} \cap\) read-only-reads (acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
\(\mathcal{O}_{j}\) )
\(\left(\right.\) dropWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{j}}\right)=\{ \}\right)\)
proof -
\{
fix \({ }^{j} \mathcal{O}_{j}\) sb \(_{j}\)
assume j -bound: \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\text {sb }}\) )
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{ts}_{\mathrm{sb}-\mathrm{j}}\) : \(\left(\operatorname{map} \mathcal{O}-\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}\right)!\mathrm{j}=\left(\mathcal{O}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}\right)\)
let ?take-sb \({ }_{j}=\left(\right.\) takeWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\mathrm{sb}}\right) \mathrm{sb}_{\mathrm{j}}\right)\)
let ? drop-sb \({ }_{j}=\left(\right.\) dropWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\text {sb }}\right) \mathrm{sb}_{\mathrm{j}}\right)\)
assume conflict: \(\mathrm{A} \cap\) read-only-reads (acquired True ?take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \(_{\mathrm{j}} \neq\{ \}\)
have False
proof -
from j-bound leq
have j-bound': j < length (map owned ts)
by auto
from j -bound have j -bound \({ }^{\prime \prime}: \mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\) by auto
from j-bound ' have j-bound \({ }^{\prime \prime \prime}: \mathrm{j}\) < length ts by simp

\section*{from conflict obtain \(\mathrm{a}^{\prime}\) where}

\section*{\(a^{\prime}-\mathrm{in}: \mathrm{a}^{\prime} \in \mathrm{A}\) and}
\(\mathrm{a}^{\prime}\)-in-j: \(\mathrm{a}^{\prime} \in\) read-only-reads (acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \({ }_{\mathrm{j}}\)
by auto
from ts-sim [rule-format, OF j-bound \({ }^{\prime \prime}\) [ \(\mathrm{ts}_{\mathrm{sb}}\)-j j-bound \({ }^{\prime \prime}\)
obtain \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{is}_{\mathrm{sbj}} \mathcal{D}_{\text {sbj }} \mathcal{D}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \vartheta_{\mathrm{sbj}}\) is \(\mathrm{S}_{\mathrm{j}}\) where
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{sbj}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\) and
suspends \({ }_{j}\) : suspends \({ }_{j}=\) ? drop-sb \(b_{j}\) and
\(\mathrm{is}_{\mathrm{j}}\) : instrs suspends \(\mathrm{s}_{\mathrm{j}} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{is}_{\mathrm{j}} @\) prog-instrs suspends \(\mathrm{j}_{\mathrm{j}}\) and
\(\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.\) outstanding-refs is-volatile-Write \(\left.{ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)\) and
\(\mathrm{ts}_{\mathrm{j}}: \mathrm{ts}!\mathrm{j}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{s}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
\(\left.\vartheta_{\text {sbj }}\right|^{\text {‘ }}\left(\right.\) dom \(\vartheta_{\text {sbj }}-\) read-tmps suspends \(\left.{ }_{j}\right),(), \mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\), release ?take-sb \(\left.{ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\right)\)
apply (cases \(\left.\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\right)\)
apply (force simp add: Let-def) done
from split-in-read-only-reads [OF a'-in-j [simplified suspends \({ }_{j}\) [symmetric]]]
obtain \(t\) v ys zs where
split-suspends \(_{\mathrm{j}}\) : suspends \(\mathrm{s}_{\mathrm{j}}=\) ys @ Read \(_{\mathrm{sb}}\) False \(\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\) (is suspends \({ }_{\mathrm{j}}=\) ? suspends)
and
\(\mathrm{a}^{\prime}\)-unacq: \(\mathrm{a}^{\prime} \notin\) acquired True ys (acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
by blast
from direct-memop-step.Ghost [where \(\vartheta=\vartheta_{\text {sb }}\) and \(\mathrm{m}=\) flush ?drop-sb m] have (Ghost A L R W\# is \(\mathrm{s}_{\mathrm{sb}}{ }^{\prime}\), \(\vartheta_{\mathrm{sb}},()\), flush ?drop-sb m, \(\mathcal{D}_{\mathrm{sb}}\), acquired True sb \(\mathcal{O}_{\text {sb }}\), release sb (dom \(\left.\mathcal{S}_{\text {sb }}\right) \mathcal{R}_{\text {sb }}\), share ?drop-sb \(\left.\mathcal{S}\right) \rightarrow\) \(\left(\mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},()\right.\), flush ? drop-sb \(\mathrm{m}, \mathcal{D}_{\mathrm{sb}}\), acquired True sb \(\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}\), augment-rels (dom (share ?drop-sb \(\mathcal{S})\) ) R (release sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) ), share ?drop-sb \(\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
from direct-computation.concurrent-step.Memop [OF
i-bound-ts' [simplified is sbb ] ts'-i [simplified is stb ] this [simplified is stb ]]
have store-step: (?ts', flush ?drop-sb m, share ?drop-sb \(\mathcal{S}) \Rightarrow_{\mathrm{d}}\) \(\left(? \mathrm{ts}^{\prime} \mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}, \vartheta_{\mathrm{sb}},(), \mathcal{D}_{\mathrm{sb}}\right.\right.\), acquired True sb \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\), augment-rels (dom (share ?drop-sb \(\mathcal{S})\) ) R (release sb (dom \(\left.\left.\mathcal{S}_{\text {sb }}\right) \mathcal{R}_{\text {sb }}\right)\) )],
(is \(-\Rightarrow_{\mathrm{d}}(? \mathrm{ts}-\mathrm{A}, ? \mathrm{~m}-\mathrm{A}, ?\) share-A \()\) )
    by ( \(\operatorname{simp}\) add: is \(_{\text {sb }}\) )
    from i-bound \({ }^{\prime}\) have i-bound \({ }^{\prime \prime}\) : i < length ?ts-A
        by simp
    from valid-program-history [OF j-bound \({ }^{\prime \prime}\) ts \(_{\text {sb-j }}\) ]
    have causal-program-history is \(_{\text {sbj }} \mathrm{sb}_{\mathrm{j}}\).
    then have cph: causal-program-history is sbj ?suspends
        apply -
        apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? \({ }^{\text {take- }} \mathrm{sb}_{\mathrm{j}}\) ] )
        apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
        apply (simp add: split-suspends \({ }_{j}\) )
        done
    from \(t s_{j}\) neq-i-j j-bound
    have ts-A-j: ?ts-A! \(=\left(\right.\) hd-prog \(p_{j}\left(y s @ R_{\text {ead }}\right.\) False a't v\# zs), is \({ }_{j}\),
        \(\left.\vartheta_{\text {sbj }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(y s @ \operatorname{Read}_{\text {sb }}\right.\) False a't v\# zs \(\left.)\right),(), \mathcal{D}_{\mathrm{j}}\),
        acquired True ? take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\), release ? take-sb \({ }_{\mathrm{j}}\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{j}}\) )
        by (simp add: split-suspends \({ }_{\mathrm{j}}\) )
    from j-bound \({ }^{\prime \prime \prime}\) i-bound' neq-i-j have j-bound \({ }^{\prime \prime \prime \prime}\) : \(\mathrm{j}<\) length ?ts-A
        by simp
    from valid-last-prog \(\left[\mathrm{OF}\right.\) j-bound \(\left.{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
    then
    have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) ?suspends \(=\mathrm{p}_{\mathrm{j}}\)
        apply -
        apply (rule last-prog-same-append [where sb=?take-sb \({ }_{\mathrm{j}}\) ])
        apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
        apply simp
        done
    from valid-reads [ OF j -bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
    have reads-consis: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
    from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing
\(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 j-bound \({ }^{\prime \prime}\)
    \(\mathrm{ts}_{\mathrm{sb}}\)-j reads-consis]
        have reads-consis-m: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{m}_{\mathrm{j}}\)
            by ( \(\operatorname{simp}\) add: m suspends \(\mathrm{f}_{\mathrm{j}}\) )
from outstanding-non-write-non-vol-reads-drop-disj [OF i-bound j-bound \({ }^{\prime \prime}\) neq-i-j ts \(_{\text {sb }}\)-i \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
have outstanding-refs is-Write \({ }_{\text {sb }}\) ?drop-sb \(\cap\) outstanding-refs is-non-volatile-Read \({ }_{\text {sb }}\) suspends \(_{\mathrm{j}}=\{ \}\)
by ( \(\operatorname{simp}\) add: suspends \({ }_{j}\) )
from reads-consistent-flush-independent [OF this reads-consis-m]
have reads-consis-flush-m: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) )
?m-A suspends \({ }^{2}\).
hence reads-consis-m-A-ys: reads-consistent True (acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) ?m-A
by (simp add: split-suspends \({ }_{\mathrm{j}}\) reads-consistent-append)
```

from valid-history [ OF j-bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have h -consis
history-consistent $\vartheta_{s b j}\left(\right.$ hd-prog $\mathrm{p}_{\mathrm{j}}\left(\right.$ ?take-sb ${ }_{\mathrm{j}} @$ suspends $\left.\mathrm{s}_{\mathrm{j}}\right)$ ) (?take-sb ${ }_{\mathrm{j}} @$ suspends $\mathrm{s}_{\mathrm{j}}$ )
apply (simp only: split-suspends $\mathrm{j}_{\mathrm{j}}$ [symmetric] suspends $\mathrm{s}_{\mathrm{j}}$ )
apply simp
done

```
    have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\)
proof -
    from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(_{\mathrm{j}} @\) ? drop-sb \(\left.\mathrm{b}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by \(\operatorname{simp}\)
    from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
    have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ? take-sb \(\mathrm{b}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    moreover
    have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{b}_{\mathrm{j}} @\) suspends \(\left._{\mathrm{j}}\right)\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\)
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
by (rule last-prog-hd-prog-append)
    ultimately show ?thesis
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    qed
from valid-write-sops [OF j-bound \({ }^{\prime \prime}\) ts \(\mathrm{ss}_{\mathrm{sb}}\)-j]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop
    by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and
    valid-sops-drop: \(\forall\) sop \(\in\) write-sops ys. valid-sop sop
        apply (simp only: write-sops-append )
        apply auto
        done
    from read-tmps-distinct [OF j-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{j}}\) ]
    have distinct-read-tmps (?take-sb \(@_{j}\) Suspends \(_{\mathrm{j}}\) )
        by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    then obtain
read-tmps-take-drop: read-tmps ?take-sb b \(_{\mathrm{j}} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
    distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
    apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
    apply (simp only: distinct-read-tmps-append)
    done
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\text {sbj }}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\) by (simp add: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis]
have no-vol-read: outstanding-refs is-volatile-Read sb \(_{\text {b }}\) ys \(=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \(_{j}\) )
from flush-store-buffer-append [
OF j-bound \({ }^{\prime \prime \prime \prime}\) is s \(_{\mathrm{j}}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] cph [simplified suspends \({ }_{\mathrm{j}}\) ]
ts-A-j [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] refl lp [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] reads-consis-m-A-ys
hist-consis' [simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop distinct-read-tmps-drop
[simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\) ?share-A]
obtain is \({ }_{j}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}^{\prime}\) : instrs ( Read \(_{\mathrm{sb}}\) False \(\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\) ) @ \(\mathrm{is}_{\mathrm{sbj}}=\)
\(\mathrm{is}_{\mathrm{j}}\) @ prog-instrs ( \(\operatorname{Read}_{\mathrm{sb}}\) False a't v \# zs) and
steps-ys: (?ts-A, ?m-A, ?share-A) \(\Rightarrow_{\mathrm{d}}{ }^{*}\)
(?ts-A[j:= (last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Ghost \(\left.\left._{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\right)\right)\) ys,
\[
\mathrm{isj}_{\mathrm{j}}{ }^{\prime},
\]
\(\left.\vartheta_{\text {sbj }}\right|^{‘}\left(\operatorname{dom} \vartheta_{\text {sbj }}-\right.\) read-tmps \(\left(\operatorname{Read}_{\text {sb }}\right.\) False a'tv \# zs) \(),()\),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired
True ? take-sb \(\left.\mathrm{j}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right), \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) ) ],
flush ys?m-A,
share ys ?share-A)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?\) shared-ys) \()\)
by (auto)
note conflict-computation \(=\) rtranclp-trans [OF rtranclp-r-rtranclp [OF steps-flush-sb, OF store-step] steps-ys]
from cph
have causal-program-history is sbj \(^{\text {( }}\) (ys @ \(\left[\operatorname{Read}_{\text {sb }}\right.\) False a't v]) @ zs)
by simp
from causal-program-history-suffix [OF this]
have \(\mathrm{cph}^{\prime}\) : causal-program-history is \(\mathrm{sbj}_{\mathrm{sbj}} \mathrm{zs}\).
interpret causal \({ }_{j}\) : causal-program-history is \(_{\text {sbj }} \mathrm{zs}\) by (rule cph')
from causal \({ }_{j}\).causal-program-history [of [], simplified, OF refl] is \({ }_{\mathrm{j}}{ }^{\prime}\)
obtain is \({ }^{\text {j }}{ }^{\prime \prime}\)
where \(\mathrm{is}_{\mathrm{j}}\) ' \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}=\) Read False \(\mathrm{a}^{\prime} \mathrm{t} \# \mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) and
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) : instrs zs @ \(\mathrm{is}_{\text {sbj }}=\mathrm{is}_{\mathrm{j}}{ }^{\prime \prime}\) @ prog-instrs zs
by clarsimp
from j-bound \({ }^{\prime \prime \prime}\)
have j-bound-ys: j < length ?ts-ys
by auto
from j-bound-ys neq-i-j
```

have ts-ys-j: ?ts-ys!j=(last-prog (hd-prog $\mathrm{p}_{\mathrm{j}}\left(\operatorname{Read}_{\mathrm{sb}}\right.$ False a't v\# zs)) ys, is ${ }_{\mathrm{j}}{ }^{\prime}$,
$\left.\vartheta_{\mathrm{sbj}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{sbj}}-\right.$ read-tmps $\left(\operatorname{Read}_{\mathrm{sb}}\right.$ False $\left.\left.\mathrm{a}^{\prime} \mathrm{t} \mathrm{v} \# \mathrm{zs}\right)\right),()$,
$\mathcal{D}_{\mathrm{j}} \vee$ outstanding-refs is-volatile-Write $_{\text {sb }}$ ys $\neq\{ \}$,
acquired True ys (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ), $\mathcal{R}_{\mathrm{j}}$ )
by auto
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe-delayed (?ts-ys,?m-ys,?shared-ys).
from safe-delayedE [OF this j-bound-ys ts-ys-j, simplified is ${ }_{j}$ ']
have $\mathrm{a}^{\prime} \in$ acquired True ys (acquired True ? take-sb $\mathcal{O}_{\mathrm{j}}$ ) $\vee$
$\mathrm{a}^{\prime} \in$ read-only (share ys (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ))
apply cases
apply (auto simp add: Let-def $\mathrm{is}_{\mathrm{sb}}$ )
done
with a'-unacq
have a'-ro: a' $\in$ read-only (share ys (share ? drop-sb $\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ))
by auto
from a'-in
have a'-not-ro: a' $\notin$ read-only (share ? drop-sb $\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ )
by (auto simp add: in-read-only-convs)
have $a^{\prime} \in \mathcal{O}_{j} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
proof -
\{
assume a-notin: $\mathrm{a}^{\prime} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
from weak-sharing-consis [ OF j -bound ${ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}$ ]
have weak-sharing-consistent $\mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$.
with weak-sharing-consistent-append [of $\mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
have weak-sharing-consistent (acquired True ? take-sb ${ }_{j} \mathcal{O}_{j}$ ) suspends ${ }_{j}$
by (auto simp add: suspends $\mathrm{j}_{\mathrm{j}}$ )
with split-suspends ${ }_{j}$
have weak-consis: weak-sharing-consistent (acquired True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$ ) ys
by (simp add: weak-sharing-consistent-append)
from all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have all-acquired ys $\subseteq$ all-acquired $\mathrm{sb}_{j}$
apply (clarsimp)
apply (clarsimp simp add: suspends ${ }_{j}\left[\right.$ ssymmetric] $^{\text {split-suspends }}{ }_{j}$ all-acquired-append)
done
with a-notin acquired-takeWhile-non-volatile-Write ${ }_{\text {sb }}\left[\right.$ of $\left.\operatorname{sb}_{j} \mathcal{O}_{j}\right]$
all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have $\mathrm{a}^{\prime} \notin$ acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\mathcal{O}_{\mathrm{j}} \cup$ all-acquired ys
by auto
from read-only-share-unowned [OF weak-consis this a'-ro]
have $\mathrm{a}^{\prime} \in$ read-only (share ?drop-sb $\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ).
with $a^{\prime}$-not-ro have False
by auto
\}

```
```

        thus ?thesis by blast
    qed
    moreover
    from A-unaquired-by-others [rule-format, OF - neq-i-j] tssb-j j-bound
    have A \cap all-acquired sb
        by (auto simp add: Let-def)
    moreover
    from A-unowned-by-others [rule-format, OF - neq-i-j] ts tsb-j j-bound
    have A \cap \mathcal{O}
        by (auto simp add: Let-def dest: all-shared-acquired-in)
    moreover note a'-in
    ultimately
    show False
        by auto
    qed
    }
thus ?thesis
by (auto simp add: Let-def)
qed
have valid-own': valid-ownership }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{tss}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ ]
have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mathrm{ (sb @ [Ghostsb A L R W])
by (auto simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by ( }\operatorname{simp}\mathrm{ add: }\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}
qed
next
show outstanding-volatile-writes-unowned-by-others ts }\mp@subsup{\textrm{s}}{\mathbf{s}}{}\mp@subsup{}{}{\prime
proof (unfold-locales)

```

```

    assume i }\mp@subsup{i}{1}{}\mathrm{ -bound: i
    assume j-bound: j < length tssmb
    assume i }\mp@subsup{\textrm{i}}{1}{}-\textrm{j}:\mp@subsup{\textrm{i}}{1}{}\not=
    assume ts-\mp@subsup{i}{1}{}: ts\mp@subsup{s}{sb}{}\mp@subsup{}{}{\prime}!\mp@subsup{i}{1}{}=(\mp@subsup{p}{1}{},\mp@subsup{\textrm{is}}{1}{},\mp@subsup{\textrm{xs}}{1}{},\mp@subsup{\textrm{sb}}{1}{},\mp@subsup{\mathcal{D}}{1}{},\mp@subsup{\mathcal{O}}{1}{},\mp@subsup{\mathcal{R}}{1}{})
    assume ts-j: ts sb !! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
    show (}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
    proof (cases \mp@subsup{i}{1}{}=i)
        case True
        with i }\mp@subsup{\textrm{i}}{1}{}-\textrm{j}\mathrm{ have i-j: i}=\textrm{j
        by simp
        from j-bound have j-bound': j < length tssb
            by (simp add: ts sb 
        hence j-bound '': j < length (map owned tssb)
            by simp
    ```
from ts-j i-j have ts-j \({ }^{\prime}: \operatorname{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from outstanding-volatile-writes-unowned-by-others
[OF i-bound j-bound \({ }^{\prime} \mathrm{i}-\mathrm{j} \mathrm{ts}_{\mathrm{sb}}\)-i ts-j\({ }^{\prime}\) ]
have \(\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right) \cap\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}=\{ \}\) 。
with ts- \(\mathrm{i}_{1}\) True i-bound show ?thesis
by (clarsimp simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) outstanding-refs-append
acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) )

\section*{next}
case False
note \(\mathrm{i}_{1}-\mathrm{i}=\) this
from \(\mathrm{i}_{1}\)-bound have \(\mathrm{i}_{1}\)-bound \({ }^{\prime}\) : \(\mathrm{i}_{1}<\) length \(\mathrm{ts}_{\text {sb }}\)
by (simp add: \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) )
from ts- \(\mathrm{i}_{1}\) False have ts- \(\mathrm{i}_{1}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}\) )
show ?thesis
proof (cases \(\mathrm{j}=\mathrm{i}\) )
case True
from \(i_{1}\)-bound \({ }^{\prime}\)
have \(\mathrm{i}_{1}\)-bound \({ }^{\prime \prime}: \mathrm{i}_{1}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
by simp
from outstanding-volatile-writes-unowned-by-others
[OF \(\mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{i}\)-bound \(\mathrm{i}_{1}-\mathrm{i}\) ts- \(\mathrm{i}_{1}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}\) - i ]
have \(\left(\mathcal{O}_{\text {sb }} \cup\right.\) all-acquired sb\() \cap\) outstanding-refs is-volatile-Write sb \(_{\text {sb }} \operatorname{sb}_{1}=\{ \}\).
moreover
from A-unused-by-others [rule-format, OF - False [symmetric]] False ts-i \(\mathrm{i}_{1} \mathrm{i}_{1}\)-bound
have \(\mathrm{A} \cap\) outstanding-refs is-volatile-Write \({ }_{s b} \mathrm{sb}_{1}=\{ \}\)
by (auto simp add: Let-def \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
ultimately
show ?thesis
using ts-j True ts sb \(^{\prime}\)
by (auto \(\operatorname{simp}\) add: i-bound \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) all-acquired-append)
next
case False
from j-bound have \(j\)-bound \({ }^{\prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ts-j False have \(\mathrm{ts}^{\mathrm{j}} \mathrm{j}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from outstanding-volatile-writes-unowned-by-others
[OF \(\mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{j}\)-bound \({ }^{\prime} \mathrm{i}_{1}-\mathrm{j}\) ts- \(\mathrm{i}_{1}{ }^{\prime}\) ts-j\({ }^{\prime}\) ]
show \(\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right) \cap\) outstanding-refs is-volatile-Write \(\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{1}=\{ \}\).
qed
qed
qed
next
show read-only-reads-unowned \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)

\section*{proof}
fix n m
fix \(\mathrm{p}_{\mathrm{n}} \operatorname{is}_{\mathrm{n}} \mathcal{O}_{\mathrm{n}} \mathcal{R}_{\mathrm{n}} \mathcal{D}_{\mathrm{n}} \vartheta_{\mathrm{n}} \mathrm{sb}_{\mathrm{n}} \mathrm{p}_{\mathrm{m}}\) is \(_{\mathrm{m}} \mathcal{O}_{\mathrm{m}} \mathcal{R}_{\mathrm{m}} \mathcal{D}_{\mathrm{m}} \vartheta_{\mathrm{m}} \mathrm{sb}_{\mathrm{m}}\)
assume \(n\)-bound: \(\mathrm{n}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
and m-bound: \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
and neq- \(\mathrm{n}-\mathrm{m}: \mathrm{n} \neq \mathrm{m}\)
and nth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
and mth: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\)
from n-bound have \(n\)-bound': \(\mathrm{n}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}\) )
from \(m\)-bound have \(m\)-bound': \(m<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
show \(\left(\mathcal{O}_{\mathrm{m}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{m}}\right) \cap\)
read-only-reads (acquired True (takeWhile (Not o is-volatile-Write \({ }_{s b}\) ) \(\operatorname{sb}_{\mathrm{n}}\) ) \(\mathcal{O}_{\mathrm{n}}\) )
\(\left(\right.\) dropWhile \(^{(N o t} \circ\) is-volatile-Write \(\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{n}}\right)=\)
\{\}
proof (cases m=i)
case True
with neq-n-m have neq-n-i: \(\mathrm{n} \neq \mathrm{i}\)
by auto
with n -bound nth i-bound have \(\mathrm{nth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth \({ }^{\prime}\) ts \(_{s_{s b}-i}\) ]
moreover
from A-no-read-only-reads-by-others [rule-format, OF - neq-n-i [symmetric]] n-bound \({ }^{\prime}\) \(n t h{ }^{\prime}\)
have \(\mathrm{A} \cap\) read-only-reads (acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{n}}\) ) \(\mathcal{O}_{n}\) )
\(\left(\right.\) dropWhile \(^{(N o t} \circ\) is-volatile-Write \(\left.\left.{ }_{s b}\right) \operatorname{sb}_{\mathrm{n}}\right)=\)
\{\}
by auto
ultimately
show ?thesis
using True \(\mathrm{ts}_{\mathrm{sb}-\mathrm{i}} \mathrm{nth}{ }^{\prime} \mathrm{mth} \mathrm{n}\)-bound \({ }^{\prime} \mathrm{m}\)-bound \({ }^{\prime}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) all-acquired-append)
next
case False
note neq-m-i \(=\) this
with m-bound mth i-bound have \(\mathrm{mth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\) by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
show ?thesis
proof (cases \(\mathrm{n}=\mathrm{i}\) )
case True
note read-only-reads-unowned [OF i-bound m-bound' neq-m-i [symmetric] \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\)
mth \({ }^{\prime}\)
then show ?thesis
using True neq-m-i \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) nth mth n -bound \({ }^{\prime} \mathrm{m}\)-bound \({ }^{\prime}\)
apply (case-tac outstanding-refs (is-volatile-Write \({ }_{\text {sb }}\) ) sb \(=\{ \}\) )
apply (clarsimp simp add: outstanding-vol-write-take-drop-appends
acquired-append read-only-reads-append \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\right)+\)
done
next
case False
with n-bound nth i-bound have \(n t h ': \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}\) )
from read-only-reads-unowned [OF n-bound' m-bound' neq-n-m nth' mth' False neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
next
show ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have \(\forall \mathrm{j}<\) length \(^{\text {ts }}{ }_{\text {sb }} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
\(\left(\right.\) let \(\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\)
in \(\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired \(\left.\mathrm{sb}{ }^{\prime}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\right)\)
proof -
\{
fix j \(p_{j} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
assume ts \(_{\text {sb }}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
have \(\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb \() \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\)
proof -
\{
fix \(a^{\prime}\)
assume \(\mathrm{a}^{\prime}\)-in-i: \(\mathrm{a}^{\prime} \in\left(\mathcal{O}_{\text {sb }} \cup\right.\) all-acquired sb \()\)
assume \(\mathrm{a}^{\prime}\)-in- \(\mathrm{j}: \mathrm{a}^{\prime} \in\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)\)
have False
proof -
from \(a^{\prime}\)-in-i have \(a^{\prime} \in\left(\mathcal{O}_{s b} \cup\right.\) all-acquired \(\left.s b\right) \vee a^{\prime} \in A\)
by (simp add: sb' all-acquired-append)
then show False
proof
assume \(\mathrm{a}^{\prime} \in\left(\mathcal{O}_{\text {sb }} \cup\right.\) all-acquired sb\()\)
with ownership-distinct [OF i-bound j-bound neq-i-j ts \(\mathrm{s}_{\mathbf{s b}}-\mathrm{i} \mathrm{ts}_{\mathbf{s b}}-\mathrm{j}\) ] \(\mathrm{a}^{\prime}\)-in-j
show ?thesis
by auto
next
assume \(a^{\prime} \in A\)
moreover
have j -bound': \(\mathrm{j}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
using j -bound by auto
from A-unowned-by-others [rule-format, OF - neq-i-j] ts sbb-j \(^{\text {j }}\)-bound
obtain \(\mathrm{A} \cap\) acquired True (takeWhile (Not \(\circ\) is-volatile-Write sb ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}}=\{ \}\) and \(\mathrm{A} \cap\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \(\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}_{\mathrm{j}}\) ) \(=\{ \}\)
by (auto simp add: Let-def)
moreover
from A-unaquired-by-others [rule-format, OF - neq-i-j] ts \(\mathrm{ss}_{\mathrm{sb}}-\mathrm{j} j\)-bound
have \(\mathrm{A} \cap\) all-acquired \(\mathrm{sb}_{\mathrm{j}}=\{ \}\)
```

    by auto
        ultimately
        show ?thesis
    using a'-in-j
    by (auto dest: all-shared-acquired-in)
        qed
    qed
    }
then show ?thesis by auto
qed
}
then show ?thesis by (fastforce simp add: Let-def)
qed
from ownership-distinct-nth-update [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ this]
show ?thesis by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{\prime}\mp@subsup{}{}{\prime}\textrm{sb}\mp@subsup{}{}{\prime}
qed
qed
have valid-hist': valid-history program-step ts ssb}\mp@subsup{}{}{\prime
proof -
from valid-history [OF i-bound ts}\mp@subsup{\textrm{ts}}{\mathbf{s}-}{}-\textrm{i}
have history-consistent }\mp@subsup{\vartheta}{\textrm{sb}}{}\mathrm{ (hd-prog p}\mp@subsup{\textrm{p}}{\textrm{sb}}{}\textrm{sb})\textrm{sb}
with valid-write-sops [OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ ]
valid-implies-valid-prog-hd [OF i-bound ts }\mp@subsup{\textrm{s}}{\mathbf{sb}}{}-\textrm{i}\mathrm{ valid]
have history-consistent }\mp@subsup{\vartheta}{\mathrm{ sb }}{}(\mathrm{ hd-prog p pb (sb@[Ghost sb A L R W]))
(sb@ [Ghost sb A L R W])
apply -
apply (rule history-consistent-appendI)
apply (auto simp add: hd-prog-append-Ghostsb
done
from valid-history-nth-update [OF i-bound this]
show ?thesis by (simp add: ts sbb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{\prime}
qed
have valid-reads': valid-reads m
proof -
from valid-reads [OF i-bound ts }\mp@subsup{\textrm{ts}}{\textrm{sb}}{}-\textrm{i}\mathrm{ ]
have reads-consistent False }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{\textrm{m}}{\textrm{sb}}{}\textrm{sb}
from reads-consistent-snoc-Ghost sb [OF this]
have reads-consistent False }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{m}{sb}{}\mathrm{ (sb @ [Ghost sb A L R W]).
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}
qed
have valid-sharing': valid-sharing }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound ts ssb
have non-volatile-writes-unshared S S sb (sb @ [Ghost sb A L R W])
by (auto simp add: non-volatile-writes-unshared-append)

```
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have consis': sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}\).
from A-shared-owned
have \(\mathrm{A} \subseteq\) dom (share ?drop-sb \(\mathcal{S}) \cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\)
by (simp add: sharing-consistent-append acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) )
moreover have \(\operatorname{dom}\) (share ?drop-sb \(\mathcal{S}\) ) \(\subseteq \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
proof
fix \(a^{\prime}\)
assume \(\mathrm{a}^{\prime}\)-in: \(\mathrm{a}^{\prime} \in \operatorname{dom}\) (share ?drop-sb \(\mathcal{S}\) )
from share-unshared-in [ \(\mathrm{OF} \mathrm{a}^{\prime}\)-in]
show \(\mathrm{a}^{\prime} \in \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\) (share sb \(\mathcal{S}_{\mathrm{sb}}\) )
proof
assume \(\mathrm{a}^{\prime} \in \operatorname{dom}\) (share ?drop-sb Map.empty)
from share-mono-in [OF this] share-append [of ?take-sb ?drop-sb]
have \(\mathrm{a}^{\prime} \in \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) )
by auto
thus? thesis
by simp
next
assume \(\mathrm{a}^{\prime} \in \operatorname{dom} \mathcal{S} \wedge \mathrm{a}^{\prime} \notin\) all-unshared ?drop-sb thus ?thesis by auto
qed
qed
ultimately
have A-subset: A \(\subseteq \operatorname{dom} \mathcal{S} \cup \operatorname{dom}\left(\right.\) share sb \(\mathcal{S}_{\text {sb }}\) ) \(\cup\) acquired True \(\operatorname{sb} \mathcal{O}_{\text {sb }}\)
by auto
have \(\mathrm{A} \subseteq \operatorname{dom}\left(\right.\) share sb \(\left.\mathcal{S}_{\mathrm{sb}}\right) \cup\) acquired True \(\mathrm{sb} \mathcal{O}_{\text {sb }}\)
proof -
\{
fix x
assume \(x-A: x \in A\)
have \(\mathrm{x} \in \operatorname{dom}\) (share sb \(\mathcal{S}_{\text {sb }}\) ) \(\cup\) acquired True sb \(\mathcal{O}_{\text {sb }}\) proof \{
assume \(\mathrm{x} \in \operatorname{dom} \mathcal{S}\)
from share-all-until-volatile-write-share-acquired [OF «sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉
i-bound \(\mathrm{ts}_{\text {sb }}\)-i this [simplified \(\mathcal{S}\) ]]
A-unowned-by-others x-A
have ?thesis
by (fastforce simp add: Let-def)
\}
with A-subset show ? thesis using x-A by auto
qed
\}
thus ?thesis by blast
qed
with consis' L-subset A-R R-acq
have sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}\) (sb @ [Ghost \({ }_{\text {sb }} \mathrm{A}\) L R W])
by (simp add: sharing-consistent-append acquired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\) )
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathbf{s b} \text {-i }}\) ]
]
show read-only-unowned \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) i subset-refl]
show unowned-shared \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next

have no-write-to-read-only-memory \(\mathcal{S}_{\text {sb }}\left(\mathrm{sb} @\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A} \mathrm{L} R \mathrm{~W}\right]\right)\)
by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}{ }^{\text {sb }}{ }^{\prime}\) )
qed
have tmps-distinct': tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have distinct-load-tmps is sbb \(^{\prime}{ }^{\prime}\) by (simp add: \(\mathrm{is}_{\mathbf{s b}}\) )
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) by (simp add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) ) next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have distinct-read-tmps (sb @ [Ghost sb \(_{\text {A L R W] }}\) )
by (auto simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{sb}^{\prime}\) ) next
from load-tmps-read-tmps-distinct [OF i-bound ts \(_{s b}-\mathrm{i}\) ]
have load-tmps is sb \(^{\prime} \cap\) read-tmps (sb @ \(\left.\left[\mathrm{Ghost}_{\text {sb }} \mathrm{ALRW}\right]\right)=\{ \}\) by (auto simp add: read-tmps-append is sb \(_{\text {sb }}\) )
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
 qed have valid-sops': valid-sops \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) proof -
from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
obtain
valid-store-sops \({ }^{\prime}: \forall\) sop \(\in\) store-sops is \({ }_{\text {sb }}{ }^{\prime}\). valid-sop sop
by (auto simp add: is sb )
from valid-write-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text { - }}\) ]
have valid-write-sops': \(\forall\) sop \(\in\) write-sops (sb@ [Ghost \({ }_{\text {sb }}\) A L R W]).
valid-sop sop
by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound valid-write-sops' valid-store-sops \({ }^{\text {T }}\) ]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\) ) qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}\) ]
obtain
dd-is: data-dependency-consistent-instrs (dom \(\vartheta_{\mathrm{sb}}{ }^{\prime}\) ) is sb \(^{\prime}\)
by (auto simp add: is sb \(\vartheta_{\text {sb }}\) )
from load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
 by (auto simp add: write-sops-append is sb )
from valid-data-dependency-nth-update [OF i-bound dd-is this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{sb}^{\prime}\) )
qed
have load-tmps-fresh \({ }^{\prime}\) : load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have load-tmps is sab \(^{\prime} \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\) by (auto simp add: is sb \(_{\text {b }}\) )
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime}\) ) qed have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from clean-no-outstanding-volatile-Write \({ }_{s b}\) [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
 by (auto simp add: outstanding-refs-append)
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{D}_{\mathrm{sb}}\) )
qed
have valid-program-history \({ }^{\prime}\) : valid-program-history \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) proof -
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have causal-program-history is \(\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}\).
then have causal': causal-program-history is s \(_{\text {sb }}{ }^{\prime}\) (sb@[Ghost \({ }_{\text {sb }}\) A L R W])
```

    by (auto simp: causal-program-history-Ghost \(\mathrm{is}_{\mathrm{sb}}\) )
    from valid-last-prog [ OF i-bound $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ ]
have last-prog $\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}$.
hence last-prog $\mathrm{p}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\right.\right.$ Ghost $_{\mathrm{sb}}$ A L R W] $)=\mathrm{p}_{\mathrm{sb}}$
by (simp add: last-prog-append-Ghost ${ }_{\text {sb }}$ )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by (simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}$ )
qed
show ?thesis
proof (cases outstanding-refs is-volatile-Write ${ }_{\mathbf{s b}} \mathrm{sb}=\{ \}$ )
case True
from True have flush-all: takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $=\mathrm{sb}$
by (auto simp add: outstanding-refs-conv)
from True have suspend-nothing: dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}=[]$
by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends $=[]$
by (simp add: suspends)
from suspends-empty is-sim have is: is $=$ Ghost A L R W\# is ${ }_{\text {sb }}{ }^{\prime}$
by (simp add: is stb )
with suspends-empty ts-i
have ts-i: ts!i $=\left(\mathrm{p}_{\mathrm{sb}}\right.$, Ghost A L R W\# is $\mathrm{sb}^{\prime}$,
$\vartheta_{\mathrm{sb}},(), \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$,release ?take-sb (dom $\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}$ )
by simp
from direct-memop-step.Ghost
have (Ghost A L R W\# is ${ }_{\text {sb }}{ }^{\prime}$,
$\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}}$,
release ?take-sb $\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow$
(is $\mathrm{s}_{\mathrm{sb}}{ }^{\prime}$,
$\vartheta_{\mathrm{sb}},(), \mathrm{m}, \mathcal{D}$, acquired True ?take-sb $\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}$,
augment-rels (dom $\mathcal{S}) \mathrm{R}$ (release ?take-sb (dom $\left.\mathcal{S}_{\text {sb }}\right) \mathcal{R}_{\text {sb }}$ ),
$\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.
from direct-computation.concurrent-step.Memop [OF i-bound' ts-i this]
have $(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}$
$\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.$,
$\vartheta_{\mathrm{sb}},(), \mathcal{D}$, acquired True ? take-sb $\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}$,
augment-rels (dom $\mathcal{S}) \mathrm{R}$ (release ?take-sb (dom $\left.\left.\mathcal{S}_{\text {sb }}\right) \mathcal{R}_{\text {sb }}\right)$ )],
$\left.\mathrm{m}, \mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$.

```

\section*{moreover}
```

from suspend-nothing
have suspend-nothing ${ }^{\prime}$ : (dropWhile (Not $\circ$ is-volatile-Write $\left.{ }_{\text {sb }}\right)$ sb $\left.{ }^{\prime}\right)=[]$

```
by (simp add: \(\mathrm{sb}^{\prime}\) )
```

have all-shared-A: $\forall \mathrm{j}$ p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta$ sb. $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow$
$\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=(\mathrm{p}$, is, $\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow$
all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) sb) $\cap \mathrm{A}=\{ \}$
proof -
\{
fix j $\mathrm{p}_{\mathrm{j}}$ is $\mathrm{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}$
assume neq-i-j: $\mathrm{i} \neq \mathrm{j}$
assume $\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
assume x-shared: $\mathrm{x} \in$ all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$ )
assume $x-A: x \in A$
have False
proof -
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
have all-shared $\mathrm{sb}_{\mathrm{j}} \subseteq$ all-acquired $\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}$.
moreover have all-shared (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ ) $\subseteq$ all-shared
$\mathrm{sb}_{\mathrm{j}}$
using all-shared-append [of (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) $\mathrm{sb}_{\mathrm{j}}$ )
(dropWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{sb}_{\mathrm{j}}$ )]
by auto
moreover
from A-unaquired-by-others [rule-format, OF - neq-i-j] jth j-bound
have $\mathrm{A} \cap$ all-acquired $\mathrm{sb}_{\mathrm{j}}=\{ \}$ by auto
moreover

```
        from A-unowned-by-others [rule-format, OF - neq-i-j] jth j-bound
        have \(A \cap \mathcal{O}_{j}=\{ \}\)
    by (auto dest: all-shared-acquired-in)
        ultimately
        show False
    using x -A x -shared
    by blast
        qed
    \}
    thus ?thesis by blast
qed
hence all-shared-L: \(\forall \mathrm{j}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow\)
    \(\mathrm{ts}_{\text {sb }}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
    all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cap \mathrm{L}=\{ \}\)
    using L-subset by blast
        have all-shared-A: \(\forall \mathrm{j}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb. \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow\)
```

            ts
            all-shared (takeWhile (Not ○ is-volatile-Write sb
        proof -
    {
    fix j p pisj }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\vartheta}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\textrm{x
    assume j-bound: j < length ts sb
    assume jth: ts scb
        assume neq-i-j: i f= j
    assume x-shared: x }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write (sb
    assume x-A: x }\in\textrm{A
    have False
    proof -
        from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
        have all-shared sb
        moreover have all-shared (takeWhile (Not ○ is-volatile-Write sb
    using all-shared-append [of (takeWhile (Not o is-volatile-Write 
    (dropWhile (Not o is-volatile-Write sb
    by auto
        moreover
        from A-unaquired-by-others [rule-format, OF - neq-i-j] jth j-bound
        have }\textrm{A}\cap\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{j}{}={}\mathrm{ by auto
        moreover
    from A-unowned-by-others [rule-format, OF - neq-i-j] jth j-bound
    have }\textrm{A}\cap\mp@subsup{\mathcal{O}}{j}{}={
            by (auto dest: all-shared-acquired-in)
        ultimately
        show False
    using x-A x-shared
    by blast
    qed
    }
thus ?thesis by blast
qed
hence all-shared-L: }\forall\textrm{j p}\mathrm{ is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. j < length tssb \longrightarrow m f j \longrightarrow
ts
all-shared (takeWhile (Not ○ is-volatile-Write sb})\textrm{sb})\cap\textrm{L}={
using L-subset by blast
have all-unshared-R: }\forall\textrm{j}\mathrm{ p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. j < length ts sb \longrightarrow i f = j
tssb ! j = (p, is,\vartheta, sb, \mathcal{D, O},\mathcal{R})\longrightarrow
all-unshared (takeWhile (Not \circ is-volatile-Write sb) sb) \capR={}
proof -
{

```

```

    assume j-bound: j < length ts sb
    ```
\(\mathrm{sb}_{\mathrm{j}}\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathcal{\vartheta}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume \(x\)-unshared: \(x \in\) all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
assume \(x-R: x \in R\)
have False
proof -
from unshared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
have all-unshared \(\mathrm{sb}_{\mathrm{j}} \subseteq\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\).
moreover have all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ) \(\subseteq\)
all-unshared \(\mathrm{sb}_{\mathrm{j}}\)
using all-unshared-append [of (takeWhile (Not o is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
(dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \({ }_{\mathrm{j}}\) )]
by auto
moreover
note ownership-distinct [OF i-bound j-bound neq-i-j \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} j \mathrm{jth}\) ]
ultimately
show False
using R-acq x-R x-unshared acquired-all-acquired [of True sb \(\mathcal{O}_{\mathrm{sb}}\) ] by blast
qed
\}
thus ?thesis by blast
qed
have all-acquired-R: \(\forall \mathrm{j}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} . \mathrm{j}<\) length \(^{\mathrm{ts}} \mathrm{s}_{\mathrm{sb}} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow\)
\(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=(\mathrm{p}\), is, \(\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cap \mathrm{R}=\{ \}\)
proof -
\{
fix j \(\mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume x -acq: \(\mathrm{x} \in\) all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \(\mathrm{sb}_{\mathrm{bb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
assume \(x-R: x \in R\)
have False
proof -
from x -acq have \(\mathrm{x} \in\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
using all-acquired-append [of takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{j}\)
dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ]
by auto
moreover
note ownership-distinct [OF i-bound j-bound neq-i-j ts \(\mathrm{s}_{\mathrm{sb}}-\mathrm{i} j\) jth]
ultimately
show False
using R-acq x-R acquired-all-acquired [of True sb \(\mathcal{O}_{\text {sb }}\) ]
```

by blast
qed
}
thus ?thesis by blast
qed
have all-shared-R: }\forall\textrm{j}\mathrm{ p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb. j < length ts sb \longrightarrow i\not= j
ts
all-shared (takeWhile (Not \circ is-volatile-Write }\mp@subsup{}{sb}{}\mathrm{ ) sb) }\cap\textrm{R}={
proof -
{

```

```

    assume j-bound: j < length ts sb
    assume jth: ts tsb
            assume neq-i-j: i }=\textrm{j
    assume x-shared: x }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write sb) sb j)
    assume x-R: x }\in
    have False
    proof -
        from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
        have all-shared sbb
        moreover have all-shared (takeWhile (Not \circ is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{})\subseteq\mathrm{ all-shared
    sbj
using all-shared-append [of (takeWhile (Not o is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{b}}{}\mathrm{ ) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
(dropWhile (Not o is-volatile-Write sb) sb j}\mathrm{ )]
by auto
moreover
note ownership-distinct [OF i-bound j-bound neq-i-j ts sb-i jth]
ultimately
show False
using R-acq x-R x-shared acquired-all-acquired [of True sb }\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{}\mathrm{ ]
by blast
qed
}
thus ?thesis by blast
qed
note share-commute =
share-all-until-volatile-write-append-Ghostsb [OF True <ownership-distinct ts sb
«sharing-consis }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mathrm{ 〉
i-bound ts sb-i all-shared-L all-shared-A all-acquired-R all-unshared-R all-shared-R]
from D
have \mathcal{D}
by (auto simp: outstanding-refs-append)

```
have \(\forall \mathrm{a} \in \mathrm{R} .\left(\mathrm{a} \in\left(\operatorname{dom}\left(\operatorname{share} \operatorname{sb} \mathcal{S}_{\text {sb }}\right)\right)\right)=(\mathrm{a} \in \operatorname{dom} \mathcal{S})\)
proof -
fix a
assume \(a-R: a \in R\)
have \(\left(\mathrm{a} \in\left(\operatorname{dom}\left(\operatorname{share} \mathrm{sb} \mathcal{S}_{\mathrm{sb}}\right)\right)\right)=(\mathrm{a} \in \operatorname{dom} \mathcal{S})\)
proof -
from a-R R-acq acquired-all-acquired [of True sb \(\mathcal{O}_{\text {sb }}\) ]
have \(\mathrm{a} \in \mathcal{O}_{\mathrm{sb}} \cup\) all-acquired sb
by auto
from share-all-until-volatile-write-thread-local' \({ }^{[O F}\) ownership-distinct-ts \(s_{s b}\) sharing-consis- \(\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\)-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) this] suspend-nothing
show ?thesis by (auto simp add: domIff \(\mathcal{S}\) )
qed
\}
then show ?thesis by auto
qed
from augment-rels-shared-exchange [OF this]
have rel-commute:
augment-rels \((\operatorname{dom} \mathcal{S}) \mathrm{R}\) (release sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)=\)
release (sb @ [Ghost \({ }_{\text {sb }}\) A L R W]) (dom \(\mathcal{S}_{\mathrm{sb}}\) ) \(\mathcal{R}_{\mathrm{sb}}\)
by (clarsimp simp add: release-append \(\mathcal{S}_{\text {sb }}\) )

```

    (ts[i := ( }\mp@subsup{\textrm{p}}{\textrm{sb}}{},\mp@subsup{\textrm{is}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}
    \vartheta sb
                augment-rels (dom S S R (release ?take-sb (dom }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{})\mp@subsup{\mathcal{R}}{\mathrm{ sb }}{})\mathrm{ )],
                m,\mathcal{S}}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
    apply (rule sim-config.intros)
                apply (simp add: m ts tsb}\mp@subsup{}{}{\prime
    flush-all-until-volatile-write-append-Ghost-commute [OF i-bound ts tsb-i])
apply (clarsimp simp add: }\mathcal{S}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\mp@subsup{\textrm{sb}}{}{\prime}}{}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mathrm{ share-commute)
using leq
apply (simp add: ts sb
using i-bound i-bound' ts-sim ts-i True (\mathcal{D}
apply (clarsimp simp add: Let-def nth-list-update
outstanding-refs-conv ts }\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\vartheta}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{D}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mathrm{ suspend-nothing' flush-all
rel-commute
acquired-append split: if-split-asm)
done
ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct'
valid-sops'
valid-dd' load-tmps-fresh' enough-flushs'
valid-program-history' valid' m}\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
by auto
next
case False

```
```

then obtain r where r-in: r f set sb and volatile-r: is-volatile-Write sb r
by (auto simp add: outstanding-refs-conv)
from takeWhile-dropWhile-real-prefix
[OF r-in, of (Not O is-volatile-Write sb), simplified, OF volatile-r]
obtain a' v'sb" A" L" R" W" sop' where
sb-split: sb = takeWhile (Not o is-volatile-Write sb )sb @ Write sb True a' sop' v}\mp@subsup{v}{}{\prime}\mp@subsup{A}{}{\prime\prime}\mp@subsup{L}{}{\prime\prime
R" W"\# sb"
and
drop: dropWhile (Not o is-volatile-Write esb) sb = Write sb True a'sop' v' A"L" R" W"\#
sb"
apply (auto)
subgoal for y ys
apply (case-tac y)
apply auto
done
done
from drop suspends have suspends: suspends = Write sb True a' sop'v' A" L" R" W"\#
sb"
by simp

```
    have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by auto
moreover
have Write \({ }_{\text {sb }}\) True \(a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime \prime} \mathrm{L}^{\prime \prime} \mathrm{R}^{\prime \prime} \mathrm{W}^{\prime \prime} \in\) set sb
    by (subst sb-split) auto
note drop-app \(=\) dropWhile-append 1
[OF this, of (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ), simplified]
from takeWhile-append1 [where \(\mathrm{P}=\) Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\), OF r -in] volatile-r
have takeWhile-app:

is-volatile-Write \({ }_{\text {sb }}\) ) sb)
    by simp
note share-commute \(=\) share-all-until-volatile-write-append-Ghostsb \({ }_{\text {sb }}\) [ OF False i-bound
\(\mathrm{ts}_{\mathrm{sb} \text { - }}\) ]
from \(\mathcal{D}\)
 by (auto simp: outstanding-refs-append)
have \(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply (simp add: m flush-all-until-volatile-write-append-Ghost-commute [OF i-bound \(\left.\left.\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\right] \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime}\right)\)
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}^{\prime}} \mathrm{ts}_{\mathrm{sb}^{\prime}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \boldsymbol{\vartheta}_{\mathrm{sb}}{ }^{\prime}\) share-commute)
using leq
apply (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
using i-bound i-bound' ts-sim ts-i is-sim \(\mathcal{D}^{\prime}\)
apply (clarsimp simp add: Let-def nth-list-update is-sim drop-app read-tmps-append suspends
prog-instrs-append-Ghost \(t_{s b}\) instrs-append-Ghostsb hd-prog-append-Ghost sb \(_{\text {sb }}\) drop is \(\mathrm{sb}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \vartheta_{\mathrm{sb}}{ }^{\prime} \mathcal{D}_{\mathrm{sb}}{ }^{\prime}\) takeWhile-app split: if-split-asm)
done
ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' valid-dd \({ }^{\prime}\) valid-sops \({ }^{\prime}\) load-tmps-fresh' enough-flushs \({ }^{\prime}\)
valid-program-history' valid' \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply )
qed
qed
next
case (StoreBuffer i \(\mathrm{p}_{\mathrm{sb}}\) is \(\mathrm{s}_{\mathrm{sb}} \vartheta_{\mathrm{sb}}\) sb \(\left.\mathcal{D}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathcal{R}_{\mathrm{sb}} \mathrm{sb}^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\right)\)
then obtain
\(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}=\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \boldsymbol{v}_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}{ }^{\prime}, \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\right)\right]\) and
i -bound: \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) and
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}\right.\), is \(\left.\mathrm{s}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\) and
flush: \(\left(\mathrm{m}_{\mathrm{sb}}, \mathrm{sb}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \rightarrow_{\mathrm{f}}\)
\(\left(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}_{\mathrm{sb}}{ }^{\prime}, \mathcal{R}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
by auto
from sim obtain
\(\mathrm{m}: \mathrm{m}=\) flush-all-until-volatile-write \(\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}: \mathcal{S}=\) share-all-until-volatile-write \(\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}\) and
leq: length \(\mathrm{ts}_{\mathrm{sb}}=\) length ts and
ts-sim: \(\forall \mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}}\).
let \(\left(\mathrm{p}, \mathrm{is}_{\mathrm{sb}}, \mathcal{\vartheta}, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}\);
suspends \(=\) dropWhile (Not \(\circ\) is-volatile-Write \(\left.{ }_{\text {sb }}\right)\) sb
in \(\exists\) is \(\mathcal{D}\). instrs suspends @ \(\mathrm{is}_{\text {sb }}=\) is @ prog-instrs suspends \(\wedge\)
\[
\mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee \text { outstanding-refs is-volatile-Write }{ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}\right) \wedge
\]
ts ! \(\mathrm{i}=\) (hd-prog p suspends, is, \(\left.\vartheta\right|^{6}\) (dom \(\vartheta\) - read-tmps suspends), (), \(\mathcal{D}\),
acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb) \(\mathcal{O}_{\text {sb }}\), release ( takeWhile \(^{(N o t} \circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}\right)\)
by cases blast
from i-bound leq have i-bound': \(\mathrm{i}<\) length ts
by auto
have split-sb: sb \(=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb @ dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb
(is sb = ?take-sb@?drop-sb)
by simp
from ts-sim [rule-format, OF i-bound] \(\mathrm{ts}_{\mathrm{sb}}\)-i obtain suspends is \(\mathcal{D}\) where suspends: suspends \(=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb and is-sim: instrs suspends \(@ \mathrm{is}_{\mathrm{sb}}=\) is @ prog-instrs suspends and
\(\mathcal{D}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \(\left.{ }_{\text {sb }} \mathrm{sb} \neq\{ \}\right)\) and
ts-i: ts ! \(\mathrm{i}=\)
(hd-prog \(\mathrm{p}_{\mathrm{sb}}\) suspends, is,
\(\left.\vartheta_{\text {sb }}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\text {sb }}-\right.\) read-tmps suspends \(),(), \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\text {sb }}\), release ?take-sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) )
by (auto simp add: Let-def)
from flush-step-preserves-valid [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{-}}\)- flush valid]
have valid': valid ts \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from flush obtain \(r\) where \(s b: s b=r \# s^{\prime}\)
by (cases) auto
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have history-consistent \(\vartheta_{\mathrm{sb}}\) (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\) ) sb.
then
have hist-consis': history-consistent \(\vartheta_{\mathrm{sb}}\) (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\) ) \(\mathrm{sb}^{\prime}\)
by (auto simp add: sb intro: history-consistent-hd-prog
split: memref.splits option.splits)
from valid-history-nth-update [OF i-bound this]
have valid-hist \({ }^{\prime}\) : valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have dist-sb': distinct-read-tmps \(\mathrm{sb}^{\prime}\)
by (simp add: sb split: memref.splits)
have tmps-distinct \({ }^{\prime}:\) tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) - \(]\)
have distinct-load-tmps is sb \(_{\text {sb }}\).
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
next
from read-tmps-distinct-nth-update [OF i-bound dist-sb]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
next
from load-tmps-read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have load-tmps is \(\mathrm{sb}_{\mathrm{sb}} \cap\) read-tmps \(\mathrm{sb}^{\prime}=\{ \}\)
by (auto simp add: sb split: memref.splits)
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
qed
from load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have load-tmps is \({ }_{\text {sb }} \cap \bigcup\) (fst ' write-sops sb') \(=\{ \}\)
by (auto simp add: sb split: memref.splits)
from valid-data-dependency-nth-update
[OF i-bound data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i] this]
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
```

from valid-store-sops [OF i-bound $\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}} \mathrm{i}$ ] valid-write-sops [OF i -bound $\mathrm{ts}_{\mathrm{sb}^{-}} \mathrm{i}$ ]
valid-sops-nth-update [OF i-bound]
have valid-sops': valid-sops $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
by (cases r) (auto simp add: sb ts sb )

```
    have load-tmps-fresh \({ }^{\prime}\) : load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    proof -
        from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) ]
        have load-tmps is \({ }_{\text {sb }} \cap \operatorname{dom} \vartheta_{\text {sb }}=\{ \}\).
        from load-tmps-fresh-nth-update [OF i-bound this]
        show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}\) )
    qed
    have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
    proof -
    from clean-no-outstanding-volatile-Write \({ }_{s b}\) [OF i-bound ts \(\left._{\text {sb }}-\mathrm{i}\right]\)
    have \(\neg \mathcal{D}_{\text {sb }} \longrightarrow\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}^{\prime}=\{ \}\)
by (auto simp add: sb split: if-split-asm)
    from enough-flushs-nth-update [OF i-bound this]
    show ?thesis
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) )
    qed
    show ?thesis
    proof (cases r)
        case ( Write \(_{\text {sb }}\) volatile a sop v A L R W)
        from flush this
        have \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\left(\mathrm{m}_{\mathrm{sb}}(\mathrm{a}:=\mathrm{v})\right)\)
by cases (auto simp add: sb)
    have non-volatile-owned: \(\neg\) volatile \(\longrightarrow \mathrm{a} \in \mathcal{O}_{\text {sb }}\)
    proof (cases volatile)
case True thus ?thesis by simp
    next
case False
with outstanding-non-volatile-refs-owned-or-read-only [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have \(a \in \mathcal{O}_{\text {sb }}\)
    by (simp add: sb Write \({ }_{\text {sb }}\) )
thus ?thesis by simp
    qed
    have a-unowned-by-others:
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-, \mathcal{O}_{\mathrm{j}},-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\) in a \(\notin \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\) ) proof (unfold Let-def, clarify del: notI)
fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}}\) is \(\mathrm{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
assume neq: \(\mathrm{i} \neq \mathrm{j}\)
assume ts-j: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show a \(\notin \mathcal{O}_{j} \cup\) all-acquired sb \(_{j}\)
proof (cases volatile)
case True
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound neq \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}-\mathrm{j}\right]\)
show ?thesis by (simp add: sb Write \({ }_{\text {sb }}\) True)
next
case False
with non-volatile-owned
have \(\mathrm{a} \in \mathcal{O}_{\text {sb }}\)
by simp
with ownership-distinct [OF i-bound j-bound neq \(\mathrm{ts}_{\mathrm{sb}}\)-i ts-j]
show ?thesis
by blast
qed
qed
from valid-reads [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have reads-consis: reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}\).
\{
fix j
\(\operatorname{fix~}_{\mathrm{j}} \operatorname{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
assume ts \(_{\text {sb }}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sbj}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
have a \(\notin\) outstanding-refs is-Write \({ }_{\mathbf{s b}}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) proof
assume \(\mathrm{a} \in\) outstanding-refs is-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
hence \(\mathrm{a} \in\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) )
\(\mathrm{sb}_{\mathrm{j}}\) )
by (simp add: outstanding-refs-is-non-volatile-Write sb \(^{\text {b-takeWhile-conv) }}\)
hence a \(\in\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) sb \(_{j}\)
using outstanding-refs-append [of - (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ) (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )]
by auto
with non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]]
have a \(\in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto
with a-unowned-by-others [rule-format, OF j-bound neq-i-j] \(\mathrm{ts}_{\mathbf{s b}}-\mathrm{j}\)
```

    show False
    ```
    by auto
qed
    \}
    note a-notin-others \(=\) this
from a-notin-others
have a-notin-others':
\(\forall \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} . \mathrm{i} \neq \mathrm{j} \longrightarrow\)
(let \(\left(-,-,-, \mathrm{sb}_{\mathrm{j}},-,-,-\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}\) in a \(\notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ))
by (fastforce simp add: Let-def)
obtain D f where sop: sop=(D,f) by (cases sop) auto
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ] sop sb Write \({ }_{\text {sb }}\)
obtain D-tmps: \(\mathrm{D} \subseteq\) dom \(\vartheta_{\mathrm{sb}}\) and f-v: f \(\vartheta_{\mathrm{sb}}=\mathrm{v}\) and
D-sb': \(\mathrm{D} \cap\) read-tmps sb \({ }^{\prime}=\{ \}\)
by auto
let \(? \vartheta=\left(\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\text { read-tmps sb}\right)^{\prime}\right)\)
from D-tmps D-sb'
have D -tmps': \(\mathrm{D} \subseteq\) dom ?ध
by auto

have valid-sop sop
by (auto simp add: sb Write \({ }_{\text {sb }}\) )
from this [simplified sop]
interpret valid-sop (D,f) .
from D-tmps D-sb'
have \(\left(\left(\operatorname{dom} \vartheta_{\text {sb }}-\right.\right.\) read-tmps sb \(\left.) \cap \mathrm{D}\right)=\mathrm{D}\)
by blast
with valid-sop [OF refl D-tmps] valid-sop [OF refl D-tmps] f-v
have \(\mathrm{f}-\mathrm{v}\) : f ? \(\mathrm{V}=\mathrm{v}\)
by auto
have valid-program-history \({ }^{\prime}\) : valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have causal-program-history is \(_{\text {sb }} \mathrm{sb}\).
then have causal': causal-program-history is sb \(_{\text {sb }}\) sb \(^{\prime}\)
by (simp add: sb Write \({ }_{\text {sb }}\) causal-program-history-def)
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by (simp add: sb Write \({ }_{\text {sb }}\) )
from valid-program-history-nth-update [OF i-bound causal' this]
```

show ?thesis
by (simp add: tssb
qed

```
            show ?thesis
            proof (cases volatile)
case True
note volatile \(=\) this
from flush Write sb volatile
obtain
    \(\mathcal{O}_{s b}{ }^{\prime}: \mathcal{O}_{s b}{ }^{\prime}=\mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}\) and
    \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}} \oplus \mathrm{W} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) and
        \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\) Map.empty
    by cases (auto simp add: sb)
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
obtain
A-shared-owned: \(\mathrm{A} \subseteq \operatorname{dom} \mathcal{S}_{\mathrm{sb}} \cup \mathcal{O}_{\mathrm{sb}}\) and
L-subset: \(\mathrm{L} \subseteq \mathrm{A}\) and
\(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\) and
R-owned: \(\mathrm{R} \subseteq \mathcal{O}_{\mathrm{sb}}\)
by (clarsimp simp add: sb Write \({ }_{\text {sb }}\) volatile)
from sb Write \({ }_{\text {sb }}\) True have take-empty: takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}=[]\) by (auto simp add: outstanding-refs-conv)
from sb Write sb True have suspend-all: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(=\mathrm{sb}\) by (auto simp add: outstanding-refs-conv)
hence suspends-all: suspends \(=\mathrm{sb}\)
by (simp add: suspends)
from is-sim
have is-sim: Write True a (D, f) A L R W\# instrs sb \({ }^{\prime}\) @ is \(\mathrm{s}_{\mathrm{sb}}=\) is @ prog-instrs sb \({ }^{\prime}\) by (simp add: True Writesb suspends-all sb sop)
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
interpret causal-program-history is sb sb .
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have last-prog: last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
from causal-program-history [of [Write sb \(^{\text {True a }}\) (D, f) v A L R W] sb \(]\) is-sim
obtain is' where
is: is \(=\) Write True a (D, f) A L R W\# is' and
is \({ }^{\prime}\)-sim: instrs sb \({ }^{\prime} i^{1} \mathrm{~s}_{\mathrm{sb}}=\) is \({ }^{\prime} @\) prog-instrs sb \({ }^{\prime}\)
by (auto simp add: sb Write sb \(^{\text {b }}\) volatile sop)
from causal-program-history have
```

    causal-program-history-sb': causal-program-history is \(\mathrm{sb} \mathrm{sb}^{\prime}\)
    apply -
    apply (rule causal-program-history.intro)
    apply (auto simp add: sb Write \({ }_{\text {sb }}\) )
    done
    ```
from ts-i have ts-i: ts ! \(\mathrm{i}=\)
(hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\), Write True a (D, f) A L R W\# is', ? ?, (), D, acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),
            release ?take-sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
    by (simp add: suspends-all sb Write sb is)
let ?ts \({ }^{\prime}=\operatorname{ts}\left[\mathrm{i}:=\left(h d-\right.\right.\) prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\), is', ? \({ }^{\prime}\), (), True, acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\), Map.empty)]
from i-bound \({ }^{\prime}\) have ts'-i: ?ts \(!\mathrm{i}=\left(h d-p r o g ~ \mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\right.\), is', ? \(\vartheta\), (),True, acquired True ?take-sb \(\mathcal{O}_{\text {sb }} \cup \mathrm{A}-\mathrm{R}\), Map.empty) by simp
from no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
have a-not-ro: a \(\notin\) read-only \(\mathcal{S}_{\text {sb }}\) by (clarsimp simp add: sb Write \({ }_{\text {sb }}\) volatile)

\section*{\{}
fix j
\(\operatorname{fix}_{\mathrm{p}} \mathrm{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
have a \(\notin\) unforwarded-non-volatile-reads (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ) \(\}\) proof
let ?take-sb \({ }_{j}=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{\mathrm{j}}\)
let ? drop-sb \(b_{j}=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
assume a-in: \(\mathrm{a} \in\) unforwarded-non-volatile-reads ?drop-sb \(\mathrm{b}_{\mathrm{j}}\{ \}\)
from a-unowned-by-others [rule-format, OF j-bound neq-i-j] ts \(\mathrm{s}_{\mathbf{s b}}-\mathrm{j}\)
obtain a-unowned: a \(\notin \mathcal{O}_{\mathrm{j}}\) and a-unacq: a \(\notin\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto
with all-acquired-append [of ?take-sb \({ }_{j} \quad\) ?drop-sb \(\left.{ }_{j}\right]\) ac-quired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\left[\right.\) of \(\operatorname{sb}_{j} \mathcal{O}_{j}\) ]
have a-unacq-take: a \(\notin\) acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\)
by (auto simp add: )
note nvo-j \(=\) outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-drop-j: non-volatile-owned-or-read-only True (share ?take-sb \(\mathcal{S}_{\mathrm{sb}}\) )
(acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \({ }_{\mathrm{j}}\).
```

note consis-j $=$ sharing-consis [OF j-bound $\mathrm{ts}_{\mathrm{sb}} \mathrm{j}$ ]
with sharing-consistent-append [of $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb $\mathrm{Sb}_{\mathrm{j}}$ ?drop-sb ${ }_{\mathrm{j}}$ ]
obtain consis-take-j: sharing-consistent $\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}}$ ?take-sb ${ }_{\mathrm{j}}$ and
consis-drop-j: sharing-consistent (share ?take-sb $\mathcal{S}_{\mathrm{sb}}$ )
(acquired True ?take-sb $\mathcal{O}_{\mathrm{j}}$ ) ?drop-sb ${ }_{\mathrm{j}}$
by auto
from in-unforwarded-non-volatile-reads-non-volatile-Read ${ }_{\mathbf{s b}}[\mathrm{OF} \mathrm{a-in}]$
have $\mathrm{a}-\mathrm{in}$ ': $\mathrm{a} \in$ outstanding-refs is-non-volatile-Read ${ }_{\mathrm{sb}}$ ? ${ }^{\text {drop-sb }}{ }_{\mathrm{j}}$.
note reads-consis-j $=$ valid-reads [ OF j -bound $\mathrm{ts}_{\mathrm{sb}} \mathrm{j}$ ]
from reads-consistent-drop [OF this]
have reads-consis-drop-j:
reads-consistent True (acquired True ?take-sb $\boldsymbol{\mathcal { O }}_{\mathrm{j}}$ ) (flush ?take-sb $\mathrm{m}_{\mathrm{sb}}$ ) ?drop-sb ${ }_{\mathrm{j}}$.

```
    from read-only-share-all-shared [of a ?take-sb \(\mathcal{S}_{\text {sb }}\) ] a-not-ro
    all-shared-acquired-or-owned [OF consis-take-j]
    all-acquired-append [of ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\) ] a-unowned a-unacq
have a-not-ro-j: a \(\notin\) read-only (share ?take-sb \({ }_{j} \mathcal{S}_{\text {sb }}\) )
    by auto
from ts-sim [rule-format, OF j -bound] \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j} \mathrm{j}\)-bound
obtain suspends \(\mathrm{j}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}}\) where
    suspends \(\mathrm{s}_{\mathrm{j}}\) : suspends \(\mathrm{j}_{\mathrm{j}}=\) ? \(\mathrm{drop}^{2}-\mathrm{sb}_{\mathrm{j}}\) and
    \(\mathrm{is}_{\mathrm{j}}\) : instrs suspends \(\mathrm{s}_{\mathrm{j}} @ i \mathrm{is}_{\mathrm{sbj}}=\mathrm{is} \mathrm{is}_{\mathrm{j}} @\) prog-instrs suspends \(\mathrm{s}_{\mathrm{j}}\) and
    \(\mathcal{D}_{\mathrm{j}}: \mathcal{D}_{\mathrm{sbj}}=\left(\mathcal{D}_{\mathrm{j}} \vee\right.\) outstanding-refs is-volatile-Write \(\left.\mathrm{s}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}} \neq\{ \}\right)\) and
    \(\mathrm{ts}_{\mathrm{j}}: \mathrm{ts}!\mathrm{j}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\mathrm{s}_{\mathrm{j}}\), \(\mathrm{is}_{\mathrm{j}}\),
    \(\left.\vartheta_{\mathrm{j}}\right|^{6}\left(\right.\) dom \(\vartheta_{\mathrm{j}}\) - read-tmps suspends \(\left.\mathrm{j}_{\mathrm{j}}\right),()\),
    \(\mathcal{D}_{\mathrm{j}}\), acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\) )
    by (auto simp: Let-def)
from valid-last-prog [ OF j -bound \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\) have last-prog: last-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\).
from j-bound i-bound \({ }^{\prime}\) leq have j -bound-ts': \(\mathrm{j}<\) length ts
    by simp
    from read-only-read-acquired-unforwarded-acquire-witness [OF nvo-drop-j con-
sis-drop-j
    a-not-ro-j a-unacq-take a-in]

\section*{have False}
proof
assume \(\exists\) sop \(a^{\prime}\) v ys zs A L R W.
?drop-sb \({ }_{j}=\) ys @ Write \({ }_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop v A LRW \(\# \mathrm{zs} \wedge \mathrm{a} \in \mathrm{A} \wedge\)
\(a \notin\) outstanding-refs is-Write sb \(^{\text {ys }}\) ys \(\mathrm{a}^{\prime} \neq \mathrm{a}\)
with suspends \({ }_{j}\)
obtain \(\mathrm{a}^{\prime}\) sop' \(\mathrm{v}^{\prime}\) ys \(\mathrm{zs}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
split-suspendsj: suspends \(\mathrm{s}_{\mathrm{j}}=\mathrm{ys}\) @ Write \(_{\mathrm{sb}}\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}^{\prime}\) (is suspends \({ }_{j}=\) ?suspends) and
\(a-A^{\prime}: a \in A^{\prime}\) and
no-write: a \(\notin\) outstanding-refs is-Write \({ }_{\text {sb }}\) (ys @ \(\left[W_{r i t e}^{s b}\right.\) True \(\left.\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{W}^{\prime}\right)\)
by (auto simp add: outstanding-refs-append)
from last-prog
have lp: last-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}=\mathrm{p}_{\mathrm{j}}\)
apply -
apply (rule last-prog-same-append [where \(\mathrm{sb}=\) ? take-sb \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric \(^{2}\) suspends \({ }_{j}\) )
apply simp
done
from sharing-consis [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]
have sharing-consis-j: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
then have \(A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}\)
by (simp add: sharing-consistent-append [of - ? ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\), simplified]
suspends \(_{j}\) [symmetric] split-suspends \({ }_{j}\) sharing-consistent-append)
from valid-program-history [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{j}\) ]
have causal-program-history \(\mathrm{is}_{\mathrm{sbj}} \mathrm{sb}_{\mathrm{j}}\).
then have cph: causal-program-history is sbj ?suspends
apply -
apply (rule causal-program-history-suffix [where \(\mathrm{sb}=\) ? \({ }^{\text {take- }} \mathrm{sb}_{\mathrm{j}}\) ] )
apply (simp only: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply (simp add: split-suspends \({ }_{\mathrm{j}}\) )
done
from valid-reads [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}\)-j]
have reads-consis-j: reads-consistent False \(\mathcal{O}_{j} \mathrm{~m}_{\mathrm{sb}} \mathrm{sb}\).
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) )
j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb \(_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) m suspends \(\mathrm{s}_{\mathrm{j}}\) by (simp add: m suspends \({ }_{\mathrm{j}}\) )
hence reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ) \(m\left(y s @\left[W_{r i t e}^{s b}\right.\right.\) True \(\left.\left.a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)\)
by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) reads-consistent-append)
from valid-write-sops [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j}\) @?suspends). valid-sop sop
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \({ }_{\mathrm{j}}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and valid-sops-drop: \(\forall\) sop \(\in\) write-sops ( \(y s @\left[W r i t e{ }_{\text {sb }}\right.\) True \(\left.\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}{ }^{\prime}\right]\) ). valid-sop sop
apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j} @_{\text {suspends }}^{j}\) )
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}[\) symmetric \(]\) suspends \(_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h -consis:
history-consistent \(\vartheta_{\mathrm{j}}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) (?take-sb \(\mathrm{j}_{\mathrm{j}}\) @suspends \(\mathrm{j}_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
apply simp
done
have last-prog-hd-prog: last-prog (hd-prog \(\left.\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\right)\) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\left(\mathrm{hd}-\operatorname{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left({ }^{(\text {take-sb }} \mathrm{j}_{\mathrm{j}} @\right.\) ? drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\)
by simp
from last-prog-hd-prog-append' [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \({ }_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric suspends \(\mathrm{s}_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{j}_{\mathrm{j}} @\) suspends j\(\left.)\right)\) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends \({ }_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\mathrm{j}}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{s}_{\mathrm{j}}\) ) suspends \(\mathrm{s}_{\mathrm{j}}\)
by (simp add: split-suspendsj \(\left[\right.\) symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read \({ }_{\text {sb }}\)
\(\left(y s @\left[W_{r i t e}^{s b}\right.\right.\) True a' \(\left.\left.\operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\right)=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \({ }_{j}\) )
have acq-simp:
acquired True (ys @ \(\left[\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime} \mathrm{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ] \()\)
(acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) \(=\)
acquired True ys (acquired True ? take-sb \(\left.\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\right) \cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
by (simp add: acquired-append)
from flush-store-buffer-append [where \(\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Write}_{\text {sb }}\right.\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ] and \(\mathrm{sb}^{\prime}=\mathrm{zs}^{\prime}\), simplified,

OF j-bound-ts' \({ }^{\prime} \mathrm{is}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{s}_{\mathrm{j}}\) ] cph [simplified suspends \(\left.\mathrm{s}_{\mathrm{j}}\right] \mathrm{ts}_{\mathrm{j}}[\) simplified split-suspends \({ }_{j}\) ]
refl lp [simplified split-suspends \({ }_{\mathrm{j}}\) ] reads-consis-ys
hist-consis' \({ }^{\prime}\) simplified split-suspends \({ }_{\mathrm{j}}\) ] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-vol-read], where
\(\mathcal{S}=\mathcal{S}]\)
obtain \(\mathrm{is}_{\mathrm{j}}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : instrs \(\mathrm{zs}^{\prime} @ \mathrm{is}_{\mathrm{sbj}}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs \(\mathrm{zs}{ }^{\prime}\) and
steps-ys: (ts, m, \(\mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}\)
(ts \([\mathrm{j}:=(\) last-prog
(hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime}\) sop \(^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \# \mathrm{zs}\) ) \()\) ( \(\mathrm{ys} @\left[\right.\) Write \(_{\text {sb }}\)
True \(\mathrm{a}^{\prime} \mathrm{spp}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \mathrm{J}\) ),
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\mathrm{j}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{j}}-\right.\) read-tmps zs'\()\),
(), True, acquired True ys (acquired True ?take-sb \(\mathcal{O}_{j}\) ) \(\cup \mathrm{A}^{\prime}-\)
\(\left.\left.R^{\prime}, \mathcal{R}_{\mathrm{j}}{ }^{\prime}\right)\right]\),
flush (ys@[Write \({ }_{\text {sb }}\) True \(\left.\left.\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}{ }^{\prime}\right]\right) \mathrm{m}\), share ( \(\mathrm{ys} @\left[W_{r i t e}\right.\) sb True \(\mathrm{a}^{\prime}\) sop \(\left.\left.\left.^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\right) \mathcal{S}\right)\)
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}(? \mathrm{ts}-\mathrm{ys}, ? \mathrm{~m}-\mathrm{ys}, ?\) shared-ys))
by (auto simp add: acquired-append outstanding-refs-append)
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j ts-i
have ts-ys-i: ?ts-ys!i = (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\), Write True a (D, f) A L R W\# is', ? \({ }^{\text {, }}\), (), \(\mathcal{D}\),
acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),release ?take-sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) )
by simp
note conflict-computation \(=\) steps-ys
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe: safe-delayed (?ts-ys,?m-ys,?shared-ys).
with safe-delayedE [OF safe i-bound-ys ts-ys-i]
have a-unowned:
\(\forall \mathrm{j}<\) length ?ts-ys. \(\mathrm{i} \neq \mathrm{j} \longrightarrow\left(\right.\) let \(\left(\mathcal{O}_{\mathrm{j}}\right)=\) map owned ?ts-ys! j in a \(\left.\notin \mathcal{O}_{\mathrm{j}}\right)\)
apply cases
apply (auto simp add: Let-def sb)
done
from a-A' a-unowned [rule-format, of j ] neq-i-j j-bound leq \(\mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
show False
```

by (auto simp add: Let-def)
next
assume \existsA L R W ys zs. ?drop-sbj = ys @ Ghostsb ALR W\# zs ^a\inA Aa\not\in
outstanding-refs is-Write sb ys
with suspendsj
obtain ys zs' A' L' R' W' where

```

```

and
a-A': a }\in\mp@subsup{A}{}{\prime}\mathrm{ and
no-write: a \& outstanding-refs is-Write sb (ys @ [Ghostsb A' L' R' W\])

    by (auto simp add: outstanding-refs-append)
        from last-prog
        have lp: last-prog pj suspendsj
    apply -
    apply (rule last-prog-same-append [where sb=?take-sb;}]\mathrm{ )
    apply (simp only: split-suspendsj [symmetric] suspends}\mp@subsup{\textrm{s}}{\textrm{j}}{}\mathrm{ )
    apply simp
    done
    from valid-program-history [OF j-bound ts sb-j]
    have causal-program-history is isbj sbj.
    then have cph: causal-program-history is isbj ?suspends
    apply -
apply (rule causal-program-history-suffix [where sb=?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}]\mathrm{ )
apply (simp only: split-suspendsj [symmetric] suspendsj)
apply (simp add: split-suspendsj)
done
from valid-reads [OF j-bound ts scb-j]
have reads-consis-j: reads-consistent False }\mp@subsup{\mathcal{O}}{j}{}\mp@subsup{\textrm{m}}{\textrm{sb}}{}\mathrm{ sb
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing
S sb ts sb
j-bound tssb-j this]
have reads-consis-m-j: reads-consistent True (acquired True ?take-sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ )m suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{
by (simp add: m suspends}\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )

```
    hence reads-consis-ys: reads-consistent True (acquired True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) )

by (simp add: split-suspends \(\mathrm{j}_{\mathrm{j}}\) reads-consistent-append)
from valid-write-sops [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have \(\forall\) sop \(\in\) write-sops (?take-sb \({ }_{j} @\) ?suspends). valid-sop sop by (simp add: split-suspendsj [symmetric] suspends \({ }_{\mathrm{j}}\) )
then obtain valid-sops-take: \(\forall\) sop \(\in\) write-sops ?take-sb \({ }_{j}\). valid-sop sop and valid-sops-drop: \(\forall\) sop \(\in\) write-sops ( \(\mathrm{ys} @\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W} \mathrm{J}\right]\) ). valid-sop sop apply (simp only: write-sops-append)
apply auto
done
from read-tmps-distinct [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have distinct-read-tmps (?take-sb \({ }_{j} @_{\text {suspends }}^{j}\) )
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
then obtain
read-tmps-take-drop: read-tmps ?take-sb \({ }_{j} \cap\) read-tmps suspends \({ }_{j}=\{ \}\) and
distinct-read-tmps-drop: distinct-read-tmps suspends \({ }_{j}\)
apply (simp only: split-suspends \({ }_{j}\left[\right.\) symmetric suspends \(_{\mathrm{j}}\) )
apply (simp only: distinct-read-tmps-append)
done
from valid-history [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have h -consis:
history-consistent \(\vartheta_{\mathrm{j}}\left(\right.\) hd-prog \(\mathrm{p}_{\mathrm{j}}\left(?\right.\) take-sb \(_{\mathrm{j}} @\) suspends \(\left.\mathrm{s}_{\mathrm{j}}\right)\) ) (?take-sb \(\mathrm{j}_{\mathrm{j}}\) @suspends \(\mathrm{j}_{\mathrm{j}}\) )
apply (simp only: split-suspends \({ }_{j}\) [symmetric] suspends \({ }_{j}\) )
apply simp
done
from sharing-consis [ OF j -bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\) ]
have sharing-consis-j: sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
then have \(A^{\prime}-R^{\prime}: A^{\prime} \cap R^{\prime}=\{ \}\)
by (simp add: sharing-consistent-append [of - - ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\), simplified]
suspends \({ }_{j}\) [symmetric] split-suspends \({ }_{j}\) sharing-consistent-append)
have last-prog-hd-prog: last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{s}_{\mathrm{j}}=\left(\mathrm{hd}-\mathrm{prog} \mathrm{p}_{\mathrm{j}}\right.\) suspends \(\left._{\mathrm{j}}\right)\) proof -
from last-prog have last-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ? take-sb \(\mathrm{j}_{\mathrm{j}}\) @?drop-sb \(\left.\mathrm{j}_{\mathrm{j}}\right)=\mathrm{p}_{\mathrm{j}}\) by simp
from last-prog-hd-prog-append \({ }^{\prime}\) [OF h-consis] this
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\) hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\)
by (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(_{\mathrm{j}}\) )
moreover
have last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\left(\right.\) ?take-sb \(\mathrm{b}_{\mathrm{j}} @\) suspends j\()\) ) ?take-sb \(\mathrm{j}_{\mathrm{j}}=\)
last-prog (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(\mathrm{j}_{\mathrm{j}}\) ) ?take-sb \(\mathrm{b}_{\mathrm{j}}\)
apply (simp only: split-suspends \(\mathrm{s}_{\mathrm{j}}\) [symmetric] suspends \(\mathrm{s}_{\mathrm{j}}\) )
by (rule last-prog-hd-prog-append)
ultimately show ?thesis
by (simp add: split-suspends \(\mathrm{s}_{\mathrm{j}}\left[\right.\) symmetric] suspends \(\mathrm{j}_{\mathrm{j}}\) )
qed
from history-consistent-appendD [OF valid-sops-take read-tmps-take-drop
h-consis] last-prog-hd-prog
have hist-consis': history-consistent \(\vartheta_{\mathrm{j}}\) (hd-prog \(\mathrm{p}_{\mathrm{j}}\) suspends \(_{\mathrm{j}}\) ) suspends \(\mathrm{j}_{\mathrm{j}}\)
by (simp add: split-suspends \({ }_{j}\left[\right.\) symmetric] suspends \({ }_{j}\) )
from reads-consistent-drop-volatile-writes-no-volatile-reads
[OF reads-consis-j]
have no-vol-read: outstanding-refs is-volatile-Read \({ }_{\text {sb }}\)
\(\left(\mathrm{ys}_{\mathrm{s}}\right.\) [Ghost \(\left.\left.\mathrm{sb}_{\mathrm{sb}} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\right]\right)=\{ \}\)
by (auto simp add: outstanding-refs-append suspends \({ }_{j}\) [symmetric]
split-suspends \(_{j}\) )
have acq-simp:
acquired True (ys @ [Ghost sb \(\mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ' \(]\) )
(acquired True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) \(=\)
acquired True ys (acquired True ? take-sb \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ) \(\cup \mathrm{A}^{\prime}-\mathrm{R}^{\prime}\)
by (simp add: acquired-append)
from flush-store-buffer-append \(\left[\right.\) where \(\mathrm{sb}=\mathrm{ys} @\left[\mathrm{Ghost}_{\text {sb }} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\right]\) and \(\mathrm{sb}^{\prime}=\mathrm{zs}^{\prime}\), simplified,
OF j-bound-ts' is \(_{\mathrm{j}}\) [simplified split-suspends \({ }_{\mathrm{j}}\) ] cph [simplified suspends \(\mathrm{s}_{\mathrm{j}}\) ]
\(\mathrm{ts}_{\mathrm{j}}\) [simplified split-suspends \(\mathrm{j}_{\mathrm{j}}\) ] refl lp [simplified split-suspends \(\mathrm{j}_{\mathrm{j}}\) ] reads-consis-ys
hist-consis' [simplified split-suspendsj] valid-sops-drop
distinct-read-tmps-drop [simplified split-suspends \({ }_{\mathrm{j}}\) ]
no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-vol-read], where \(\mathcal{S}=\mathcal{S}]\)
obtain is \({ }^{\prime}{ }^{\prime} \mathcal{R}_{\mathrm{j}}{ }^{\prime}\) where
\(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\) : instrs \(\mathrm{zs}^{\prime} @ \mathrm{is}_{\text {sbj }}=\mathrm{is}_{\mathrm{j}}{ }^{\prime} @\) prog-instrs zs' and
steps-ys: (ts, m, \(\mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}\)
(ts \([\mathrm{j}:=\) (last-prog
 \(\mathrm{is}_{\mathrm{j}}{ }^{\prime}\),
\(\left.\vartheta_{\mathrm{j}}\right|^{\prime}\left(\operatorname{dom} \vartheta_{\mathrm{j}}-\right.\) read-tmps zs'\()\),
(),
\(\mathcal{D}_{\mathrm{j}} \vee\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) ys \(\neq\{ \}\), acquired True ys (acquired True ? take-sb \(\mathcal{O}_{j}\) ) \(\cup A^{\prime}-R^{\prime}, \mathcal{R}_{j}{ }^{\prime}\) )],
flush (ys@[Ghost \(\left.\left.{ }_{\text {sb }} A^{\prime} L^{\prime} R^{\prime} W\right]\right) m\), share (ys@[Ghost \({ }_{\text {sb }} A^{\prime}\)
\(\mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}\) ]) \(\mathcal{S}\) )
(is \((-,-,-) \Rightarrow_{\mathrm{d}}{ }^{*}\) (?ts-ys,?m-ys,?shared-ys))
by (auto simp add: acquired-append outstanding-refs-append)
from i-bound' have i-bound-ys: i < length ?ts-ys
by auto
from i-bound' neq-i-j ts-i
have ts-ys-i: ?ts-ys!i = (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}\), Write True a (D, f) A L R W\# is', ? \({ }^{\text {, }}\), (), \(\mathcal{D}\),
acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
by simp
note conflict-computation \(=\) steps-ys
from safe-reach-safe-rtrancl [OF safe-reach conflict-computation]
have safe: safe-delayed (?ts-ys,?m-ys,?shared-ys).
with safe-delayedE [OF safe i-bound-ys ts-ys-i]
have a-unowned:
```

j < length ?ts-ys. i\not=j }\longrightarrow(\mathrm{ let }(\mp@subsup{\mathcal{O}}{\textrm{j}}{})=\mathrm{ map owned ?ts-ys!j in a }\not\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}
apply cases
apply (auto simp add: Let-def sb)
done
from a-A' a-unowned [rule-format, of j] neq-i-j j-bound leq A'-R'
show False
by (auto simp add: Let-def)
qed
then show False
by simp
qed
}
note a-notin-unforwarded-non-volatile-reads-drop = this

```
have valid-reads': valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
    fix j \(\mathrm{p}_{\mathrm{j}} \mathrm{is} \mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
    assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
    assume ts- \(\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
    show reads-consistent False \(\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}_{\mathrm{j}}\)
    proof (cases \(\mathrm{i}=\mathrm{j}\) )
        case True
        from reads-consis ts-j j-bound sb show ?thesis
                            by (clarsimp \(\operatorname{simp}\) add: True \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}\) Write \(_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) volatile
reads-consistent-pending-write-antimono)
    next
        case False
        from j-bound have j -bound': \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
        by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
            moreover from ts-j False have ts-j\({ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
            using j-bound by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
            ultimately have consis-m: reads-consistent False \(\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\)
                by (rule valid-reads)
            from a-unowned-by-others [rule-format, OF j-bound' False] ts-j'
            have a-unowned: \(\mathfrak{a} \notin \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
                by \(\operatorname{simp}\)
            let ?take-sb \({ }_{j}=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{\mathrm{j}}\)
            let ?drop-sb \(\mathrm{b}_{\mathrm{j}}=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
            from a-unowned acquired-reads-all-acquired [of True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) ]
            all-acquired-append [of ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\) ]
            have a-not-acq-reads: a \(\notin\) acquired-reads True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\)
            by auto
            moreover
            note a-unfw = a-notin-unforwarded-non-volatile-reads-drop [OF j-bound' ts-j \({ }^{\prime}\) False]
            ultimately
            show ?thesis
using reads-consistent-mem-eq-on-unforwarded-non-volatile-reads-drop [where \(\mathrm{W}=\{ \}\) and
\(\mathrm{A}=\) unforwarded-non-volatile-reads ? drop-sb \(\mathrm{b}_{\mathrm{j}}\{ \} \cup\) acquired-reads True ?take-sb \(\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) and \(\mathrm{m}^{\prime}=\left(\mathrm{m}_{\mathrm{sb}}(\mathrm{a}:=\mathrm{v})\right), \mathrm{OF}--\) consis- m\(]\) by (fastforce simp add: \(\mathrm{m}_{\mathrm{sb}}\) )
qed
qed
have valid-own': valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
proof
fix \(\mathrm{j} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{p}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show non-volatile-owned-or-read-only False \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{j}=\mathrm{i}\) )
case True
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have non-volatile-owned-or-read-only False
\(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\left(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right) \mathrm{sb}^{\prime}\)
by (auto simp add: sb Write sb volatile non-volatile-owned-or-read-only-pending-write-antimono)
then show?thesis
using True i-bound \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}-\mathrm{j}\)
by (auto \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )

\section*{next}
case False
from j-bound have j -bound \({ }^{\prime}: \mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) ')
with \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}-\mathrm{j}\) False i-bound
have \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
note nvo \(=\) outstanding-non-volatile-refs-owned-or-read-only \(\left[\mathrm{OF} j\right.\)-bound \(\left.{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{j}\right]\)
from read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i] R-owned
have \(\mathrm{R} \cap\) read-only \(\mathcal{S}_{\text {sb }}=\{ \}\)
by auto
with read-only-reads-unowned [OF j-bound \({ }^{\prime}\) i-bound False \(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{j} \mathrm{ts}_{\mathbf{s b}}-\mathrm{i}\right]\) L-subset have \(\forall\) a read-only-reads
(acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\mathcal{O}_{\mathrm{j}}\) )
(dropWhile (Not o is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ).
\(\mathrm{a} \in\) read-only \(\mathcal{S}_{\mathrm{sb}} \longrightarrow \mathrm{a} \in\) read-only \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
by (auto simp add: in-read-only-convs sb Write \({ }_{\text {sb }}\) volatile)
from non-volatile-owned-or-read-only-read-only-reads-eq' [OF nvo this]
have non-volatile-owned-or-read-only False \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\).
thus ?thesis by (simp add: \(\mathcal{S}_{\text {sb }}\) )
qed
qed
next
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
fix \(\mathrm{i}_{1} \mathrm{j} \mathrm{p}_{1}\) is \(\mathcal{O}_{1} \mathcal{R}_{1} \mathcal{D}_{1} \mathrm{xs}_{1} \mathrm{sb}_{1} \mathrm{p}_{\mathrm{j}}\) is \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume \(i_{1}\)-bound: \(i_{1}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume j-bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume \(i_{1}-j: i_{1} \neq j\)
assume ts- \(\mathrm{i}_{1}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
assume \(\mathrm{ts}^{\mathrm{j}} \mathrm{j}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show \(\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right) \cap\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{1}=\{ \}\)
proof (cases \(i_{1}=i\) )
case True
from \(\mathrm{i}_{1}-\mathrm{j}\) True have neq- \(\mathrm{i}-\mathrm{j}: \mathrm{i} \neq \mathrm{j}\)
by auto
from \(j\)-bound have \(j\)-bound \({ }^{\prime}: ~ j<\) length ts \(_{\text {sb }}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ts-j neq-i-j have ts-j \({ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound' neq-i-j
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ts-j\(\left.{ }^{\prime}\right]\) ts-i \(\mathrm{i}_{1}\) i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i True show ? thesis
by (clarsimp simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) sb Write \({ }_{\mathrm{sb}}\) volatile)

\section*{next}
case False
note \(\mathrm{i}_{1}-\mathrm{i}=\) this
from \(i_{1}\)-bound have \(i_{1}\)-bound \({ }^{\prime}\) : \(i_{1}<\) length ts \(_{\text {sb }}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime}{ }^{\prime}\) sb)
hence \(\mathrm{i}_{1}\)-bound \({ }^{\prime \prime}: \mathrm{i}_{1}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
by auto
from ts-i \(\mathrm{i}_{1}\) False have \(\mathrm{ts}-\mathrm{i}_{1}\) : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) sb)
show ?thesis
proof (cases \(\mathrm{j}=\mathrm{i}\) )
case True
from outstanding-volatile-writes-unowned-by-others [ \(\mathrm{OF} \mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{i}\)-bound \(\mathrm{i}_{1}-\mathrm{i}\) ts- \(\mathrm{i}_{1}{ }^{\prime}\)
\(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]\)
have \(\left(\mathcal{O}_{\text {sb }} \cup\right.\) all-acquired sb\() \cap\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \operatorname{sb}_{1}=\{ \}\).
then show ?thesis
using True \(i_{1}-\mathrm{i}\) ts-j \(\mathrm{ts}_{\mathrm{sb}^{\prime}}\)-i i-bound
by (auto simp add: sb Write \({ }_{\text {sb }}\) volatile \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime} \mathcal{O}_{\text {sb }}{ }^{\prime}\) )
next
case False
from \(j\)-bound have \(j\)-bound \({ }^{\prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ts-j False have \(\mathrm{ts}^{\mathrm{j}} \mathrm{j}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from outstanding-volatile-writes-unowned-by-others
[OF \(i_{1}\)-bound \({ }^{\prime} j\)-bound \({ }^{\prime} i_{1}-j\) ts- \(i_{1}{ }^{\prime}\) ts- \({ }^{\prime}\) ]
show \(\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right) \cap\) outstanding-refs is-volatile-Write \(\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}_{1}=\{ \} \cdot\) qed
```

    qed
    qed
    next
show read-only-reads-unowned tssb}\mp@subsup{}{}{\prime
proof
fix n m

```

```

        assume n-bound: n < length tssb
        and m-bound: m < length ts sb}\mp@subsup{}{\mathrm{ ' }}{
        and neq-n-m: n\not=m
        and nth: ts sb
    ```

```

    from n-bound have n-bound': n < length ts tsb by (simp add: ts scb
    from m-bound have m-bound': m < length ts sb by (simp add: ts sb )
    show ( (\mathcal{O}\cup \cup all-acquired sb
                read-only-reads (acquired True (takeWhile (Not o is-volatile-Write (sb) sb n) (\mp@subsup{\mathcal{O}}{n}{})
                    (dropWhile (Not o is-volatile-Write sb})\mp@subsup{\operatorname{sb}}{\textrm{n}}{})
                {}
    proof (cases m=i)
        case True
        with neq-n-m have neq-n-i: n\not=i
    by auto
        with n-bound nth i-bound have nth': ts sb}!\textrm{n}=(\mp@subsup{\textrm{p}}{\textrm{n}}{},\mp@subsup{\textrm{is}}{\textrm{n}}{},\mp@subsup{\vartheta}{\textrm{n}}{},\mp@subsup{\textrm{sb}}{\textrm{n}}{},\mp@subsup{\mathcal{D}}{\textrm{n}}{},\mp@subsup{\mathcal{O}}{\textrm{n}}{},\mp@subsup{\mathcal{R}}{\textrm{n}}{}
    by (auto simp add: ts sb )
        note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth' ts sb-i
        then
        show ?thesis
    using True ts sb-i neq-n-i nth mth n-bound' m-bound' L-subset
    ```

```

        next
        case False
        note neq-m-i = this
        with m-bound mth i-bound have mth': ts smb
    by (auto simp add: ts sb ')
        show ?thesis
        proof (cases n=i)
    case True
    from read-only-reads-append [of ( }\mp@subsup{\mathcal{O}}{\mathbf{sb}}{}\cup\textrm{A}-\textrm{R})\mathrm{ (takeWhile (Not ○ is-volatile-Write}\mp@subsup{\textrm{s}}{\mathbf{s}}{}\mathrm{ )
    sbn
(dropWhile (Not \circ is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb }\mp@subsup{\textrm{s}}{\textrm{n}}{}\mathrm{ )]
have read-only-reads
(acquired True (takeWhile (Not o is-volatile-Write }\mp@subsup{\textrm{sb}}{\textrm{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{n}}{})(\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\cup\textrm{U}-\textrm{R})\mathrm{ )
(dropWhile (Not o is-volatile-Write sb}) s\mp@subsup{b}{n}{})\subseteq\mathrm{ read-only-reads ( }\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\cup\textrm{A
R) sb
by auto
with ts sb-i nth mth neq-m-i n-bound' True
read-only-reads-unowned [OF i-bound m-bound' False [symmetric] ts ts-i mth]
show ?thesis

```
by (auto \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) Write \(_{\text {sb }}\) volatile)
next
case False
with n -bound nth i-bound have \(\mathrm{nth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from read-only-reads-unowned [OF n-bound'm-bound' neq-n-m nth' mth'] False neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
next
show ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
fix \(i_{1}\) j p \(_{1}\) is \(\mathcal{O}_{1} \mathcal{O}_{1} \mathcal{D}_{1} \mathrm{xs}_{1} \mathrm{sb}_{1} \mathrm{p}_{\mathrm{j}}\) is \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume \(\mathrm{i}_{1}\)-bound: \(\mathrm{i}_{1}<\) length \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\)
assume j-bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume \(\mathrm{i}_{1}-\mathrm{j}: \mathrm{i}_{1} \neq \mathrm{j}\)
assume ts-i \({ }_{1}: \mathrm{ts}_{\text {sb }}{ }^{\prime} \mathrm{l}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
assume ts-j: \(\mathrm{ts}_{\mathrm{sb}}!\) j \(=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show \(\left(\mathcal{O}_{1} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{1}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}\)
proof (cases \(i_{1}=i\) )
case True
with \(\mathrm{i}_{1}-\mathrm{j}\) have \(\mathrm{i}-\mathrm{j}: \mathrm{i} \neq \mathrm{j}\)
by simp
from j-bound have j -bound \({ }^{\prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
hence j -bound \({ }^{\prime \prime}: \mathrm{j}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
by simp
from ts-j i-j have ts- \(\mathrm{j}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}\) )

show ?thesis
using \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) True \(\mathrm{ts}-\mathrm{i}_{1} \mathrm{i}\)-bound \(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) sb Write \(\mathrm{sb}_{\mathrm{s}}\) volatile)
next
case False
note \(\mathrm{i}_{1}-\mathrm{i}=\) this
from \(i_{1}\)-bound have \(i_{1}\)-bound': \(i_{1}<\) length ts \(_{\text {sb }}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
hence \(\mathrm{i}_{1}\)-bound \({ }^{\prime \prime}: \mathrm{i}_{1}<\) length (map owned \(\mathrm{ts}_{\mathrm{sb}}\) )
by simp
from ts- \(\mathrm{i}_{1}\) False have \(\mathrm{ts}-\mathrm{i}_{1}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}\) )
show ?thesis
proof (cases \(\mathrm{j}=\mathrm{i}\) )
case True
from ownership-distinct [OF \(\mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{i}\)-bound \(\mathrm{i}_{1}-\mathrm{i}\) ts- \(\left.\mathrm{i}_{1}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]\)
```

show ?thesis
using $\mathrm{ts}_{\mathrm{sb}}$-i True ts-j i-bound $\mathcal{O}_{\mathrm{sb}}{ }^{\prime}$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ sb Write $_{\text {sb }}$ volatile)
next
case False
from j-bound have j -bound': j < length $\mathrm{ts}_{\text {sb }}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ )
from ts-j False have ts-j${ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
from ownership-distinct [ $\mathrm{OF}_{\mathrm{i}_{1} \text {-bound }}{ }^{\prime} \mathrm{j}$-bound ${ }^{\prime} \mathrm{i}_{1}-\mathrm{j}$ ts- $\mathrm{i}_{1}{ }^{\prime}$ ts-j${ }^{\prime}$ ]
show ?thesis .
qed
qed
qed
qed
have valid-sharing': valid-sharing $\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ ts $_{\mathrm{sb}}{ }^{\prime}$
proof (intro-locales)
show outstanding-non-volatile-writes-unshared $\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (unfold-locales)
fix $\mathrm{j} \mathrm{p}_{\mathrm{j}}$ is $\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j -bound: $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
assume jth : $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show non-volatile-writes-unshared $\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$ sb $_{\mathrm{j}}$
proof (cases $\mathrm{i}=\mathrm{j}$ )
case True
with outstanding-non-volatile-writes-unshared [OF i-bound $\mathrm{ts}_{\mathrm{sb}} \mathrm{i}$ ]
i-bound jth $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ show ?thesis
by (clarsimp simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ sb Write $_{\text {sb }}$ volatile)
next
case False
from j -bound have j -bound': j < length $\mathrm{ts}_{\mathrm{sb}}$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
from jth False have jth ': $\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
from outstanding-non-volatile-writes-unshared [OF j-bound ${ }^{\prime}$ jth']
have unshared: non-volatile-writes-unshared $\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{j}$.
have $\forall \mathrm{a} \in \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus \mathrm{W} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}$. a $\notin$ outstanding-refs
is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$
proof -
\{
fix a
assume a-in: a $\in \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}$
hence $a-R: a \in R$
by clarsimp
assume $\mathrm{a}-\mathrm{in}-\mathrm{j}: \mathrm{a} \in$ outstanding-refs is-non-volatile-Write ${ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}$
have False
proof -
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF

```
```

        outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth ']]
        a-in-j
    have a }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb}\mp@subsup{\textrm{s}}{\textrm{j}}{
        by auto
        moreover
        with ownership-distinct [OF i-bound j-bound' False tssb-i jth'] a-R R-owned
    show False
        by blast
    qed
    }
thus ?thesis by blast
qed
from non-volatile-writes-unshared-no-outstanding-non-volatile-Write sb
[OF unshared this]
show ?thesis .
qed
qed
next
show sharing-consis ( }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (unfold-locales)
fix j p j isj }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{Xs}}\mp@subsup{\textrm{xj}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{
assume j-bound: j < length tssbb
assume jth: ts scb}\mp@subsup{}{}{\prime}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{xs}}{\textrm{j}}{\prime},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
show sharing-consistent ( }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\oplus}{\textrm{W}}{\prime}R\mp@subsup{\ominus}{\textrm{A}}{}L)\mp@subsup{\mathcal{O}}{j}{}\mp@subsup{sbb}{j}{
proof (cases i=j)
case True
with i-bound jth ts scb
show ?thesis
by (clarsimp simp add: ts ssb
next
case False
from j-bound have j-bound': j < length ts sb
by (auto simp add: ts sb ')
from jth False have jth': ts stb ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\textrm{xs}}{\textrm{j}}{},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
by (auto simp add: ts sb ')
from sharing-consis [OF j-bound' jth']
have consis: sharing-consistent }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\mathrm{ .
have acq-cond: all-acquired sb}\mp@subsup{\textrm{j}}{\textrm{j}}{}\cap\operatorname{dom}\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}-\operatorname{dom}(\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\oplus\mp@subsup{\textrm{w}}{}{\prime}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})={
proof -
{
fix a
assume a-acq: a }\in\mathrm{ all-acquired sbj
assume a }\in\operatorname{dom}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{
assume a-L: a }\in
have False

```
```

    proof -
    from ownership-distinct [OF i-bound j-bound' False ts ssb-i jth']
    have }\textrm{A}\cap\mathrm{ all-acquired sb}\mp@subsup{\textrm{s}}{\textrm{j}}{={{}
        by (auto simp add: sb Write sb volatile)
    with a-acq a-L L-subset
    show False
        by blast
    qed
    }
thus ?thesis
by auto
qed
have uns-cond: all-unshared sbj
proof -
{
fix a
assume a-uns: a }\in\mathrm{ all-unshared sbj
assume a }\not\in\textrm{L
assume a-R: a }\in
have False
proof -
from unshared-acquired-or-owned [OF consis] a-uns
have a }\in\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{\textrm{j}}{}\cup\mp@subsup{\mathcal{O}}{\textrm{j}}{}\mathrm{ by auto
with ownership-distinct [OF i-bound j-bound' False ts scb-i jth'] R-owned a-R
show False
by blast
qed
}
thus ?thesis
by auto
qed
from sharing-consistent-preservation [OF consis acq-cond uns-cond]
show ?thesis
by (simp add: ts sb
qed
qed
next
show read-only-unowned ( }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})\mp@subsup{\textrm{ts}}{\mathbf{sb}}{}\mp@subsup{}{}{\prime
proof

```

```

        assume j-bound: j < length ts sbb
    ```

```

        show }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cap\mathrm{ read-only ( }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})={
        proof (cases i=j)
            case True
            from read-only-unowned [OF i-bound ts }\mp@subsup{\textrm{s}}{\mathbf{sb}}{}\mathrm{ -i] R-owned A-R
            have (\mathcal{O}
    by (auto simp add: in-read-only-convs )
with jth ts scb-i i-bound True

```
show ?thesis
by (auto \(\operatorname{simp}\) add: \(\mathcal{O}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}\) )
next
case False
from j-bound have j-bound': \(\mathrm{j}<\) length ts \(_{\text {sb }}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) ')
with False j th have \(\mathrm{jth}^{\prime}: \operatorname{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) ')
from read-only-unowned [OF j-bound' jth \(]\)
have \(\mathcal{O}_{\mathrm{j}} \cap\) read-only \(\mathcal{S}_{\text {sb }}=\{ \}\).
moreover
from ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}-\mathrm{i}\) jth \(]\) R-owned
have \(\left(\mathcal{O}_{\text {sb }} \cup \mathrm{A}\right) \cap \mathcal{O}_{\mathrm{j}}=\{ \}\)
by (auto simp add: sb Write \({ }_{\text {sb }}\) volatile)
moreover note R-owned A-R
ultimately show ?thesis
by (fastforce simp add: in-read-only-convs split: if-split-asm)
qed
qed
next
show unowned-shared \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
show \(-\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot \operatorname{set} \mathrm{ts}_{\text {sb }}\right) \subseteq \operatorname{dom}\left(\mathcal{S}_{\text {sb }} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
proof -
have \(\mathrm{s}: ~ \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})\right.\) ' set \(\left.\mathrm{ts}_{\mathrm{sb}}\right)=\)
\(\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot\right.\) set \(\left.\mathrm{ts}_{\text {sb }}\right) \cup \mathrm{A}-\mathrm{R}\)
apply (unfold \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
apply (rule acquire-release-ownership-nth-update [OF R-owned i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ])
apply (rule local.ownership-distinct-axioms)
done
note unowned-shared L-subset A-R
then
show ?thesis
apply (simp only: s)
apply auto
done
qed
qed
next
show no-outstanding-write-to-read-only-memory \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof
fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show no-write-to-read-only-memory \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
with \(j\) th \(\mathrm{ts}_{\mathrm{sb}}\)-i i-bound no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathbf{s b}}-\mathrm{i}\) ] show ?thesis
by (auto simp add: \(\mathrm{sb} \mathrm{ts}_{\text {sb }}{ }^{\prime}\) Write \(_{\text {sb }}\) volatile)
next
case False
from j-bound have j -bound \({ }^{\prime}\) : \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
with False j th have \(\mathrm{jth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) ')
from no-outstanding-write-to-read-only-memory [OF j-bound' jth']
have nw: no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
have \(\mathrm{R} \cap\) outstanding-refs is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}=\{ \}\)
proof -
note dist \(=\) ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False ts stb \(_{\text {sbi }}\) jth \(]\)
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth \({ }^{\prime}\) ]
dist
have outstanding-refs is-non-volatile-Write \({ }_{\text {sb }} \operatorname{sb}_{\mathrm{j}} \cap \mathcal{O}_{\mathrm{sb}}=\{ \}\)
by auto
moreover
from outstanding-volatile-writes-unowned-by-others [OF j-bound' i-bound
False [symmetric] jth \({ }^{\prime}\) ts \(_{\text {sb-i }}\) ]
have outstanding-refs is-volatile-Write \({ }_{\text {sb }} \operatorname{sb}_{\mathrm{j}} \cap \mathcal{O}_{\text {sb }}=\{ \}\)
by auto
ultimately have outstanding-refs is-Write \({ }_{\text {sb }} \operatorname{sb}_{j} \cap \mathcal{O}_{\text {sb }}=\{ \}\)
by (auto simp add: misc-outstanding-refs-convs)
with R-owned
show ?thesis by blast
qed
then
have \(\forall \mathrm{a} \in\) outstanding-refs is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{j}}\).
\(\mathrm{a} \in\) read-only \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \longrightarrow \mathrm{a} \in\) read-only \(\mathcal{S}_{\mathrm{sb}}\)
by (auto simp add: in-read-only-convs)
from no-write-to-read-only-memory-read-only-reads-eq [OF nw this] show ?thesis .
qed
qed
qed
from direct-memop-step.WriteVolatile [OF]
have (Write True a (D, f) A L R W\# is',
\(? \vartheta,(), \mathrm{m}, \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb (dom \(\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}, \mathcal{S}\right) \rightarrow\)
(is \({ }^{\prime}, ? \vartheta,(), \mathrm{m}(\mathrm{a}:=\mathrm{v})\), True, acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\), Map.empty, \(\mathcal{S}\)
\(\left.\oplus_{W} R \ominus_{\mathrm{A}} \mathrm{L}\right)\)
by (simp add: \(\mathrm{f}-\mathrm{v}^{\prime}[\) symmetric \(\left.]\right)\)
from direct-computation.Memop [OF i-bound' ts-i this]
have store-step:
\((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(? \mathrm{ts}^{\prime}, \mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
```

have sb'split:
$\mathrm{sb}^{\prime}=$ takeWhile (Not $\circ$ is-volatile-Write $\mathrm{sb}_{\mathrm{sb}}$ ) $\mathrm{sb}^{\prime} @$
dropWhile (Not o is-volatile-Write ${ }_{\text {sb }}$ ) sb ${ }^{\prime}$
by simp
from reads-consis
have no-vol-reads: outstanding-refs is-volatile-Read $\mathrm{Re}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\{ \}$
by (simp add: sb Write ${ }_{\text {sb }}$ True)
hence outstanding-refs is-volatile-Read ${ }_{\mathbf{s b}}$ (takeWhile (Not $\circ$ is-volatile-Write $_{\text {sb }}$ ) sb')
$=\{ \}$
by (auto simp add: outstanding-refs-conv dest: set-takeWhileD)
moreover
have outstanding-refs is-volatile-Write ${ }_{\text {sb }}$
$\left(\right.$ takeWhile $\left.^{(\text {Not } \circ \text { is-volatile-Write }}{ }_{\text {sb }}\right)$ sb $\left.{ }^{\prime}\right)=\{ \}$
proof -
have $\forall \mathrm{r} \in$ set ( takeWhile ( $^{(N o t} \circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ ) sb ${ }^{\prime}$ ). $\neg$ (is-volatile-Write ${ }_{\text {sb }} \mathrm{r}$ )
by (auto dest: set-takeWhileD)
thus? ?thesis
by (simp add: outstanding-refs-conv)
qed
ultimately
have no-volatile:
outstanding-refs is-volatile (takeWhile (Not $\circ$ is-volatile-Write $\left._{\text {sb }}\right) \mathrm{sb}^{\prime}$ ) $=\{ \}$
by (auto simp add: outstanding-refs-conv is-volatile-split)

```

\section*{moreover}
```

from no-vol-reads have $\forall \mathrm{r} \in$ set sb $^{\prime} . \neg$ is-volatile-Read ${ }_{\text {sb }} \mathrm{r}$ by (fastforce simp add: outstanding-refs-conv is-volatile-Read ${ }_{\mathbf{s b}}$-def split: memref.splits)
hence $\forall \mathrm{r} \in$ set sb' $^{\prime}$. (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) $\mathrm{r}=($ Not $\circ$ is-volatile) r by (auto simp add: is-volatile-split)
hence takeWhile-eq: (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb $\left.{ }^{\prime}\right)=$ (takeWhile (Not o is-volatile) sb')
apply -
apply (rule takeWhile-cong)
apply auto
done
from leq
have leq': length $\mathrm{ts}_{\mathrm{sb}}=$ length $? \mathrm{ts}^{\prime}$ by simp
hence i -bound-ts': i < length ?ts' using i-bound by simp
from is'-sim
have is'-sim-split:
instrs
(takeWhile (Not $\circ$ is-volatile-Write ${ }_{\text {sb }}$ ) sb' @

```
dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(@_{\text {is }}^{\text {sb }}=\)
is' @ prog-instrs (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb' @
dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb')
by (simp add: sb'-split [symmetric])
from reads-consistent-flush-all-until-volatile-write [OF «valid-ownership-and-sharing \(\mathcal{S}_{\text {sb }}\) \(\mathrm{ts}_{\mathrm{sb}}\) )
i -bound \(\mathrm{ts}_{\mathrm{sb}}\)-i reads-consis]
have reads-consistent True (acquired True ?take-sb \(\mathcal{O}_{\text {sb }}\) ) m (Write \(\mathrm{s}_{\mathrm{sb}}\) True a (D,f) v A L R W\#sb')
by ( simp add: m sb Write \(_{\mathrm{sb}}\) volatile)
hence reads-consistent True (acquired True ?take-sb \(\left.\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right)(\mathrm{m}(\mathrm{a}:=\mathrm{v})) \mathrm{sb}^{\prime}\) by simp
from reads-consistent-takeWhile [OF this]
have r-consis': reads-consistent True (acquired True ?take-sb \(\left.\mathcal{O}_{s b} \cup A-R\right)(m(a:=v))\) (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \()\).
from last-prog have last-prog-sb': last-prog \(\mathrm{p}_{\mathbf{s b}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\) by (simp add: sb Write \({ }_{\text {sb }}\) )
from valid-write-sops [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
have \(\forall\) sop \(\in\) write-sops sb'. valid-sop sop by (auto simp add: sb Write \({ }_{\text {sb }}\) )
hence valid-sop': \(\forall\) sop \(\in\) write-sops (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) sb'). valid-sop sop
by (fastforce dest: set-takeWhileD simp add: in-write-sops-conv)
from no-volatile
have no-volatile-Read \({ }_{\text {sb }}\) :
outstanding-refs is-volatile-Read \({ }_{\mathbf{s b}}\left(\operatorname{takeWhile}\left(\right.\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\mathbf{s b}}\right) \mathrm{sb}^{\prime}\right)=\) \{\}
by (auto simp add: outstanding-refs-conv is-volatile-Read \({ }_{\mathbf{s b}}\)-def split: memref.splits)
from flush-store-buffer-append [OF i-bound-ts' is'-sim-split, simplified,
OF causal-program-history-sb' ts'-i refl last-prog-sb' r-consis' hist-consis' valid-sop \({ }^{\prime}\) dist-sb' no-volatile-Read \({ }_{\text {sb }}\)-volatile-reads-consistent [OF no-volatile-Read \({ }_{\text {sb }}\) ], where \(\left.\mathcal{S}=\left(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right]\)
```

obtain is" where
is"-sim: instrs (dropWhile (Not o is-volatile-Write sb ) sb)
is" @ prog-instrs (dropWhile (Not o is-volatile-Write sb) sb) and
steps: (?ts', m(a := v), \mathcal{S}\oplus\textrm{w}R}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})=>\mp@subsup{|}{\textrm{d}}{}\mp@subsup{}{}{*
(ts[i := (last-prog (hd-prog p pbb (dropWhile (Not o is-volatile-Write sb
(takeWhile (Not o is-volatile-Write sb) sb'),
is",

```
```

                    \vartheta 点 |
                    read-tmps (dropWhile (Not o is-volatile-Write sb
    (), True, acquired True (takeWhile (Not ० is-volatile-Write sb
(acquired True ?take-sb }\mp@subsup{\mathcal{O}}{sb}{}\cup\textrm{A}-\textrm{R}\mathrm{ ),
release (takeWhile (Not O is-volatile-Write sb) sb')
(dom (\mathcal{S}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})) Map.empty)],
flush (takeWhile (Not o is-volatile-Write esb) sb') (m(a := v)),
share (takeWhile (Not o is-volatile-Write sb) sb
by (auto)
note sim-flush = r-rtranclp-rtranclp [OF store-step steps]
moreover
note flush-commute =
flush-flush-all-until-volatile-write-Write }\mp@subsup{e}{sb}{}\mathrm{ -volatile-commute [OF i-bound ts }\mp@subsup{\textrm{s}}{\mathbf{sb}}{}\mathrm{ -i [simplified
sb Write sb True]
outstanding-refs-is-Write sb-takeWhile-disj a-notin-others']
from last-prog-hd-prog-append' [where sb=(takeWhile (Not \circ is-volatile-Write sb) sb
and sb'=(dropWhile (Not o is-volatile-Write esb) sb}\mp@subsup{}{}{\prime})\mathrm{ ,
simplified sb'-split [symmetric], OF hist-consis' last-prog-sb']
have last-prog-eq:
last-prog (hd-prog p pb (dropWhile (Not \circ is-volatile-Write sb
(takeWhile (Not o is-volatile-Write sb) sb
hd-prog p sb (dropWhile (Not o is-volatile-Write esb) sb})
have take-emtpy: takeWhile (Not \circ is-volatile-Write }\mp@subsup{}{\textrm{sb}}{})(\textrm{r}\#\textrm{sb})=[
by (simp add: Write sb True)
have dist-sb': }\forall\textrm{i}\mathrm{ p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
i < length ts sb }
ts sb ! i = (p, is, \vartheta, sb,\mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow
(all-shared (takeWhile (Not o is-volatile-Write sb) sb) U
all-unshared (takeWhile (Not \circ is-volatile-Write sb) sb) \cup
all-acquired (takeWhile (Not ० is-volatile-Write sb) sb)) \cap
(all-shared (takeWhile (Not \circ is-volatile-Write sb) sb})
all-unshared (takeWhile (Not ० is-volatile-Write sb) sb') \cup
all-acquired (takeWhile (Not \circ is-volatile-Write esb}) sb`))
{}
proof -
{
fix j pj isj }\mp@subsup{\mathcal{O}}{\textrm{j}}{\mp@code{R}}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\vartheta}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\textrm{x
assume j-bound: j < length ts sb
assume jth: ts sbb}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{i}}{\textrm{j}}{},\mp@subsup{\vartheta}{\textrm{j}}{,},\mp@subsup{\textrm{Sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
assume x-shared: x }\in\mathrm{ all-shared (takeWhile (Not ० is-volatile-Write sb) sb j) U
all-unshared (takeWhile (Not o is-volatile-Writesb) sb j) U
all-acquired (takeWhile (Not o is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{s}}{}\mathrm{ ) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )

```
```

assume x-sb': x (all-shared (takeWhile (Not o is-volatile-Write mb) sb})
all-unshared (takeWhile (Not ० is-volatile-Write sb) sb') \cup
all-acquired (takeWhile (Not o is-volatile-Write sb) sb
have False
proof (cases i=j)
case True with x-shared ts tsb-i jth show False by (simp add: sb volatile Writesb
next
case False
from x-shared all-shared-acquired-or-owned [OF sharing-consis [OF j-bound
jth]]
unshared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
all-shared-append [of (takeWhile (Not o is-volatile-Write sbb sb j)
(dropWhile (Not \circ is-volatile-Write sb}) s\mp@subsup{b}{j}{})
all-unshared-append [of (takeWhile (Not O is-volatile-Write sb) sb
(dropWhile (Not o is-volatile-Write sb) sbj
all-acquired-append [of (takeWhile (Not ० is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb }\mp@subsup{\textrm{b}}{\textrm{j}}{}\mathrm{ )
(dropWhile (Not \circ is-volatile-Write sb) sb j)]
have x }\in\mathrm{ all-acquired sbj }\cup\mp@subsup{\mathcal{O}}{j}{
by auto
moreover
from x-sb' all-shared-acquired-or-owned [OF sharing-consis [OF i-bound ts sb-i]]
unshared-acquired-or-owned [OF sharing-consis [OF i-bound ts
all-shared-append [of (takeWhile (Not o is-volatile-Write sb) sb
(dropWhile (Not * is-volatile-Write sb) sb
all-unshared-append [of (takeWhile (Not \circ is-volatile-Write sb) sb')
(dropWhile (Not \circ is-volatile-Write sb) sb')]
all-acquired-append [of (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{}\mathrm{ ) sb')
(dropWhile (Not o is-volatile-Write sb) sb
have x }\in\mathrm{ all-acquired sb }\cup\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{
by (auto simp add: sb Write sb volatile)
moreover
note ownership-distinct [OF i-bound j-bound False ts ts-i jth]
ultimately show False by blast
qed
}
thus ?thesis by blast
qed
have dist-R-L-A: }\forall\textrm{j}\textrm{p}\mathrm{ is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb
j < length ts sb }\longrightarrow\textrm{i}\not=\textrm{j}
tssb ! j = (p, is,\vartheta, sb, \mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow
(all-shared sb \cup all-unshared sb \cup all-acquired sb) \cap (R\cupL\cupA)={}
proof -
{
fix j pj isj }\mp@subsup{\mathcal{O}}{\textrm{j}}{\mp@code{R}}\mp@subsup{\mathcal{R}}{\textrm{j}}{}\mp@subsup{\mathcal{D}}{\textrm{j}}{}\mp@subsup{\vartheta}{\textrm{j}}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{}\textrm{x
assume j-bound: j < length ts sb
assume neq-i-j: i\not= j
assume jth: ts sbb}!\textrm{j}=(\mp@subsup{\textrm{p}}{\textrm{j}}{,},\mp@subsup{\textrm{i}}{\textrm{j}}{},\mp@subsup{\vartheta}{\textrm{j}}{,},\mp@subsup{\textrm{Sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
assume x-shared: x }\in\mathrm{ all-shared sbj }
all-unshared sb}\mp@subsup{j}{j}{}

```
```

                    all-acquired sbj
        assume x-R-L-A: x }\inR\cupR\cupL\cup
        have False
        proof -
            from x-shared all-shared-acquired-or-owned [OF sharing-consis [OF j-bound
    jth]]
unshared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
have x }\in\mathrm{ all-acquired sb}\mp@subsup{\textrm{s}}{\textrm{j}}{}\cup\mp@subsup{\mathcal{O}}{\textrm{j}}{
by auto
moreover
from x-R-L-A R-owned L-subset
have x }\in\mathrm{ all-acquired sb }\cup\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{
by (auto simp add: sb Writesb volatile)
moreover
note ownership-distinct [OF i-bound j-bound neq-i-j ts sb-i jth]
ultimately show False by blast
qed
}
thus ?thesis by blast
qed
from local.ownership-distinct-axioms have ownership-distinct ts tsb .
from local.sharing-consis-axioms have sharing-consis }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mathrm{ .
note share-commute=
share-all-until-volatile-write-flush-commute [OF take-empty «ownership-distinct
ts
have rel-commute-empty:
release (takeWhile (Not o is-volatile-Write sb})\mathrm{ ) sb') (dom S U R - L) Map.empty =
release (takeWhile (Not ○ is-volatile-Write sb) sb')(dom S S sb }\cup\textrm{R}-\textrm{L}
Map.empty
proof -
{
fix a
assume a-in: a }\in\mathrm{ all-shared (takeWhile (Not ○ is-volatile-Write
have (a\in(dom S \cupR-L))=(a\in(dom S Sbb }\cup\textrm{R}-\textrm{L})
proof -
from all-shared-acquired-or-owned [OF sharing-consis [OF i-bound ts tsbi]] a-in
all-shared-append [of (takeWhile (Not \circ is-volatile-Write esb) sb}\mathrm{ ) (dropWhile
(Not o is-volatile-Write esb) sb
have a }\in\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb
by (auto simp add: sb Write sb volatile)
from share-all-until-volatile-write-thread-local [OF <ownership-distinct ts sb >
<sharing-consis }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mathrm{ \ i-bound ts stb
have S a = \mathcal{Sba}
by (auto simp add: sb Write sb volatile }\mathcal{S}\mathrm{ )
then show ?thesis
by (auto simp add: domIff)
qed

```
```

    }
    then show ?thesis
    apply -
    apply (rule release-all-shared-exchange)
    apply auto
    done
    qed
{

```

```

assume jth: ts ssb}!\mp@code{j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{},\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\vartheta}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
assume j-bound: j < length ts sb
assume neq: i = j
have release (takeWhile (Not ० is-volatile-Write sb ) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
(dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\cup\textrm{R}-\textrm{L})\mp@subsup{\mathcal{R}}{\textrm{j}}{
= release (takeWhile (Not o is-volatile-Write }\mp@subsup{}{\mathbf{sb}}{})\mp@subsup{\textrm{sb}}{\textrm{j}}{}
(dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mp@subsup{\mathcal{R}}{\textrm{j}}{
proof -
{
fix a
assume a-in: a }\in\mathrm{ all-shared (takeWhile (Not o is-volatile-Write esb) sb }\mp@subsup{\textrm{j}}{\textrm{j}}{}\mathrm{ )
have}(\textrm{a}\in(\operatorname{dom}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\cup\textrm{R}-\textrm{L}))=(\textrm{a}\in\operatorname{dom}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}
proof -
from ownership-distinct [OF i-bound j-bound neq ts sb-i jth]
have A-dist: A \cap (\mathcal{O}
by (auto simp add: sb Write sb volatile)
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]] a-in
all-shared-append [of (takeWhile (Not \circ is-volatile-Write sb) sb
(dropWhile (Not o is-volatile-Write sb ) sb j}\mathrm{ )]
have a-in: a }\in\mp@subsup{\mathcal{O}}{j}{}\cup\mathrm{ all-acquired sb
by auto
with ownership-distinct [OF i-bound j-bound neq ts sto-i jth]
have a }\not\in(\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb) by auto
with A-dist R-owned A-R A-shared-owned L-subset a-in
obtain a }\not\inR\mathrm{ and a }\not\in
by fastforce
then show ?thesis by auto
qed
}
then
show ?thesis
apply -
apply (rule release-all-shared-exchange)
apply auto
done
qed

```
\}
note release-commute \(=\) this
have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right.\right.\right.\), Map.empty \(\left.\left.)\right], \mathrm{m}_{\mathrm{sb}}(\mathrm{a}:=\mathrm{v}), \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\)

(takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb'),
is \({ }^{\prime \prime}\), \(\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\operatorname{dom} \vartheta_{\mathrm{sb}}-\right.\)
read-tmps (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \({ }^{\prime}\) ), (),True, acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb')
(acquired True ? take-sb \(\mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}\) ), release (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) sb')
(dom ( \(\left.\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) ) Map.empty)],
flush (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb') ( \(\mathrm{m}(\mathrm{a}:=\mathrm{v})\) ), share (takeWhile (Not o is-volatile-Write \({ }_{s b}\) ) sb') \(\left.\left(\mathcal{S} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\right)\)
apply (rule sim-config.intros)
apply (simp add: flush-commute m)
apply (clarsimp simp add: \(\mathcal{S}_{\mathrm{sb}^{\prime}}{ }^{\prime} \mathcal{S}\) share-commute simp del: restrict-restrict)
using leq
apply simp
using i-bound i-bound' ts-sim \(\mathcal{D}\)
apply (clarsimp simp add: Let-def nth-list-update is'"-sim last-prog-eq sb Write \({ }_{\text {sb }}\) volatile \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
rel-commute-empty
split: if-split-asm )
apply (rule conjI)
apply blast
apply clarsimp
apply (frule (2) release-commute)
apply clarsimp
apply fastforce
done

\section*{ultimately}
show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct \({ }^{\prime}\)
valid-dd' valid-sops' load-tmps-fresh' enough-flushs \({ }^{\prime}\)
valid-program-history' valid \(^{\prime}\)
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply simp add: \(\mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\) )
next
case False
note non-vol \(=\) this
from flush Write \(_{\text {sb }}\) False
obtain
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and
\(\mathcal{S}_{\mathrm{sb}}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\) and
\[
\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}
\]
by cases (auto simp add: sb)
from non-volatile-owned non-vol have a-owned: \(\mathrm{a} \in \mathcal{O}_{\text {sb }}\) by simp

\section*{\{}
fix j
fix \(\mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{sbj}} \mathcal{O}_{\mathrm{j}} \mathcal{D}_{\mathrm{sbj}} \vartheta_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(^{\text {ts }}{ }_{\mathrm{sb}}\)
assume \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{j}}, \mathrm{Sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{sbj}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
have a \(\notin\) unforwarded-non-volatile-reads (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \({ }_{j}\) ) \(\}\)
proof
let ?take-sb \({ }_{j}=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb \(_{j}\)
let ? drop-sb \({ }_{j}=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\)
assume a-in: a \(\in\) unforwarded-non-volatile-reads ?drop-sb \({ }_{j}\{ \}\)
from a-unowned-by-others [rule-format, OF j-bound neq-i-j] ts \(\mathrm{sb}_{\mathrm{sb}}\) - j
obtain a-unowned: a \(\notin \mathcal{O}_{\mathrm{j}}\) and a-unacq: a \(\notin\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto
with all-acquired-append [of ?take-sb \({ }_{j}\) ?drop-sb \(\left.{ }_{j}\right]\) ac-
quired-takeWhile-non-volatile-Write \({ }_{\text {sb }}\left[\right.\) of \(\operatorname{sb}_{j} \mathcal{O}_{j}\) ]
have a-unacq-take: a \(\notin\) acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\)
by (auto)
note nvo-j \(=\) outstanding-non-volatile-refs-owned-or-read-only [OF j-bound \(\mathrm{ts}_{\mathrm{sb}}\) - j ]
from non-volatile-owned-or-read-only-drop [OF nvo-j]
have nvo-drop-j: non-volatile-owned-or-read-only True (share ?take-sb \(\mathcal{S}_{\mathrm{sb}}\) ) (acquired True ?take-sb \(\mathcal{O}_{j}\) ) ?drop-sb \({ }_{j}\).
from in-unforwarded-non-volatile-reads-non-volatile-Read \({ }_{\mathbf{s b}}\) [OF a-in]
have a-in': a \(\in\) outstanding-refs is-non-volatile-Read \({ }_{\mathrm{sb}}\) ?drop-sb \(\mathrm{sb}_{\mathrm{j}}\).
from non-volatile-owned-or-read-only-outstanding-refs [OF nvo-drop-j] a-in'
have \(\mathrm{a} \in\) acquired True ?take-sb \(\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \cup\) all-acquired ?drop-sb \(\mathrm{S}_{\mathrm{j}} \cup\) read-only-reads (acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \(\mathrm{j}_{\mathrm{j}}\) by (auto simp add: misc-outstanding-refs-convs)
moreover
from acquired-append [of True ?take-sb \({ }_{\mathrm{j}}\) ? drop- \(^{\mathrm{sb}} \mathrm{b}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ] acquired-all-acquired [of True ?take-sb \({ }_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}\) ]
all-acquired-append [of ?take-sb \({ }_{j}\) ?drop-sb \({ }_{j}\) ]
have acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}} \cup\) all-acquired ?drop-sb \(\mathcal{S}_{\mathrm{j}} \subseteq \mathcal{O}_{\mathrm{j}} \cup\) all-acquired sb \(\mathrm{j}_{\mathrm{j}}\)
by auto
ultimately
have a \(\in\) read-only-reads (acquired True ?take-sb \(\mathcal{O}_{\mathrm{j}}\) ) ?drop-sb \({ }_{\mathrm{j}}\) using a-owned ownership-distinct [OF i-bound j-bound neq-i-j \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i} \mathrm{ts}_{\mathrm{sb}}\)-j] by auto
with read-only-reads-unowned [OF j-bound i-bound neq-i-j [symmetric] \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{j}} \mathrm{ts}_{\mathbf{s b}}-\mathrm{i}\) ] a-owned
show False
by auto
qed
\(\}\) note a-notin-unforwarded-non-volatile-reads-drop \(=\) this
```

have valid-reads': valid-reads $\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$
proof (unfold-locales)
fix $\mathrm{j} \mathrm{p}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}$
assume j-bound: $\mathrm{j}<$ length ts $_{\text {sb }}{ }^{\prime}$
assume ts- $\mathrm{j}:$ ts $_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show reads-consistent False $\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}_{\mathrm{j}}$
proof (cases $\mathrm{i}=\mathrm{j}$ )
case True
from reads-consis ts-j j-bound sb show ?thesis
by (clarsimp simp add: True $\mathrm{m}_{\mathrm{sb}}{ }^{\prime}$ Write $_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}$ False
reads-consistent-pending-write-antimono)
next
case False
from $j$-bound have $j$-bound ${ }^{\prime}$ : $\mathrm{j}<$ length ts $_{\text {sb }}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}$ )
moreover from ts-j False have $\mathrm{ts}^{\mathrm{j}} \mathrm{j}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
using j -bound by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
ultimately have consis-m: reads-consistent False $\mathcal{O}_{\mathrm{j}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}$
by (rule valid-reads)
from a-unowned-by-others [rule-format, OF j-bound' False] ts-j'
have a-unowned: $\mathrm{a} \notin \mathcal{O}_{\mathrm{j}} \cup$ all-acquired $\mathrm{sb}_{\mathrm{j}}$
by simp
let ?take-sb ${ }_{\mathrm{j}}=$ takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathrm{sb}}$ ) $\mathrm{sb}_{\mathrm{j}}$
let ?drop-sb ${ }_{j}=$ dropWhile (Not $\circ$ is-volatile-Write ${ }_{s b}$ ) $\mathrm{sb}_{\mathrm{j}}$
from a-unowned acquired-reads-all-acquired [of True ?take-sb $\mathrm{O}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}}$ ]
all-acquired-append [of ?take-sb ${ }_{j}$ ?drop-sb ${ }_{j}$ ]
have a-not-acq-reads: a $\notin$ acquired-reads True ?take-sb ${ }_{j} \mathcal{O}_{\mathrm{j}}$
by auto
moreover

```
        note a-unfw = a-notin-unforwarded-non-volatile-reads-drop [OF j-bound' ts-j \({ }^{\prime}\) False]
        ultimately
        show ?thesis
            using reads-consistent-mem-eq-on-unforwarded-non-volatile-reads-drop [where
\(\mathrm{W}=\{ \}\) and
    A=unforwarded-non-volatile-reads ?drop-sb \(\left\} \cup\right.\) acquired-reads True ?take-sb \({ }_{j} \mathcal{O}_{\mathrm{j}}\) and
    \(\mathrm{m}^{\prime}=\left(\mathrm{m}_{\mathrm{sb}}(\mathrm{a}:=\mathrm{v})\right), \mathrm{OF}--\) consis-m]
            by (fastforce simp add: \(\mathrm{m}_{\mathrm{sb}}\) )
```

    qed
    qed
have valid-own': valid-ownership }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts tsbi] sb
have non-volatile-owned-or-read-only False }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{\textrm{sb}}{}{\prime
by (auto simp add: Write sb False)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by (simp add: t\mp@subsup{s}{\textrm{sb}}{\prime}}\mp@subsup{}{}{\prime}\mp@subsup{W}{}{\prime
qed
next
show outstanding-volatile-writes-unowned-by-others ts sb}\mp@subsup{}{}{\prime
proof -
from sb
have out: outstanding-refs is-volatile-Write }\mp@subsup{}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{}{\prime}\subseteq\mathrm{ outstanding-refs is-volatile-Write sb
sb
by (auto simp add: Write}\mp@subsup{e}{\mathrm{ sb }}{}\mathrm{ False)
have acq: all-acquired sb}\mp@subsup{}{}{\prime}\subseteq\mathrm{ all-acquired sb
by (auto simp add: Write sb False sb)
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}-\textrm{i}\mathrm{ out acq]
show ?thesis by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{}{}{\prime}\mathrm{ Write esb False }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mathrm{ )
qed
next
show read-only-reads-unowned tssb}\mp@subsup{}{}{\prime
proof -
have ro: read-only-reads (acquired True (takeWhile (Not \circ is-volatile-Writesb) sb
O
(dropWhile (Not o is-volatile-Write sb) sb
\subseteq read-only-reads (acquired True (takeWhile (Not o is-volatile-Write esb) sb) \mathcal { O } _ { \mathrm { sb } } )
(dropWhile (Not o is-volatile-Write esb) sb)
by (auto simp add: sb Write sb non-vol)
have }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb}\mp@subsup{}{}{\prime}\subseteq\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb
by (auto simp add: sb Write esb non-vol)
from read-only-reads-unowned-nth-update [OF i-bound ts sb-i ro this]
show ?thesis
by (simp add: ts sb
qed
next
show ownership-distinct tssb}\mp@subsup{}{}{\prime
proof -
have acq: all-acquired sb}\mp@subsup{}{}{\prime}\subseteq\mathrm{ all-acquired sb
by (auto simp add: Write sb False sb)
with ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound ts tsbi]
show ?thesis by (simp add: ts sbb}\mp@subsup{}{}{\prime}\mp@subsup{W}{\mathrm{ Write }}{\mathrm{ sb }
qed

```
qed
have valid-sharing': valid-sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb Write sb \(_{\text {b }}\) False)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \text { - }}\) ]
have sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb Write \({ }_{\text {sb }}\) False)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - i ]
]
show read-only-unowned \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) by (simp add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\text {sb }}\)-i]
have no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb Write \({ }_{\text {sb }}\) False)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}{ }^{\mathrm{sb}}\) )
qed
from is-sim
obtain is-sim: instrs (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) @ is \(\mathrm{s}_{\mathrm{sb}}=\) is @ prog-instrs (dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb)
by (simp add: suspends sb Write \({ }_{\text {sb }}\) False)
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by blast

\section*{moreover}
note flush-commute \(=\)
flush-all-until-volatile-write-Write \({ }_{s b}\)-non-volatile-commute [OF i-bound \(\mathrm{ts}_{\mathbf{s b}}\) - [simplified sb Write \({ }_{\text {sb }}\) non-vol]
outstanding-refs-is-Write \({ }_{\text {sb-}}\)-takeWhile-disj a-notin-others']
note share-commute \(=\)
share-all-until-volatile-write-update-sb [of sb' sb, OF - i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i, simplified sb Write \(_{\text {sb }}\) False, simplified]
have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}_{\mathrm{sb}}(\mathrm{a}:=\mathrm{v}), \mathcal{S}_{\mathrm{sb}}\right) \sim\)
(ts,m,S)
apply (rule sim-config.intros)
apply (simp add: m flush-commute)
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\text {sb }}\) ' share-commute)
using leq
apply simp
using i-bound i-bound \({ }^{\prime}\) is-sim ts-i ts-sim \(\mathcal{D}\)
apply (clarsimp simp add: Let-def nth-list-update suspends sb Write \({ }_{\text {sb }}\) False \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) split: if-split-asm )
done
ultimately
show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' tmps-distinct' \(\mathrm{m}_{\text {sb }}{ }^{\prime}\)
valid-dd' valid-sops' load-tmps-fresh' enough-flushs' valid-program-history' valid \({ }^{\prime}\) \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
qed
next
case \(\left(\operatorname{Read}_{\text {sb }}\right.\) volatile a t v)
from flush this obtain \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\) and \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}\)
by cases (auto simp add: sb)
\[
\text { have valid-own': valid-ownership } \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}
\] proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - \(]\) ] sb
have non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: \(\operatorname{Read}_{\mathrm{sb}}\) )
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \operatorname{Read}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
qed
next
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from sb
have out: outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}^{\prime} \subseteq\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb by (auto simp add: \(\operatorname{Read}_{\text {bb }}\) )
have acq: all-acquired \(\mathrm{sb}^{\prime} \subseteq\) all-acquired sb by (auto simp add: \(\operatorname{Read}_{\mathrm{sb}} \mathrm{sb}\) )
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound \(\mathrm{ts}_{\text {sb }}\)-i out acq]
show ?thesis by (simp add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \operatorname{Read}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\right)\)
qed next
show read-only-reads-unowned \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have ro: read-only-reads (acquired True (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb') \(\mathcal{O}_{\text {sb }}\) ) (dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb') \(\subseteq\) read-only-reads (acquired True (takeWhile (Not o is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) \(\mathcal{O}_{\mathrm{sb}}\) ) (dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb) by (auto simp add: sb \(\operatorname{Read}_{\mathrm{sb}}\) )
have \(\mathcal{O}_{\text {sb }} \cup\) all-acquired \(\mathrm{sb}^{\prime} \subseteq \mathcal{O}_{\text {sb }} \cup\) all-acquired sb by (auto simp add: sb Read \(_{\text {sb }}\) )
from read-only-reads-unowned-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{s}_{\mathrm{sb}}}\)-i ro this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime}{ }^{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed next
show ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have acq: all-acquired \(\mathrm{sb}^{\prime} \subseteq\) all-acquired sb by (auto simp add: \(\operatorname{Read}_{\text {sb }} \mathrm{sb}\) )
with ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\) ]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}{ }^{\prime} \operatorname{Read}_{\text {sb }} \mathcal{O}_{\text {sb }}\) )
qed
qed
have valid-sharing \({ }^{\prime}\) : valid-sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\) proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) - ]
have non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb \(\operatorname{Read}_{\text {sb }}\) )
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by \(\left(\operatorname{simp}\right.\) add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\) next
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have sharing-consistent \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) sb \(^{\prime}\)
by (auto simp add: sb Read \(_{\text {sb }}\) )
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )

\section*{next}

]
show read-only-unowned \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by \(\left(\operatorname{simp}\right.\) add: \(\left.\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\) next
from no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathbf{s b}_{\mathbf{b}}-\mathrm{i}}\) ]
have no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb Read \({ }_{\text {sb }}\) )
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\left.\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\right)\)
qed
have valid-reads \({ }^{\prime}\) : valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-reads [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have reads-consistent False \(\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}^{\prime}\) by (simp add: sb Read sb )
from valid-reads-nth-update [OF i-bound this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-program-history \({ }^{\prime}\) : valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}} \mathrm{i}}\) ]
have causal-program-history is \({ }_{\text {sb }} \mathrm{sb}\).
then have causal': causal-program-history is sb \(^{\text {sb }}{ }^{\prime}\)
by (simp add: sb Read sb causal-program-history-def)
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\) - ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by (simp add: sb \(\operatorname{Read}_{\text {sb }}\) )
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}\) )
qed
from is-sim
have is-sim: instrs (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) @ is \(\mathrm{s}_{\mathrm{sb}}=\)
is @ prog-instrs (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb)
by ( \(\operatorname{simp}\) add: sb Read \(_{\text {sb }}\) suspends)
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have \(\vartheta_{\mathrm{sb}}-\mathrm{v}\) : \(\vartheta_{\mathrm{sb}} \mathrm{t}=\) Some v
by (simp add: history-consistent-access-last-read sb \(\operatorname{Read}_{\text {sb }}\) split:option.splits)
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by blast
moreover
note flush-commute \(=\) flush-all-until-volatile-write-Read \({ }_{\mathbf{s b}}\)-commute \(\left[\mathrm{OF}\right.\) i-bound \(\mathrm{ts}_{\mathbf{s b}}-\mathrm{i}\) [simplified sb \(\left.\operatorname{Read}_{\text {sb }}\right]\) ]
note share-commute \(=\)
share-all-until-volatile-write-update-sb [of \(\mathrm{sb}^{\prime} \mathrm{sb}, \mathrm{OF}\) - i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i, simplified sb \(\operatorname{Read}_{\mathrm{sb}}\), simplified]
have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\right)\right], \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply (simp add: m flush-commute)
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) share-commute)
using leq
apply simp
using i-bound i-bound' ts-sim ts-i is-sim \(\mathcal{D}\)
apply (clarsimp simp add: Let-def nth-list-update sb suspends \(\operatorname{Read}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime}\) split: if-split-asm)
done
ultimately show ?thesis
using valid-own \({ }^{\prime}\) valid-hist \({ }^{\prime}\) valid-reads \({ }^{\prime}\) valid-sharing' tmps-distinct \({ }^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime}\) valid-dd \({ }^{\prime}\) valid-sops \({ }^{\prime}\) load-tmps-fresh' enough-flushs \({ }^{\prime}\) valid-sharing \({ }^{\prime}\) valid-program-history' valid \(^{\prime}\) \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
next
case ( \(\operatorname{Prog}_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\) )
from flush this obtain \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}}\) and \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\) and \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\mathcal{R}_{\mathrm{sb}}\)
by cases (auto simp add: sb)
have valid-own': valid-ownership \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound \(\left.\mathrm{ts}_{\mathbf{s b}}-\mathrm{i}\right]\) sb
have non-volatile-owned-or-read-only False \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\) sb \(^{\prime}\)
by (auto simp add: \(\operatorname{Prog}_{\text {sb }}\) )
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ? thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \operatorname{Prog}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
qed
next
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from sb
have out: outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}^{\prime} \subseteq\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb by (auto simp add: \(\operatorname{Prog}_{\text {sb }}\) )
have acq: all-acquired \(\mathrm{sb}^{\prime} \subseteq\) all-acquired sb by (auto simp add: Prog \(_{\text {sb }}\) sb)
```

    from outstanding-volatile-writes-unowned-by-others-store-buffer
    [OF i-bound ts tsb-i out acq]
    show ?thesis by (simp add: ts sb ' }\mp@subsup{\operatorname{Prog}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}
    qed
next
show read-only-reads-unowned tssb}\mp@subsup{}{}{\prime
proof -
have ro: read-only-reads (acquired True (takeWhile (Not o is-volatile-Write sb
(dropWhile (Not o is-volatile-Write sb ) sb
\subseteq read-only-reads (acquired True (takeWhile (Not o is-volatile-Write esb) sb) \mathcal { O } _ { sb } )
(dropWhile (Not o is-volatile-Write sb) sb)
by (auto simp add: sb Progsb)
have }\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb'}\subseteq\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb
by (auto simp add: sb Progsb)
from read-only-reads-unowned-nth-update [OF i-bound ts sb-i ro this]
show ?thesis
by (simp add: ts sb}\mp@subsup{}{}{\prime
qed
next
show ownership-distinct tssb}\mp@subsup{}{}{\prime
proof -
have acq: all-acquired sb
by (auto simp add: Progsb sb)
with ownership-distinct-instructions-read-value-store-buffer-independent
[OF i-bound ts sb-i]
show ?thesis by (simp add: ts sb}\mp@subsup{}{\mathrm{ ' }}{}\mp@subsup{\operatorname{Prog}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}
qed
qed
have valid-sharing': valid-sharing }\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (intro-locales)
from outstanding-non-volatile-writes-unshared [OF i-bound ts sb-i]
have non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{\textrm{sb}}{}{\prime
by (auto simp add: sb Prog
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
by (simp add: ts sb}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}
next
from sharing-consis [OF i-bound ts sb-i
have sharing-consistent }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{\textrm{sb}}{}{\prime
by (auto simp add: sb Progsb)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
by (simp add: ts tsb}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound ts tsb-i]
]
show read-only-unowned }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathrm{ ts }}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime
by (simp add: }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\mathrm{ sb}}{\prime}\mp@subsup{}{}{\prime}
next

```
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - i ]
have no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (auto simp add: sb \(\operatorname{Prog}_{\text {sb }}\) )
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (simp add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) )
qed
have valid-reads': valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-reads [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - ]
have reads-consistent False \(\mathcal{O}_{\text {sb }} \mathrm{m}_{\text {sb }}\) sb \(^{\prime}\)
by (simp add: sb Prog \({ }_{\text {sb }}\) )
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-program-history': valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-program-history [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\) ]
have causal-program-history is \(\mathrm{s}_{\mathrm{sb}} \mathrm{sb}\).
then have causal': causal-program-history is \(\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}^{\prime}\) by (simp add: sb Prog \({ }_{\text {sb }}\) causal-program-history-def)
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{-}\) ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{2} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by (simp add: sb Prog \({ }_{\text {sb }}\) )
from last-prog-to-last-prog-same [OF this]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\).
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
from is-sim
have is-sim: instrs (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb') @ is \(\mathrm{s}_{\mathrm{sb}}=\) is @ prog-instrs (dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb)
by (simp add: suspends sb \(\operatorname{Prog}_{\mathrm{sb}}\) )
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by blast
moreover note flush-commute \(=\) flush-all-until-volatile-write-Prog \({ }_{s b}\)-commute [OF i-bound
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\left[\right.\) simplified sb \(\left.\operatorname{Prog}_{\mathrm{sb}}\right]\) ]
note share-commute \(=\)
share-all-until-volatile-write-update-sb [of \(\mathrm{sb}^{\prime} \mathrm{sb}, \mathrm{OF}\) - i -bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\)-i, simplified sb \(\operatorname{Prog}_{\text {sb }}\), simplified]
have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply (simp add: m flush-commute)
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) share-commute)
using leq
apply simp
using i-bound i-bound' ts-sim ts-i is-sim \(\mathcal{D}\)
apply (clarsimp simp add: Let-def nth-list-update sb suspends \(\operatorname{Prog}_{\mathrm{sb}} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) split: if-split-asm)
done
ultimately show ?thesis
using valid-own' valid-hist \({ }^{\prime}\) valid-reads' valid-sharing' tmps-distinct \({ }^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime}\)
valid-dd \({ }^{\prime}\) valid-sops \({ }^{\prime}\) load-tmps-fresh \({ }^{\prime}\) enough-flushs' valid-sharing'
valid-program-history \({ }^{\prime}\) valid \(^{\prime}\)
\(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
by (auto \(\operatorname{simp}\) del: fun-upd-apply)
next
case (Ghost \(_{\text {sb }}\) A L R W)
from flush Ghost \({ }_{\text {sb }}\) obtain
\(\mathcal{O}_{\mathrm{sb}}{ }^{\prime}: \mathcal{O}_{\mathrm{sb}}{ }^{\prime}=\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\) and
\(\mathcal{S}_{\mathrm{sb}}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) and \(\mathcal{R}_{\mathrm{sb}}{ }^{\prime}: \mathcal{R}_{\mathrm{sb}}{ }^{\prime}=\) augment-rels \(\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathrm{R} \mathcal{R}_{\mathrm{sb}}\) and
\(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\)
by cases (auto simp add: sb)
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{-\mathrm{i}}}\) ]
obtain
A-shared-owned: \(\mathrm{A} \subseteq \operatorname{dom} \mathcal{S}_{\text {sb }} \cup \mathcal{O}_{\text {sb }}\) and
L-subset: \(\mathrm{L} \subseteq \mathrm{A}\) and
\(\mathrm{A}-\mathrm{R}: \mathrm{A} \cap \mathrm{R}=\{ \}\) and
R-owned: \(\mathrm{R} \subseteq \mathcal{O}_{\text {sb }}\)
by (clarsimp simp add: sb Ghost \({ }_{\text {sb }}\) )
have valid-own': valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
proof
fix \(\mathrm{j} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{p}_{\mathrm{j}}\)
assume j-bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume ts \(_{\text {sb }}{ }^{\prime}-\mathrm{j}: \mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show non-volatile-owned-or-read-only False \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{j}=\mathrm{i}\) )
```

    case True
    from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound ts tsb-i]
    have non-volatile-owned-or-read-only False ( (\mathcal{Sbb}}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})(\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\cup\textrm{A}-\textrm{R}) s\mp@subsup{b}{}{\prime
    by (auto simp add: sb Ghostsb non-volatile-owned-or-read-only-pending-write-antimono)
    then show ?thesis
    using True i-bound ts sb '-j
    by (auto simp add: t\mp@subsup{t}{sb}{}\mp@subsup{}{}{\prime}}\mp@subsup{\mathcal{S}}{\textrm{sb}}{\prime}\mp@subsup{}{}{\prime}\textrm{sb}\mp@subsup{\mathcal{O}}{\textrm{sb}}{}\mp@subsup{}{}{\prime}
    next
    case False
    from j-bound have j-bound': j < length ts sb
        by (auto simp add: ts sb ')
    with ts sb'-j False i-bound
    have tsmb-j: ts stb
        by (auto simp add: ts sb ')
    note nvo = outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' ts sb-j]
    from read-only-unowned [OF i-bound ts tsb
    have R \cap read-only }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}={
        by auto
    with read-only-reads-unowned [OF j-bound' i-bound False ts sb-j ts sb-i] L-subset
    have }\forall\textrm{a}\in\mathrm{ read-only-reads
        (acquired True (takeWhile (Not o is-volatile-Write sb) sb j
    (dropWhile (Not o is-volatile-Write sb) sb j}\mathrm{ ).
    a }\in\mathrm{ read-only }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\longrightarrow\textrm{a}\in\mathrm{ read-only ( }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
        by (auto simp add: in-read-only-convs sb Ghostsb)
    from non-volatile-owned-or-read-only-read-only-reads-eq' [OF nvo this]
    have non-volatile-owned-or-read-only False (\mathcal{S}
    thus ?thesis by (simp add: }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\mp@subsup{}{}{\prime}\mathrm{ )
    qed
    qed
next
show outstanding-volatile-writes-unowned-by-others ts }\mp@subsup{\textrm{s}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (unfold-locales)

```

```

    assume i in-bound: i
    assume j-bound: j < length ts sb
    assume i }\mp@subsup{\textrm{i}}{1}{}-\textrm{j}:\mp@subsup{\textrm{i}}{1}{}\not=\textrm{j
    assume ts-1 i
    ```

```

    show ( }\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired }\mp@subsup{\textrm{sb}}{\textrm{j}}{})\cap\mathrm{ outstanding-refs is-volatile-Write }\mp@subsup{\mathrm{ sb }}{}{\mathrm{ sb}
    proof (cases i i_i)
        case True
        from in -j True have neq-i-j: i\not=j
            by auto
        from j-bound have j-bound': j < length ts sb
            by (simp add: ts sb
        from ts-j neq-i-j have ts-j': ts stb
    ```
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from outstanding-volatile-writes-unowned-by-others [OF i-bound j-bound' neq-i-j
\(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ts-j \(\mathrm{j}^{\prime}\) ts- \(\mathrm{i}_{1}\) i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i True show ?thesis
by (clarsimp simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) sb Ghost \({ }_{\text {sb }}\) )
next
case False
note \(\mathrm{i}_{1}-\mathrm{i}=\) this
from \(i_{1}\)-bound have \(i_{1}\)-bound': \(i_{1}<\) length \(\mathrm{ts}_{\text {sb }}\) by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) )
hence \(\mathrm{i}_{1}\)-bound \({ }^{\prime \prime}: \mathrm{i}_{1}<\) length (map owned \(\mathrm{ts}_{\text {sb }}\) )
by auto
from ts-i \(\mathrm{i}_{1}\) False have \(\mathrm{ts}-\mathrm{i}_{1}\) ': \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}\) )
show ?thesis
proof (cases \(\mathrm{j}=\mathrm{i}\) )
case True
from outstanding-volatile-writes-unowned-by-others [ \(\mathrm{OF} \mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{i}\)-bound \(\mathrm{i}_{1}\) - i ts- \(\mathrm{i}_{1}{ }^{\prime}\) \(\left.\mathrm{ts}_{\text {sb }}-\mathrm{i}\right]\)
have \(\left(\mathcal{O}_{\mathrm{sb}} \cup\right.\) all-acquired sb\() \cap\) outstanding-refs is-volatile- Write \(_{\mathrm{sb}} \mathrm{sb}_{1}=\{ \}\).
then show? thesis
using True \(\mathrm{i}_{1}-\mathrm{i}\) ts-j \(\mathrm{ts}_{\mathrm{sb}}\) - i -bound
by (auto simp add: sb Ghostsb \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
next
case False
from j -bound have j -bound': \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from ts-j False have ts-j\({ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}\) )
from outstanding-volatile-writes-unowned-by-others
[ \(\mathrm{OF} \mathrm{i}_{1}\)-bound \({ }^{\prime} \mathrm{j}\)-bound \({ }^{\prime} \mathrm{i}_{1}-\mathrm{j}\) ts- \(\mathrm{i}_{1}{ }^{\prime}\) ts-j\({ }^{\text {' }}\) ]
show \(\left(\mathcal{O}_{\mathrm{j}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{j}}\right) \cap\) outstanding-refs is-volatile-Write \({ }_{\text {sb }} \mathrm{sb}_{1}=\{ \}\).
qed
qed
qed
next
show read-only-reads-unowned ts sb \({ }^{\prime}\)
proof
fix n m
fix \(\mathrm{p}_{\mathrm{n}} \mathrm{is}_{\mathrm{n}} \mathcal{O}_{\mathrm{n}} \mathcal{R}_{\mathrm{n}} \mathcal{D}_{\mathrm{n}} \vartheta_{\mathrm{n}} \mathrm{sb}_{\mathrm{n}} \mathrm{p}_{\mathrm{m}}\) is \(\mathrm{m}_{\mathrm{m}} \mathcal{O}_{\mathrm{m}} \mathcal{R}_{\mathrm{m}} \mathcal{D}_{\mathrm{m}} \vartheta_{\mathrm{m}} \mathrm{sb}_{\mathrm{m}}\)
assume n-bound: \(\mathrm{n}<\) length \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\)
and m -bound: \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
and neq- \(\mathrm{n}-\mathrm{m}: \mathrm{n} \neq \mathrm{m}\)
and nth: \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{n}=\left(\mathrm{p}_{\mathrm{n}}, \mathrm{is}_{\mathrm{n}}, \vartheta_{\mathrm{n}}, \mathrm{sb}_{\mathrm{n}}, \mathcal{D}_{\mathrm{n}}, \mathcal{O}_{\mathrm{n}}, \mathcal{R}_{\mathrm{n}}\right)\)
and mth: \(\mathrm{ts}_{\mathrm{sb}}\) ! \(\mathrm{m}=\left(\mathrm{p}_{\mathrm{m}}, \mathrm{is}_{\mathrm{m}}, \vartheta_{\mathrm{m}}, \mathrm{sb}_{\mathrm{m}}, \mathcal{D}_{\mathrm{m}}, \mathcal{O}_{\mathrm{m}}, \mathcal{R}_{\mathrm{m}}\right)\)
from \(n\)-bound have n -bound': \(\mathrm{n}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from m -bound have m -bound': \(\mathrm{m}<\) length \(\mathrm{ts}_{\mathrm{sb}}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
show \(\left(\mathcal{O}_{\mathrm{m}} \cup\right.\) all-acquired \(\left.\mathrm{sb}_{\mathrm{m}}\right) \cap\)
read-only-reads (acquired True (takeWhile (Noto is-volatile-Write \({ }_{\mathbf{s b}}\) ) sb \({ }_{\mathrm{n}}\) ) \(\mathcal{O}_{\mathrm{n}}\) ) \(\left(\right.\) dropWhile \(\left(\right.\) Not \(\circ\) is-volatile-Write \(\left.\left.{ }_{\text {sb }}\right) \operatorname{sb}_{\mathrm{n}}\right)=\) \{\}
```

    proof (cases m=i)
        case True
        with neq-n-m have neq-n-i: n}=\textrm{i
            by auto
        with n-bound nth i-bound have nth': ts mb}!\textrm{n}=(\mp@subsup{\textrm{p}}{\textrm{n}}{},\mp@subsup{\textrm{is}}{\textrm{n}}{},\mp@subsup{\vartheta}{\textrm{n}}{},\mp@subsup{\textrm{sb}}{\textrm{n}}{},\mp@subsup{\mathcal{D}}{\textrm{n}}{},\mp@subsup{\mathcal{O}}{\textrm{n}}{},\mp@subsup{\mathcal{R}}{\textrm{n}}{}
        by (auto simp add: ts sb
        note read-only-reads-unowned [OF n-bound' i-bound neq-n-i nth' ts smb-i]
        then
        show ?thesis
            using True ts ssb-i neq-n-i nth mth n-bound' m-bound' L-subset
        by (auto simp add: ts sb
    next
    case False
    note neq-m-i = this
    with m-bound mth i-bound have mth': ts sb
        by (auto simp add: tssb}\mp@subsup{}{\mathrm{ ' }}{
    show ?thesis
    proof (cases n=i)
        case True
    from read-only-reads-append [of (\mathcal{O}
    sbn
(dropWhile (Not o is-volatile-Write sb
have read-only-reads
(acquired True (takeWhile (Not o is-volatile-Write sb) sb n})(\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cupA-R)
(dropWhile (Not o is-volatile-Write sb}) s\mp@subsup{b}{n}{})\subseteqread-only-reads ( (\mathcal{Osb}\cup\cupA - R
sb
by auto
with ts sb-i nth mth neq-m-i n-bound' True
read-only-reads-unowned [OF i-bound m-bound' False [symmetric] ts sb-i mth']
show ?thesis
by (auto simp add: ts sbb}\mp@subsup{}{}{\prime
next
case False
with n-bound nth i-bound have nth': ts sb !n = (pm,is
by (auto simp add: ts ssb
from read-only-reads-unowned [OF n-bound'm-bound' neq-n-m nth'mth'] False
neq-m-i
show ?thesis
by (clarsimp)
qed
qed
qed
next
show ownership-distinct ts sbb
proof (unfold-locales)

```

```

    assume i }\mp@subsup{i}{1}{}\mathrm{ -bound: }\mp@subsup{i}{1}{}<length tssmb'
    assume j-bound: j < length tssmb
    ```
```

assume $\mathrm{i}_{1}-\mathrm{j}: \mathrm{i}_{1} \neq \mathrm{j}$
assume ts- $\mathrm{i}_{1}: \mathrm{ts}_{\mathrm{sb}}!_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
assume ts-j: ts $\mathrm{sbb}_{\mathrm{b}}{ }^{\prime} \mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
show $\left(\mathcal{O}_{1} \cup\right.$ all-acquired $\left.\mathrm{sb}_{1}\right) \cap\left(\mathcal{O}_{\mathrm{j}} \cup\right.$ all-acquired $\left.\mathrm{sb}_{\mathrm{j}}\right)=\{ \}$
proof (cases $\mathrm{i}_{1}=\mathrm{i}$ )
case True
with $\mathrm{i}_{1}-\mathrm{j}$ have $\mathrm{i}-\mathrm{j}: \mathrm{i} \neq \mathrm{j}$
by simp
from j -bound have j -bound': $\mathrm{j}<$ length $\mathrm{ts}_{\text {sb }}$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\text {' }}$ )
hence j -bound ${ }^{\prime \prime}$ : j < length (map owned $\mathrm{ts}_{\text {sb }}$ )
by simp
from ts-j i-j have ts-j${ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{isj}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by ( simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$ )
from ownership-distinct [OF i-bound j-bound ${ }^{\prime} \mathrm{i}-\mathrm{j} \mathrm{ts}_{\mathrm{sb}}-\mathrm{its}$ ts ${ }^{\text {j }}$ ]
show ?thesis
using $\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}$ True ts- $\mathrm{i}_{1} \mathrm{i}$-bound $\mathcal{O}_{\mathrm{sb}}{ }^{\prime}$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}$ Ghost sb )
next
case False
note $\mathrm{i}_{1}-\mathrm{i}=$ this
from $i_{1}$-bound have $i_{1}$-bound': $i_{1}<$ length $\mathrm{ts}_{\text {sb }}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
hence $\mathrm{i}_{1}$-bound ${ }^{\prime \prime}: \mathrm{i}_{1}<$ length (map owned $\mathrm{ts}_{\text {sb }}$ )
by simp
from ts-i $\mathrm{i}_{1}$ False have ${\mathrm{ts}-\mathrm{i}_{1}}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{l}_{1}=\left(\mathrm{p}_{1}, \mathrm{is}_{1}, \mathrm{xs}_{1}, \mathrm{sb}_{1}, \mathcal{D}_{1}, \mathcal{O}_{1}, \mathcal{R}_{1}\right)$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
show ?thesis
proof (cases $\mathrm{j}=\mathrm{i}$ )
case True
from ownership-distinct [ $\mathrm{OF} \mathrm{i}_{1}$-bound ${ }^{\prime} \mathrm{i}$-bound $\mathrm{i}_{1}-\mathrm{i}$ ts- $\left.\mathrm{i}_{1}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]$
show ?thesis
using $\mathrm{ts}_{\mathrm{sb}}$-i True ts-j i-bound $\mathcal{O}_{\mathrm{sb}}{ }^{\prime}$
by (auto simp add: $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathrm{sb}$ Ghost $_{\mathrm{sb}}$ )
next
case False
from j-bound have $j$-bound': $\mathrm{j}<$ length $\mathrm{ts}_{\mathrm{sb}}$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
from ts-j False have ts-j${ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)$
by ( $\operatorname{simp}$ add: $\mathrm{ts}_{\mathrm{sb}}$ )
from ownership-distinct [ $\mathrm{OF}_{\mathrm{i}} \mathrm{i}_{1}$-bound ${ }^{\prime} \mathrm{j}$-bound ${ }^{\prime} \mathrm{i}_{1}-\mathrm{j}$ ts- $\mathrm{i}_{1}{ }^{\prime}$ ts-j${ }^{\prime}$ ]
show ?thesis .
qed
qed
qed
qed
have valid-sharing': valid-sharing ( $\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}$ ) $\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}$

```
proof (intro-locales)
show outstanding-non-volatile-writes-unshared \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
fix \(\mathrm{j} \mathrm{p} \mathrm{p}_{\mathrm{j}}\) is \(\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{acq}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length ts \(_{\text {sb }}{ }^{\prime}\)
assume jth: \(\mathrm{ts}_{\mathrm{sb}^{\prime}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show non-volatile-writes-unshared \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True

i-bound jth \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}}\) i show ?thesis

next
case False
from j-bound have j-bound \({ }^{\prime}\) : \(\mathrm{j}<\) length ts \(_{\text {sb }}\) by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from j th False have \(\mathrm{jth}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\) by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
from j-bound jth i-bound False
have j : non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\) apply -
apply (rule outstanding-non-volatile-writes-unshared)
apply (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
done
from jth False have \(\mathrm{jth}^{\prime}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
from outstanding-non-volatile-writes-unshared [OF j-bound' jth]
have unshared: non-volatile-writes-unshared \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
have \(\forall \mathrm{a} \in \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus \mathrm{w} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) - dom \(\mathcal{S}_{\mathrm{sb}} . \mathrm{a} \notin\) outstanding-refs is-non-volatile-Write \({ }_{\text {sb }}\) \(\mathrm{sb}_{\mathrm{j}}\)
proof -
\{
fix a
assume a-in: \(\mathrm{a} \in \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus{ }_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\)
hence \(a-R: a \in R\)
by clarsimp
assume a-in-j: a \(\in\) outstanding-refs is-non-volatile-Write \({ }_{s b} \operatorname{sb}_{j}\)
have False proof -
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes [OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth 1\(]\) a-in-j
have \(\mathrm{a} \in \mathcal{O}_{\mathrm{j}} \cup\) all-acquired \(\mathrm{sb}_{\mathrm{j}}\)
by auto

\section*{moreover}
with ownership-distinct [OF i-bound j-bound' False ts stbil \(^{\text {-i }}\) jth \(]\) a-R R-owned show False
by blast
```

qed
}
thus ?thesis by blast
qed
from non-volatile-writes-unshared-no-outstanding-non-volatile-Write sb
[OF unshared this]
show ?thesis .
qed
qed
next
show sharing-consis ( }\mp@subsup{\mathcal{S}}{\mathbf{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mp@subsup{\textrm{ts}}{\textrm{sb}}{}\mp@subsup{}{}{\prime
proof (unfold-locales)

```

```

    assume j-bound: j < length ts sb
    assume jth: ts scb
    show sharing-consistent ( }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{\prime}R\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mp@subsup{\mathcal{O}}{j}{}\mp@subsup{\textrm{sb}}{\textrm{j}}{
    proof (cases i=j)
        case True
        with i-bound jth ts sb-i sharing-consis [OF i-bound ts cs-i]
        show ?thesis
            by (clarsimp simp add: ts sbb
    next
        case False
        from j-bound have j-bound': j < length ts sb
        by (auto simp add: ts sb}\mp@subsup{}{\mathrm{ ' }}{
        from jth False have jth': ts stb ! j = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,}\mp@subsup{\textrm{is}}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{xS}}{\textrm{j}}{,},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{},\mp@subsup{\mathcal{O}}{\textrm{j}}{},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
        by (auto simp add: ts sb ')
    from sharing-consis [OF j-bound' jth ]
    have consis: sharing-consistent }\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\mathcal{O}}{\textrm{j}}{\textrm{sb}}\mp@subsup{\textrm{j}}{\textrm{j}}{
        have acq-cond: all-acquired sb}\mp@subsup{\textrm{g}}{\textrm{j}}{}\cap\operatorname{dom}\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}-\operatorname{dom}(\mp@subsup{\mathcal{S}}{\textrm{sb}}{}\mp@subsup{\oplus}{\textrm{W}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})={
        proof -
        {
    fix a
assume a-acq: a }\in\mathrm{ all-acquired sbj
assume a }\in\operatorname{dom}\mp@subsup{\mathcal{S}}{\mathrm{ sb}}{
assume a-L: a }\in\textrm{L
have False
proof -
from ownership-distinct [OF i-bound j-bound' False ts sb-i jth]
have A \cap all-acquired sb }\mp@subsup{\textrm{j}}{\textrm{G}}{={}{
by (auto simp add: sb Ghostsb)
with a-acq a-L L-subset
show False
by blast
qed
}

```
thus ?thesis
by auto
qed
have uns-cond: all-unshared \(\operatorname{sb}_{\mathrm{j}} \cap \operatorname{dom}\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)-\operatorname{dom} \mathcal{S}_{\mathrm{sb}}=\{ \}\)
proof -
\{
fix a
assume a-uns: \(a \in\) all-unshared \(\mathrm{sb}_{\mathrm{j}}\)
assume a \(\notin \mathrm{L}\)
assume \(a-R: a \in R\)
have False
proof -
from unshared-acquired-or-owned [OF consis] a-uns
have \(\mathrm{a} \in\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\) by auto
with ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False ts \(_{\text {sb }}-\mathrm{i} j\) jth \(]\) R-owned a-R show False
by blast
qed
\}
thus ?thesis
by auto
qed
from sharing-consistent-preservation [OF consis acq-cond uns-cond]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
qed
next
show unowned-shared \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (unfold-locales)
show \(-\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot\right.\) set \(\left.\mathrm{ts}_{\text {sb }}\right) \subseteq \operatorname{dom}\left(\mathcal{S}_{\text {sb }} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
proof -
have \(\mathrm{s}: ~ \bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O})\right.\) ' \(\left.\operatorname{set} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\right)=\)
\(\bigcup\left((\lambda(-,-,-,-,-, \mathcal{O},-) . \mathcal{O}) \cdot\right.\) set \(\left.\mathrm{ts}_{\text {sb }}\right) \cup \mathrm{A}-\mathrm{R}\)
apply (unfold \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
apply (rule acquire-release-ownership-nth-update [OF R-owned i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ])
apply (rule local.ownership-distinct-axioms)
done
note unowned-shared L-subset A-R
then
show ?thesis
apply (simp only: s)
apply auto
done
qed
qed
next
show read-only-unowned \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof
fix \(\mathrm{j}_{\mathrm{j}}\) is \(\mathrm{S}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(^{\mathrm{ts}} \mathrm{sb}^{\prime}{ }^{\prime}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show \(\mathcal{O}_{\mathrm{j}} \cap\) read-only \(\left(\mathcal{S}_{\text {sb }} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\{ \}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
from read-only-unowned [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have \(\left(\mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right) \cap\) read-only \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)=\{ \}\)
by (auto simp add: in-read-only-convs )
with jth \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) i-bound True
show ?thesis
by (auto simp add: \(\mathcal{O}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}\) )
next
case False
from j -bound have j -bound \({ }^{\prime}\) : \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
with False jth have jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
from read-only-unowned [OF j-bound \({ }^{\prime}\) jth']
have \(\mathcal{O}_{\mathrm{j}} \cap\) read-only \(\mathcal{S}_{\mathrm{sb}}=\{ \}\).
moreover
from ownership-distinct [OF i-bound j-bound \({ }^{\prime}\) False \(\mathrm{ts}_{\mathrm{sb}^{-}}\)-i jth \(]\)R-owned
have \(\left(\mathcal{O}_{\text {sb }} \cup \mathrm{A}\right) \cap \mathcal{O}_{\mathrm{j}}=\{ \}\)
by (auto simp add: sb Ghostsb)
moreover note R-owned A-R
ultimately show ?thesis
by (fastforce simp add: in-read-only-convs split: if-split-asm)
qed
qed
next
show no-outstanding-write-to-read-only-memory \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\) ts \(_{\mathrm{sb}}{ }^{\prime}\)
proof
fix \(\mathrm{j} \mathrm{p}_{\mathrm{j}}\) is \(\mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \mathrm{xs}_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
show no-write-to-read-only-memory ( \(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) ) sb \(\mathrm{sb}_{\mathrm{j}}\)
proof (cases \(\mathrm{i}=\mathrm{j}\) )
case True
with \(\mathrm{jth}^{\text {ts }} \mathrm{sb}_{\mathrm{sb}}\)-i i -bound no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
show ?thesis
by (auto simp add: \(\mathrm{sb} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) Ghost \(_{\text {sb }}\) )
next
case False
from j -bound have j -bound': \(\mathrm{j}<\) length \(\mathrm{ts}_{\text {sb }}\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
with False jth have jth : \(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \mathrm{xs}_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
by (auto simp add: \(\mathrm{ts}_{\text {sb }}\) )
from no-outstanding-write-to-read-only-memory [OF j-bound' \({ }^{\text {jth }}\) ]
have nw: no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
have \(R \cap\) outstanding-refs is-Write \({ }_{s b} \operatorname{sb}_{j}=\{ \}\)
proof -
note dist \(=\) ownership-distinct [ OF i-bound \(j\)-bound \({ }^{\prime}\) False \(\mathrm{ts}_{\mathrm{sb}}\)-i jth \({ }^{\prime}\) ]
from non-volatile-owned-or-read-only-outstanding-non-volatile-writes
[OF outstanding-non-volatile-refs-owned-or-read-only [OF j-bound' jth 1 ]]
dist
have outstanding-refs is-non-volatile-Write \({ }_{\text {sb }} \operatorname{sb}_{\mathrm{j}} \cap \mathcal{O}_{\text {sb }}=\{ \}\)
by auto
moreover
from outstanding-volatile-writes-unowned-by-others [OF j-bound' i-bound False [symmetric] jth \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have outstanding-refs is-volatile-Write \({ }_{\mathbf{s b}} \mathrm{sb}_{\mathrm{j}} \cap \mathcal{O}_{\mathrm{sb}}=\{ \}\)
by auto
ultimately have outstanding-refs is-Write stb \(_{\text {sb }} \operatorname{sb}_{j} \cap \mathcal{O}_{\text {sb }}=\{ \}\)
by (auto simp add: misc-outstanding-refs-convs)
with R -owned
show ?thesis by blast
qed
then
have \(\forall \mathrm{a} \in\) outstanding-refs is-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{j}}\).
\(\mathrm{a} \in\) read-only \(\left(\mathcal{S}_{\mathrm{sb}} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right) \longrightarrow \mathrm{a} \in\) read-only \(\mathcal{S}_{\mathrm{sb}}\)
by (auto simp add: in-read-only-convs)
from no-write-to-read-only-memory-read-only-reads-eq [OF nw this]
show ?thesis .
qed
qed
qed
have valid-reads': valid-reads \(\mathrm{m}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}^{\prime}}{ }^{\prime}\)
proof -
from valid-reads [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have reads-consistent False \(\left(\mathcal{O}_{s b} \cup \mathrm{~A}-\mathrm{R}\right) \mathrm{m}_{\mathrm{sb}} \mathrm{sb}^{\prime}\)
by (simp add: sb Ghostsb)
from valid-reads-nth-update [OF i-bound this]
show ?thesis by ( simp add: \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{O}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-program-history': valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-program-history [ OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\) ]
have causal-program-history is \(\mathrm{s}_{\mathrm{sb}} \mathrm{sb}\).
then have causal': causal-program-history is sb \(\mathrm{sb}^{\prime}\)
by (simp add: sb Ghostsb causal-program-history-def)
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{-}\) ]
have last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
hence last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}^{\prime}=\mathrm{p}_{\mathrm{sb}}\)
by (simp add: sb Ghostsb)
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
from is-sim
have is-sim: instrs (dropWhile (Not \(\circ\) is-volatile-Write \(\left.{ }_{s b}\right)\) sb') @ is \(\mathrm{s}_{\mathrm{sb}}=\) is @ prog-instrs (dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb')
by (simp add: sb Ghostsb suspends)
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by blast
moreover
note flush-commute \(=\)
flush-all-until-volatile-write-Ghost sb \(^{-c o m m u t e}\) [ OF i-bound \(\mathrm{ts}_{\mathbf{s b}^{-\mathrm{i}}}\) [simplified sb Ghost \({ }_{\mathbf{s b}}\) ]]
have dist-R-L-A: \(\forall \mathrm{j}\) p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}\).
\(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{i} \neq \mathrm{j} \longrightarrow\)
\(\mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow\)
(all-shared (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cup\)
all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) \(\cup\)
all-acquired \(\left(\right.\) takeWhile \(\left.\left.\left.^{\left(\text {Not }^{\circ} \circ \text { is-volatile-Write }\right.}{ }_{\text {sb }}\right) \mathrm{sb}\right)\right) \cap(\mathrm{R} \cup \mathrm{L} \cup \mathrm{A})=\{ \}\)
proof -
\{
fix \(\mathrm{jp}_{\mathrm{j}} \mathrm{is}_{\mathrm{j}} \mathcal{O}_{\mathrm{j}} \mathcal{R}_{\mathrm{j}} \mathcal{D}_{\mathrm{j}} \vartheta_{\mathrm{j}} \mathrm{sb}_{\mathrm{j}} \mathrm{x}\)
assume j -bound: \(\mathrm{j}<\) length \(\mathrm{ts}_{\mathrm{sb}}\)
assume neq-i-j: \(\mathrm{i} \neq \mathrm{j}\)
assume \(\mathrm{jth}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{j}=\left(\mathrm{p}_{\mathrm{j}}, \mathrm{is}_{\mathrm{j}}, \vartheta_{\mathrm{j}}, \mathrm{sb}_{\mathrm{j}}, \mathcal{D}_{\mathrm{j}}, \mathcal{O}_{\mathrm{j}}, \mathcal{R}_{\mathrm{j}}\right)\)
assume x-shared: \(\mathrm{x} \in\) all-shared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) \(\cup\)
all-unshared (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) ) \(\cup\)
all-acquired (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )
assume \(x\)-R-L-A: \(x \in R \cup L \cup A\)
have False
proof -
from x-shared all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
unshared-acquired-or-owned [OF sharing-consis [OF j-bound jth]]
all-shared-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) (dropWhile
(Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )]
all-unshared-append [of (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb \({ }_{\mathrm{j}}\) ) (dropWhile
(Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb \(_{\mathrm{j}}\) )]
all-acquired-append [of (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}_{\mathrm{j}}\) ) (dropWhile
(Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) \(\mathrm{sb}_{\mathrm{j}}\) )]
have \(\mathrm{x} \in\) all-acquired \(\mathrm{sb}_{\mathrm{j}} \cup \mathcal{O}_{\mathrm{j}}\)
by auto
moreover
```

            from x-R-L-A R-owned L-subset
            have x\in all-acquired sb \cup (\mathcal{Osb}
                by (auto simp add: sb Ghostsb)
            moreover
            note ownership-distinct [OF i-bound j-bound neq-i-j ts sb-i jth]
            ultimately show False by blast
        qed
    }
    thus ?thesis by blast
    qed

```
```

{

```

```

assume jth: ts sblj = ( }\mp@subsup{\textrm{p}}{\textrm{j}}{,}\mp@subsup{\textrm{is}}{\textrm{j}}{},\mp@subsup{\vartheta}{\textrm{j}}{\textrm{j}},\mp@subsup{\textrm{sb}}{\textrm{j}}{},\mp@subsup{\mathcal{D}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{O}}{\textrm{j}}{\textrm{j}},\mp@subsup{\mathcal{R}}{\textrm{j}}{}
assume j-bound: j < length tssb
assume neq: i }=\textrm{j
have release (takeWhile (Not o is-volatile-Write sb ) sb }\mp@subsup{}{\textrm{j}}{}\mathrm{ )
(dom }\mp@subsup{\mathcal{S}}{\mathrm{ sb }}{}\cup\textrm{R}-\textrm{L})\mp@subsup{\mathcal{R}}{\textrm{j}}{
= release (takeWhile (Not ० is-volatile-Write (
(dom }\mp@subsup{\mathcal{S}}{\textrm{sb}}{})\mp@subsup{\mathcal{R}}{\textrm{j}}{
proof -
{
fix a
assume a-in: a }\in\mathrm{ all-shared (takeWhile (Not ० is-volatile-Write }\mp@subsup{\textrm{sb}}{\mathbf{s}}{}\mathrm{ ) sb }\mp@subsup{\textrm{b}}{\textrm{j}}{}\mathrm{ )
have (a\in(dom \mathcal{S}
proof -
from ownership-distinct [OF i-bound j-bound neq ts sb-i jth]
have A-dist: }\textrm{A}\cap(\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sb
by (auto simp add: sb Ghostsb)
from all-shared-acquired-or-owned [OF sharing-consis [OF j-bound jth]] a-in
all-shared-append [of (takeWhile (Not o is-volatile-Write sb ) sb }\mp@subsup{}{\textrm{j}}{}\mathrm{ )
(dropWhile (Not o is-volatile-Write sb
have a-in: a }\in\mp@subsup{\mathcal{O}}{\textrm{j}}{}\cup\mathrm{ all-acquired sbj
by auto
with ownership-distinct [OF i-bound j-bound neq ts tsb-i jth]
have a }\not\in(\mp@subsup{\mathcal{O}}{\mathrm{ sb }}{}\cup\mathrm{ all-acquired sb) by auto
with A-dist R-owned A-R A-shared-owned L-subset a-in
obtain a }\not\in\textrm{R}\mathrm{ and a }\not\in\textrm{L
by fastforce
then show ?thesis by auto
qed
}
then
show ?thesis
apply -
apply (rule release-all-shared-exchange)

```
```

            apply auto
            done
        qed
    }
    note release-commute = this

```
from ownership-distinct-axioms have ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}\).
from sharing-consis-axioms have sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\).
note share-commute \(=\) share-all-until-volatile-write-Ghost \({ }_{\text {sb }}\)-commute \([\mathrm{OF}\)
〈ownership-distinct \(\mathrm{ts}_{\mathrm{sb}}\) 〉
«sharing-consis \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 i-bound \(\mathrm{ts}_{\mathrm{sb}}\) - [simplified sb Ghost \({ }_{\mathrm{sb}}\) ] dist-R-L-A]
    have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}^{\prime}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}} \cup \mathrm{A}-\mathrm{R}\right.\right.\right.\),augment-rels (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathrm{R}\)
\(\left.\left.\left.\mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply (simp add: m flush-commute)
apply (clarsimp \(\operatorname{simp}\) add: \(\mathcal{S} \mathcal{S}_{\text {sb }}{ }^{\prime}\) share-commute)
using leq
apply simp
using i-bound i-bound' ts-sim ts-i is-sim \(\mathcal{D}\)
apply (clarsimp simp add: Let-def nth-list-update sb suspends Ghost \({ }_{\mathrm{sb}} \mathcal{R}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
    split: if-split-asm)
        apply (rule conjI)
        apply fastforce
        apply clarsimp
        apply (frule (2) release-commute)
        apply clarsimp
        apply auto
done
    ultimately
    show ?thesis
using valid-own' valid-hist ' valid-reads' valid-sharing' tmps-distinct'
    valid-dd' valid-sops \({ }^{\prime}\) load-tmps-fresh' enough-flushs \({ }^{\prime}\)
    valid-program-history' valid \({ }^{\prime}\)
    \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply simp add: \(\mathcal{O}_{\text {sb }}{ }^{\prime} \mathcal{R}_{\text {sb }}{ }^{\prime}\) )
    qed
next
    case (Program i p \(\mathrm{p}_{\mathrm{sb}}\) is \(\mathrm{sbb}_{\mathrm{sb}} \vartheta_{\mathrm{sb}}\) \(\mathcal{D}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}} \mathcal{R}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime}\) mis)
    then obtain
            \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}=\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}, \mathrm{is}_{\mathrm{sb}} @ \operatorname{mis}, \vartheta_{\mathrm{sb}}, \mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\)
and
    i-bound: i \(<\) length ts \(_{\text {sb }}\) and
    \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}: \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}}, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\) and
    prog: \(\vartheta_{\mathrm{sb}} \vdash \mathrm{p}_{\mathrm{sb}} \rightarrow_{\mathrm{p}}\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}, \mathrm{mis}\right)\) and
    \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime}: \mathcal{S}_{\mathrm{sb}}{ }^{\prime}=\mathcal{S}_{\mathrm{sb}}\) and
    \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}: \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}_{\mathrm{sb}}\)
    by auto

\section*{from sim obtain}
\(\mathrm{m}: \mathrm{m}=\) flush－all－until－volatile－write \(\mathrm{ts}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}\) and
\(\mathcal{S}: \mathcal{S}=\) share-all-until-volatile-write \(\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}\) and
leq: length \(\mathrm{ts}_{\mathrm{sb}}=\) length ts and
ts-sim: \(\forall \mathrm{i}<\) length \(\mathrm{ts}_{\text {sb }}\).
let \(\left(\mathrm{p}, \mathrm{is}_{\mathrm{sb}}, \vartheta, \mathrm{sb}, \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}\right)=\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}\);
suspends \(=\) dropWhile (Not \(\circ\) is-volatile-Write \(\left.{ }_{\text {sb }}\right) \mathrm{sb}\)
in \(\exists\) is \(\mathcal{D}\). instrs suspends @ is \(_{\text {sb }}=\) is @ prog-instrs suspends \(\wedge\)
\[
\mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee \text { outstanding-refs is-volatile-Write }{ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}\right) \wedge
\]
ts ! \(\mathrm{i}=\)
(hd-prog p suspends,
is,
\(\left.\vartheta\right|^{6}(\operatorname{dom} \vartheta-\) read-tmps suspends \(),()\),
\(\mathcal{D}\),
acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb) \(\mathcal{O}_{s b}\), release (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb) (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}\) )
by cases blast
from i-bound leq have i-bound': i < length ts
by auto
have split-sb: sb \(=\) takeWhile (Not \(\circ\) is-volatile-Write \(s\) ) sb @ dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb
(is \(\mathrm{sb}=\) ?take-sb@?drop-sb)
by simp
from ts-sim [rule-format, OF i-bound] \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) obtain suspends is \(\mathcal{D}\) where suspends: suspends \(=\) dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) ) sb and is-sim: instrs suspends @ is \(\mathrm{s}_{\mathrm{sb}}=\) is @ prog-instrs suspends and \(\mathcal{D}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \(\left.{ }_{\mathrm{sb}} \mathrm{sb} \neq\{ \}\right)\) and ts-i: ts ! \(\mathrm{i}=\)
(hd-prog \(\mathrm{p}_{\mathrm{sb}}\) suspends, is,
\(\left.\vartheta_{\mathrm{sb}}\right|^{6}\left(\right.\) dom \(\vartheta_{\mathrm{sb}}-\) read-tmps suspends \(),(), \mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\),
release ?take-sb (dom \(\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\) )
by (auto simp add: Let-def)
from prog-step-preserves-valid [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\)-i prog valid]
have valid': valid \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}\) )
have valid-own': valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)
show outstanding-non-volatile-refs-owned-or-read-only \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from outstanding-non-volatile-refs-owned-or-read-only [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have non-volatile-owned-or-read-only False \(\mathcal{S}_{\text {sb }} \mathcal{O}_{\text {sb }}\left(\operatorname{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\)
by (auto simp add: non-volatile-owned-or-read-only-append)
from outstanding-non-volatile-refs-owned-or-read-only-nth-update [OF i-bound this]
show ?thesis by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
qed
next
show outstanding-volatile-writes-unowned-by-others \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have out: outstanding-refs is-volatile-Write \({ }_{s b}\left(\operatorname{sb} @\left[\operatorname{Prog}_{s b} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right) \subseteq\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) sb
by (auto simp add: outstanding-refs-conv )
from outstanding-volatile-writes-unowned-by-others-store-buffer
[OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) this]

qed
next
show read-only-reads-unowned \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
have ro: read-only-reads (acquired True (takeWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) )
\(\left.\left.\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right) \mathcal{O}_{\mathrm{sb}}\right)\)
(dropWhile (Not \(\circ\) is-volatile-Write \(\left.{ }_{s b}\right)\left(\operatorname{sb@} @ \operatorname{Prog}_{s b} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime}\right.\) mis \(\left.]\right)\) )
\(\subseteq\) read-only-reads (acquired True (takeWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ) sb) \(\mathcal{O}_{\mathrm{sb}}\) )
(dropWhile (Not o is-volatile-Write \({ }_{\text {sb }}\) ) sb)
apply (case-tac outstanding-refs (is-volatile-Write \({ }_{\text {sb }}\) ) sb \(=\{ \}\) )
apply (simp-all add: outstanding-vol-write-take-drop-appends acquired-append read-only-reads-append )
done
have \(\mathcal{O}_{\mathrm{sb}} \cup\) all-acquired \(\left(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right) \subseteq \mathcal{O}_{\mathrm{sb}} \cup\) all-acquired sb by (auto simp add: all-acquired-append)
from read-only-reads-unowned-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i ro this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) )
qed
next show ownership-distinct \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\) proof -
from ownership-distinct-instructions-read-value-store-buffer-independent
\(\left[\right.\) OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\), where \(\left.\mathrm{sb}{ }^{\prime}=\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right]\)
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) all-acquired-append)
qed
qed
from valid-last-prog [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have last-prog: last-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}=\mathrm{p}_{\mathrm{sb}}\).
have valid-hist \({ }^{\prime}\) : valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) proof -
from valid-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have history-consistent \(\vartheta_{\mathrm{sb}}\) (hd-prog \(\mathrm{p}_{\mathrm{sb}} \mathrm{sb}\) ) sb.
from history-consistent-append- Prog \(_{s b}\) [OF prog this last-prog]
have hist-consis': history-consistent \(\vartheta_{\mathrm{sb}}\) (hd-prog \(\left.\mathrm{p}_{\mathrm{sb}}{ }^{\prime}\left(\operatorname{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \operatorname{mis}\right]\right)\right)\) (sb@[Prog \(\left.{ }_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\) ).
from valid-history-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
have valid-reads \({ }^{\prime}\) : valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from valid-reads [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have reads-consistent False \(\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathrm{sb}\).
from reads-consistent-snoc- Prog \(_{s b}\) [OF this]
have reads-consistent False \(\mathcal{O}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \quad\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\).
from valid-reads-nth-update [OF i-bound this]
show ?thesis by (simp add: \(\mathrm{ts}_{\text {sb }}\) )
qed
have valid-sharing': valid-sharing \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof (intro-locales)

have non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\)
by (auto simp add: non-volatile-writes-unshared-append)
from outstanding-non-volatile-writes-unshared-nth-update [OF i-bound this]
show outstanding-non-volatile-writes-unshared \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from sharing-consis [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have sharing-consistent \(\mathcal{S}_{\mathrm{sb}} \mathcal{O}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\)
by (auto simp add: sharing-consistent-append)
from sharing-consis-nth-update [OF i-bound this]
show sharing-consis \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from read-only-unowned-nth-update [OF i-bound read-only-unowned [OF i-bound
\(\left.\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]\) ]
show read-only-unowned \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
next
from unowned-shared-nth-update [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\)-i subset-refl]
show unowned-shared \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sb }}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\) )
next
from no-outstanding-write-to-read-only-memory [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i]
have no-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\)
by (simp add: no-write-to-read-only-memory-append)
from no-outstanding-write-to-read-only-memory-nth-update [OF i-bound this]
show no-outstanding-write-to-read-only-memory \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have tmps-distinct \({ }^{\prime}\) : tmps-distinct \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\)
proof (intro-locales)
from load-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}-\mathrm{i}}\) ]
have distinct-load-tmps is sb .
with distinct-load-tmps-prog-step [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i prog valid]
have distinct-load-tmps (is stb \(\left._{\text {sb }} @ m i s\right)\)
by (auto simp add: distinct-load-tmps-append)
from load-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
next
from read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{s}}-\mathrm{i}}\) ]
have distinct-read-tmps (sb@[ \(\left.\left.\operatorname{Prog}_{s b} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\)
by (simp add: distinct-read-tmps-append)
from read-tmps-distinct-nth-update [OF i-bound this]
show read-tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
next
from load-tmps-read-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
distinct-load-tmps-prog-step [OF i-bound \(\mathrm{ts}_{\mathbf{s b}^{\boldsymbol{b}}}\)-i prog valid]
have load-tmps (is s.sb \(^{@}\) mis) \(\cap\) read-tmps \(\left(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)=\{ \}\)
by (auto simp add: read-tmps-append load-tmps-append)
from load-tmps-read-tmps-distinct-nth-update [OF i-bound this]
show load-tmps-read-tmps-distinct \(\mathrm{ts}_{\mathbf{s b}}{ }^{\prime}\) by (simp add: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) )
qed
have valid-dd': valid-data-dependency \(\mathrm{ts}_{\text {sb }}{ }^{\prime}\)
proof -
from data-dependency-consistent-instrs [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have data-dependency-consistent-instrs (dom \(\vartheta_{\mathrm{sb}}\) ) is \(\mathrm{s}_{\mathrm{sb}}\).
with valid-data-dependency-prog-step [OF i-bound \(\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\)-i prog valid]
load-tmps-write-tmps-distinct [OF i-bound \(\mathrm{ts}_{\mathbf{s b}} \mathbf{- i}\) ]
obtain
data-dependency-consistent-instrs (dom \(\vartheta_{\mathrm{sb}}\) ) (is \(\left.\mathrm{is}_{\mathrm{sb}} @ m i s\right)\)
load-tmps (is s \(\left._{\text {sb }} @ m i s\right) \cap \bigcup\left(\right.\) fst \(^{\prime}\) write-sops \(\left.\left(\operatorname{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right)=\{ \}\)
by (force simp add: load-tmps-append data-dependency-consistent-instrs-append
write-sops-append)
from valid-data-dependency-nth-update [OF i-bound this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathbf{s b}}\) )
qed
have load-tmps-fresh': load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from load-tmps-fresh [OF i-bound \(\mathrm{ts}_{\mathrm{sb}} \mathrm{i}\) ]
load-tmps-fresh-prog-step [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}\)-i prog valid]
have load-tmps (is s. \(\left._{\text {sb }} @ m i s\right) \cap\) dom \(\vartheta_{\text {sb }}=\{ \}\)
by (auto simp add: load-tmps-append)
from load-tmps-fresh-nth-update [OF i-bound this]
show ?thesis
by (simp add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
have enough-flushs': enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from clean-no-outstanding-volatile-Write \({ }_{s b}\) [OF i-bound ts \(\left._{s b}-\mathrm{i}\right]\)
have \(\neg \mathcal{D}_{\text {sb }} \longrightarrow\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\left(\operatorname{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)=\{ \}\)
by (auto simp add: outstanding-refs-append)
from enough-flushs-nth-update [OF i-bound this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\text {sb }}\) )
qed
have valid-sops \({ }^{\prime}\) : valid-sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from valid-store-sops [OF i-bound \(\mathrm{ts}_{\mathrm{sb}^{\mathrm{b}}}\) - ] valid-sops-prog-step [OF prog]
valid-implies-valid-prog[OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) valid]
have valid-store: \(\forall\) sop \(\in\) store-sops (is sb \(@ m\) mis). valid-sop sop
by (auto simp add: store-sops-append)
from valid-write-sops [OF i-bound \(\mathrm{ts}_{\mathbf{s b}-\mathrm{i}}\) ]
have \(\forall\) sop \(\in\) write-sops \(\left(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\text {sb }} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\). valid-sop sop
by (auto simp add: write-sops-append)
from valid-sops-nth-update [OF i-bound this valid-store]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
have valid-program-history': valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
proof -
from valid-program-history [OF i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have causal-program-history is \(\mathrm{s}_{\mathrm{sb}} \mathrm{sb}\).
from causal-program-history- Prog \(_{\text {sb }}\) [OF this]
have causal': causal-program-history (is sb \(@ m i s)\) ( \(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\) ).
from last-prog-append-Prog \({ }_{\text {sb }}\)
have last-prog \(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)=\mathrm{p}_{\mathrm{sb}}{ }^{\prime}\).
from valid-program-history-nth-update [OF i-bound causal' this]
show ?thesis
by ( \(\operatorname{simp}\) add: \(\mathrm{ts}_{\mathrm{sb}}\) )
qed
show ?thesis
proof (cases outstanding-refs is-volatile-Write \({ }_{\mathbf{s b}} \mathrm{sb}=\{ \}\) )
case True
from True have flush-all: takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) \(\mathrm{sb}=\mathrm{sb}\)
by (auto simp add: outstanding-refs-conv)
from True have suspend-nothing: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) \(\mathrm{sb}=[]\)
by (auto simp add: outstanding-refs-conv)
hence suspends-empty: suspends \(=[]\)
by (simp add: suspends)
from suspends-empty is-sim have is: is \(=\) is \(_{\text {sb }}\)
by ( \(\operatorname{simp}\) )
from ts-i have ts-i: ts ! \(\mathrm{i}=\left(\mathrm{p}_{\mathrm{sb}}, \mathrm{is}_{\mathrm{sb}}, \vartheta_{\mathrm{sb}},()\right.\),
\(\mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb \(\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\)
by (simp add: suspends-empty is)
from direct-computation.Program [OF i-bound' ts-i prog]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.\), is \(\mathrm{s}_{\mathrm{sb}} @ \operatorname{mis}, \vartheta_{\mathrm{sb}},()\),
\(\mathcal{D}\), acquired True ?take-sb \(\mathcal{O}_{\mathrm{sb}}\), release ?take-sb (dom \(\left.\left.\left.\left.\mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)\).
moreover
note flush-commute \(=\) flush-all-until-volatile-write-append-Prog \({ }_{\text {sb }}\)-commute \([\mathrm{OF}\) i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
from True
have suspend-nothing':
\(\left(\operatorname{dropWhile}(\right.\) Not \(\circ\) is-volatile-Write \(\left.s b)\left(\mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right)=[]\)
by (auto simp add: outstanding-refs-conv)
note share-commute \(=\)
share-all-until-volatile-write-update-sb [OF share-append-Prog \({ }_{s b}\) i-bound \(\mathrm{ts}_{\mathbf{s b}-\mathrm{i}}\) ]
from \(\mathcal{D}\)
have \(\mathcal{D}^{\prime}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \({ }_{\mathrm{sb}}\left(\operatorname{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \operatorname{mis}\right]\right)\) \(\neq\{ \})\)
by (auto simp: outstanding-refs-append)
```

have $\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{is}_{\mathrm{sb}} @ \operatorname{mis}, \vartheta_{\mathrm{sb}}, \mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\right.$,
$\left.\mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim$
$\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}\right.\right.\right.$, is $\mathrm{s}_{\mathrm{sb}} @ \operatorname{mis}, \vartheta_{\mathrm{sb}},(), \mathcal{D}$,
acquired True (takeWhile (Not $\circ$ is-volatile-Write ${ }_{\mathbf{s b}}$ )
$\left.\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right) \mathcal{O}_{\mathrm{sb}}$,
release ( $\left.\left.\left.\left.\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right) \quad\left(\operatorname{dom} \mathcal{S}_{\mathrm{sb}}\right) \mathcal{R}_{\mathrm{sb}}\right)\right], \mathrm{m}, \mathcal{S}\right)$
apply (rule sim-config.intros)
apply (simp add: $m$ flush-commute)
apply (clarsimp simp add: $\mathcal{S} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}$ 'share-commute)
using leq
apply simp

```
using i-bound i-bound \({ }^{\prime}\) ts-sim ts-i \(\mathcal{D}^{\prime}\)
apply (clarsimp simp add: Let-def nth-list-update flush-all suspend-nothing \({ }^{\prime} \operatorname{Prog}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\)
release-append-Prog \({ }_{\text {sb }}\) release-append
split: if-split-asm)
done
ultimately show ?thesis
using valid-own' valid-hist \({ }^{\prime}\) valid-reads' valid-sharing' tmps-distinct \({ }^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime}\)
valid-dd' valid-sops' load-tmps-fresh' enough-flushs' valid-sharing'
valid-program-history' valid \({ }^{\prime}\)
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply simp add: acquired-append-Prog \({ }_{\text {sb }}\) re-lease-append-Prog \({ }_{\text {sb }}\) release-append flush-all)
next
case False
then obtain \(r\) where \(r-i n: r \in\) set sb and volatile-r: is-volatile-Write \({ }_{s b} r\)
by (auto simp add: outstanding-refs-conv)
from takeWhile-dropWhile-real-prefix
[OF r-in, of (Not \(\circ\) is-volatile-Write \({ }_{\mathrm{sb}}\) ), simplified, OF volatile-r]
obtain \(\mathrm{a}^{\prime} \mathrm{v}^{\prime} \mathrm{sb}^{\prime \prime} \operatorname{sop}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}\) where
sb-split: \(s b=\) takeWhile (Not \(\circ\) is-volatile-Write \({ }_{s b}\) ) sb @ Write \({ }_{\text {sb }}\) True \(a^{\prime} \operatorname{sop}^{\prime} v^{\prime} A^{\prime} L^{\prime} R^{\prime}\) \(\mathrm{W}^{\prime} \#\) sb"
and
drop: dropWhile (Not \(\circ\) is-volatile-Write \({ }_{\mathbf{s b}}\) ) sb \(=\) Write \(_{\text {sb }}\) True \(\mathrm{a}^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \#\) sb"
apply (auto)
subgoal for \(y\)
apply (case-tac y)
apply auto
done
done
from drop suspends have suspends': suspends \(=\) Write \(_{\text {sb }}\) True \(a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime}\) \(\mathrm{W}^{\prime} \# \mathrm{sb}^{\prime \prime}\)
by simp
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}(\mathrm{ts}, \mathrm{m}, \mathcal{S})\) by auto
moreover
note flush-commute \(=\) flush-all-until-volatile-write-append-Prog \({ }_{\text {sb }}\)-commute \([\mathrm{OF}\) i-bound \(\mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\) ]
have Write sb True \(a^{\prime} \operatorname{sop}^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime} \in\) set sb
by (subst sb-split) auto
from dropWhile-append 1 [OF this, of (Not \(\circ\) is-volatile-Write \({ }_{\text {sb }}\) )]
have drop-app-comm:
\(\left.\left(\operatorname{dropWhile}^{(\operatorname{Not} \circ \text { is-volatile-Write }} \mathrm{sbb}\right)\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\right)\right)=\) dropWhile (Not o is-volatile-Write \({ }_{s b}\) ) sb @ [ \(\left.\operatorname{Prog}_{s b} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right]\)
by simp
note share-commute \(=\)
share-all-until-volatile-write-update-sb \(\left[\mathrm{OF}\right.\) share-append-Prog \(\left.{ }_{s b} \mathrm{i}-\mathrm{bound} \mathrm{ts}_{\mathrm{sb}}-\mathrm{i}\right]\)
from \(\mathcal{D}\)
have \(\mathcal{D}^{\prime}: \mathcal{D}_{\mathrm{sb}}=\left(\mathcal{D} \vee\right.\) outstanding-refs is-volatile-Write \({ }_{\text {sb }}\left(\operatorname{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime}{ }^{\prime} \mathrm{mis}\right]\right)\) \(\neq\{ \}\) )
by (auto simp: outstanding-refs-append)
have \(\left(\mathrm{ts}_{\mathrm{sb}}\left[\mathrm{i}:=\left(\mathrm{p}_{\mathrm{sb}}{ }^{\prime}, \mathrm{is} \mathrm{s}_{\mathrm{sb}} @ \mathrm{mis}, \vartheta_{\mathrm{sb}}, \mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}} \mathrm{p}_{\mathrm{sb}}{ }^{\prime} \mathrm{mis}\right], \mathcal{D}_{\mathrm{sb}}, \mathcal{O}_{\mathrm{sb}}, \mathcal{R}_{\mathrm{sb}}\right)\right]\right.\), \(\left.\mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim\) \((\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config.intros)
apply (simp add: m flush-commute)
apply (clarsimp simp add: \(\mathcal{S} \mathcal{S}_{\text {sb }}\) 'share-commute)
using leq
apply simp
using i-bound i-bound' ts-sim ts-i is-sim suspends suspends' [simplified suspends] \(\mathcal{D}^{\prime}\)
apply (clarsimp simp add: Let-def nth-list-update Prog \(_{\text {sb }}\)
drop-app-comm instrs-append prog-instrs-append
read-tmps-append hd-prog-append-Prog \({ }_{s b}\) acquired-append-Prog \({ }_{s b}\) re-lease-append-Prog \({ }_{\text {sb }}\) release-append \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) split: if-split-asm)
done
ultimately show ?thesis
using valid-own' valid-hist' valid-reads' valid-sharing' \({ }^{\prime}\) tmps-distinct' \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime}\)
valid-dd' valid-sops \({ }^{\prime}\) load-tmps-fresh' enough-flushs' valid-sharing \({ }^{\prime}\)
valid-program-history' valid \({ }^{\prime}\)
\(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
by (auto simp del: fun-upd-apply)
qed
qed
qed
theorem (in xvalid-program) concurrent-direct-steps-simulates-store-buffer-history-steps:
assumes step-sb: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
assumes valid-own: valid-ownership \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}\)
assumes valid-sb-reads: valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\)
assumes valid-hist: valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}\)
assumes valid-sharing: valid-sharing \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}\)
assumes tmps-distinct: tmps-distinct ts tsb
assumes valid-sops: valid-sops \(\mathrm{ts}_{\mathrm{sb}}\)
assumes valid-dd: valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}\)
assumes load-tmps-fresh: load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}\)
assumes enough-flushs: enough-flushs \(\mathrm{ts}_{\mathrm{sb}}\)
assumes valid-program-history: valid-program-history \(\mathrm{ts}_{\text {sb }}\)
assumes valid: valid ts \(\mathrm{ts}_{\mathrm{sb}}\)
shows \(\Lambda \mathrm{ts} \mathcal{S} \mathrm{m} .\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Longrightarrow\) safe-reach-direct safe-delayed ( \(\mathrm{ts}, \mathrm{m}, \mathcal{S}\) )
valid-ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) valid-history program-step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
\(\wedge\)
valid-sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) tmps-distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) valid-data-dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\)
valid-sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) load-tmps-fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) enough-flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\)
valid-program-history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\) valid \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \wedge\)
\(\left(\exists \mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime} .(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\right)\)
using step－sb valid－own valid－sb－reads valid－hist valid－sharing tmps－distinct valid－sops valid－dd load－tmps－fresh enough－flushs valid－program－history valid
proof（induct rule：converse－rtranclp－induct－sbh－steps）
case refl thus ？case
by auto
next
case（step ts \({ }_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime \prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime \prime}\) ）
note first \(=\left\{\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime \prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime \prime}\right)\right\rangle\)
note \(\operatorname{sim}=\left\langle\left(\mathrm{t}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\right\rangle\)
note safe－reach \(=\langle\) safe－reach－direct safe－delayed（ts，m， \(\mathcal{S})\rangle\)
note valid－own \(=\) «valid－ownership \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}\) 〉
note valid－reads \(=\) 〔valid－reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉
note valid－hist \(=\) 〈valid－history program－step \(\left.\mathrm{ts}_{\mathrm{sb}_{\mathrm{b}}}\right\rangle\)
note valid－sharing \(=\) 〔valid－sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉
note tmps－distinct \(=\left\langle\right.\) tmps－distinct \(\left.\mathrm{ts}_{\mathrm{sb}}\right\rangle\)
note valid－sops \(=\) 〔valid－sops ts \(\left._{\text {sb }}\right\rangle\)
note valid－dd \(=\) 〔valid－data－dependency \(\mathrm{ts}_{\mathrm{sb}}\) 〉
note load－tmps－fresh \(=\) 〈load－tmps－fresh \(\mathrm{ts}_{\mathrm{sb}}\) 〉
note enough－flushs \(=\left\langle\right.\) enough－flushs \(\left.\mathrm{ts}_{\mathrm{sb}}\right\rangle\)
note valid－prog－hist \(=\) 〈valid－program－history \(\mathrm{ts}_{\mathrm{sb}}\) 〉
note valid \(=\left\langle\right.\) valid \(\left.\mathrm{ts}_{\text {sb }}\right\rangle\)
from concurrent－direct－steps－simulates－store－buffer－history－step［OF first
valid－own valid－reads valid－hist valid－sharing tmps－distinct valid－sops valid－dd
load－tmps－fresh enough－flushs valid－prog－hist valid sim safe－reach］
obtain \(\mathrm{ts}^{\prime \prime} \mathrm{m}{ }^{\prime \prime} \mathcal{S}^{\prime \prime}\) where
valid－own＂：valid－ownership \(\mathcal{S}_{\text {sb }}{ }^{\prime \prime}\) ts \(_{\text {sb }}\)＂and
valid－reads＂：valid－reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime \prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}\) and valid－hist＂：valid－history program－step \(\mathrm{ts}_{\text {sb }}{ }^{\prime \prime}\) and
valid－sharing \({ }^{\prime \prime}\) ：valid－sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime \prime}\) ts \(_{\text {sb }}{ }^{\prime \prime}\) and
tmps－dist＂：tmps－distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}\) and
valid－dd \({ }^{\prime \prime}\) ：valid－data－dependency \(\mathrm{ts}_{\text {sb }}{ }^{\prime \prime}\) and valid－sops＂：valid－sops \(\mathrm{ts}_{\text {sb }}{ }^{\prime \prime}\) and
load－tmps－fresh＂：load－tmps－fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}\) and
enough－flushs＂：enough－flushs \(\mathrm{ts}_{\text {sb }}{ }^{\prime \prime}\) and
valid－prog－hist＂：valid－program－history \(\mathrm{ts}_{\mathrm{sb}}\)＂and
valid \({ }^{\prime \prime}\) ：valid \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}\) and
steps：\((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}\left(\mathrm{ts}^{\prime \prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\) and
sim：\(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime \prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime \prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime \prime}\right) \sim\left(\mathrm{ts}^{\prime \prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\)
by blast
from step．hyps（3）［OF sim safe－reach－steps［OF safe－reach steps］valid－own＂valid－reads \({ }^{\prime \prime}\) valid－hist＂valid－sharing＂
tmps－dist＂valid－sops＂valid－dd＂load－tmps－fresh＂enough－flushs＂valid－prog－hist＂valid＂ ］
obtain \(\mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime}\) where
valid：valid－ownership \(\mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid－reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid－history program－step \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid－sharing \(\mathcal{S}_{\text {sb }}{ }^{\prime}\) ts \(_{\text {sb }}{ }^{\prime}\) tmps－distinct \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid－data－dependency \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid－sops \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) load－tmps－fresh \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) enough－flushs \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
valid－program－history \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) and
last：\(\left(\mathrm{ts}^{\prime \prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\) and
\(\operatorname{sim}^{\prime}:\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
by blast
note steps also note last
finally show ？case
using valid sim＇
by blast
qed
sublocale initial \({ }_{\text {sb }} \subseteq\) tmps－distinct..\(^{\text {．}}\)
locale xvalid－program－progress \(=\) program－progress + xvalid－program
theorem（in xvalid－program－progress）concurrent－direct－execution－simulates－store－buffer－history－execution： assumes exec－sb：\(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow{ }_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
assumes init：initial \({ }_{\text {sb }} \operatorname{ts}_{\text {sb }} \mathcal{S}_{\text {sb }}\)
assumes valid：valid \(\mathrm{ts}_{\mathrm{sb}}\)
assumes sim：\(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
assumes safe：safe－reach－direct safe－free－flowing（ts，m，S
shows \(\exists \mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime} .(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge\)
\(\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
proof－
from init interpret ini：initial \({ }_{\text {sb }}\) ts \(_{\text {sb }} \mathcal{S}_{\mathrm{sb}}\) ．
from safe－free－flowing－implies－safe－delayed＇［OF init sim safe］
have safe－delayed：safe－reach－direct safe－delayed（ts，m， \(\mathcal{S}\) ）．
from local．ini．valid－ownership－axioms have valid－ownership \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－reads－axioms have valid－reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－history－axioms have valid－history program－step \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－sharing－axioms have valid－sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．tmps－distinct－axioms have tmps－distinct \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－sops－axioms have valid－sops \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－data－dependency－axioms have valid－data－dependency \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．load－tmps－fresh－axioms have load－tmps－fresh \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．enough－flushs－axioms have enough－flushs \(\mathrm{ts}_{\mathrm{sb}}\) ．
from local．ini．valid－program－history－axioms have valid－program－history \(\mathrm{ts}_{\mathrm{sb}}\) ．
from concurrent－direct－steps－simulates－store－buffer－history－steps［OF exec－sb
〈valid－ownership \(\mathcal{S}_{\text {sb }} \mathrm{ts}_{\text {sb }}\) 〉
〈valid－reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 〈valid－history program－step \(\mathrm{ts}_{\mathrm{sb}}\) 〉
〈valid－sharing \(\mathcal{S}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sb}}\) 〉 〈tmps－distinct \(\mathrm{ts}_{\mathrm{sb}}\) 〉 〈valid－sops \(\mathrm{ts}_{\mathrm{sb}}\) 〉

«valid－program－history \(\mathrm{ts}_{\mathrm{sb}}\) 〉 valid sim safe－delayed］
show ？thesis by auto
qed
```

lemma filter-is-Write $_{\mathbf{s b}}$-Cons-Write $_{\mathbf{s b}}$ : filter is-Write ${ }_{\mathbf{s b}} \mathrm{xs}=$ Write $_{\mathbf{s b}}$ volatile a sop v A L
R W\#ys
$\Longrightarrow \exists \mathrm{rs}$ rws. $\left(\forall \mathrm{r} \in\right.$ set rs. is-Read ${ }_{\text {sb }} r \vee$ is- Prog $_{\text {sb }} \mathrm{r} \vee$ is-Ghost $\left.{ }_{\text {sb }} \mathrm{r}\right) \wedge$
$\mathrm{xs}=\mathrm{rs} @ W_{r i t e}^{s b}$ volatile a sop v A L R W\#rws $\wedge$ ys=filter is-Write ${ }_{\text {sb }}$ rws
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
note feq $=\left\langle\right.$ filter is- Write $_{s b}(\mathrm{x} \# \mathrm{xs})=$ Write $_{\text {sb }}$ volatile a sop v A L R W\# ys $\rangle$
show ?case
proof (cases x)
case ( Write $_{\text {sb }}$ volatile $\mathrm{a}^{\prime}$ sop $^{\prime} \mathrm{v}^{\prime} \mathrm{A}^{\prime} \mathrm{L}^{\prime} \mathrm{R}^{\prime} \mathrm{W}^{\prime}$ )
with feq obtain volatile ${ }^{\prime}=$ volatile $\mathrm{a}^{\prime}=\mathrm{a} \mathrm{v}^{\prime}=\mathrm{v}$ sop ${ }^{\prime}=\operatorname{sop} \mathrm{A}^{\prime}=\mathrm{A} \mathrm{L}^{\prime}=\mathrm{L} \mathrm{R}^{\prime}=\mathrm{R} \mathrm{W}^{\prime}=\mathrm{W}$
ys $=$ filter is-Write ${ }_{s b} \mathrm{xs}$
by auto
thus ?thesis
apply -
apply (rule-tac $\mathrm{x}=[]$ in exI)
apply (rule-tac $x=x$ in exI)
apply (simp add: Write ${ }_{\text {sb }}$ )
done
next
case ( Read $_{\text {sb }}$ volatile $\mathrm{a}^{\prime} \mathrm{t}^{\prime}$ v )
from feq have filter is- Write $_{\text {sb }} \mathrm{xs}=$ Write $_{\text {sb }}$ volatile a sop v A L R W\#ys
by (simp add: $\operatorname{Read}_{\mathrm{sb}}$ )
from Cons.hyps [OF this] obtain rs rws where
$\forall r \in$ set rs. is-Read ${ }_{\text {sb }} r \vee$ is- Prog $_{s b} r \vee$ is-Ghostsb $r$ and
xs=rs @ Writesb volatile a sop v A L R W\# rws and
ys=filter is-Write ${ }_{\text {sb }}$ rws
by clarsimp
then show ?thesis
apply -
apply (rule-tac $\mathrm{x}=\operatorname{Read}_{\mathrm{sb}}$ volatile $\mathrm{a}^{\prime} \mathrm{a}^{\prime} \mathrm{t}^{\prime} \mathrm{v}^{\prime} \# \mathrm{rs}$ in exI)
apply (rule-tac $x=r w s$ in exI)
apply (simp add: Read $_{\mathrm{sb}}$ )
done
next
case ( $\operatorname{Prog}_{\mathrm{sb}} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}$ )
from feq have filter is-Write ${ }_{\text {sb }} \mathrm{xs}=$ Write $_{\text {sb }}$ volatile a sop v A L R W\#ys
by (simp add: $\operatorname{Prog}_{\text {sb }}$ )
from Cons.hyps [OF this] obtain rs rws where
$\forall r \in$ set rs. is-Read ${ }_{\text {sb }} r \vee{\text { is- }-\operatorname{Prog}_{s b} r} \vee$ is-Ghost ${ }_{\text {sb }} r$ and
xs=rs @ Write ${ }_{\text {sb }}$ volatile a sop v A L R W\# rws and
ys=filter is-Write ${ }_{\text {sb }}$ rws
by clarsimp
then show ?thesis
apply -
apply (rule-tac $\mathrm{x}=\operatorname{Prog}_{\text {sb }} \mathrm{p}_{1} \mathrm{p}_{2}$ mis\#rs in exI)
apply (rule-tac $x=r w s$ in exI)
apply (simp add: Prog ${ }_{\text {sb }}$ )

```
```

        done
    next
    case (Ghost sb A' L' R' W')
    from feq have filter is-Write sb xs = Write sb volatile a sop v A L R W # ys
        by (simp add: Ghost sb)
    from Cons.hyps [OF this] obtain rs rws where
        \forallr set rs. is-Read sb r V is--Prog
        xs=rs @ Write
        ys=filter is-Write sb rws
        by clarsimp
    then show ?thesis
        apply -
        apply (rule-tac x=Ghost sb A' L' R' W'#rs in exI)
        apply (rule-tac x=rws in exI)
        apply (simp add: Ghost sb)
        done
    qed
    qed
lemma filter-is-Write sb-empty: filter is-Write
\Longrightarrow ( \forall \mathrm { r } \in set xs. is-Readsb r \vee \mp@code { i s - P r o g }
proof (induct xs)
case Nil thus ?case by simp
next
case (Cons x xs)
note feq = «filter is-Write sb (x\#xs) = []>
show ?case
proof (cases x)
case (Write sb volatile' a' v}\mp@subsup{}{}{\prime}\mathrm{ )
with feq have False
by simp
thus ?thesis ..
next
case (Readsb a' v')
from feq have filter is-Write sb xs = []
by (simp add: Readsb)
from Cons.hyps [OF this] obtain
\forallr\in set xs. is-Readsb r V is-Prog
by clarsimp
then show ?thesis
by (simp add: Readsb
next
case (Prog
from feq have filter is-Write }\mp@subsup{}{\mathrm{ sb }}{}\mathrm{ xs = [
by (simp add: Prog
from Cons.hyps [OF this] obtain
\forallr\in set xs. is-Read sb r r is-Prog
by clarsimp
then show ?thesis
by (simp add: Prog

```
```

next
case (Ghost sb A' L' R' W')
from feq have filter is-Write }\mp@subsup{\mathrm{ sb xs }}{\mathrm{ x }}{=[]
by (simp add: Ghost sb)
from Cons.hyps [OF this] obtain
\forallr set xs. is-Read sb r V is-Prog
by clarsimp
then show ?thesis
by (simp add: Ghost sb
qed
qed
lemma flush-reads-program: }\bigwedge\mathcal{O}\mathcal{S}\mathcal{R}
r \in set sb. is-Read
\exists\mathcal{O}}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime}.(\textrm{m},\textrm{sb},\mathcal{O},\mathcal{R},\mathcal{S})\mp@subsup{->}{\textrm{f}}{*
proof (induct sb)
case Nil thus ?case by auto
next
case (Cons x sb)
note <br>forallr\inset (x \# sb). is-Read cor r V is-Prog

```

```

r V is-Progsb r V is-Ghostsb r
by (cases x) auto

```
```

{
assume is-Read sb x
then obtain volatile a t v where x: x= Read sb volatile a t v
by (cases x) auto
have (m,Read sb volatile a t v \#sb,\mathcal{O},\mathcal{R},\mathcal{S})}\mp@subsup{->}{\textrm{f}}{}(\textrm{m},\textrm{sb},\mathcal{O},\mathcal{R},\mathcal{S}
by (rule Read
also
from Cons.hyps [OF sb] obtain }\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ acq}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime
where (m, sb,\mathcal{O},\mathcal{R},\mathcal{S})->\mp@subsup{->}{\textrm{f}}{*}(\textrm{m},[],\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime})\mathrm{ by blast}
finally
have ?case
by (auto simp add: x)
}
moreover
{
assume is- - Prog
then obtain p1 p
by (cases x) auto
have (m, Prog
by (rule Prog
also
from Cons.hyps [OF sb] obtain }\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime}\mathrm{ acq'
where (m, sb,\mathcal{O},\mathcal{R},\mathcal{S})->\mp@subsup{f}{}{*}(\textrm{m},[],\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime},\mathcal{S})}\mathrm{ ) by blast

```
```

    finally
    have ?case
        by (auto simp add: x)
    }
moreover
{
assume is-Ghostsb x
then obtain A L R W where x: x= Ghost sb A L R W
by (cases x) auto
have (m,Ghost sb A L R W\#sb,\mathcal{O},\mathcal{R},\mathcal{S})}\mp@subsup{->}{\textrm{f}}{(m,sb,\mathcal{O}\cup\textrm{A}
R},\mathcal{S}\oplus\mp@subsup{\textrm{w}}{\textrm{W}}{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L}
by (rule Ghost)
also
from Cons.hyps [OF sb] obtain }\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{S}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}\mathrm{ acq'
where (m, sb,\mathcal{O}\cupA - R ,augment-rels (dom \mathcal{S})R\mathcal{R},\mathcal{S}\oplus\textrm{W}R\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})\mp@subsup{->}{\textrm{f}}{*
[],\mathcal{O}}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime})\mathrm{ by blast
finally
have ?case
by (auto simp add: x)
}
ultimately show ?case
using x by blast
qed

```
lemma flush-progress: \(\exists \mathrm{m}^{\prime} \mathcal{O}^{\prime} \mathcal{S}^{\prime} \mathcal{R}^{\prime} .(\mathrm{m}, \mathrm{r} \# \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
proof (cases r)
    case \(\left(W_{\text {Write }}^{\text {sb }}\right.\) volatile a sop v A L R W)
    from flush-step. Write \({ }_{\text {sb }}\) [OF refl refl refl, of m volatile a sop v A L R W sb \(\left.\mathcal{O} \mathcal{R} \mathcal{S}\right]\)
    show ?thesis
        by (auto simp add: Write \({ }_{\text {sb }}\) )
next
    case \(\left(\right.\) Read \(_{\text {sb }}\) volatile a \(\left.\mathrm{t} v\right)\)
    from flush-step. Read \(_{\text {sb }}[\) of m volatile a \(\mathrm{t} \mathrm{v} \operatorname{sb} \mathcal{O} \mathcal{R} \mathcal{S}]\)
    show ?thesis
        by (auto simp add: Read sb )
next
    case ( \(\operatorname{Prog}_{s b} \mathrm{p}_{1} \mathrm{p}_{2} \mathrm{mis}\) )
    from flush-step. \(\operatorname{Prog}_{\text {sb }}\left[\right.\) of \(\mathrm{m} \mathrm{p}_{1} \mathrm{p}_{2}\) mis \(\left.\operatorname{sb} \mathcal{O} \mathcal{R} \mathcal{S}\right]\)
    show ?thesis
        by (auto simp add: \(\operatorname{Prog}_{s b}\) )
next
    case \(\left(\right.\) Ghost \(_{\text {sb }}\) A L R W)
    from flush-step.Ghost [of m A L R W sb \(\mathcal{O} \mathcal{R} \mathcal{S}]\)
    show ?thesis
        by (auto simp add: Ghost \({ }_{\text {sb }}\) )
qed
lemma flush-empty:
assumes steps: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows \(\mathrm{sb}=[] \Longrightarrow \mathrm{m}^{\prime}=\mathrm{m} \wedge \mathrm{sb}^{\prime}=[] \wedge \mathcal{O}^{\prime}=\mathcal{O} \wedge \mathcal{R}^{\prime}=\mathcal{R} \wedge \mathcal{S}^{\prime}=\mathcal{S}\)
using steps
apply (induct rule: converse-rtranclp-induct5)
apply (auto elim: flush-step.cases)
done
lemma flush-append:
assumes steps: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows \(\bigwedge \mathrm{xs} .(\mathrm{m}, \mathrm{sb} @ \mathrm{xs}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m}^{\prime}, \mathrm{sb}^{\prime} @ \mathrm{xs}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
using steps
proof (induct rule: converse-rtranclp-induct5)
case refl thus ?case by auto
next
case (step \(\mathrm{m} \operatorname{sb} \mathcal{O} \mathcal{R} \mathcal{S} \mathrm{m}^{\prime \prime} \mathrm{sb}^{\prime \prime} \mathcal{O}^{\prime \prime} \mathcal{R}^{\prime \prime} \mathcal{S}^{\prime \prime}\) )
note first \(=\left\langle(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime \prime}, \mathrm{sb}^{\prime \prime}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\right\rangle\)
note rest \(=\left\langle\left(\mathrm{m}^{\prime \prime}, \mathrm{sb}^{\prime \prime}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right) \rightarrow_{\mathrm{f}}^{*}\left(\mathrm{~m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\right\rangle\)
from step.hyps (3) have append-rest: ( \(\left.\mathrm{m}^{\prime \prime}, \mathrm{sb}^{\prime \prime @ x s,} \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m}^{\prime}\right.\),
\(\left.\mathrm{sb}^{\prime} @ \mathrm{xs}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\).
from first show ?case
proof (cases)
case ( Write \(_{\text {sb }}\) volatile A R W L a sop v)
then obtain \(\mathrm{sb}: \mathrm{sb}=\mathrm{Write}_{\mathrm{sb}}\) volatile a sop v A L R W\#sb" and \(\mathrm{m}^{\prime \prime}: \mathrm{m}^{\prime \prime}=\mathrm{m}(\mathrm{a}:=\mathrm{v})\)
and
\(\mathcal{O}^{\prime \prime}: \mathcal{O}^{\prime \prime}=(\) if volatile then \(\mathcal{O} \cup \mathrm{A}-\mathrm{R}\) else \(\mathcal{O})\) and
\(\mathcal{R}^{\prime \prime}: \mathcal{R}^{\prime \prime}=(\) if volatile then Map.empty else \(\mathcal{R})\) and
\(\mathcal{S}^{\prime \prime}: \mathcal{S}^{\prime \prime}=\left(\right.\) if volatile then \(\mathcal{S} \oplus \mathrm{w} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) else \(\left.\mathcal{S}\right)\)
by auto
have \(\left(\mathrm{m}, \mathrm{Write}_{\text {sb }}\right.\) volatile a sop v A L R W\#sb" \(\left.@ x s, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}\)
\(\left(\mathrm{m}(\mathrm{a}:=\mathrm{v}), \mathrm{sb}{ }^{\prime \prime} @ \mathrm{xs}\right.\), if volatile then \(\mathcal{O} \cup \mathrm{A}-\mathrm{R}\) else \(\mathcal{O}\), if volatile then Map.empty else
\(\mathcal{R}\),
if volatile then \(\mathcal{S} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\) else \(\left.\mathcal{S}\right)\)
apply (rule flush-step.Write \({ }_{\text {sb }}\) )
apply auto
done
hence \((\mathrm{m}, \mathrm{sb} @ \mathrm{xs}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime \prime}, \mathrm{sb}{ }^{\prime \prime} @ \mathrm{xs}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\)
by (simp add: \(\mathrm{sb} \mathrm{m}^{\prime \prime} \mathcal{O}^{\prime \prime} \mathcal{R}^{\prime \prime} \mathcal{S}^{\prime \prime}\) )
also note append-rest
finally show ?thesis .
next
case \(\left(\operatorname{Read}_{\mathrm{sb}}\right.\) volatile at v\()\)
then obtain sb : \(\mathrm{sb}=\operatorname{Read}_{\mathrm{sb}}\) volatile at \(\mathrm{v} \# \mathrm{sb}^{\prime \prime}\) and \(\mathrm{m}^{\prime \prime}: \mathrm{m}^{\prime \prime}=\mathrm{m}\)
and \(\mathcal{O}^{\prime \prime}: \mathcal{O}^{\prime \prime}=\mathcal{O}\) and \(\mathcal{S}^{\prime \prime}: \mathcal{S}^{\prime \prime}=\mathcal{S}\) and \(\mathcal{R}^{\prime \prime}: \mathcal{R}^{\prime \prime}=\mathcal{R}\)
by auto
have \(\left(\mathrm{m}_{,}\right.\)Read \(_{\mathrm{sb}}\) volatile a \(\left.\mathrm{t} \mathrm{v} \# \mathrm{sb}^{\prime \prime} @ \mathrm{xs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right) \rightarrow_{\mathrm{f}}\left(\mathrm{m}, \mathrm{sb}^{\prime \prime} @ \mathrm{xs}, \mathcal{O}, \mathcal{R}, \mathcal{S}\right)\)
by (rule flush-step.Read \({ }_{\text {sb }}\) )
hence ( \(\mathrm{m}, \mathrm{sb} @ \mathrm{xs}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime \prime}, \mathrm{sb}{ }^{\prime \prime} @ \mathrm{xs}, \mathcal{O}^{\prime \prime}, \mathcal{R}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\)
by ( \(\operatorname{simp}\) add: \(\mathrm{sb} \mathrm{m}^{\prime \prime} \mathcal{O}^{\prime \prime} \mathcal{R}^{\prime \prime} \mathcal{S}^{\prime \prime}\) )
also note append-rest
finally show ?thesis .
```

next
case (Prog

```

```

        and }\mp@subsup{\mathcal{O}}{}{\prime\prime}:\mp@subsup{\mathcal{O}}{}{\prime\prime}=\mathcal{O}\mathrm{ and }\mp@subsup{\mathcal{S}}{}{\prime\prime}:\mp@subsup{\mathcal{S}}{}{\prime\prime}=\mathcal{S}\mathrm{ and }\mp@subsup{\mathcal{R}}{}{\prime\prime}:\mp@subsup{\mathcal{R}}{}{\prime\prime}=\mathcal{R
        by auto
    ```

```

        by (rule flush-step.Prog
    hence (m,sb@xs,\mathcal{O},\mathcal{R},\mathcal{S})\mp@subsup{->}{\textrm{f}}{}(\mp@subsup{\textrm{m}}{}{\prime\prime},\mp@subsup{\textrm{sb}}{}{\prime\prime}@\textrm{xs},\mp@subsup{\mathcal{O}}{}{\prime\prime},\mp@subsup{\mathcal{R}}{}{\prime\prime},\mp@subsup{\mathcal{S}}{}{\prime\prime})
        by (simp add: sb m" }\mp@subsup{\mathcal{O}}{}{\prime\prime}\mp@subsup{\mathcal{R}}{}{\prime\prime}\mp@subsup{\mathcal{S}}{}{\prime\prime}
    also note append-rest
    finally show ?thesis .
    next
case (Ghost A L R W)
then obtain sb: sb=Ghost sb A L R W\#sb"' and m": m"=m
and }\mp@subsup{\mathcal{O}}{}{\prime\prime}:\mp@subsup{\mathcal{O}}{}{\prime\prime}=\mathcal{O}\cup\textrm{A}-\textrm{R}\mathrm{ and }\mp@subsup{\mathcal{S}}{}{\prime\prime}:\mp@subsup{\mathcal{S}}{}{\prime\prime}=\mathcal{S}\oplus\textrm{w}R\mp@subsup{\ominus}{\textrm{A}}{}\quad\textrm{L}\mathrm{ and
\mp@subsup{\mathcal{R}}{}{\prime\prime}:\mp@subsup{\mathcal{R}}{}{\prime\prime}=\mathrm{ augment-rels (dom S S) R }\mathcal{R}
by auto
have (m,Ghost sb A L R W\#sb''@xs,\mathcal{O},\mathcal{R},\mathcal{S})\mp@subsup{->}{\textrm{f}}{(m,sb}\mp@subsup{}{}{\prime\prime}@xs,\mathcal{O}\cup\textrm{A}
(dom S S R \mathcal{R},\mathcal{S}\oplus\mp@subsup{\textrm{w}}{}{\textrm{R}}\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L})
by (rule flush-step.Ghost)
hence (m,sb@xs,\mathcal{O},\mathcal{R},\mathcal{S})\mp@subsup{->}{\textrm{f}}{}(\mp@subsup{\textrm{m}}{}{\prime\prime},\mp@subsup{\textrm{sb}}{}{\prime\prime}@xs,\mp@subsup{\mathcal{O}}{}{\prime\prime},\mp@subsup{\mathcal{R}}{}{\prime\prime},\mp@subsup{\mathcal{S}}{}{\prime\prime})
by (simp add: sb m"}\mp@subsup{}{}{\prime\prime}\mp@subsup{\mathcal{O}}{}{\prime\prime}\mp@subsup{\mathcal{R}}{}{\prime\prime}\mp@subsup{\mathcal{S}}{}{\prime\prime}
also note append-rest
finally show ?thesis .
qed
qed

```
lemmas store-buffer-step-induct \(=\)
    store-buffer-step.induct [split-format (complete),
    consumes 1, case-names SBWrite \({ }_{\text {sb }}\) ]
theorem flush-simulates-filter-writes:
    assumes step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{w}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows \(\bigwedge \operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{S}_{\mathrm{h}}\). sb=filter is-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}\)
```

    \Longrightarrow
    \exists\mp@subsup{\textrm{sb}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime}.(\textrm{m},\mp@subsup{\textrm{sb}}{\textrm{h}}{},\mp@subsup{\mathcal{O}}{\textrm{h}}{},\mp@subsup{\mathcal{R}}{\textrm{h}}{},\mp@subsup{\mathcal{S}}{\textrm{h}}{})->\mp@subsup{->}{\textrm{f}}{}\mp@subsup{}{}{*}(\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\textrm{sb}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime})
    sb'=filter is-Write sb }\mp@subsup{\textrm{sb}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\wedge(\mp@subsup{\textrm{sb}}{}{\prime}=[]\longrightarrow\mp@subsup{\textrm{sb}}{\textrm{h}}{}\mp@subsup{}{}{\prime}=[]

```
using step
proof (induct rule: store-buffer-step-induct)
case \(\left(\mathrm{SBWrite}_{\mathrm{sb}} \mathrm{m}\right.\) volatile a \(\left.\mathrm{D} \mathrm{f} v \mathrm{ALR} \mathrm{W} \operatorname{sb} \mathcal{O} \mathcal{R} \mathcal{S}\right)\)

from filter-is-Write \({ }_{s b}\)-Cons-Write Sb \(_{\text {b }}\) [OF filter-Write \({ }_{s b}\) [symmetric]]
obtain rs rws where
    rs-reads: \(\forall r \in\) set rs. is-Read \({ }_{\text {sb }} r \vee\) is- \(\operatorname{Prog}_{s b} r \vee\) is-Ghost \({ }_{\text {sb }} r\) and
    \(\mathrm{sb}_{\mathrm{h}}: \mathrm{sb}_{\mathrm{h}}=\mathrm{rs} @\) Write \(_{\mathrm{sb}}\) volatile a \((\mathrm{D}, \mathrm{f})\) v A L R W\# rws and
    \(\mathrm{sb}: \mathrm{sb}=\) filter is-Write \({ }_{\text {sb }}\) rws
    by blast
from flush-reads-program [OF rs-reads] obtain \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \mathcal{R}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathrm{acq}_{\mathrm{h}}{ }^{\prime}\)
```

    where (m, rs, (\mathcal{O},\mp@subsup{\mathcal{R}}{\textrm{h}}{},\mp@subsup{\mathcal{S}}{\textrm{h}}{})\mp@subsup{->}{\textrm{f}}{}\mp@subsup{}{}{*}(\textrm{m},[],\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime})\mathrm{ by blast}
    from flush-append [OF this]
    have (m, rs@Write sb volatile a (D,f) v A L R W # rws, (\mathcal{O}
        (m, Write sb volatile a (D,f) v A L R W# rws, (\mathcal{O}}\mp@subsup{}{\textrm{\prime}}{},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime}
        by simp
    also
    from flush-step.Write sb [OF refl refl refl, of m volatile a (D,f) v A L R W rws }\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime
    S ( }\mp@subsup{}{}{\prime
obtain }\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime
where (m, Write sb volatile a (D,f) v A L R W\# rws, \mathcal{O}}\mp@subsup{}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime})\mp@subsup{->}{\textrm{f}}{}(\textrm{m}(\textrm{a}:=\textrm{v}),rws
\mathcal{O}}\mp@subsup{\textrm{h}}{}{\prime\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}
by auto
finally have steps: (m, sb }\mp@subsup{\textrm{h}}{\textrm{h}}{,}\mp@subsup{\mathcal{O}}{\textrm{h}}{},\mp@subsup{\mathcal{R}}{\textrm{h}}{},\mp@subsup{\mathcal{S}}{\textrm{h}}{})\mp@subsup{->}{\textrm{f}}{}\mp@subsup{}{}{*}(\textrm{m}(\textrm{a}:=\textrm{v}),\mathrm{ rws,}\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}
by (simp add: sbb
show ?case
proof (cases sb)
case Cons
with steps sb show ?thesis
by fastforce
next
case Nil
from filter-is-Write sb-empty [OF sb [simplified Nil, symmetric]]
have }\forallr\in\mathrm{ set rws. is-Read }\mp@subsup{\mp@code{sb}}{}{}r\vee is-Prog sb r V is-Ghost sb r.
from flush-reads-program [OF this] obtain }\mp@subsup{\mathcal{O}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime}\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime}\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime}\mp@subsup{\textrm{acqu}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime
where (m(a:=v), rws,\mathcal{O}}\mp@subsup{}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime})->\mp@subsup{->}{\textrm{f}}{*}(\textrm{m}(\textrm{a}:=\textrm{v}),[],\mp@subsup{\mathcal{O}}{\textrm{h}}{\prime\prime\prime},\mp@subsup{\mathcal{R}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime\prime})\mathrm{ by blast
with steps

```

```

        with sb Nil show ?thesis by fastforce
    qed
    qed

```
lemma bufferd-val-filter-is-Write sb \(_{\text {-eq-ext: }}\)
    buffered-val (filter is-Write sb \(^{s b}\) ) \(\mathrm{a}=\) buffered-val sb a
    by (induct sb) (auto split: memref.splits)
lemma bufferd-val-filter-is-Write sb \(^{\text {- eq }}\) :
    buffered-val (filter is-Write \({ }_{\text {sb }}\) sb) \(=\) buffered-val sb
    by (rule ext) (rule bufferd-val-filter-is-Write \({ }_{\text {sb }}\)-eq-ext)
lemma outstanding-refs-is-volatile-Write \({ }_{s b}\)-filter-writes:
    outstanding-refs is-volatile-Write \({ }_{\text {sb }}\) (filter is-Write \({ }_{\text {sb }} \mathrm{xs}\) ) \(=\)
    outstanding-refs is-volatile-Write \({ }_{s b}\) xs
    by (induct xs) (auto simp add: is-volatile-Write sb \(_{\text {sb }}\)-def split: memref.splits)

\section*{A. 6 Simulation of Store Buffer Machine without History by Store Buffer Machine with History}
theorem (in valid-program) concurrent-history-steps-simulates-store-buffer-step:
assumes step-sb: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{sb}}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes sim: \(\mathrm{ts} \sim_{h} \mathrm{ts}_{\mathrm{h}}\)
```

    shows \(\exists \mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} .\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow{ }_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \wedge \mathrm{ts}^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
    ```
proof -
    interpret sbh-computation:
    computation sbh-memop-step flush-step program-step
        \(\lambda \mathrm{p} \mathrm{p}^{\prime}\) is sb. sb @ \(\left[\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.\) is \(]\).
    from step-sb
    show ?thesis
    proof (cases rule: concurrent-step-cases)
    case (Memop i - p is \(\vartheta \operatorname{sb} \mathcal{D} \mathcal{O} \mathcal{R}-\) is \(^{\prime} \vartheta^{\prime} \mathrm{sb}^{\prime}-\mathcal{D}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime}\) )
    then obtain
        \(\mathrm{ts}^{\prime}: \mathrm{ts}^{\prime}=\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\) and
        i-bound: i < length ts and
        ts-i: ts! \(\mathrm{i}=(\mathrm{p}\), is, \(\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\) and
        step-sb: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sb}}\)
                        (is \({ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\) )
        by auto
    from sim obtain
    lts-eq: length ts \(=\) length \(t_{h}\) and
    sim-loc: \(\forall \mathrm{i}<\) length ts. \(\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.\).
                        let \((\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{h}}!\mathrm{i}\) in
                        ts! \(i=\left(p\right.\), is, \(\vartheta\), filter is-Write \(\left.{ }_{\text {sb }} \operatorname{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right) \wedge\)
                (filter is-Write \({ }_{\text {sb }} \mathrm{sb}=[] \longrightarrow \mathrm{sb}=[]\) )
        by cases (auto)
    from lts-eq i-bound have i-bound': i \(<\) length ts \(_{\mathrm{h}}\)
    by simp
    from step-sb
    show ?thesis
    proof (cases)
        case (SBReadBuffered a v volatile t)
        then obtain
is: is \(=\) Read volatile a \(\mathrm{t} \#\) is \(^{\prime}\) and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
    \(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(m^{\prime}: m^{\prime}=m\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{v})\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\) and
buf-val: buffered-val sb \(\mathrm{a}=\) Some v
by auto
from sim-loc [rule-format, OF i-bound] ts-i is
obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}\right.\), Read volatile a \(\left.\mathrm{t} \# \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and \(\mathrm{sb}: \mathrm{sb}=\) filter is-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}\) and
sb-empty: filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}=[]\)
by (auto simp add: Let-def)
from buf-val
have buf-val': buffered-val \(\mathrm{sb}_{\mathrm{h}} \mathrm{a}=\) Some v
by (simp add: bufferd-val-filter-is-Write sb \(^{\boldsymbol{b}}\)-eq sb )
let \(? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}=\left(\mathrm{p}\right.\), is \(^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\) volatile a t v\(\left.], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\)
let \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}\right]\)
from sbh-memop-step.SBHReadBuffered [OF buf-val']
have (Read volatile at \(\#\) is \(\left.{ }^{\prime}, \vartheta, \operatorname{sb}_{\mathrm{h}}, \mathrm{m}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{sbh}}\) \(\left(\mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.\) volatile a t v\(\left.], \mathrm{m}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right)\).
from sbh-computation.Memop [OF i-bound \({ }^{\prime}\) ts \(_{\mathrm{h}}\)-i this]
have step: \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\).
from sb have \(\mathrm{sb}: \mathrm{sb}=\) filter is-Write \(\mathrm{sb}_{\mathrm{sb}}\left(\mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.\) volatile a t v\(\left.]\right)\)
by simp
show ?thesis
proof (cases filter is-Write sb \(\mathrm{sb}_{\mathrm{h}}=[]\) )
case False
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}}{ } \mathrm{tts}_{\mathrm{h}}{ }^{\prime}\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb sb-empty False
apply (auto simp add: Let-def nth-list-update)
done
with step show ?thesis
by (auto simp del: fun-upd-apply simp add: \(\mathcal{S}^{\prime} \mathrm{m}^{\prime} \mathrm{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathcal{D}^{\prime} \mathrm{sb}^{\prime} \mathcal{R}^{\prime}\) )
next
case True
with sb-empty have empty: \(\mathrm{sb}_{\mathrm{h}}=[]\) by simp
from i -bound \({ }^{\prime}\) have \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathrm{i}=? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}\)
by auto
from sbh-computation.StoreBuffer [OF - this, simplified empty, simplified, OF -flush-step.Read \({ }_{s b}\), of \(m \mathcal{S}_{\mathrm{h}}\) ] i-bound \({ }^{\prime}\)
have \(\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
\(\Rightarrow_{\mathrm{sbh}} \quad\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}),[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
by (simp add: empty list-update-overwrite)
with step have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\)
\[
\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}),[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)
\]
by force
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}), \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{v}),[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
```

using sim-loc i-bound i-bound' sb empty
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp del: fun-upd-apply simp add: $\mathcal{S}^{\prime} \mathrm{m}^{\prime} \mathrm{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathcal{D}^{\prime} \mathrm{sb}^{\prime} \mathcal{R}^{\prime}$ )
qed
next
case (SBReadUnbuffered a volatile t)
then obtain
is: is $=$ Read volatile a $\mathrm{t} \# \mathrm{is}^{\prime}$ and
$\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}$ and
$\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}$ and
$\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}$ and
$\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}$ and
$\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}$ and
$\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}$ a) and
$\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}$ and
buf: buffered-val sb a = None
by auto

```
from sim-loc [rule-format, OF i-bound] ts-i is obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}, \operatorname{Read}\right.\) volatile a \(\left.\mathrm{t} \# \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
\(\mathrm{sb}: \mathrm{sb}=\) filter is- Write \(_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}\) and
sb-empty: filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}=[]\)
by (auto simp add: Let-def)
from buf
have buf': buffered-val \(\mathrm{sb}_{\mathrm{h}} \mathrm{a}=\) None
by (simp add: bufferd-val-filter-is-Write sb -eq \(s b\) )
let \(? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}=\left(\mathrm{p}\right.\), is \(^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{ma}), \mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\) volatile a \(\left.\left.\mathrm{t}(\mathrm{ma})\right], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\)
let \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}\right]\)
from sbh-memop-step.SBHReadUnbuffered [OF buf']
have (Read volatile a t \(\#\) is \(\left.^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathrm{m}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{sbh}}\)
(is \({ }^{\prime}, \vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a})), \mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\) volatile a t (ma)\(\left.], \mathrm{m}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right)\).
from sbh-computation.Memop [OF i-bound \({ }^{\prime}\) tsh \(_{\mathrm{h}}\)-i this]
have step: \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}\)
\(\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\).
moreover
from sb have \(s b: s b=\) filter is-Write \({ }_{s b}\left(\mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Read}_{\mathrm{sb}}\right.\right.\) volatile a \(\left.\left.\mathrm{t}(\mathrm{m} \operatorname{a})\right]\right)\)
by simp
show ?thesis
proof (cases filter is-Write \(\left.{ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}=[]\right)\)
case False
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m}\right.\right.\) a), \(\left.\mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\right] \sim_{\mathrm{h}}{ } \mathrm{tts}_{\mathrm{h}}{ }^{\prime}\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb sb-empty False
apply (auto simp add: Let-def nth-list-update)
done
with step show ?thesis
by (auto simp del: fun-upd-apply simp add: \(\mathcal{S}^{\prime} \mathrm{m}^{\prime} \mathrm{ts}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{D}^{\prime} \vartheta^{\prime} \mathrm{sb}^{\prime}\) )
next
case True
with sb-empty have empty: \(\mathrm{sb}_{\mathrm{h}}=[]\) by simp
from i-bound \({ }^{\prime}\) have \(? \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=? \mathrm{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}\)
by auto
from sbh-computation.StoreBuffer [OF - this, simplified empty, simplified, OF -
flush-step.Read \({ }_{\text {sb }}\), of \(\mathrm{m} \mathcal{S}_{\mathrm{h}}\) ] i-bound \({ }^{\prime}\)
have (? \(\left.\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
\(\Rightarrow_{\mathrm{sbh}} \quad\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a})),[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
by (simp add: empty)
with step have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow\) sbh \(^{*}\)
\(\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \(\left.\left.\left.^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
by force
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}), \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta\left(\mathrm{t} \mapsto \mathrm{m}\right.\right.\right.\) a), []\(, \mathcal{D}_{\mathrm{h}}\), \(\left.\left.\mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb empty
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp del: fun-upd-apply simp add: \(\mathcal{S}^{\prime} \mathrm{m}^{\prime} \operatorname{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathcal{D}^{\prime} \mathrm{sb}^{\prime} \mathcal{R}^{\prime}\) )
qed
next
case (SBWriteNonVolatile a D f A L R W)
then obtain
is: is \(=\) Write False a \((D, f)\) A L R W\#is \({ }^{\prime}\) and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(m^{\prime}: m^{\prime}=m\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\right.\) Write \(_{\text {sb }}\) False a (D, f) (f \(\vartheta\) ) A L R W]
by auto
from sim-loc [rule-format, OF i-bound] ts-i
obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}\right.\), Write False a \((\mathrm{D}, \mathrm{f})\) A L R W\#is \(\left.{ }^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
sb: \(s b=\) filter is-Write \({ }_{s b} s b_{h}\)
by (auto simp add: Let-def is)
from sbh-memop-step.SBHWriteNonVolatile
have (Write False a (D, f) A L R W\# is \(\left.{ }^{\prime}, \vartheta, \operatorname{sb}_{\mathrm{h}}, \mathrm{m}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{sbh}}\) (is \({ }^{\prime}, \vartheta\), sb \(_{\mathrm{h}} @\left[\mathrm{Write}_{\text {sb }}\right.\) False a (D, f) (f \(\vartheta\) ) A L R W], m, \(\mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\) ).
from sbh-computation.Memop [OF i-bound' tsh \(_{h}\)-i this]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}\) \(\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \(^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}} @\left[\mathrm{Write}_{\mathrm{sb}}\right.\) False a \((\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta)\) A L R W], \(\left.\left.\mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\), \(\left.\mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\).
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}\right.\right.\) @ \(\left[\right.\) Write \(_{\text {sb }}\) False a (D,f) (f \(\left.\vartheta\right)\) A L R W], \(\left.\left.\mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{h}\) \(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}} @\left[\mathrm{Write}_{\mathrm{sb}}\right.\right.\right.\) False a (D,f) (f \(\left.\vartheta\right)\) A L R W], \(\left.\left.\mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp add: \(\mathcal{S}^{\prime} \mathrm{m}^{\prime} \vartheta^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{D}^{\prime} \mathrm{ts}^{\prime} \mathrm{sb}\) )
next
case (SBWriteVolatile a D f A L R W)
then obtain
is: is \(=\) Write True a (D, f) A L R W\#is' and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb} @\left[\right.\) Write \(_{\text {sb }} \operatorname{True}\) a \((\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta)\) A L R W]
by auto
from sim-loc [rule-format, OF i-bound] ts-i is
obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}\right.\), Write \(\operatorname{True}\) a \(\left.(\mathrm{D}, \mathrm{f}) \mathrm{A} L \mathrm{R} \mathrm{W} \# \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
\(\mathrm{sb}: \mathrm{sb}=\) filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}\)
by (auto simp add: Let-def)
from sbh-computation.Memop [OF i-bound \({ }^{\prime} \mathrm{ts}_{\mathrm{h}}-\mathrm{i}\) SBHWriteVolatile
]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}\) \(\left(\operatorname{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}} @\left[\right.\right.\right.\right.\) Write \(_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta)\) A L R W], True, \(\left.\left.\mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\), \(\left.\mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\).
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta\right.\right.\), sb @ \(\left[\mathrm{Write}_{\text {sb }} \operatorname{True}\right.\) a \((\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta)\) A L R W], \(\left.\left.\mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}}\) \(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}} @\left[\mathrm{Write}_{\mathrm{sb}} \operatorname{True} \mathrm{a}(\mathrm{D}, \mathrm{f})(\mathrm{f} \vartheta)\right.\right.\right.\) A L R W],True, \(\left.\left.\mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\right]\)
```

apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb
apply (auto simp add: Let-def nth-list-update)
done

```
    ultimately show ?thesis
by (auto simp add: \(\operatorname{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathrm{m}^{\prime} \mathrm{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}\) )
    next
    case SBFence
    then obtain
is: is \(=\) Fence \#is \({ }^{\prime}\) and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
    \(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
\(\mathrm{sb}: \mathrm{sb}=[]\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=[]\)
by auto
from sim-loc [rule-format, OF i-bound] ts-i sb is obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}\right.\), Fence \(\#\) is \(\left.{ }^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
sb: [] = filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}\)
by (auto simp add: Let-def)
from filter-is-Write \({ }_{\text {sb }}\)-empty [OF sb [symmetric]]

from flush-reads-program [OF this] obtain \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathcal{R}_{\mathrm{h}}{ }^{\prime}\)
where flsh: \(\left(\mathrm{m}, \mathrm{sb}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m},[], \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) by blast
let \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\), Fence \(\#\) is \(\left.\left.^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\right]\)
from sbh-computation.store-buffer-steps [OF flsh i-bound' ts \(_{h}\)-i]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}^{*}}{ }^{*}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\).
also
from i-bound \({ }^{\prime}\) have i-bound \({ }^{\prime \prime}:\) i \(<\) length ?tsh \({ }^{\prime}\)
by auto
from i-bound \({ }^{\prime}\) have \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathrm{i}: ~ ? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}!\mathrm{i}=\left(\mathrm{p}\right.\), Fence \(\#\) is \(\left.{ }^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\)
by simp
from sbh-computation.Memop [OF i-bound \({ }^{\prime \prime}\) ts \(_{\mathrm{h}}{ }^{\prime}\)-i SBHFence] i-bound \({ }^{\prime}\)
have \(\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \({ }^{\prime}, \vartheta,[]\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime}\), Map.empty \(\left.\left.)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) by (simp)
finally


\section*{moreover}
have \(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta,[], \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta,[]\right.\right.\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime}\), Map.empty \(\left.)\right]\) apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp add: \(\operatorname{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathrm{m}^{\prime} \mathrm{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{S}^{\prime} \mathcal{R}^{\prime}\) )
next
case (SBRMWReadOnly cond t a D f ret A L R W)
then obtain
is: is \(=\) RMW a t \((D, f)\) cond ret A L R W\#is \({ }^{\prime}\) and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \mathrm{m}\) a) and
\(\mathrm{sb}: \mathrm{sb}=[]\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=[]\) and
cond: \(\neg \operatorname{cond}(\vartheta(\mathrm{t} \mapsto \mathrm{ma}))\)
by auto
from sim-loc [rule-format, OF i-bound] ts-i sb is obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}, \mathrm{RMW}\right.\) at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W\# is \(\left.{ }^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
sb: [] = filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}\)
by (auto simp add: Let-def)
from filter-is-Write sb-empty \(^{\text {[OF sb }}\) [symmetric]]
have \(\forall r \in\) set \(^{\text {sb }} \mathrm{h}\). is- \(\operatorname{Read}_{\text {sb }} \mathrm{r} V\) is- \(\operatorname{Prog}_{\text {sb }} \mathrm{r} V\) is-Ghost \({ }_{\text {sb }} r\).
from flush-reads-program [OF this] obtain \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathcal{R}_{\mathrm{h}}{ }^{\prime}\)
where flsh: \(\left(\mathrm{m}, \mathrm{sb}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m},[], \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) by blast
let \({ }^{2} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{tsh}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{RMW}\right.\right.\) at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W\# is \(\left.\left.{ }^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\right]\)
from sbh-computation.store-buffer-steps [OF flsh i-bound \({ }^{\prime} \mathrm{ts}_{\mathrm{h}}-\mathrm{i}\) ]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}^{*}}{ }^{*}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\).
also
from i-bound \({ }^{\prime}\) have i-bound \({ }^{\prime \prime}:\) i \(<\) length ?tsh \({ }^{\prime}\)
by auto
from i-bound \({ }^{\prime}\) have \(\mathrm{tsh}^{\prime}{ }^{-} \mathrm{i}\) : \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}!\mathrm{i}=(\mathrm{p}, \mathrm{RMW}\) a \(\mathrm{t}(\mathrm{D}, \mathrm{f})\) cond ret \(\mathrm{A} L \mathrm{R}\) W\#is \(\left.{ }^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\)
by \(\operatorname{simp}\)
note step \(=\) SBHRMWReadOnly \([\) where cond \(=\) cond and \(\vartheta=\vartheta\) and \(m=m\), OF cond ]
from sbh-computation.Memop [OF i-bound \({ }^{\prime \prime} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)-i step ] i-bound \({ }^{\prime}\)
have \(\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{t}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \({ }^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),[]\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime}\),Map.empty \(\left.)\right], \mathrm{m}\), \(\mathcal{S}_{\mathrm{h}}{ }^{\prime}\) )
by (simp)
finally
have \(\left(\mathrm{t}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}^{*}}{ }^{*}\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \({ }^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),[]\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime}\), Map.empty \(\left.\left.)\right], \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\).
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m}\right.\right.\) a) \(\left.,[], \mathcal{D}, \mathcal{O}, \mathcal{R})\right] \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}),[]\right.\right.\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime}\),Map.empty)]
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp add: \(\operatorname{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathrm{m}^{\prime} \mathrm{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{S}^{\prime} \mathcal{R}^{\prime}\) )
next
case (SBRMWWrite cond t a D f ret A L R W)
then obtain
is: is = RMW a t ( \(\mathrm{D}, \mathrm{f}\) ) cond ret A L R W\#is' and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a}))))\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} \mathrm{a})(\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a})))))\) and
\(\mathrm{sb}: \mathrm{sb}=[]\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=[]\) and
cond: cond \((\vartheta(\mathrm{t} \mapsto \mathrm{m} \mathrm{a}))\)
by auto
from sim-loc [rule-format, OF i-bound] ts-i sb is obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) acq \(_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}, \mathrm{RMW}\right.\) at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W\# is \(\left.{ }^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
sb: [] = filter is-Write \({ }_{\text {sb }} \mathrm{sb}_{\mathrm{h}}\)
by (auto simp add: Let-def)
from filter-is-Write sb \(_{\text {-empty }}\) [OF sb [symmetric] ]
have \(\forall r \in\) set \(^{\text {sb }}{ }_{h}\). is- \(\operatorname{Read}_{s b} r \vee\) is- \(\operatorname{Prog}_{s b} r \vee\) is-Ghost \({ }_{\text {sb }} r\).
from flush-reads-program [OF this] obtain \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathcal{R}_{\mathrm{h}}{ }^{\prime}\)
where flsh: \(\left(\mathrm{m}, \mathrm{sb}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow_{\mathrm{f}}{ }^{*}\left(\mathrm{~m},[], \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) by blast
let \(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{RMW}\right.\right.\) at \((\mathrm{D}, \mathrm{f})\) cond ret A L R W\# is \(\left.\left.{ }^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\right]\)
from sbh-computation.store-buffer-steps [OF flsh i-bound' \(\mathrm{ts}_{\mathrm{h}}\) - i ]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}^{*}}{ }^{*}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\).
also
from i-bound \({ }^{\prime}\) have i-bound \({ }^{\prime \prime}:\) i \(<\) length ? \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
by auto
from i-bound \({ }^{\prime}\) have \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}-\mathrm{i}\) : ? \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}\) !i \(=(\mathrm{p}, \mathrm{RMW}\) a \(\mathrm{t}(\mathrm{D}, \mathrm{f})\) cond ret \(\mathrm{A} L \mathrm{R}\) \(\mathrm{W} \#\) is \(\left.{ }^{\prime}, \vartheta,[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\)
by simp
note step \(=\) SBHRMWWrite [where cond \(=\) cond and \(\vartheta=\vartheta\) and \(\mathrm{m}=\mathrm{m}\), OF cond]
from sbh-computation.Memop [OF i-bound \({ }^{\prime \prime}\) ts \({ }^{\prime}\)-i step ] i-bound \({ }^{\prime}\)
have \(\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \({ }^{\prime}\),
\(\vartheta\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m}\right.\) a) \((\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a}))))),[]\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \cup \mathrm{A}-\mathrm{R}\), Map.empty \(\left.)\right]\),
\(\left.\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{ma})))), \mathcal{S}_{\mathrm{h}}{ }^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
by ( \(\operatorname{simp}\) )
finally
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \(^{\prime}\),
\(\vartheta\left(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m}\right.\) a) \((\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a})))))\), [], False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \cup \mathrm{A}-\mathrm{R}\), Map.empty \(\left.)\right]\), \(\left.\mathrm{m}(\mathrm{a}:=\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{ma})))), \mathcal{S}_{\mathrm{h}}{ }^{\prime} \oplus_{\mathrm{W}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\).
moreover
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m}\right.\right.\) a) \(\left.(\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{m} \mathrm{a}))))),[], \mathcal{D}, \mathcal{O}, \mathcal{R})\right] \sim_{\mathrm{h}}\)
\(\operatorname{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta(\mathrm{t} \mapsto \operatorname{ret}(\mathrm{m} a)(\mathrm{f}(\vartheta(\mathrm{t} \mapsto(\mathrm{ma}))))),[]\right.\right.\), False, \(\mathcal{O}_{\mathrm{h}}{ }^{\prime} \cup \mathrm{A}-\mathrm{R}\), Map.empty \(\left.)\right]\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp add: \(\operatorname{ts}^{\prime} \mathcal{O}^{\prime} \vartheta^{\prime} \mathrm{m}^{\prime} \mathrm{sb}^{\prime} \mathcal{D}^{\prime} \mathcal{S}^{\prime} \mathcal{R}^{\prime}\) )
next
case (SBGhost A L R W)
then obtain
is: is \(=\) Ghost A L R W\#is' and
\(\mathcal{O}^{\prime}: \mathcal{O}^{\prime}=\mathcal{O}\) and
\(\mathcal{R}^{\prime}: \mathcal{R}^{\prime}=\mathcal{R}\) and
\(\mathcal{S}^{\prime}: \mathcal{S}^{\prime}=\mathcal{S}\) and
\(\mathcal{D}^{\prime}: \mathcal{D}^{\prime}=\mathcal{D}\) and
\(\mathrm{m}^{\prime}: \mathrm{m}^{\prime}=\mathrm{m}\) and
\(\vartheta^{\prime}: \vartheta^{\prime}=\vartheta\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\mathrm{sb}\)
by auto
from sim-loc [rule-format, OF i-bound] ts-i is
obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) where
\(\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}\right.\), Ghost A L R W\# is \(\left., \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
\(\mathrm{sb}: \mathrm{sb}=\) filter is- \(\mathrm{Write}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}\) and
sb-empty: filter is-Write \({ }_{s b} \mathrm{sb}_{\mathrm{h}}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}=[]\)
by (auto simp add: Let-def)
let \({ }^{2} \mathrm{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}=\left(\mathrm{p}\right.\), is \({ }^{\prime}, \boldsymbol{\vartheta}, \mathrm{sb}_{\mathrm{h}} @\left[\right.\) Ghost \(_{\mathrm{sb}}\) A L R W], \(\left.\mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\)
let \({ }^{2} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\right.\) ? \(\left.\mathrm{ts}_{\mathrm{h}}-\mathrm{i}\right]\)
note step \(=\) SBHGhost
from sbh-computation.Memop [OF i-bound' ts \(_{h}\)-i step ] i-bound \({ }^{\prime}\)
have step: \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)\)
by (simp)
from sb have sb: \(\mathrm{sb}=\) filter is-Write \({ }_{\mathrm{sb}}\left(\mathrm{sb}_{\mathrm{h}} @\left[\right.\right.\) Ghost \(_{\mathrm{sb}} \mathrm{A} L\) R W] \()\)
by simp
show ?thesis
proof (cases filter is-Write \(\left.{ }_{\mathbf{s b}} \mathrm{sb}_{\mathrm{h}}=[]\right)\)
case False
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right] \sim_{\mathrm{h}}{ }{ }^{2} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb sb-empty False
apply (auto simp add: Let-def nth-list-update)
done
with step show ?thesis
by (auto simp del: fun-upd-apply simp add: \(\mathcal{S}^{\prime} \mathrm{m}^{\prime} \operatorname{ts}^{\prime} \mathcal{O}^{\prime} \mathcal{D}^{\prime} \boldsymbol{v}^{\prime} \mathrm{sb}^{\prime} \mathcal{R}^{\prime}\) )
next
case True
with sb-empty have empty: \(\mathrm{sb}_{\mathrm{h}}=[]\) by simp
from i -bound \({ }^{\prime}\) have \(? \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=? \mathrm{ts}_{\mathrm{h}} \mathrm{i}^{\mathrm{i}}{ }^{\prime}\)
by auto
from sbh-computation.StoreBuffer [OF - this, simplified empty, simplified, OF -
flush-step. Ghost, of \(m \mathcal{S}_{\mathrm{h}}\) ] i-bound \({ }^{\prime}\)
have (? \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\) )
\(\Rightarrow_{\text {sbh }}\left(\operatorname{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is \({ }^{\prime}, \boldsymbol{\vartheta},[], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}} \cup \mathrm{A}-\mathrm{R}\), augment-rels \(\left.\left.\left(\operatorname{dom} \mathcal{S}_{\mathrm{h}}\right) \mathrm{R} \mathcal{R}_{\mathrm{h}}\right)\right]\),
\(\left.\mathrm{m}, \mathcal{S}_{\mathrm{h}} \oplus_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)\)
```

        by (simp add: empty)
    with step have ( }\mp@subsup{\textrm{ts}}{\textrm{h}}{},\textrm{m},\mp@subsup{\mathcal{S}}{\textrm{h}}{})=>\mp@subsup{=>}{\textrm{sbh}}{}\mp@subsup{}{}{*
        (tsh}[\textrm{i}:=(\textrm{p},\mathrm{ is', Э, [], 疎,}\mp@subsup{\mathcal{O}}{\textrm{h}}{}\cup\textrm{A}-\textrm{R},\mathrm{ augment-rels (dom }\mp@subsup{\mathcal{S}}{\textrm{h}}{})\textrm{R}\mp@subsup{\mathcal{R}}{\textrm{h}}{})],\textrm{m},\mp@subsup{\mathcal{S}}{\textrm{h}}{
    \oplusW R }\mp@subsup{\ominus}{\textrm{A}}{\prime}\textrm{L}
by force
moreover
have ts [i := (p,is',\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})] \mp@subsup{~}{h}{}
tsh}[\textrm{i}:=(\textrm{p},\mp@subsup{\textrm{is}}{}{\prime},\vartheta,\mp@code{\vartheta},[],\mp@subsup{\mathcal{D}}{\textrm{h}}{},\mp@subsup{\mathcal{O}}{\textrm{h}}{}\cup\textrm{A}-\textrm{R},\mathrm{ augment-rels (dom }\mp@subsup{\mathcal{S}}{\textrm{h}}{})\textrm{R}\mp@subsup{\mathcal{R}}{\textrm{h}}{})
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb empty
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis

```

```

    qed
    qed
    next
case (Program i - p is \vartheta sb \mathcal{D}\mathcal{O}\mathcal{R}\mp@subsup{\textrm{p}}{}{\prime}\mathrm{ is')}
then obtain
ts': ts' = ts[i := (p', is@is',},\boldsymbol{\vartheta},\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})]\mathrm{ and
i-bound: i < length ts and
ts-i: ts ! i = (p, is, \vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\mathrm{ and}
prog-step: \vartheta\vdash p }\mp@subsup{->}{\textrm{p}}{}(\mp@subsup{\textrm{p}}{}{\prime},\mathrm{ is') and
\mathcal{S}
m': m'=m
by auto

```

\section*{from sim obtain}
```

lts-eq: length ts $=$ length $\mathrm{ts}_{\mathrm{h}}$ and
sim-loc: $\forall \mathrm{i}<$ length ts. $\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.$.
let $(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{h}}!\mathrm{i}$ in ts $!\mathrm{i}=\left(\mathrm{p}\right.$, is, $\vartheta$, filter is-Write $\left.\mathrm{sb}_{\mathrm{sb}} \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right) \wedge$ (filter is- Write $\left._{\mathrm{sb}} \mathrm{sb}=[] \longrightarrow \mathrm{sb}=[]\right)$ )
by cases auto
from sim-loc [rule-format, OF i-bound] ts-i
obtain $\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}$ acq $_{\mathrm{h}}$ where
$\mathrm{ts}_{\mathrm{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)$ and
$\mathrm{sb}: \mathrm{sb}=$ filter is- $\mathrm{Write}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}$ and
sb-empty: filter is-Write ${ }_{s b} \mathrm{sb}_{\mathrm{h}}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}=[]$
by (auto simp add: Let-def)
from lts-eq i-bound have i-bound': i < length tsh
by simp
let ${ }^{2} \operatorname{ts}_{\mathrm{h}}-\mathrm{i}^{\prime}=\left(\mathrm{p}^{\prime}\right.$, is @ is ${ }^{\prime}, \vartheta, \mathrm{sb}_{\mathrm{h}} @\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $\left.], \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)$
let ${ }^{2} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}=\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\right.$ ? $\left.\mathrm{ts}_{\mathrm{h}}-\mathrm{i}\right]$
from sbh-computation.Program [OF i-bound ${ }^{\prime} \mathrm{ts}_{\mathrm{h}}$ - i prog-step]
have step: $\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\text {sbh }}\left(? \mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right)$.

```
```

show ?thesis
proof (cases filter is-Write sb sb
case False
have ts[i := ( }\mp@subsup{\textrm{p}}{}{\prime},\mathrm{ is@is }\mp@subsup{}{}{\prime},\vartheta,\textrm{sb},\mathcal{D},\mathcal{O},\mathcal{R})]~\mp@subsup{~}{h}{}\mp@subsup{}{}{\prime}\mp@subsup{\textrm{ts}}{\textrm{h}}{}\mp@subsup{}{}{\prime
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb False sb-empty
apply (auto simp add: Let-def nth-list-update)
done
with step show ?thesis
by (auto simp add: ts' }\mp@subsup{\mathcal{S}}{}{\prime}\textrm{m}
next
case True
with sb-empty have empty: }\mp@subsup{\textrm{sb}}{\textrm{h}}{}=[] by sim
from i-bound' have ? ts}\mp@subsup{\textrm{h}}{\textrm{h}}{\prime}!\textrm{i}=?\mp@subsup{\textrm{ts}}{\textrm{h}}{}-\mp@subsup{\textrm{i}}{}{\prime
by auto
from sbh-computation.StoreBuffer [OF - this, simplified empty, simplified, OF -
flush-step.Prog}\mp@subsup{}{sb}{},\mathrm{ of m }\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mathrm{ ] i-bound'
have (?ts}\mp@subsup{}{\textrm{h}}{}\mp@subsup{}{}{\prime},\textrm{m},\mp@subsup{\mathcal{S}}{\textrm{h}}{}
\#
by (simp add: empty)
with step have ( }\mp@subsup{\textrm{ts}}{\textrm{h}}{},\textrm{m},\mp@subsup{\mathcal{S}}{\textrm{h}}{})=>\mp@subsup{\#}{\textrm{sbh}}{}\mp@subsup{}{}{*

```

```

        by force
    moreover
    ```

```

        apply (rule sim-history-config.intros)
    using lts-eq
apply simp
using sim-loc i-bound i-bound' sb empty
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp del: fun-upd-apply simp add: 疎 m' ts')
qed
next
case (StoreBuffer i - p is \vartheta sb \mathcal{D}\mathcal{O}\mathcal{R}---\mp@subsup{\textrm{sb}}{}{\prime}\mp@subsup{\mathcal{O}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime})
then obtain
ts': ts '}=\textrm{ts}[\textrm{i}:=(\textrm{p},\textrm{is},\vartheta,\mp@subsup{\textrm{sb}}{}{\prime},\mathcal{D},\mp@subsup{\mathcal{O}}{}{\prime},\mathcal{R}\mp@subsup{}{}{\prime})]\mathrm{ and
i-bound: i < length ts and
ts-i: ts ! i = (p, is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})\mathrm{ and}
sb-step: (m,sb,\mathcal{O},\mathcal{R},\mathcal{S})}\mp@subsup{->}{\textrm{w}}{(}(\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\textrm{sb}}{}{\prime},\mp@subsup{\mathcal{O}}{}{\prime},\mp@subsup{\mathcal{R}}{}{\prime},\mp@subsup{\mathcal{S}}{}{\prime}
by auto
from sim obtain
lts-eq: length ts = length tsh

```
sim-loc: \(\forall \mathrm{i}<\) length ts. \(\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.\).
let \((\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{h}}!\mathrm{i}\) in ts!i=(p,is, \(\vartheta\), filter is-Write \(\left.{ }_{\text {sb }} \mathrm{sb}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}\right) \wedge\) (filter is-Write \({ }_{\mathrm{sb}} \mathrm{sb}=[] \longrightarrow \mathrm{sb}=[]\) ))
by cases auto
from sim-loc [rule-format, OF i-bound] ts-i
obtain \(\operatorname{sb}_{\mathrm{h}} \mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{D}_{\mathrm{h}}\) acq \(_{\mathrm{h}}\) where
\(t_{s_{h}}-\mathrm{i}: \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}_{\mathrm{h}}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}\right)\) and
\(\mathrm{sb}: \mathrm{sb}=\) filter is- \(\mathrm{Write}_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}\) and
sb-empty: filter is-Write \(\mathrm{sb} \mathrm{sb}_{\mathrm{h}}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}=[]\)
by (auto simp add: Let-def)
from lts-eq i-bound have i-bound': i < length tsh
by simp
from flush-simulates-filter-writes [OF sb-step sb, of \(\mathcal{O}_{\mathrm{h}} \mathcal{R}_{\mathrm{h}} \mathcal{S}_{\mathrm{h}}\) ]
obtain \(\mathrm{sb}_{\mathrm{h}}{ }^{\prime} \mathcal{O}_{\mathrm{h}}{ }^{\prime} \mathcal{R}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime}\)
where flush': \(\left(\mathrm{m}, \mathrm{sb}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}, \mathcal{R}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \rightarrow \rightarrow^{*}\left(\mathrm{~m}^{\prime}, \mathrm{sb}_{\mathrm{h}}{ }^{\prime}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) and
\(\mathrm{sb}^{\prime}: \mathrm{sb}^{\prime}=\) filter is-Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}{ }^{\prime}\) and
sb'-empty: filter is- Write \({ }_{\mathrm{sb}} \mathrm{sb}_{\mathrm{h}}{ }^{\prime}=[] \longrightarrow \mathrm{sb}_{\mathrm{h}}{ }^{\prime}=[]\)
by auto
from sb-step obtain volatile a sop v A L R W where sb=Write sb \(_{\text {sb }}\) volatile a sop v A L R W\#sb \({ }^{\prime}\)
by cases force
from sbh-computation.store-buffer-steps [OF flush' i -bound \({ }^{\prime} \mathrm{ts}_{\mathrm{h}} \mathrm{i}-\mathrm{i}\) ]
have \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow_{\mathrm{sbh}^{*}}\left(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.\), is, \(\left.\left.\left.\vartheta, \mathrm{sb}_{\mathrm{h}}{ }^{\prime}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\right], \mathrm{m}^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\).

\section*{moreover}
have ts \(\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\), is, \(\left.\left.\boldsymbol{\vartheta}, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right] \sim_{\mathrm{h}}\)
\(\mathrm{ts}_{\mathrm{h}}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\), is, \(\left.\left.\vartheta, \mathrm{sb}_{\mathrm{h}}{ }^{\prime}, \mathcal{D}_{\mathrm{h}}, \mathcal{O}_{\mathrm{h}}{ }^{\prime}, \mathcal{R}_{\mathrm{h}}{ }^{\prime}\right)\right]\)
apply (rule sim-history-config.intros)
using lts-eq
apply simp
using sim-loc i-bound i-bound' sb sb' sb'-empty
apply (auto simp add: Let-def nth-list-update)
done
ultimately show ?thesis
by (auto simp add: ts \({ }^{\prime}\) )
qed
qed
theorem (in valid-program) concurrent-history-steps-simulates-store-buffer-steps:
assumes step-sb: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{sb}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows \(\wedge \mathrm{ts}_{\mathrm{h}} \mathcal{S}_{\mathrm{h}} . \mathrm{ts}_{\sim_{\mathrm{h}}} \mathrm{ts}_{\mathrm{h}} \Longrightarrow \exists \mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} .\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}_{\mathrm{h}}\right) \Rightarrow \mathrm{sbh}^{*}\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}_{\mathrm{h}}\right) \wedge \mathrm{ts}^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
using step-sb
proof (induct rule: converse-rtranclp-induct-sbh-steps)
```

    case refl thus ?case by auto
    next
case (step ts m S ts" m" ' S')
have first: (ts,m,\mathcal{S})=>\mp@subsup{=>}{\textrm{sb}}{}(\mp@subsup{\textrm{ts}}{}{\prime\prime},\mp@subsup{\textrm{m}}{}{\prime\prime},\mp@subsup{\mathcal{S}}{}{\prime\prime})\mathrm{ by fact}
have sim: ts }\mp@subsup{~}{h}{}\mp@subsup{t}{h}{h}\mathrm{ by fact

```

```

    obtain tsh}\mp@subsup{}{}{\prime\prime}\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}\mathrm{ where
        exec: ( }\mp@subsup{\textrm{ts}}{\textrm{h}}{\prime\prime},\textrm{m},\mp@subsup{\mathcal{S}}{\textrm{h}}{})=>\mp@subsup{=>}{\mp@subsup{\textrm{sbh}}{}{*}}{*}(\mp@subsup{\textrm{ts}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\textrm{m}}{}{\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{\prime\prime})\mathrm{ and sim: ts " }\mp@subsup{~}{\textrm{h}}{}\mp@subsup{\textrm{ts}}{\textrm{h}}{\prime\prime
        by auto
    from step.hyps (3) [OF sim, of }\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime}
    obtain }\mp@subsup{\textrm{ts}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime}\mathrm{ where exec-rest: ( }\mp@subsup{\textrm{ts}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime},\mp@subsup{\textrm{m}}{}{\prime\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{}\mp@subsup{}{}{\prime\prime})=\mp@subsup{|}{\mathrm{ sbh }}{}\mp@subsup{}{}{*}(\mp@subsup{\textrm{ts}}{\textrm{h}}{}\mp@subsup{}{}{\prime},\mp@subsup{\textrm{m}}{}{\prime},\mp@subsup{\mathcal{S}}{\textrm{h}}{\prime})\mathrm{ and sim
    tsh'
by auto
note exec also note exec-rest
finally show ?case
using sim' by blast
qed

```
theorem (in xvalid-program-progress) concurrent-direct-execution-simulates-store-buffer-execution:
assumes exec-sb: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathrm{x}\right) \Rightarrow_{\mathrm{sb}^{*}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathrm{x}^{\prime}\right)\)
assumes init: initial \({ }_{\text {sb }} \mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\text {sb }}\)
assumes valid: valid \(\mathrm{ts}_{\mathrm{sb}}\)
assumes sim: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
assumes safe: safe-reach-direct safe-free-flowing (ts,m,S)
shows \(\exists \mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime}\).
\[
\begin{aligned}
& \left(\mathrm{tts}_{\mathrm{sb},}, \mathrm{~m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow{ }_{\mathrm{sbb}}{ }^{*}\left(\mathrm{tss}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}\right) \wedge \\
& \mathrm{ts}_{\mathrm{sb}}^{\prime} \sim_{\mathrm{h}} \mathrm{tsth}^{\prime} \wedge \\
& (\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow \text { d }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge \\
& \left(\mathrm{tsh}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)
\end{aligned}
\]
proof -
from init interpret ini: initial \({ }_{\mathrm{sb}} \mathrm{t}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}\).
from concurrent-history-steps-simulates-store-buffer-steps [OF exec-sb ini.history-refl, of \(\mathcal{S}_{\mathrm{sb}}\) ]
obtain \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime}\) where
sbh: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{tsh}^{\prime}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) and
sim-sbh: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
by auto
from concurrent-direct-execution-simulates-store-buffer-history-execution [OF sbh init valid sim safe]
obtain \(\mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime}\) where
\(\mathrm{d}:(\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}^{*}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\) and
d-sim: \(\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
by auto
with sbh sim-sbh show ?thesis by blast qed
inductive sim-direct-config::
('p,'p store-buffer, 'dirty, 'owns,'rels) thread-config list \(\Rightarrow\) ('p,unit,bool,'owns','rels') thread-config list \(\Rightarrow\) bool
\[
\left(-\sim_{d}-[60,60] 100\right)
\]

\section*{where}

【length ts \(=\) length \(\mathrm{ts}_{\mathrm{d}} ;\)
\(\forall \mathrm{i}<\) length ts. \(\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.\). let \((\mathrm{p}\), is, \(\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}_{\mathrm{d}}!\) i in ts \(\left.!=\left(\mathrm{p}, \mathrm{is}, \vartheta,[], \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right)\)
】
\[
\Longrightarrow
\]
\[
\mathrm{ts} \sim_{\mathrm{d}} \mathrm{ts}_{\mathrm{d}}
\]
lemma empty-sb-sims:
assumes empty:
\(\forall \mathrm{i}\) p is xs \(\mathrm{sb} \mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}} \longrightarrow \mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]\)
assumes sim-h: \(\mathrm{ts}_{\mathrm{sb}} \sim_{h} \mathrm{ts}_{\mathrm{h}}\)
assumes sim-d: \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}_{\mathrm{h}}, \mathcal{S}_{\mathrm{h}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
shows \(\mathrm{ts}_{\mathrm{sb}} \sim_{\mathrm{d}} \mathrm{ts} \wedge \mathrm{m}_{\mathrm{h}}=\mathrm{m} \wedge\) length \(\mathrm{ts}_{\mathrm{sb}}=\) length ts
proof-
from sim-h empty
have empty':
\(\forall \mathrm{i} \mathrm{p}\) is xs sb \(\mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{h}} \longrightarrow \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]\)
apply (cases)
apply clarsimp
subgoal for i
apply (drule-tac \(x=i\) in spec)
apply (auto simp add: Let-def)
done done
from sim-h sim-config-emptyE [OF empty' sim-d]
show ?thesis
apply cases
apply clarsimp
apply (rule sim-direct-config.intros)
apply clarsimp
apply clarsimp
using empty \({ }^{\prime}\)
subgoal for i
apply (drule-tac \(x=i\) in spec)
apply (drule-tac \(x=i\) in spec)
apply (drule-tac \(x=i\) in spec)
apply (auto simp add: Let-def)
done
done
qed
lemma empty-d-sims:
assumes sim: \(\mathrm{ts}_{\mathrm{sb}} \sim_{d}\) ts
shows \(\exists \mathrm{ts}_{\mathrm{h}} . \mathrm{ts}_{\mathrm{sb}} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}} \wedge\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
```

proof -
from sim obtain
leq: length ts }\mp@subsup{\textrm{sb}}{\textrm{b}}{}=\mathrm{ length ts and
sim: }\forall\textrm{i}<<length ts tsb
(\exists\mathcal{O}}\mp@subsup{)}{}{\prime}\mp@subsup{\mathcal{D}}{}{\prime}\mp@subsup{\mathcal{R}}{}{\prime}
let (p,is, \vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R})= ts!i in
ts sb!i=(p,is,\vartheta,[],\mathcal{D}
by cases auto
define tsh where tsh }\equiv\operatorname{map}(\lambda(p,is,\vartheta,sb,\mathcal{D},\mathcal{O},\mathcal{R}).(p,is,\vartheta,[]::'a memref list,\mathcal{D},\mathcal{O},\mathcal{R})
ts
have tsmb }\mp@subsup{~}{h}{}\mp@subsup{t}{\textrm{s}}{\textrm{h}
apply (rule sim-history-config.intros)
using leq sim
apply (auto simp add: tsh
done
moreover
have empty:
\foralli p is xs sb \mathcal{O}}\mathcal{O}\mathcal{R}.\textrm{i}< length tsh \longrightarrow tsh! i=(p,is,xs,sb,\mathcal{D},\mathcal{O},\mathcal{R})\longrightarrow\textrm{sb}=[
apply (clarsimp simp add: tsh-def Let-def)
subgoal for i
apply (case-tac ts!i)
apply auto
done
done
have ( }\mp@subsup{\textrm{ts}}{\textrm{h}}{\textrm{m}},\textrm{m},\mathcal{S})~(\textrm{ts},\textrm{m},\mathcal{S}
apply (rule sim-config-emptyI [OF empty])
apply (clarsimp simp add: tsh
apply (clarsimp simp add: tsh-def Let-def)
subgoal for i
apply (case-tac ts!i)
apply auto
done
done
ultimately show ?thesis by blast
qed

```
theorem (in xvalid-program-progress) concurrent-direct-execution-simulates-store-buffer-execution-empty: assumes exec-sb: \(\left(\mathrm{tt}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathrm{x}\right) \Rightarrow \Rightarrow_{\mathrm{sb}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathrm{x}^{\prime}\right)\)
assumes init: initial \({ }_{\text {sb }} \mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}}\)
assumes valid: valid \(\mathrm{ts}_{\mathrm{sb}}\)
assumes empty:
\(\forall \mathrm{i}\) p is xs sb \(\mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \longrightarrow \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]\)
assumes sim: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
assumes safe: safe-reach-direct safe-free-flowing (ts,m,S )
shows \(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime}\).
\[
(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow_{\mathrm{d}^{*}}\left(\mathrm{ts}^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}^{\prime}\right) \wedge \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{d}} \mathrm{ts}^{\prime}
\]
proof -
from concurrent-direct-execution-simulates-store-buffer-execution [OF exec-sb init valid sim safe]
```

    obtain \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime} \mathrm{ts}^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime}\) where
        \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow{ }_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) and
        sim-h: \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\) and
        exec: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\) and
        sim: \(\left(\mathrm{tss}^{\prime}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
        by auto
    from empty-sb-sims [OF empty sim-h sim]
    obtain \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{d}} \mathrm{ts}^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime}=\mathrm{m}^{\prime}\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}=\) length \(\mathrm{ts}^{\prime}\)
        by auto
    thus ?thesis
        using exec
    by blast
    qed

```
locale \(^{\text {initial }}{ }_{d}=\) simple-ownership-distinct + read-only-unowned + unowned-shared +
fixes valid
assumes empty-is: \(\lceil i<\) length ts; ts!i \(=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\) is \(=[]\)
assumes empty-rels: \(\llbracket \mathrm{i}<\) length ts; ts! \(\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow \mathcal{R}=\) Map.empty
assumes valid-init: valid (map \((\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}, \mathrm{is}, \vartheta,[], \mathcal{D}, \mathcal{O}, \mathcal{R}))\) ts)
locale empty-store-buffers \(=\)
fixes ts::('p,'p store-buffer,bool,owns,rels) thread-config list
assumes empty-sb: \(\llbracket \mathrm{i}<\) length ts; ts \(!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow \mathrm{sb}=[]\)
lemma initial-d-sb:
    assumes init: initiald \({ }_{d}\) ts \(\mathcal{S}\) valid
    shows initial \({ }_{\mathrm{sb}}(\operatorname{map}(\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}\), is, \(\vartheta,[], \mathcal{D}, \mathcal{O}, \mathcal{R}))\) ts) \(\mathcal{S}\)
        (is initial \({ }_{\text {sb }}\) ?map \(\mathcal{S}\) )
proof -
    from init interpret ini: initial \(_{d}\) ts \(\mathcal{S}\).
    show ?thesis
    proof (intro-locales)
        show simple-ownership-distinct ?map
        apply (clarsimp simp add: simple-ownership-distinct-def)
        subgoal for i j
        apply (case-tac ts!i)
        apply (case-tac ts!j)
        apply (cut-tac \(\mathrm{i}=\mathrm{i}\) and \(\mathrm{j}=\mathrm{j}\) in ini.simple-ownership-distinct)
        apply clarsimp
        apply clarsimp
        apply clarsimp
        apply assumption
        apply assumption
        apply auto
        done
        done
    next
        show read-only-unowned \(\mathcal{S}\) ?map
```

    apply (clarsimp simp add: read-only-unowned-def)
    subgoal for i
    apply (case-tac ts!i)
    apply (cut-tac i=i in ini.read-only-unowned)
    apply clarsimp
    apply assumption
    apply auto
    done
    done
    next
show unowned-shared S ?map
apply (clarsimp simp add: unowned-shared-def')
apply (rule ini.unowned-shared ')
apply clarsimp
subgoal for a i
apply (case-tac ts!i)
apply auto
done
done
next
show initial }\mp@subsup{l}{\mathrm{ sb}}{\mathrm{ -axioms ?map}
apply (unfold-locales)
subgoal for i
apply (case-tac ts!i)
apply simp
done
subgoal for i
apply (case-tac ts!i)
apply clarsimp
apply (rule-tac i=i in ini.empty-is)
apply clarsimp
apply fastforce
done
subgoal for i
apply (case-tac ts!i)
apply clarsimp
apply (rule-tac i=i in ini.empty-rels)
apply clarsimp
apply fastforce
done
done
qed
qed

```
theorem (in xvalid-program-progress) store-buffer-execution-result-sequential-consistent:
assumes exec-sb: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}, \mathrm{x}\right) \Rightarrow_{\mathrm{sb}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathrm{x}^{\prime}\right)\)
assumes empty': empty-store-buffers \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\)
assumes sim: \(\mathrm{ts}_{\mathrm{sb}} \sim_{\mathrm{d}}\) ts
assumes init: initial \({ }_{d}\) ts \(\mathcal{S}\) valid
assumes safe: safe-reach-direct safe-free-flowing (ts,m,S \()\)
shows \(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime}\).
\[
(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}\right) \wedge \mathrm{ts}_{\mathrm{sb}}^{\prime} \sim_{\mathrm{d}} \mathrm{ts}^{\prime}
\]
proof -
from empty \({ }^{\prime}\)
have empty':
\(\forall \mathrm{i}\) p is xs sb \(\mathcal{D} \mathcal{O} \mathcal{R} . \mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \longrightarrow \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]\) by (auto simp add: empty-store-buffers-def)
define \(\mathrm{ts}_{\mathrm{h}}\) where \(\mathrm{ts}_{\mathrm{h}} \equiv \operatorname{map}(\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}, \mathrm{is}, \vartheta,[]::\) 'a memref list, \(\mathcal{D}, \mathcal{O}, \mathcal{R}))\) ts
from initial-d-sb [OF init]
have init-h:initial \({ }_{\text {sb }} \operatorname{ts}_{\mathrm{h}} \mathcal{S}\)
by (simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def)
from initial \({ }_{d} \cdot\) valid-init [OF init]
have valid-h: valid \(t_{s h}\)
by ( simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def)
from sim obtain
leq: length \(\mathrm{ts}_{\mathrm{sb}}=\) length ts and
sim: \(\forall \mathrm{i}<\) length ts \(_{\text {sb }}\).
\(\left(\exists \mathcal{O}^{\prime} \mathcal{D}^{\prime} \mathcal{R}^{\prime}\right.\).
\[
\begin{aligned}
& \text { let }(\mathrm{p}, \text { is, }, \vartheta \text {,sb }, \mathcal{D}, \mathcal{O}, \mathcal{R})=\mathrm{ts}!\mathrm{i} \text { in } \\
& \left.\mathrm{ts}_{\mathrm{sb}}!\mathrm{i}=\left(\mathrm{p}, \mathrm{is}, \vartheta,[], \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right)
\end{aligned}
\]
by cases auto
have sim-h: \(\mathrm{ts}_{\mathrm{sb}} \sim_{h} \mathrm{ts}_{\mathrm{h}}\)
apply (rule sim-history-config.intros)
using leq sim
apply (auto simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def Let-def leq)
done
from concurrent-history-steps-simulates-store-buffer-steps [OF exec-sb sim-h, of \(\mathcal{S}\) ]
obtain \(\mathrm{ts}_{\mathrm{h}}{ }^{\prime} \mathcal{S}_{\mathrm{h}}{ }^{\prime}\) where steps-h: \(\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right)\) and sim-h': \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{h}}{ }^{\prime}\)
by auto

\section*{moreover}
have empty:
\(\forall \mathrm{i}\) p is xs sb \(\mathcal{D} \mathcal{O} \mathcal{R}\). \(\mathrm{i}<\) length \(\mathrm{ts}_{\mathrm{h}} \longrightarrow \mathrm{ts}_{\mathrm{h}}!\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \longrightarrow \mathrm{sb}=[]\)
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def Let-def)
subgoal for i
apply (case-tac ts!i)
apply auto
done
done
have \(\operatorname{sim}^{\prime}:\left(\mathrm{ts}_{\mathrm{h}}, \mathrm{m}, \mathcal{S}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
apply (rule sim-config-emptyI [OF empty])
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def)
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{h}}\)-def Let-def)
subgoal for i
apply (case-tac ts!i)
```

    apply auto
    done
    done

```
from concurrent-direct-execution-simulates-store-buffer-history-execution [OF steps-h init-h valid-h sim' safe]
obtain \(\mathrm{ts}^{\prime} \mathrm{m}^{\prime \prime} \mathcal{S}^{\prime \prime}\) where steps: ( \(\left.\mathrm{ts}, \mathrm{m}, \mathcal{S}\right) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\)
and \(\operatorname{sim}^{\prime}:\left(\mathrm{ts}_{\mathrm{h}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}_{\mathrm{h}}{ }^{\prime}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime \prime}, \mathcal{S}^{\prime \prime}\right)\)
by blast
from empty-sb-sims [OF empty' \(\operatorname{sim}\)-h' \(\operatorname{sim}\) ] steps
show ?thesis
by auto
qed
locale initial \(_{V}=\) simple-ownership-distinct + read-only-unowned + unowned-shared + fixes valid
assumes empty-is: \(\llbracket i<\) length ts; ts! \(\mathrm{i}=(\mathrm{p}, \mathrm{is}, \mathrm{xs}, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) \rrbracket \Longrightarrow\) is=[]
assumes valid-init: valid (map \((\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}, \mathrm{is}, \vartheta,[], \mathcal{D}, \mathcal{O}\), Map.empty \())\) ts)
theorem (in xvalid-program-progress) store-buffer-execution-result-sequential-consistent':
assumes exec-sb: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}, \mathrm{x}\right) \Rightarrow_{\mathrm{sb}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathrm{x}^{\prime}\right)\)
assumes empty': empty-store-buffers ts \(_{\text {sb }}{ }^{\prime}\)
assumes sim: \(\mathrm{ts}_{\mathrm{sb}} \sim_{\mathrm{d}}\) ts
assumes init: initialv ts \(\mathcal{S}\) valid
assumes safe: safe-reach-virtual safe-free-flowing (ts,m,S )
shows \(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime}\).
\[
(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow_{\mathrm{v}}^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{d}} \mathrm{ts}^{\prime}
\]
proof -
define \(\mathrm{ts}_{\mathrm{d}}\) where \(\mathrm{ts}_{\mathrm{d}}==(\operatorname{map}(\lambda(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}\), Map.empty::rels \())\) ts)
have rem-ts: remove-rels \(\mathrm{ts}_{\mathrm{d}}=\) ts
apply (rule nth-equalityI)
apply ( simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def remove-rels-def)
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def remove-rels-def)
subgoal for i
apply (case-tac ts!i)
apply clarsimp
done
done
from sim
have \(\operatorname{sim}^{\prime}: \mathrm{ts}_{\mathrm{sb}} \sim_{d} \mathrm{ts}_{\mathrm{d}}\)
apply cases
apply (rule sim-direct-config.intros)
apply (auto simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def)
done
have init': initial \(_{d} \operatorname{ts}_{\mathrm{d}} \mathcal{S}\) valid
proof (intro-locales)
from init have simple-ownership-distinct ts
by (simp add: initialv-def)
then
show simple-ownership-distinct \(\mathrm{ts}_{\mathrm{d}}\)
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def simple-ownership-distinct-def)
subgoal for \(\mathrm{i} j\)
apply (case-tac ts!i)
apply (case-tac ts! j)
apply force
done
done
next
from init have read-only-unowned \(\mathcal{S}\) ts
by (simp add: initialv-def)
then show read-only-unowned \(\mathcal{S}\) ts
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def read-only-unowned-def)
subgoal for i
apply (case-tac ts!i)
apply force
done
done
next
from init have unowned-shared \(\mathcal{S}\) ts
by (simp add: initialv-def)
then
show unowned-shared \(\mathcal{S}\) ts \({ }_{\mathrm{d}}\)
apply (clarsimp simp add: \(\mathrm{ts}_{\mathrm{d}}\)-def unowned-shared-def)
apply force
done
next
have eq: \(((\lambda(\mathrm{p}\), is \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}\), is, \(\vartheta,[], \mathcal{D}, \mathcal{O}, \mathcal{R})) \circ\)
\(\left(\lambda\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}^{\prime}\right) .(\mathrm{p}\right.\), is, \(\vartheta,(), \mathcal{D}, \mathcal{O}\), Map.empty)))
\(=\left(\lambda\left(\mathrm{p}\right.\right.\), is, \(\vartheta\), sb \(\left., \mathcal{D}, \mathcal{O}, \mathcal{R}^{\prime}\right) .(\mathrm{p}\), is, \(\vartheta,[], \mathcal{D}, \mathcal{O}\), Map.empty \(\left.)\right)\)
apply (rule ext)
subgoal for x
apply (case-tac x )
apply auto
done
done
from init have initial \({ }_{v}\)-axioms ts valid
by (simp add: initial \({ }_{v}\)-def)

\section*{then}
show initial \({ }_{d}\)-axioms ts \(_{d}\) valid
apply (clarsimp simp add: ts \(_{d}\)-def initial \({ }_{v}\)-axioms-def initial \({ }_{d}\)-axioms-def eq)
apply (rule conjI)
apply clarsimp
subgoal for i
apply (case-tac ts!i)
apply force
done
apply clarsimp
subgoal for i
apply (case-tac ts!i)
apply force
done
done
qed
\[
\{
\]
fix \(\mathrm{ts}_{\mathrm{d}}{ }^{\prime} \mathrm{m}^{\prime} \mathcal{S}^{\prime}\)
assume exec: \(\left(\mathrm{ts}_{\mathrm{d}}, \mathrm{m}, \mathcal{S}\right) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
have safe-free-flowing \(\left(\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
proof -
from virtual-simulates-direct-steps [OF exec]
have exec-v: \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{v}}{ }^{*}\left(\right.\) remove-rels \(\left.\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
by (simp add: rem-ts)
have eq: map (owned o
\((\lambda(\mathrm{p}\), is \(, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R}) .(\mathrm{p}\), is, \(\vartheta,(), \mathcal{D}, \mathcal{O},())))\)
\[
\mathrm{ts}_{\mathrm{d}}{ }^{\prime}=\operatorname{map} \text { owned } \mathrm{ts}_{\mathrm{d}}{ }^{\prime}
\]
by auto
from exec-v safe
have safe-free-flowing (remove-rels \(\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\) )
by (auto simp add: safe-reach-def)
then show ?thesis
by (auto simp add: safe-free-flowing-def remove-rels-def owned-def eq)
qed
\}
hence safe': safe-reach-direct safe-free-flowing \(\left(\mathrm{ts}_{\mathrm{d}}, \mathrm{m}, \mathcal{S}\right)\)
by (simp add: safe-reach-def)
from store-buffer-execution-result-sequential-consistent [OF exec-sb empty \({ }^{\prime}\) sim \({ }^{\prime}\) init \(^{\prime}\) safe ]
obtain ts \(_{\mathrm{d}}{ }^{\prime} \mathcal{S}^{\prime}\) where
exec-d: \(\left(\mathrm{ts}_{\mathrm{d}}, \mathrm{m}, \mathcal{S}\right) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\) and \(\operatorname{sim}-\mathrm{d}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{d}} \mathrm{ts}_{\mathrm{d}}{ }^{\prime}\)
by blast
from virtual-simulates-direct-steps [OF exec-d]
have \((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow_{\mathrm{v}}{ }^{*}\left(\right.\) remove-rels \(\left.\mathrm{ts}_{\mathrm{d}}{ }^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
by (simp add: rem-ts)
moreover
from sim-d
have \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{d}\) remove-rels \(\mathrm{ts}_{\mathrm{d}}{ }^{\prime}\)
apply (cases)
apply (rule sim-direct-config.intros)
apply (auto simp add: remove-rels-def)
done
ultimately show ?thesis
by auto
qed

\section*{A. 7 Plug Together the Two Simulations}
corollary (in xvalid-program) concurrent-direct-steps-simulates-store-buffer-step:
assumes step-sb: \(\left(\mathrm{tt}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sb}}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
assumes sim-h: \(\mathrm{ts}_{\mathrm{sb}} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{sbh}}\)
assumes sim: \(\left(\mathrm{ts}_{\mathrm{sbh}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sbh}}\right) \sim(\mathrm{ts}, \mathrm{m}, \mathcal{S})\)
assumes valid-own: valid-ownership \(\mathcal{S}_{\text {sbh }} \mathrm{ts}_{\text {sbh }}\)
assumes valid-sb-reads: valid-reads \(\mathrm{m}_{\mathrm{sb}} \mathrm{ts}_{\mathrm{sbh}}\)
assumes valid-hist: valid-history program-step \(\mathrm{t}_{\text {sbh }}\)
assumes valid-sharing: valid-sharing \(\mathcal{S}_{\text {sbh }} \mathrm{ts}_{\text {sbh }}\)
assumes tmps-distinct: tmps-distinct \(\mathrm{ts}_{\text {sbh }}\)
assumes valid-sops: valid-sops \(\mathrm{ts}_{\text {sbh }}\)
assumes valid-dd: valid-data-dependency \(\mathrm{ts}_{\text {sbh }}\)
assumes load-tmps-fresh: load-tmps-fresh \(\mathrm{ts}_{\mathrm{sbh}}\)
assumes enough-flushs: enough-flushs \(\mathrm{ts}_{\text {sbh }}\)
assumes valid-program-history: valid-program-history \(\mathrm{ts}_{\mathrm{sbh}}\)
assumes valid: valid \(\mathrm{ts}_{\text {sbh }}\)
assumes safe-reach: safe-reach-direct safe-delayed (ts,m,S)
shows \(\exists \mathrm{ts}_{\text {sbh }}{ }^{\prime} \mathcal{S}_{\text {sbh }}\).
\(\left(\mathrm{ts}_{\mathrm{sbh}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sbh}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sbh}}\right) \wedge \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{sbh}}{ }^{\prime} \wedge\)
valid-ownership \(\mathcal{S}_{\text {sbh }}{ }^{\prime} \mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) valid-reads \(\mathrm{m}_{\text {sb }}{ }^{\prime} \mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\)
valid-history program-step \(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime} \wedge\)
valid-sharing \(\mathcal{S}_{\text {sbh }}{ }^{\prime} \mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) tmps-distinct \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) valid-data-dependency \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) valid-sops \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) load-tmps-fresh \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) enough-flushs \(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime} \wedge\) valid-program-history \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\) valid \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \wedge\)
\[
\left(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime} .(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow \mathrm{d}^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge\right.
\]
\[
\left.\left(\mathrm{ts}_{\mathrm{sbh}}, \mathrm{~m}_{\mathrm{sb}}^{\prime}, \mathcal{S}_{\mathrm{sbh}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\right)
\]
proof -
from concurrent-history-steps-simulates-store-buffer-step [OF step-sb sim-h]
obtain \(\mathrm{ts}_{\text {sbh }}{ }^{\prime} \mathcal{S}_{\text {sbh }}{ }^{\prime}\) where
steps-h: \(\left(\mathrm{ts}_{\mathrm{sbh}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sbh}}\right) \Rightarrow_{\mathrm{sbh}}{ }^{*}\left(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sbh}}{ }^{\prime}\right)\) and
sim- \(h^{\prime}: \mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \sim_{\mathrm{h}} \mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}\)
by blast

\section*{moreover}
from concurrent-direct-steps-simulates-store-buffer-history-steps [OF steps-h valid-own valid-sb-reads valid-hist valid-sharing tmps-distinct valid-sops valid-dd load-tmps-fresh enough-flushs valid-program-history valid sim safe-reach]
obtain \(\mathrm{m}^{\prime}\) ts \({ }^{\prime} \mathcal{S}^{\prime}\) where
\((\mathrm{ts}, \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}\right)\left(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sbh}}{ }^{\prime}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\)
valid-ownership \(\mathcal{S}_{\mathrm{sbh}}{ }^{\prime} \mathrm{ts}_{\mathrm{sb}}{ }^{\prime}\) valid-reads \(\mathrm{m}_{\mathrm{sb}}{ }^{\prime} \mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}\) valid-history program-step \(\mathrm{ts}_{\mathrm{sbh}}{ }^{\prime}\) valid-sharing \(\mathcal{S}_{\text {sbh }}{ }^{\prime} \mathrm{ts}_{\text {sbh }}{ }^{\prime}\) tmps-distinct \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) valid-data-dependency \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) valid-sops \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) load-tmps-fresh \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) enough-flushs \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) valid-program-history \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\) valid \(\mathrm{ts}_{\text {sbh }}{ }^{\prime}\)
by blast
ultimately
show ?thesis
by blast
qed
lemma conj-commI: \(\mathrm{P} \wedge \mathrm{Q} \Longrightarrow \mathrm{Q} \wedge \mathrm{P}\)
by \(\operatorname{simp}\)
lemma def-to-eq: \(\mathrm{P}=\mathrm{Q} \Longrightarrow \mathrm{P} \equiv \mathrm{Q}\)
by simp
context xvalid-program
begin

\section*{definition}
invariant ts \(\mathcal{S} \mathrm{m} \equiv\) valid-ownership \(\mathcal{S}\) ts \(\wedge\) valid-reads m ts \(\wedge\) valid-history program-step ts \(\wedge\) valid-sharing \(\mathcal{S}\) ts \(\wedge\) tmps-distinct ts \(\wedge\) valid-data-dependency ts \(\wedge\) valid-sops ts \(\wedge\) load-tmps-fresh ts \(\wedge\) enough-flushs ts \(\wedge\) valid-program-history ts \(\wedge\) valid ts
definition ownership-inv \(\equiv\) valid-ownership
definition sharing-inv \(\equiv\) valid-sharing
definition temporaries-inv ts \(\equiv\) tmps-distinct ts \(\wedge\) load-tmps-fresh ts
definition history-inv ts \(\mathrm{m} \equiv\) valid-history program-step ts \(\wedge\) valid-program-history ts \(\wedge\) valid-reads m ts
definition data-dependency-inv ts \(\equiv\) valid-data-dependency ts \(\wedge\) load-tmps-fresh ts \(\wedge\) valid-sops ts
definition barrier-inv \(\equiv\) enough-flushs
lemma invariant-grouped-def: invariant ts \(\mathcal{S} \mathrm{m} \equiv\)
ownership-inv \(\mathcal{S}\) ts \(\wedge\) sharing-inv \(\mathcal{S}\) ts \(\wedge\) temporaries-inv ts \(\wedge\) data-dependency-inv ts \(\wedge\)
history-inv ts \(\mathrm{m} \wedge\) barrier-inv ts \(\wedge\) valid ts
apply (rule def-to-eq)
apply (auto simp add: ownership-inv-def sharing-inv-def barrier-inv-def tempo-raries-inv-def history-inv-def data-dependency-inv-def invariant-def)

\section*{done}
theorem (in xvalid-program) simulation':
assumes step-sb: \(\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \Rightarrow_{\mathrm{sbh}}\left(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}{ }^{\prime}\right)\)
assumes \(\operatorname{sim}:\left(\mathrm{ts}_{\mathrm{sb}}, \mathrm{m}_{\mathrm{sb}}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}_{\mathrm{s}}, \mathrm{m}, \mathcal{S}\right)\)
assumes inv: invariant \(\mathrm{ts}_{\mathrm{sb}} \mathcal{S}_{\mathrm{sb}} \mathrm{m}_{\mathrm{sb}}\)
assumes safe-reach: safe-reach-direct safe-delayed (ts,m, \(\mathcal{S}\) )
shows invariant \(\mathrm{ts}_{\mathrm{sb}}{ }^{\prime} \mathcal{S}_{\mathrm{sb}}{ }^{\prime} \mathrm{m}_{\mathrm{sb}}{ }^{\prime} \wedge\)
\[
\left(\exists \mathrm{ts}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime} .(\mathrm{ts}, \mathrm{~m}, \mathcal{S}) \Rightarrow_{\mathrm{d}}{ }^{*}\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right) \wedge\left(\mathrm{ts}_{\mathrm{sb}}^{\prime}, \mathrm{m}_{\mathrm{sb}}{ }^{\prime}, \mathcal{S}_{\mathrm{sb}}\right) \sim\left(\mathrm{ts}^{\prime}, \mathrm{m}^{\prime}, \mathcal{S}^{\prime}\right)\right)
\]
using inv sim safe-reach
apply (unfold invariant-def)
apply (simp only: conj-assoc)
apply (rule concurrent-direct-steps-simulates-store-buffer-history-step [OF step-sb])
apply simp-all

\section*{done}
lemmas (in xvalid-program) simulation \(=\) conj-commI [OF simulation]

\section*{end}
end

\section*{A. 8 PIMP}
```

theory PIMP
imports ReduceStoreBufferSimulation
begin
datatype expr = Const val | Mem bool addr | Tmp sop
| Unop val }=>\mathrm{ val expr
| Binop val }=>\mathrm{ val }=>\mathrm{ val expr expr
datatype stmt =
Skip
| Assign bool expr expr tmps }=>\mathrm{ owns tmps }=>\mathrm{ owns tmps }=>\mathrm{ owns tmps }
owns
| CAS expr expr expr tmps }=>\mathrm{ owns tmps }=>\mathrm{ owns tmps }=>\mathrm{ owns tmps }=>\mathrm{ owns
| Seq stmt stmt
| Cond expr stmt stmt
| While expr stmt

```
            | SGhost tmps \(\Rightarrow\) owns tmps \(\Rightarrow\) owns tmps \(\Rightarrow\) owns tmps \(\Rightarrow\) owns
            | SFence
```

primrec used-tmps:: expr $\Rightarrow$ nat - number of temporaries used
where
used-tmps (Const v) $=0$
$\mid$ used-tmps $($ Mem volatile addr $)=1$
$\mid$ used-tmps $($ Tmp sop $)=0$
| used-tmps (Unop fe) $=$ used-tmps e
| used-tmps (Binop f e $\mathrm{e}_{1} \mathrm{e}_{2}$ ) $=$ used-tmps $\mathrm{e}_{1}+$ used-tmps $\mathrm{e}_{2}$
primrec issue-expr:: tmp $\Rightarrow$ expr $\Rightarrow$ instr list — load operations
where
issue-expr t (Const v) $=[]$
|issue-expr t (Mem volatile a) $=[$ Read volatile a t$]$
|issue-expr t $($ Tmp sop $)=[]$
|issue-expr t (Unop fe) $=$ issue-expr te
|issue-expr t (Binop f $\left.e_{1} e_{2}\right)=$ issue-expr $t e_{1} @$ issue-expr $\left(t+\left(\right.\right.$ used-tmps $\left.\left.e_{1}\right)\right) e_{2}$
primrec eval-expr:: $\mathrm{tmp} \Rightarrow \operatorname{expr} \Rightarrow$ sop - calculate result
where
eval-expr $\mathrm{t}($ Const v$)=(\{ \}, \lambda \vartheta . \mathrm{v})$

```
\(\mid\) eval-expr \(\mathrm{t}(\) Mem volatile a\()=(\{\mathrm{t}\}, \lambda \vartheta\). the \((\vartheta \mathrm{t}))\)
|eval-expr t \((\mathrm{Tmp}\) sop \()=\) sop
- trick to enforce sop to be sensible in the current context, without having to include wellformedness constraints
\(\mid\) eval-expr t \((\operatorname{Unop} \mathrm{fe})=\left(\operatorname{let}\left(\mathrm{D}, \mathrm{f}_{\mathrm{e}}\right)=\right.\) eval-expr tein \(\left.\left(\mathrm{D}, \lambda \vartheta . \mathrm{f}\left(\mathrm{f}_{\mathrm{e}} \vartheta\right)\right)\right)\) |eval-expr t (Binop f \(\mathrm{e}_{1} \mathrm{e}_{2}\) ) \(=\) (let \(\left(\mathrm{D}_{1}, \mathrm{f}_{1}\right)=\) eval-expr \(\mathrm{t}_{1}\);
\(\left(\mathrm{D}_{2}, \mathrm{f}_{2}\right)=\) eval-expr \(\left(\mathrm{t}+\left(\right.\right.\) used-tmps \(\left.\left.\mathrm{e}_{1}\right)\right) \mathrm{e}_{2}\)
in \(\left.\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}, \lambda \vartheta . \mathrm{f}\left(\mathrm{f}_{1} \vartheta\right)\left(\mathrm{f}_{2} \vartheta\right)\right)\right)\)
primrec valid-sops-expr:: nat \(\Rightarrow\) expr \(\Rightarrow\) bool

\section*{where}
valid-sops-expr t (Const v) \(=\) True
|valid-sops-expr t (Mem volatile a) \(=\) True
\(\mid\) valid-sops-expr \(\mathrm{t}(\operatorname{Tmp}\) sop \()=\left(\left(\forall \mathrm{t}^{\prime} \in\right.\right.\) fst sop. \(\left.\mathrm{t}^{\prime}<\mathrm{t}\right) \wedge\) valid-sop sop \()\)
\(\mid\) valid-sops-expr t (Unop fe) \(=\) valid-sops-expr t e
\(\mid\) valid-sops-expr t (Binop f \(\left.\mathrm{e}_{1} \mathrm{e}_{2}\right)=\left(\right.\) valid-sops-expr \(\mathrm{t} \mathrm{e}_{1} \wedge\) valid-sops-expr \(\left.\mathrm{t} \mathrm{e}_{2}\right)\)
```

primrec valid-sops-stmt:: nat $\Rightarrow$ stmt $\Rightarrow$ bool
where
valid-sops-stmt t Skip $=$ True
|valid-sops-stmt t (Assign volatile a e A L R W) $=$ (valid-sops-expr t a $\wedge$ valid-sops-expr
t e)
$\mid$ valid-sops-stmt t (CAS a $\left.c_{e} s_{e} A L R W\right)=\left(\right.$ valid-sops-expr ta valid-sops-expr t $c_{e} \wedge$
valid-sops-expr t $\mathrm{s}_{\mathrm{e}}$ )
$\mid$ valid-sops-stmt t (Seq s1 $\mathrm{s}_{2}$ ) $=$ (valid-sops-stmt t $\mathrm{s}_{1} \wedge$ valid-sops-stmt t s $\mathrm{s}_{2}$ )
$\mid$ valid-sops-stmt t (Cond e $\mathrm{s}_{1} \mathrm{~s}_{2}$ ) $=$ (valid-sops-expr t e $\wedge$ valid-sops-stmt $\mathrm{t} \mathrm{s}_{1} \wedge$
valid-sops-stmt t s2)
|valid-sops-stmt t (While e s) $=$ (valid-sops-expr t e $\wedge$ valid-sops-stmt t s)
$\mid$ valid-sops-stmt t (SGhost A L R W) $=$ True
$\mid$ valid-sops-stmt t SFence $=$ True

```
```

type-synonym stmt-config $=$ stmt $\times$ nat
consts isTrue:: val $\Rightarrow$ bool
inductive stmt-step:: tmps $\Rightarrow$ stmt-config $\Rightarrow$ stmt-config $\times$ instrs $\Rightarrow$ bool
$\left(-\vdash-\rightarrow_{\mathrm{s}}-[60,60,60] 100\right)$
for $\vartheta$
where

```

AssignAddr:
\(\forall\) sop. a \(\neq\) Tmp sop \(\Longrightarrow\)
    \(\vartheta \vdash\left(\right.\) Assign volatile a e A L R W, t) \(\rightarrow_{\mathrm{s}}\)
            ((Assign volatile (Tmp (eval-expr ta)) e A L R W, t + used-tmps a), issue-expr t
a)

\section*{| Assign: \\ \(\mathrm{D} \subseteq \operatorname{dom} \vartheta \Longrightarrow\)}
```

$\vartheta \vdash\left(\right.$ Assign volatile $(\operatorname{Tmp}(\mathrm{D}, \mathrm{a}))$ e A L R W, t) $\rightarrow_{\mathrm{s}}$
((Skip, t + used-tmps e),
issue-expr te@[Write volatile (a $\vartheta)($ eval-expr te) (A $\vartheta)(\mathrm{L} \vartheta)(\mathrm{R} \vartheta)(\mathrm{W} \vartheta)]$ )

```
```

| CASAddr:
sop. a }\not=\textrm{Tmp sop \Longrightarrow
\vartheta\vdash (CAS a ce se A L R W, t) }\mp@subsup{->}{\textrm{s}}{
((CAS (Tmp (eval-expr t a)) ce se A L R W, t + used-tmps a), issue-expr t a)

```
| CASComp:
    \(\forall\) sop. \(\mathrm{c}_{\mathrm{e}} \neq\) Tmp sop \(\Longrightarrow\)
        \(\vartheta \vdash\left(\operatorname{CAS}\left(\operatorname{Tmp}\left(\mathrm{D}_{\mathrm{a}}, \mathrm{a}\right)\right) \mathrm{c}_{\mathrm{e}} \mathrm{s}_{\mathrm{e}}\right.\) A L R W, t) \(\rightarrow_{\mathrm{s}}\)
                            ((CAS (Tmp ( \(\left.\left.\mathrm{D}_{\mathrm{a}}, \mathrm{a}\right)\right)\left(\operatorname{Tmp}\left(\right.\right.\) eval-expr \(\left.\left.\mathrm{t} \mathrm{c}_{\mathrm{e}}\right)\right) \mathrm{s}_{\mathrm{e}}\) A L R W, \(\mathrm{t}+\) used-tmps \(\left.\mathrm{c}_{\mathrm{e}}\right)\),
issue-expr \(t \mathrm{c}_{\mathrm{e}}\) )
| CAS:
    \(\llbracket \mathrm{D}_{\mathrm{a}} \subseteq \operatorname{dom} \vartheta ; \mathrm{D}_{\mathrm{c}} \subseteq \operatorname{dom} \vartheta ;\) eval-expr \(\mathrm{t} \mathrm{s}_{\mathrm{e}}=(\mathrm{D}, \mathrm{f}) \rrbracket\)
        \(\Longrightarrow\)
        \(\vartheta \vdash\left(\operatorname{CAS}\left(\operatorname{Tmp}\left(\mathrm{D}_{\mathrm{a}}, \mathrm{a}\right)\right)\left(\operatorname{Tmp}\left(\mathrm{D}_{\mathrm{c}}, \mathrm{c}\right)\right) \mathrm{s}_{\mathrm{e}} \mathrm{A} \mathrm{L} R \mathrm{~W}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\)
            ((Skip, Suc ( \(\mathrm{t}+\) used-tmps \(\left.\mathrm{s}_{\mathrm{e}}\right)\) ), issue-expr \(\mathrm{t} \mathrm{s}_{\mathrm{e}}\) @
            [RMW (a \(\vartheta)\left(\mathrm{t}+\mathrm{used}-\mathrm{tmps} \mathrm{s}_{\mathrm{e}}\right)(\mathrm{D}, \mathrm{f})\left(\lambda \vartheta\right.\). the \(\left(\vartheta\left(\mathrm{t}+\right.\right.\) used-tmps \(\left.\left.\left.\mathrm{s}_{\mathrm{e}}\right)\right)=\mathrm{c} \vartheta\right)\left(\lambda \mathrm{v}_{1}\right.\)
\(\mathrm{v}_{2} . \mathrm{v}_{1}\) )
            \((\mathrm{A} \vartheta)(\mathrm{L} \vartheta)(\mathrm{R} \vartheta)(\mathrm{W} \vartheta)])\)
| Seq:
    \(\vartheta \vdash\left(\mathrm{s}_{1}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}_{1}{ }^{\prime}, \mathrm{t}^{\prime}\right)\right.\), is \()\)
    \(\Longrightarrow\)
    \(\vartheta \vdash\left(\operatorname{Seq~s} \mathrm{s}_{1} \mathrm{~s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\left(\left(\operatorname{Seq~s}_{1}{ }^{\prime} \mathrm{s}_{2}, \mathrm{t}^{\prime}\right)\right.\), is\()\)
| SeqSkip:
    \(\vartheta \vdash\left(\right.\) Seq Skip \(\left.\mathrm{s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}_{2}, \mathrm{t}\right),[]\right)\)
| Cond:
    \(\forall\) sop. e \(\neq\) Tmp sop
        \(\xrightarrow[\vartheta \vdash]{\longrightarrow}\left(\right.\) Cond e s \(\left.\mathrm{s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\)
        ((Cond (Tmp (eval-expr te\()) \mathrm{s}_{1} \mathrm{~s}_{2}, \mathrm{t}+\) used-tmps e), issue-expr \(\left.\mathrm{t} e\right)\)
| CondTrue:
    \(\llbracket \mathrm{D} \subseteq \operatorname{dom} \vartheta\); isTrue (e \(\vartheta\) )】
        \(\Longrightarrow\)
        \(\vartheta \vdash\left(\operatorname{Cond}(\operatorname{Tmp}(\mathrm{D}, \mathrm{e})) \mathrm{s}_{1} \mathrm{~s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}_{1}, \mathrm{t}\right),[]\right)\)
| CondFalse:
    \(\llbracket \mathrm{D} \subseteq \operatorname{dom} \vartheta ; \neg\) isTrue (e \(\vartheta\) ) \(\rrbracket\)
    \(\Longrightarrow\)
    \(\vartheta \vdash\left(\operatorname{Cond}(\operatorname{Tmp}(\mathrm{D}, \mathrm{e})) \mathrm{s}_{1} \mathrm{~s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}_{2}, \mathrm{t}\right),[]\right)\)
| While:
\(\vartheta \vdash\left(\right.\) While e s, t) \(\rightarrow_{s}\)
((Cond e (Seq s (While e s)) Skip, t),[])
| SGhost:
\(\vartheta \vdash\left(\right.\) SGhost A L R W, t) \(\rightarrow_{\mathrm{s}}((\) Skip, t\(),[\operatorname{Ghost}(\mathrm{A} \vartheta)(\mathrm{L} \vartheta)(\mathrm{R} \vartheta)(\mathrm{W} \vartheta)])\)
| SFence:
\(\vartheta \vdash(\) SFence, t\() \rightarrow_{\mathrm{s}}((\) Skip, t\(),[\) Fence \()\)
inductive-cases stmt-step-cases [cases set]:
\(\vartheta \vdash(\) Skip, t\() \rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash\left(\right.\) Assign volatile a e A L R W, t) \(\rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash\left(\mathrm{CAS}\right.\) a \(\left.\mathrm{c}_{\mathrm{e}} \mathrm{s}_{\mathrm{e}} \mathrm{ALRW} \mathrm{W}, \mathrm{t}\right) \rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash\left(\operatorname{Seq~s}_{1} \mathrm{~s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash\left(\right.\) Cond e s \(\left.\mathrm{s}_{2}, \mathrm{t}\right) \rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash(\) While es, t\() \rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash\left(\right.\) SGhost A L R W, t) \(\rightarrow_{\mathrm{s}} \mathrm{c}\)
\(\vartheta \vdash(\) SFence, t\() \rightarrow_{\mathrm{s}} \mathrm{c}\)
lemma valid-sops-expr-mono: \(\wedge \mathrm{t} \mathrm{t}^{\prime}\). valid-sops-expr \(\mathrm{t} \mathrm{e} \Longrightarrow \mathrm{t} \leq \mathrm{t}^{\prime} \Longrightarrow\) valid-sops-expr \(\mathrm{t}^{\prime}\) e
by (induct e) auto
lemma valid-sops-stmt-mono: \(\Lambda \mathrm{t} \mathrm{t}^{\prime}\). valid-sops-stmt \(\mathrm{t} \mathrm{s} \Longrightarrow \mathrm{t} \leq \mathrm{t}^{\prime} \Longrightarrow\) valid-sops-stmt t's
by (induct s) (auto intro: valid-sops-expr-mono)
lemma valid-sops-expr-valid-sop: \(\wedge \mathrm{t}\). valid-sops-expr \(\mathrm{t} \mathrm{e} \Longrightarrow\) valid-sop (eval-expr t e )
proof (induct e)
case (Unop fe)
then obtain valid-sops-expr t e
by simp
from Unop.hyps [OF this]
have vs: valid-sop (eval-expr te)
by simp
obtain D g where eval-e: eval-expr \(\mathrm{t} \mathrm{e}=(\mathrm{D}, \mathrm{g})\)
by (cases eval-expr t e)
interpret valid-sop (D,g)
using vs eval-e
by simp
show ?case
apply (clarsimp simp add: Let-def valid-sop-def eval-e)
apply (drule valid-sop [OF refl])
apply simp
done
next
```

case (Binop f $\mathrm{e}_{1} \mathrm{e}_{2}$ )
then obtain v1: valid-sops-expr $t e_{1}$ and $v 2$ : valid-sops-expr $t e_{2}$
by simp
with Binop.hyps (1) [of t] Binop.hyps (2) [of ( $\mathrm{t}+\mathrm{used}-\mathrm{tmps} \mathrm{e}_{1}$ )]
valid-sops-expr-mono [OF v2, of ( $\mathrm{t}+$ used-tmps $\mathrm{e}_{1}$ )]
obtain vs1: valid-sop (eval-expr $\mathrm{t}_{1}$ ) and vs2: valid-sop (eval-expr ( $\mathrm{t}+$ used-tmps $\mathrm{e}_{1}$ )
$\mathrm{e}_{2}$ )
by auto
obtain $\mathrm{D}_{1} \mathrm{~g}_{1}$ where eval- $\mathrm{e}_{1}$ : eval-expr $\mathrm{t} \mathrm{e}_{1}=\left(\mathrm{D}_{1}, \mathrm{~g}_{1}\right)$
by (cases eval-expr $\mathrm{te}_{1}$ )
obtain $D_{2} g_{2}$ where eval- $\mathrm{e}_{2}$ : eval-expr $\left(\mathrm{t}+\right.$ used-tmps $\left.\mathrm{e}_{1}\right) \mathrm{e}_{2}=\left(\mathrm{D}_{2}, \mathrm{~g}_{2}\right)$
by (cases eval-expr ( $\mathrm{t}+$ used-tmps $\mathrm{e}_{1}$ ) $\mathrm{e}_{2}$ )
interpret vs1: valid-sop ( $\mathrm{D}_{1}, \mathrm{~g}_{1}$ )
using vs1 eval- $\mathrm{e}_{1}$ by auto
interpret vs2: valid-sop ( $\mathrm{D}_{2}, \mathrm{~g}_{2}$ )
using vs2 eval- $\mathrm{e}_{2}$ by auto
\{
fix $\vartheta::$ nat $\Rightarrow$ val option
assume D1: $\mathrm{D}_{1} \subseteq \operatorname{dom} \vartheta$
assume $\mathrm{D} 2: \mathrm{D}_{2} \subseteq \operatorname{dom} \vartheta$
have $\mathrm{f}\left(\mathrm{g}_{1} \vartheta\right)\left(\mathrm{g}_{2} \vartheta\right)=\mathrm{f}\left(\mathrm{g}_{1}\left(\left.\vartheta\right|^{*}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)\right)\left(\mathrm{g}_{2}\left(\left.\vartheta\right|^{6}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)\right)$
proof -
from vs1.valid-sop [OF refl D1]
have $\mathrm{g}_{1} \vartheta=\mathrm{g}_{1}\left(\left.\vartheta\right|^{6} \mathrm{D}_{1}\right)$.
also
from D1 have $\mathrm{D}^{\prime}: \mathrm{D}_{1} \subseteq \operatorname{dom}\left(\left.\vartheta\right|^{6}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)$
by auto
have $\left.\left.\vartheta\right|^{6}\left(D_{1} \cup D_{2}\right)\right|^{6} D_{1}=\left.\vartheta\right|^{6} D_{1}$
apply (rule ext)
apply (auto simp add: restrict-map-def)
done
with vs1.valid-sop [OF refl D1]
have $\mathrm{g}_{1}\left(\left.\vartheta\right|^{6} \mathrm{D}_{1}\right)=\mathrm{g}_{1}\left(\left.\vartheta\right|^{6}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)$
by auto
finally have $\mathrm{g} 1: \mathrm{g}_{1}\left(\left.\vartheta\right|^{\prime}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)=\mathrm{g}_{1} \vartheta$
by simp
from vs2.valid-sop [OF refl D2]
have $\mathrm{g}_{2} \vartheta=\mathrm{g}_{2}\left(\left.\vartheta\right|^{6} \mathrm{D}_{2}\right)$.
also
from $D 2$ have $D^{\prime}: D_{2} \subseteq \operatorname{dom}\left(\left.\vartheta\right|^{\prime}\left(D_{1} \cup D_{2}\right)\right)$
by auto
have $\left.\left.\vartheta\right|^{6}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right|^{6} \mathrm{D}_{2}=\left.\vartheta\right|^{6} \mathrm{D}_{2}$
apply (rule ext)
apply (auto simp add: restrict-map-def)
done
with vs2.valid-sop [OF refl D2]
have $\mathrm{g}_{2}\left(\left.\vartheta\right|^{6} \mathrm{D}_{2}\right)=\mathrm{g}_{2}\left(\left.\vartheta\right|^{6}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)$
by auto
finally have $\mathrm{g} 2: \mathrm{g}_{2}\left(\left.\vartheta\right|^{\prime}\left(\mathrm{D}_{1} \cup \mathrm{D}_{2}\right)\right)=\mathrm{g}_{2} \vartheta$

```
```

by simp
from g1 g2 show ?thesis by simp
qed
}
note lem=this
show ?case
apply (clarsimp simp add: Let-def valid-sop-def eval-- }\mp@subsup{e}{1}{}\mathrm{ eval-- }\mp@subsup{e}{2}{}
apply (rule lem)
by auto
qed (auto simp add: valid-sop-def)
lemma valid-sops-expr-eval-expr-in-range:
t. valid-sops-expr t e \Longrightarrow }\Longrightarrow\mp@subsup{\textrm{t}}{}{\prime}\in\mathrm{ fst (eval-expr t e). t' }<\textrm{t}+\mathrm{ used-tmps e
proof (induct e)
case (Unop f e)
thus ?case
apply (cases eval-expr t e)
apply auto
done
next
case (Binop f e e e e
then obtain v1: valid-sops-expr t e e
by simp
from valid-sops-expr-mono [OF v2]
have v2': valid-sops-expr (t + used-tmps e}\mp@subsup{e}{1}{})\mp@subsup{e}{2}{
by auto
from Binop.hyps (1) [OF v1] Binop.hyps (2) [OF v2']
show ?case
apply (cases eval-expr t e e
apply (cases eval-expr (t + used-tmps e}\mp@subsup{\textrm{e}}{1}{})\mp@subsup{\textrm{e}}{2}{}
apply auto
done
qed auto

```
lemma stmt-step-tmps-count-mono:
    assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right), \mathrm{is}\right)\)
    shows \(\mathrm{t} \leq \mathrm{t}^{\prime}\)
using step
by (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}=\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is) arbitrary: \(\mathrm{s} \mathrm{t} \mathrm{s} \mathrm{s}^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct) force +
lemma valid-sops-stmt-invariant:
assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\)
shows valid-sops-stmt t s \(\Longrightarrow\) valid-sops-stmt t's'
using step
proof (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}==\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\), is \()\) arbitrary: \(\mathrm{st} \mathrm{s}^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct)
case AssignAddr thus ?case by
(force simp add: valid-sops-expr-valid-sop intro: valid-sops-stmt-mono valid-sops-expr-mono
dest: valid-sops-expr-eval-expr-in-range)
next
case Assign thus ?case by simp
next
case CASAddr thus ?case by
(force simp add: valid-sops-expr-valid-sop intro: valid-sops-stmt-mono valid-sops-expr-mono dest: valid-sops-expr-eval-expr-in-range)
next
case CASComp thus ?case by
(force simp add: valid-sops-expr-valid-sop intro: valid-sops-stmt-mono valid-sops-expr-mono
dest: valid-sops-expr-eval-expr-in-range)
next
case CAS thus ?case by simp
next
case Seq thus ?case by (force intro: valid-sops-stmt-mono dest: stmt-step-tmps-count-mono)
next
case SeqSkip thus ?case by auto
next
case Cond thus ?case
by (fastforce simp add: valid-sops-expr-valid-sop intro: valid-sops-stmt-mono
dest: valid-sops-expr-eval-expr-in-range)
next
case CondTrue thus ?case by force
next
case CondFalse thus ?case by force
next
case While thus ?case by auto
next
case SGhost thus ?case by simp
next
case SFence thus ?case by simp
qed
lemma map-le-restrict-map-eq: \(\left.\mathrm{m}_{1} \subseteq_{\mathrm{m}} \mathrm{m}_{2} \Longrightarrow \mathrm{D} \subseteq \operatorname{dom} \mathrm{m}_{1} \Longrightarrow \mathrm{~m}_{2}\right|^{6} \mathrm{D}=\left.\mathrm{m}_{1}\right|^{‘} \mathrm{D}\)
apply (rule ext)
apply (force simp add: restrict-map-def map-le-def)
done
lemma sbh-step-preserves-load-tmps-bound:
assumes step: (is, \(\mathcal{O}, \mathcal{D}, \vartheta, \mathrm{sb}, \mathcal{S}, \mathrm{m}) \rightarrow_{\text {sbh }}\left(\mathrm{is}^{\prime}, \mathcal{O}^{\prime}, \mathcal{D}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{S}^{\prime}, \mathrm{m}\right)\)
assumes less: \(\forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{n}\)
shows \(\forall \mathrm{i} \in\) load-tmps is'. \(\mathrm{i}<\mathrm{n}\)
using step less
by cases auto
lemma sbh-step-preserves-read-tmps-bound:
assumes step: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes less-is: \(\forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{n}\)
assumes less-sb: \(\forall \mathrm{i} \in\) read-tmps sb. \(\mathrm{i}<\mathrm{n}\)
shows \(\forall \mathrm{i} \in\) read-tmps \(\mathrm{sb}^{\prime} . \mathrm{i}<\mathrm{n}\)
using step less-is less-sb
by cases (auto simp add: read-tmps-append)
lemma sbh-step-preserves-tmps-bound:
assumes step: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes less-dom: \(\forall \mathrm{i} \in \operatorname{dom} \vartheta\). i \(<\mathrm{n}\)
assumes less-is: \(\forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{n}\)
shows \(\forall \mathrm{i} \in \operatorname{dom} \vartheta^{\prime} . \mathrm{i}<\mathrm{n}\)
using step less-dom less-is
by cases (auto simp add: read-tmps-append)
lemma flush-step-preserves-read-tmps:
assumes step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}\right)\)
assumes less-sb: \(\forall \mathrm{i} \in\) read-tmps sb. \(\mathrm{i}<\mathrm{n}\)
shows \(\forall \mathrm{i} \in\) read-tmps \(\mathrm{sb}^{\prime} . \mathrm{i}<\mathrm{n}\)
using step less-sb
by cases (auto simp add: read-tmps-append)
lemma flush-step-preserves-write-sops:
assumes step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}\right)\)
assumes less-sb: \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb). i \(<\) t
shows \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb'). \(\mathrm{i}<\mathrm{t}\)
using step less-sb
by cases (auto simp add: read-tmps-append)
lemma issue-expr-load-tmps-range':
\(\bigwedge \mathrm{t}\). load-tmps (issue-expr t e\()=\{\mathrm{i} . \mathrm{t} \leq \mathrm{i} \wedge \mathrm{i}<\mathrm{t}+\) used-tmps e \(\}\)
apply (induct e)
apply (force simp add: load-tmps-append) + done
lemma issue-expr-load-tmps-range:
\(\bigwedge \mathrm{t} . \forall \mathrm{i} \in\) load-tmps (issue-expr t e\() . \mathrm{t} \leq \mathrm{i} \wedge \mathrm{i}<\mathrm{t}+\) (used-tmps e)
by (auto simp add: issue-expr-load-tmps-range')
lemma stmt-step-load-tmps-range':
assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\)
shows load-tmps is \(=\left\{\mathrm{i} . \mathrm{t} \leq \mathrm{i} \wedge \mathrm{i}<\mathrm{t}^{\prime}\right\}\)
using step
apply (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}==\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is) arbitrary: \(\mathrm{st} \mathrm{s} \mathrm{s}^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct)
apply (force simp add: load-tmps-append simp add: issue-expr-load-tmps-range') + done
```

lemma stmt-step-load-tmps-range:
assumes step: \vartheta\vdash (s, t) }\mp@subsup{->}{\textrm{s}}{}((\mp@subsup{\textrm{s}}{}{\prime},\textrm{t}),\mathrm{ ,is)
shows }\forall\textrm{i}\in\mathrm{ load-tmps is. }\textrm{t}\leq\textrm{i}\wedge\textrm{i}<\mp@subsup{\textrm{t}}{}{\prime
using stmt-step-load-tmps-range' [OF step]
by auto

```
lemma distinct-load-tmps-issue-expr: \(\wedge\) t. distinct-load-tmps (issue-expr te)
    apply (induct e)
    apply (auto simp add: distinct-load-tmps-append dest!: issue-expr-load-tmps-range
[rule-format])
    done
lemma max-used-load-tmps: \(\mathrm{t}+\) used-tmps e \(\notin\) load-tmps (issue-expr t e)
proof -
    from issue-expr-load-tmps-range [rule-format, of \(\mathrm{t}+\mathrm{used}\)-tmps e]
    show ?thesis
        by auto
qed
lemma stmt-step-distinct-load-tmps:
    assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\)
    shows distinct-load-tmps is
    using step
    apply (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}==\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\) arbitrary: \(\mathrm{st} \mathrm{s}{ }^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct)
        apply (force simp add: distinct-load-tmps-append distinct-load-tmps-issue-expr
max-used-load-tmps)+
    done
lemma store-sops-issue-expr [simp]: \(\Lambda\) t. store-sops (issue-expr te) \(=\{ \}\)
    apply (induct e)
    apply (auto simp add: store-sops-append)
    done
lemma stmt-step-data-store-sops-range:
    assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\)
    assumes valid: valid-sops-stmt t s
    shows \(\forall(\mathrm{D}, \mathrm{f}) \in\) store-sops is. \(\forall \mathrm{i} \in \mathrm{D} . \mathrm{i}<\mathrm{t}^{\prime}\)
using step valid
proof (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}==\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\) arbitrary: \(\mathrm{st} \mathrm{s}{ }^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct)
    case AssignAddr
    thus ?case
        by auto
next
    case (Assign D volatile a e)
```

    thus ?case
    apply (cases eval-expr t e)
            apply (auto simp add: store-sops-append intro: valid-sops-expr-eval-expr-in-range
    [rule-format])
done
next
case CASAddr
thus ?case
by auto
next
case CASComp
thus ?case
by auto
next
case (CAS - - D f a A L R)
thus ?case
by (fastforce simp add: store-sops-append dest: valid-sops-expr-eval-expr-in-range
[rule-format])
next
case Seq
thus ?case
by (force intro: valid-sops-stmt-mono )
next
case SeqSkip thus ?case by simp
next
case Cond thus ?case
by auto
next
case CondTrue thus ?case by auto
next
case CondFalse thus ?case by auto
next
case While thus ?case by auto
next
case SGhost thus ?case by auto
next
case SFence thus ?case by auto
qed
lemma sbh-step-distinct-load-tmps-prog-step:
assumes step: }\vartheta\vdash(\textrm{s},\textrm{t})\mp@subsup{->}{\textrm{s}}{}((\mp@subsup{\textrm{s}}{}{\prime},\textrm{t}),\mathrm{ ,is')
assumes load-tmps-le: }\forall\textrm{i}\in\mathrm{ load-tmps is. i < t
assumes read-tmps-le: }\forall\textrm{i}\in\mathrm{ read-tmps sb. i < t
shows distinct-load-tmps is '}\wedge(\mathrm{ load-tmps is' }\cap\mathrm{ load-tmps is = {})^
(load-tmps is'\cap read-tmps sb) ={}
proof -
from stmt-step-load-tmps-range [OF step] stmt-step-distinct-load-tmps [OF step]
load-tmps-le read-tmps-le
show ?thesis
by force

```
qed
```

lemma data-dependency-consistent-instrs-issue-expr:
\t T. data-dependency-consistent-instrs T (issue-expr t e)
apply (induct e)
apply (auto simp add: data-dependency-consistent-instrs-append
dest!: issue-expr-load-tmps-range [rule-format]
)
done

```
lemma dom-eval-expr:
    \(\Lambda \mathrm{t} . \llbracket\) valid-sops-expr \(\mathrm{t} \mathrm{e} ; \mathrm{x} \in\) fst (eval-expr t\() \rrbracket \Longrightarrow \mathrm{x} \in\{\mathrm{i} . \mathrm{i}<\mathrm{t}\} \cup\) load-tmps (issue-expr
te)
proof (induct e)
    case Const thus ?case by simp
next
    case Mem thus ?case by simp
next
    case Tmp thus ?case by simp
next
    case (Unop fe)
    thus ?case
        by (cases eval-expr te) auto
next
    case (Binop fe1 e2)
    then obtain valid1: valid-sops-expr t e1 and valid2: valid-sops-expr t e2
        by auto
    from valid-sops-expr-mono [OF valid2] have valid2': valid-sops-expr (t+used-tmps e1)
e2
        by auto
    from Binop.hyps (1) [OF valid1] Binop.hyps (2) [OF valid2] Binop.prems
    show ?case
    apply (case-tac eval-expr te1)
    apply (case-tac eval-expr (t+used-tmps e1) e2)
    apply (auto simp add: load-tmps-append issue-expr-load-tmps-range')
    done
qed
lemma Cond-not-s 1 : \(\mathrm{s}_{1} \neq\) Cond e \(\mathrm{s}_{1} \mathrm{~s}_{2}\)
    by (induct \(\mathrm{s}_{1}\) ) auto
lemma Cond-not-s 2 : \(\mathrm{s}_{2} \neq\) Cond e \(\mathrm{s}_{1} \mathrm{~s}_{2}\)
    by (induct \(\mathrm{s}_{2}\) ) auto
lemma Seq-not-s \(\mathrm{s}_{1}: \mathrm{s}_{1} \neq \operatorname{Seq}_{1} \mathrm{~s}_{2}\)
    by (induct \(\mathrm{s}_{1}\) ) auto
```

lemma Seq-not-s2: s2
by (induct s2) auto
lemma prog-step-progress:
assumes step: }\vartheta\vdash(\textrm{s},\textrm{t})\mp@subsup{->}{\textrm{s}}{}((\mp@subsup{\textrm{s}}{}{\prime},\mp@subsup{\textrm{t}}{}{\prime}),\mathrm{ is }
shows ( }\mp@subsup{\textrm{s}}{}{\prime},\mp@subsup{\textrm{t}}{}{\prime})\not=(\textrm{s},\textrm{t})\vee\mathrm{ is }\not=[
using step
proof (induct x==(s,t) y==((s',t'),is) arbitrary: st s't' is rule: stmt-step.induct)
case (AssignAddr a ---- - t) thus ?case
by (cases eval-expr t a) auto
next
case Assign thus ?case by auto
next
case (CASAddr a - - - - - t) thus ?case by (cases eval-expr t a) auto
next
case (CASComp ce------ - t) thus ?case by (cases eval-expr t ce) auto
next
case CAS thus ?case by auto
next
case (Cond e - - t) thus ?case by (cases eval-expr t e) auto
next
case CondTrue thus ?case using Cond-not-s, by auto
next
case CondFalse thus ?case using Cond-not-s2 by auto
next
case Seq thus ?case by force
next
case SeqSkip thus ?case using Seq-not-\mp@subsup{s}{2}{}}\mathrm{ by auto
next
case While thus ?case by auto
next
case SGhost thus ?case by auto
next
case SFence thus ?case by auto
qed
lemma stmt-step-data-dependency-consistent-instrs:
assumes step: }\vartheta\vdash(\textrm{s},\textrm{t})\mp@subsup{->}{\textrm{s}}{}((\mp@subsup{\textrm{s}}{}{\prime},\textrm{t}),\mathrm{ ,is }
assumes valid: valid-sops-stmt t s
shows data-dependency-consistent-instrs ({i. i < t }) is
using step valid
proof (induct x==(s,t) y==(( (s', t'),is) arbitrary: st s't' is T rule: stmt-step.induct)
case AssignAddr
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append
data-dependency-consistent-instrs-issue-expr load-tmps-append
dest: dom-eval-expr)
next
case Assign
thus ?case

```
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append data-dependency-consistent-instrs-issue-expr load-tmps-append dest: dom-eval-expr)
next
case CASAddr
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append data-dependency-consistent-instrs-issue-expr load-tmps-append dest: dom-eval-expr)

\section*{next}
case CASComp
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append data-dependency-consistent-instrs-issue-expr load-tmps-append dest: dom-eval-expr)
next
case CAS
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append data-dependency-consistent-instrs-issue-expr load-tmps-append dest: dom-eval-expr)
next
case Seq
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append)

\section*{next}
case SeqSkip thus ?case by auto
next
case Cond
thus ?case
by (fastforce simp add: simp add: data-dependency-consistent-instrs-append data-dependency-consistent-instrs-issue-expr load-tmps-append dest: dom-eval-expr)
next
case CondTrue thus ?case by auto
next
case CondFalse thus ?case by auto
next
case While
thus ?case by auto
next
case SGhost thus ?case by auto
next
case SFence thus ?case by auto qed
lemma sbh-valid-data-dependency-prog-step:
assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\),is \()\)
assumes store-sops-le: \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t}\)
assumes write-sops-le: \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb). \(\mathrm{i}<\mathrm{t}\)
assumes valid: valid-sops-stmt t s
shows data-dependency-consistent-instrs ( \(\{\mathrm{i} . \mathrm{i}<\mathrm{t}\})\) is \(^{\prime} \wedge\)
load-tmps is' \(\cap \bigcup\) (fst ' store-sops is \()=\{ \} \wedge\)
load-tmps is' \(\cap \bigcup\) (fst ' write-sops sb) \(=\{ \}\)
proof -
from stmt-step-data-dependency-consistent-instrs [OF step valid]
stmt-step-load-tmps-range [OF step]
store-sops-le write-sops-le
show ?thesis
by fastforce
qed
lemma sbh-load-tmps-fresh-prog-step:
assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\), is \()\)
assumes tmps-le: \(\forall \mathrm{i} \in \operatorname{dom} \vartheta . \mathrm{i}<\mathrm{t}\)
shows load-tmps is \({ }^{\prime} \cap \operatorname{dom} \vartheta=\{ \}\)

\section*{proof -}
from stmt-step-load-tmps-range [OF step] tmps-le
show ?thesis
apply auto
subgoal for x
apply (drule-tac \(x=x\) in bspec )
apply assumption
apply (drule-tac \(x=x\) in bspec )
apply fastforce
apply simp
done
done
qed
lemma sbh-valid-sops-prog-step:
assumes step: \(\vartheta \vdash(\mathrm{s}, \mathrm{t}) \rightarrow_{\mathrm{s}}\left(\left(\mathrm{s}^{\prime}, \mathrm{t}\right)\right.\), is \()\)
assumes valid: valid-sops-stmt t s
shows \(\forall\) sop \(\in\) store-sops is. valid-sop sop
using step valid
proof (induct \(\mathrm{x}==(\mathrm{s}, \mathrm{t}) \mathrm{y}==\left(\left(\mathrm{s}^{\prime}, \mathrm{t}\right), \mathrm{is}\right)\) arbitrary: \(\mathrm{st} \mathrm{s} \mathrm{s}^{\prime} \mathrm{t}^{\prime}\) is rule: stmt-step.induct)
case AssignAddr
thus ?case by auto
next
case Assign
thus ?case
by (auto simp add: store-sops-append valid-sops-expr-valid-sop)
next
case CASAddr
thus ?case by auto
next
case CASComp
thus ?case by auto
```

next
case CAS
thus ?case
by (fastforce simp add: store-sops-append dest: valid-sops-expr-valid-sop)
next
case Seq
thus ?case
by (force intro: valid-sops-stmt-mono )
next
case SeqSkip thus ?case by simp
next
case Cond thus ?case
by auto
next
case CondTrue thus ?case by auto
next
case CondFalse thus ?case by auto
next
case While thus ?case by auto
next
case SGhost thus ?case by auto
next
case SFence thus ?case by auto
qed
primrec prog-configs:: 'a memref list }=>\mathrm{ ' 'a set
where
prog-configs [] = {}
|prog-configs (x\#xs) = (case x of

```

```

                    |- = prog-configs xs)
    lemma prog-configs-append: $\bigwedge$ ys. prog-configs $(x s @ y s)=$ prog-configs xs $\cup$ prog-configs ys by (induct xs) (auto split: memref.splits)
lemma prog-configs-in1: $\operatorname{Prog}_{s b} \mathrm{p}_{1} \mathrm{p}_{2}$ is $\in$ set $\mathrm{xs} \Longrightarrow \mathrm{p}_{1} \in$ prog-configs xs by (induct xs) (auto split: memref.splits)
lemma prog-configs-in2: $\operatorname{Prog}_{s b} \mathrm{p}_{1} \mathrm{p}_{2}$ is $\in$ set $\mathrm{xs} \Longrightarrow \mathrm{p}_{2} \in$ prog-configs xs by (induct xs) (auto split: memref.splits)
lemma prog-configs-mono: $\bigwedge$ ys. set xs $\subseteq$ set ys $\Longrightarrow$ prog-configs xs $\subseteq$ prog-configs ys by (induct xs) (auto split: memref.splits simp add: prog-configs-append prog-configs-in1 prog-configs-in2)
locale separated-tmps $=$
fixes ts
assumes valid-sops-stmt: $\llbracket \mathrm{i}<$ length ts; ts! $=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket$
$\Longrightarrow$ valid-sops-stmt t s

```
assumes valid-sops-stmt-sb: \(\left\lceil i<l e n g t h ~ t s ; ~ t s!i=((s, t), i s, \vartheta, s b, \mathcal{D}, \mathcal{O}) ;\left(s^{\prime}, \mathrm{t}\right) \in\right.\) prog-configs sb】
\(\Longrightarrow\) valid-sops-stmt t's \({ }^{\prime}\)
assumes load-tmps-le: \(\llbracket \mathrm{i}<\) length ts; ts!i \(=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket\)
\(\Longrightarrow \forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{t}\)
assumes read-tmps-le: \(\llbracket \mathrm{i}<\) length ts; ts! \(\mathrm{i}=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket\)
\(\Longrightarrow \forall \mathrm{i} \in\) read-tmps sb. \(\mathrm{i}<\mathrm{t}\)
assumes store-sops-le: \(\llbracket \mathrm{i}<\) length ts; ts!i \(=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket\)
\(\Longrightarrow \forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t}\)
assumes write-sops-le: \(\llbracket \mathrm{i}<\) length ts; ts!i \(=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket\)
\(\Longrightarrow \forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb). \(\mathrm{i}<\mathrm{t}\)
assumes tmps-le: \(\llbracket \mathrm{i}<\) length ts; ts!i \(=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}) \rrbracket\)
\(\Longrightarrow \operatorname{dom} \vartheta \cup\) load-tmps is \(=\{\mathrm{i} . \mathrm{i}<\mathrm{t}\}\)
lemma (in separated-tmps)
tmps-le':
assumes i-bound: i < length ts
assumes ts-i: ts \(!i=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O})\)
shows \(\forall \mathrm{i} \in \operatorname{dom} \vartheta . \mathrm{i}<\mathrm{t}\)
using tmps-le [OF i-bound ts-i] by auto
lemma (in separated-tmps) separated-tmps-nth-update:
\(\llbracket i<\) length ts; valid-sops-stmt t s; \(\forall\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right) \in\) prog-configs sb. valid-sops-stmt \(\mathrm{t}^{\prime} \mathrm{s}^{\prime} ;\)
\(\forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{t} ; \forall \mathrm{i} \in\) read-tmps sb. \(\mathrm{i}<\mathrm{t}\);
\(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t} ; \forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb). \(\mathrm{i}<\mathrm{t}\); dom \(\vartheta \cup\) load-tmps is \(=\{\mathrm{i} . \mathrm{i}<\mathrm{t}\} \rrbracket\)
\(\Longrightarrow\)
separated-tmps \((\mathrm{ts}[\mathrm{i}:=((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O})])\)
apply (unfold-locales)
apply (force intro: valid-sops-stmt simp add: nth-list-update split: if-split-asm) apply (fastforce intro: valid-sops-stmt-sb simp add: nth-list-update split: if-split-asm)
apply (fastforce intro: load-tmps-le [rule-format] simp add: nth-list-update split: if-split-asm)
apply (fastforce intro: read-tmps-le [rule-format] simp add: nth-list-update split: if-split-asm)
apply (fastforce intro: store-sops-le [rule-format] simp add: nth-list-update split: if-split-asm)
apply (fastforce intro: write-sops-le [rule-format] simp add: nth-list-update split: if-split-asm)
apply (fastforce dest: tmps-le [rule-format] simp add: nth-list-update split: if-split-asm) done
lemma hd-prog-app-in-first: \(\bigwedge\) ys. Prog \(_{\text {sb }} p p^{\prime}\) is \(\in\) set xs \(\Longrightarrow\) hd-prog \(q\) (xs @ ys) \(=\) hd-prog q xs
by (induct xs) (auto split: memref.splits)
lemma hd-prog-app-in-eq: \(\bigwedge\) ys. \(\operatorname{Prog}_{\text {sb }} p p^{\prime}\) is \(\in\) set \(\mathrm{xs} \Longrightarrow\) hd-prog \(q\) xs \(=\) hd-prog x xs by (induct xs) (auto split: memref.splits)
lemma hd-prog-app-notin-first: \(\bigwedge y s . \forall p p^{\prime}\) is. Prog \(_{\text {sb }} p p^{\prime}\) is \(\notin\) set xs \(\Longrightarrow\) hd-prog \(q\) (xs @ ys) \(=\) hd-prog q ys
by (induct xs) (auto split: memref.splits)
lemma union-eq-subsetD: \(\mathrm{A} \cup \mathrm{B}=\mathrm{C} \Longrightarrow \mathrm{A} \cup \mathrm{B} \subseteq \mathrm{C} \wedge \mathrm{C} \subseteq \mathrm{A} \cup \mathrm{B}\)
by auto
lemma prog-step-preserves-separated-tmps:
assumes i-bound: \(\mathrm{i}<\) length ts
assumes ts-i: ts! \(\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O})\)
assumes prog-step: \(\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.\), is \()\)
assumes sep: separated-tmps ts
shows separated-tmps
\[
\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}, \mathrm{is} @ \mathrm{is}^{\prime}, \vartheta, \mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime} \text { is }\right], \mathcal{D}, \mathcal{O}\right)\right]\right)
\]
proof -
obtain s t where p : \(\mathrm{p}=(\mathrm{s}, \mathrm{t})\) by (cases p )
obtain \(s^{\prime} t^{\prime}\) where \(p^{\prime}: p^{\prime}=\left(s^{\prime}, t^{\prime}\right)\) by (cases \(\left.p^{\prime}\right)\)
note ts-i \(=\) ts-i \([\) simplified p\(]\)
note step \(=\) prog-step \([\) simplified \(p \mathrm{p}]\)
interpret separated-tmps ts by fact
have separated-tmps \(\left(\mathrm{ts}\left[\mathrm{i}:=\left(\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\right.\right.\), is @ \(\mathrm{is}^{\prime}, \vartheta\), \(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}}(\mathrm{s}, \mathrm{t})\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\) is \(\left.\left.\left.], \mathcal{D}, \mathcal{O}\right)\right]\right)\)
proof (rule separated-tmps-nth-update [OF i-bound])
from stmt-step-load-tmps-range [OF step] load-tmps-le [OF i-bound ts-i]
stmt-step-tmps-count-mono [OF step]
show \(\forall \mathrm{i} \in\) load-tmps (is @ is \({ }^{\prime}\) ). \(\mathrm{i}<\mathrm{t}^{\prime}\)
by (auto simp add: load-tmps-append)
next
from read-tmps-le [OF i-bound ts-i] stmt-step-tmps-count-mono [OF step]
show \(\forall \mathrm{i} \in\) read-tmps ( sb @ \(\left[\operatorname{Prog}_{\mathrm{sb}}(\mathrm{s}, \mathrm{t})\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\) is \(\rceil\) ). \(\mathrm{i}<\mathrm{t}^{\prime}\)
by (auto simp add: read-tmps-append)
next
from stmt-step-data-store-sops-range [OF step] stmt-step-tmps-count-mono [OF step] store-sops-le [OF i-bound ts-i] valid-sops-stmt [OF i-bound ts-i]
show \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops (is @ is \(\left.{ }^{\prime}\right)\) ). \(\mathrm{i}<\mathrm{t}^{\prime}\) by (fastforce simp add: store-sops-append)
next from stmt-step-tmps-count-mono [OF step] write-sops-le [OF i-bound ts-i]
show \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops (sb @ \(\left[\operatorname{Prog}_{\text {sb }}(\mathrm{s}, \mathrm{t})\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\) is \(\left.]\right)\) ). \(\mathrm{i}<\mathrm{t}^{\prime}\) by (fastforce simp add: write-sops-append)
next
from tmps-le [OF i-bound ts-i]
have dom \(\vartheta \cup\) load-tmps is \(=\{\mathrm{i} . \mathrm{i}<\mathrm{t}\}\) by simp
with stmt-step-load-tmps-range \({ }^{\prime}\) [OF step] stmt-step-tmps-count-mono [OF step]
show dom \(\vartheta \cup\) load-tmps (is @ is \(\left.{ }^{\prime}\right)=\left\{\mathrm{i} . \mathrm{i}<\mathrm{t}^{\prime}\right\}\)
apply (clarsimp simp add: load-tmps-append)
apply rule
apply (drule union-eq-subsetD)
apply fastforce
apply clarsimp
subgoal for x
apply (case-tac \(\mathrm{t} \leq \mathrm{x}\) )
apply simp
apply (subgoal-tac \(\mathrm{x}<\mathrm{t}\) )
apply fastforce
apply fastforce
done
done
next
from valid-sops-stmt-invariant [OF prog-step [simplified p p] valid-sops-stmt [OF
i-bound ts-i]]
show valid-sops-stmt t's'.
next
show \(\forall\left(s^{\prime}, t^{\prime}\right) \in \operatorname{prog}-c o n f i g s\left(s b @\left[\operatorname{Prog}_{s b}(\mathrm{~s}, \mathrm{t})\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\right.\) is \(\left.]\right)\). valid-sops-stmt t's'
proof -
\{
fix \(s_{1} t_{1}\)
assume cfgs: \(\left(\mathrm{s}_{1}, \mathrm{t}_{1}\right) \in \operatorname{prog}\)-configs \(\left(\mathrm{sb} @\left[\operatorname{Prog}_{\mathrm{sb}}(\mathrm{s}, \mathrm{t})\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)\right.\right.\) is \(\left.]\right)\)
have valid-sops-stmt \(\mathrm{t}_{1} \mathrm{~s}_{1}\)
proof -
from valid-sops-stmt [OF i-bound ts-i]
have valid-sops-stmt t s.
moreover
from valid-sops-stmt-invariant [OF prog-step [simplified p p \(]\) valid-sops-stmt [OF
i-bound ts-i]]
have valid-sops-stmt t's'.
moreover
note valid-sops-stmt-sb [OF i-bound ts-i]
ultimately
show ?thesis
using cfgs
by (auto simp add: prog-configs-append)
qed
\}
thus ?thesis
by auto
qed
qed
then
show ?thesis
by ( \(\operatorname{simp}\) add: p p )
qed
lemma flush-step-sb-subset:
assumes step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}\right)\)
shows set \(\mathrm{sb}^{\prime} \subseteq\) set sb
using step
apply (induct \(\mathrm{c} 1==(\mathrm{m}, \mathrm{sb}, \mathcal{O}) \mathrm{c} 2==\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}\right)\) arbitrary: \(\mathrm{msb} \mathcal{O} \mathrm{acq} \mathrm{m}^{\prime} \mathrm{sb}^{\prime} \mathcal{O}^{\prime} \mathrm{acq}\) rule: flush-step.induct)
apply auto

\section*{done}
lemma flush-step-preserves-separated-tmps:
assumes i-bound: \(\mathrm{i}<\) length ts
assumes ts-i: ts! \(1=(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
assumes flush-step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S} \mathcal{S}^{\prime}\right)\)
assumes sep: separated-tmps ts
shows separated-tmps \(\left(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\)
proof -
obtain st where \(\mathrm{p}: \mathrm{p}=(\mathrm{s}, \mathrm{t})\) by (cases p )
note ts-i \(=\) ts-i [simplified p ]
interpret separated-tmps ts by fact
have separated-tmps ( \(\left.\mathrm{ts}\left[\mathrm{i}:=\left((\mathrm{s}, \mathrm{t}), \mathrm{is}, \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}\right)\right]\right)\)
proof (rule separated-tmps-nth-update [OF i-bound])
from load-tmps-le [OF i-bound ts-i]
show \(\forall \mathrm{i} \in\) load-tmps is. \(\mathrm{i}<\mathrm{t}\).
next from flush-step-preserves-read-tmps [OF flush-step read-tmps-le [OF i-bound ts-i] ] show \(\forall \mathrm{i} \in\) read-tmps sb'. \(\mathrm{i}<\mathrm{t}\).
next
from store-sops-le [OF i-bound ts-i]
show \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t}\).
next
from flush-step-preserves-write-sops [OF flush-step write-sops-le [OF i-bound ts-i]] show \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb'). i \(<\mathrm{t}\).
next
from tmps-le [OF i-bound ts-i]
show dom \(\vartheta \cup\) load-tmps is \(=\{\mathrm{i} . \mathrm{i}<\mathrm{t}\}\)
by auto
next
from valid-sops-stmt [OF i-bound ts-i]
show valid-sops-stmt t s.
next
from valid-sops-stmt-sb [OF i-bound ts-i] flush-step-sb-subset [OF flush-step] show \(\forall\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right) \in\) prog-configs sb'. valid-sops-stmt \(\mathrm{t}^{\prime} \mathrm{s}^{\prime}\) by (auto dest!: prog-configs-mono)
qed
then
show ?thesis by (simp add: p )
qed
lemma sbh-step-preserves-store-sops-bound:
assumes step: (is, \(\vartheta\), sb,m, \(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes store-sops-le: \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t}\)
shows \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is'). \(\mathrm{i}<\mathrm{t}\)
using step store-sops-le
by cases auto
lemma sbh-step-preserves-write-sops-bound:
assumes step: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes store-sops-le: \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is). \(\mathrm{i}<\mathrm{t}\)
assumes write-sops-le: \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb). \(\mathrm{i}<\mathrm{t}\)
shows \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb'). i \(<\mathrm{t}\)
using step store-sops-le write-sops-le
by cases (auto simp add: write-sops-append)
lemma sbh-step-prog-configs-eq:
assumes step: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows prog-configs \(\mathrm{sb}^{\prime}=\) prog-configs sb
using step
apply (cases)
apply (auto simp add: prog-configs-append)
done
lemma sbh-step-preserves-tmps-bound':
assumes step: (is, \(\vartheta, \mathrm{sb}, \mathrm{m}, \mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\left(\right.\) is \(\left.^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
shows dom \(\vartheta \cup\) load-tmps is \(=\) dom \(\vartheta^{\prime} \cup\) load-tmps is \({ }^{\prime}\)
using step
apply cases
apply (auto simp add: read-tmps-append)
done
lemma sbh-step-preserves-separated-tmps:
assumes i-bound: i < length ts
assumes ts-i: ts \(!i=(p, i s, \vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})\)
assumes memop-step: (is, \(\vartheta\), sb, m, \(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\)
\[
\left(\text { is }{ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)
\]
assumes instr: separated-tmps ts
shows separated-tmps ( \(\left.\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\)
proof -
obtain s t where p : \(\mathrm{p}=(\mathrm{s}, \mathrm{t})\) by (cases p )
note ts-i \(=\) ts-i [simplified p ]
interpret separated-tmps ts by fact
have separated-tmps ( \(\left.\mathrm{ts}\left[\mathrm{i}:=\left((\mathrm{s}, \mathrm{t}), \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\)
proof (rule separated-tmps-nth-update [OF i-bound])
from sbh-step-preserves-load-tmps-bound [OF memop-step load-tmps-le [OF i-bound ts-i] \(]\)
show \(\forall \mathrm{i} \in\) load-tmps is \({ }^{\prime} . \mathrm{i}<\mathrm{t}\).
next
from sbh-step-preserves-read-tmps-bound [OF memop-step load-tmps-le [OF i-bound ts-i]
read-tmps-le [OF i-bound ts-i]]
show \(\forall \mathrm{i} \in\) read-tmps sb'. \(\mathrm{i}<\mathrm{t}\).
next
from sbh-step-preserves-store-sops-bound [OF memop-step store-sops-le [OF i-bound ts-i]]
show \(\forall \mathrm{i} \in \bigcup\) (fst ' store-sops is'). \(\mathrm{i}<\mathrm{t}\).
next
from sbh-step-preserves-write-sops-bound [OF memop-step store-sops-le [OF i-bound ts-i]
write-sops-le [OF i-bound ts-i]]
show \(\forall \mathrm{i} \in \bigcup\) (fst ' write-sops sb'). i \(<\mathrm{t}\).
next
from sbh-step-preserves-tmps-bound \({ }^{\prime}\) [OF memop-step] tmps-le [OF i-bound ts-i]
show dom \(\vartheta^{\prime} \cup\) load-tmps is \({ }^{\prime}=\{\mathrm{i} . \mathrm{i}<\mathrm{t}\}\)
by auto
next
from valid-sops-stmt [OF i-bound ts-i]
show valid-sops-stmt t s.
next
from valid-sops-stmt-sb [OF i-bound ts-i] sbh-step-prog-configs-eq [OF memop-step]
show \(\forall\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right) \in\) prog-configs \(\mathrm{sb}^{\prime}\). valid-sops-stmt \(\mathrm{t}^{\prime} \mathrm{s}^{\prime}\)
by auto
qed
then show ?thesis
by (simp add: \(p\) )
qed

\section*{definition}
valid-pimp ts \(\equiv\) separated-tmps ts
lemma prog-step-preserves-valid:
assumes i-bound: \(\mathrm{i}<\) length ts
assumes ts-i: ts! \(i=(p\), is, \(\vartheta\), sb::stmt-config store-buffer, \(\mathcal{D}, \mathcal{O}, \mathcal{R})\)
assumes prog-step: \(\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.\), is \()\)
assumes valid: valid-pimp ts
shows valid-pimp ( \(\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}^{\prime}\right.\right.\), is \(@ i s^{\prime}, \vartheta, \mathrm{sb} @\left[\operatorname{Prog}_{\text {sb }} \mathrm{p} \mathrm{p}^{\prime}\right.\) is \(\left.\left.\left.\rceil, \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]\right)\)
using prog-step-preserves-separated-tmps [OF i-bound ts-i prog-step] valid
by (auto simp add: valid-pimp-def)
lemma flush-step-preserves-valid:
assumes i-bound: \(\mathrm{i}<\) length ts
assumes ts-i: ts! \(\mathrm{i}=(\mathrm{p}, \mathrm{is}, \vartheta\), sb::stmt-config store-buffer, \(\mathcal{D}, \mathcal{O}, \mathcal{R})\)
assumes flush-step: \((\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)\)
assumes valid: valid-pimp ts
shows valid-pimp (ts \(\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\), is \(\left.\left.\left., \vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\)
using flush-step-preserves-separated-tmps [OF i-bound ts-i flush-step] valid
by (auto simp add: valid-pimp-def)
lemma sbh-step-preserves-valid:
assumes i-bound: i < length ts
assumes ts-i: ts! \(i=(p, i s, \vartheta\), sb::stmt-config store-buffer, \(\mathcal{D}, \mathcal{O}, \mathcal{R})\)
assumes memop-step: (is, \(\vartheta\), sb, m, \(\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{sbh}}\)
\[
\left(\text { is }{ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)
\]
assumes valid: valid-pimp ts
shows valid-pimp ( \(\left.\mathrm{ts}\left[\mathrm{i}:=\left(\mathrm{p}, \mathrm{is}^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)\)

\section*{using}
sbh-step-preserves-separated-tmps [OF i-bound ts-i memop-step] valid
by (auto simp add: valid-pimp-def)
lemma hd-prog-prog-configs: hd-prog p sb \(=\mathrm{p} \vee\) hd-prog p sb \(\in\) prog-configs sb by (induct sb) (auto split:memref.splits)
interpretation PIMP: xvalid-program-progress stmt-step \(\lambda(\mathrm{s}, \mathrm{t})\). valid-sops-stmt t s valid-pimp
proof
fix \(\vartheta \mathrm{p} \mathrm{p}^{\prime}\) is \({ }^{\prime}\)
assume step: \(\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.\), is \()\)
obtain s t where \(\mathrm{p}: \mathrm{p}=(\mathrm{s}, \mathrm{t})\)
by (cases p)
obtain \(s^{\prime} t^{\prime}\) where \(p^{\prime}: p^{\prime}=\left(s^{\prime}, t^{\prime}\right)\)
by (cases p)
from prog-step-progress [OF step [simplified p p \({ }^{\text {T] }}\) ]
show \(\mathrm{p}^{\prime} \neq \mathrm{p} \vee\) is \(^{\prime} \neq[]\)
by (simp add: \(\mathrm{p} \mathrm{p}^{\prime}\) )
next
fix \(\vartheta \mathrm{p} \mathrm{p}^{\prime}\) is \({ }^{\prime}\)
assume step: \(\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.\), is \()\)
and valid-stmt: \((\lambda(\mathrm{s}, \mathrm{t})\). valid-sops-stmt t s\() \mathrm{p}\)
obtain st where p: \(\mathrm{p}=(\mathrm{s}, \mathrm{t})\)
by (cases p)
obtain \(s^{\prime} t^{\prime}\) where \(p^{\prime}: p^{\prime}=\left(s^{\prime}, t^{\prime}\right)\)
by (cases p)
from valid-sops-stmt-invariant [OF step [simplified p p \(]\) valid-stmt [simplified \(p\), simplified]]
have valid-sops-stmt \(\mathrm{t}^{\prime} \mathrm{s}\) '.
then show \((\lambda(\mathrm{s}, \mathrm{t})\). valid-sops-stmt t s\() \mathrm{p}^{\prime}\) by (simp add: \(\left.\mathrm{p}{ }^{\prime}\right)\)
next
fix its p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}\)
assume i-bound: i < length ts
and ts- \(\mathrm{i}: \mathrm{ts}!\mathrm{i}=(\mathrm{p}\), is, \(\vartheta\), sb:: \((\) stmt \(\times\) nat \()\) memref list, \(\mathcal{D}, \mathcal{O}, \mathcal{R})\)
and valid: valid-pimp ts
from valid have separated-tmps ts by (simp add: valid-pimp-def)
then interpret separated-tmps ts .
obtain st where \(\mathrm{p}: \mathrm{p}=(\mathrm{s}, \mathrm{t})\)
by (cases p )
from valid-sops-stmt [OF i-bound ts-i [simplified p\(]\) ]
show ( \(\lambda(\mathrm{s}, \mathrm{t})\). valid-sops-stmt t s) p by (auto simp add: p )
next
fix its p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta\) sb
assume i-bound: i < length ts and ts-i: ts \(!\mathrm{i}=(\mathrm{p}\), is, \(\vartheta\), sb::(stmt \(\times\) nat \()\) memref list, \(\mathcal{D}, \mathcal{O}, \mathcal{R})\) and valid: valid-pimp ts
```

    from valid have separated-tmps ts
    by (simp add: valid-pimp-def)
    then interpret separated-tmps ts .
    obtain s t where p: p = (s,t)
        by (cases p)
    from hd-prog-prog-configs [of p sb] valid-sops-stmt [OF i-bound ts-i [simplified p]]
    valid-sops-stmt-sb [OF i-bound ts-i [simplified p]]
    show ( }\lambda(\textrm{s},\textrm{t}).\mathrm{ . valid-sops-stmt t s) (hd-prog p sb)
        by (auto simp add: p)
    next
fix i ts p is \mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb p' is'
assume i-bound: i < length ts
and ts-i: ts ! i = (p, is, \vartheta, sb, \mathcal{D},\mathcal{O},\mathcal{R})
and step: }\vartheta\vdash\textrm{p}\mp@subsup{->}{\textrm{s}}{}(\mp@subsup{\textrm{p}}{}{\prime},\mathrm{ is ')
and valid: valid-pimp ts
show distinct-load-tmps is '}^
load-tmps is'\cap load-tmps is ={} ^
load-tmps is'\cap read-tmps sb ={}
proof -
obtain s t where p: p=(s,t) by (cases p)
obtain s't' where p}\mp@subsup{p}{}{\prime}:\mp@subsup{p}{}{\prime}=(\mp@subsup{s}{}{\prime},\mp@subsup{t}{}{\prime})\mathrm{ by (cases p')
note ts-i = ts-i [simplified p]
note step = step [simplified p p}
from valid
interpret separated-tmps ts
by (simp add: valid-pimp-def)
from sbh-step-distinct-load-tmps-prog-step [OF step load-tmps-le [OF i-bound ts-i]
read-tmps-le [OF i-bound ts-i]]
show ?thesis .
qed
next
fix i ts p is }\mathcal{O}\mathcal{R}\mathcal{D}\vartheta sb p' is
assume i-bound: i < length ts
and ts-i: ts ! i = (p, is,\vartheta, sb,\mathcal{D},\mathcal{O},\mathcal{R})
and step: \vartheta\vdash p }\mp@subsup{->}{\textrm{s}}{}(\mp@subsup{\textrm{p}}{}{\prime},\mathrm{ is )
and valid: valid-pimp ts
show data-dependency-consistent-instrs (dom \vartheta load-tmps is) is '}^
load-tmps is'\cap\bigcup(fst'store-sops is)}={}
load-tmps is'\cap $fst' write-sops sb) = {}
proof -
    obtain s t where p: p=(s,t) by (cases p)
    obtain s't' where p': p
    note ts-i = ts-i [simplified p]
    note step = step [simplified p p}
    from valid
    interpret separated-tmps ts
        by (simp add: valid-pimp-def)
```
from sbh-valid-data-dependency-prog-step [OF step store-sops-le [OF i-bound ts-i] write-sops-le [OF i-bound ts-i] valid-sops-stmt [OF i-bound ts-i]] tmps-le [OF i-bound ts-i] show ?thesis by auto
qed
next
fix its p is \(\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}^{\prime} \mathrm{is}^{\prime}$
assume i-bound: i $<$ length ts
and ts-i: ts ! $\mathrm{i}=(\mathrm{p}$, is, $\vartheta, \mathrm{sb}, \mathcal{D}, \mathcal{O}, \mathcal{R})$
and step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.$, is')
and valid: valid-pimp ts
show load-tmps is' $\cap \operatorname{dom} \vartheta=\{ \}$
proof -
obtain st where $\mathrm{p}: \mathrm{p}=(\mathrm{s}, \mathrm{t})$ by (cases p )
obtain $\mathrm{s}^{\prime} \mathrm{t}^{\prime}$ where $\mathrm{p}^{\prime}: \mathrm{p}^{\prime}=\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)$ by (cases p )
note ts-i $=$ ts-i [simplified p ]
note step $=$ step [simplified p p]
from valid
interpret separated-tmps ts
by (simp add: valid-pimp-def)
from sbh-load-tmps-fresh-prog-step [OF step tmps-le' [OF i-bound ts-i]] show ?thesis .
qed
next
fix $\vartheta \mathrm{p} \mathrm{p}^{\prime}$ is
assume step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.$, is)
and valid: $(\lambda(\mathrm{s}, \mathrm{t})$. valid-sops-stmt t s) p
show $\forall$ sop $\in$ store-sops is. valid-sop sop
proof -
obtain st where $\mathrm{p}: \mathrm{p}=(\mathrm{s}, \mathrm{t})$ by (cases p )
obtain $\mathrm{s}^{\prime} \mathrm{t}^{\prime}$ where $\mathrm{p}^{\prime}: \mathrm{p}^{\prime}=\left(\mathrm{s}^{\prime}, \mathrm{t}^{\prime}\right)$ by (cases $\mathrm{p}^{\prime}$ )
note step $=$ step [simplified p p]
from sbh-valid-sops-prog-step [OF step valid [simplified p,simplified]]
show ?thesis .
qed
next
fix its p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb}^{\prime}$ is ${ }^{\prime}$
assume i-bound: $\mathrm{i}<$ length ts
and ts-i: ts ! $\mathrm{i}=(\mathrm{p}$, is, $\vartheta$, , sb::stmt-config store-buffer, $\mathcal{D}, \mathcal{O}, \mathcal{R})$
and step: $\vartheta \vdash \mathrm{p} \rightarrow_{\mathrm{s}}\left(\mathrm{p}^{\prime}\right.$, is $\left.{ }^{\prime}\right)$
and valid: valid-pimp ts
from prog-step-preserves-valid [OF i-bound ts-i step valid]
show valid-pimp (ts[i:=( $\mathrm{p}^{\prime}$, is @ is $\mathrm{s}^{\prime}, \vartheta$, sb @ $\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $\left.\left.\left.\left.{ }^{\prime}\right], \mathcal{D}, \mathcal{O}, \mathcal{R}\right)\right]\right)$.
next
fix itsp is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \mathrm{sb} \mathcal{S} \mathrm{mm}^{\prime} \mathrm{sb}^{\prime} \mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{S}^{\prime}$
assume i-bound: i < length ts
and ts-i: ts ! $\mathrm{i}=(\mathrm{p}, \mathrm{is}, \boldsymbol{\vartheta}$, sb::stmt-config store-buffer, $\mathcal{D}, \mathcal{O}, \mathcal{R})$
and step: $(\mathrm{m}, \mathrm{sb}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\mathrm{f}}\left(\mathrm{m}^{\prime}, \mathrm{sb}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)$
and valid: valid-pimp ts
thm flush-step-preserves-valid [OF ]
from flush-step-preserves-valid [OF i-bound ts-i step valid]
show valid-pimp $\left(\operatorname{ts}\left[\mathrm{i}:=\left(\mathrm{p}\right.\right.\right.$, is, $\left.\left.\left.\vartheta, \mathrm{sb}^{\prime}, \mathcal{D}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$.

## next

fix i ts p is $\mathcal{O} \mathcal{R} \mathcal{D} \vartheta \operatorname{sb} \mathcal{S} \mathrm{m}$ is $\mathcal{O}^{\prime} \mathcal{R}^{\prime} \mathcal{D}^{\prime} \vartheta^{\prime} \mathrm{sb}^{\prime} \mathcal{S}^{\prime} \mathrm{m}^{\prime}$
assume i-bound: i $<$ length ts and ts- $\mathrm{i}:$ ts $!\mathrm{i}=(\mathrm{p}$, is, $\vartheta$, sb::stmt-config store-buffer, $\mathcal{D}, \mathcal{O}, \mathcal{R})$
and step: (is, $\vartheta$, sb, m, $\mathcal{D}, \mathcal{O}, \mathcal{R}, \mathcal{S}) \rightarrow_{\text {sbh }}$

$$
\left(\text { is }{ }^{\prime}, \vartheta^{\prime}, \mathrm{sb}^{\prime}, \mathrm{m}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}, \mathcal{S}^{\prime}\right)
$$

and valid: valid-pimp ts
from sbh-step-preserves-valid [OF i-bound ts-i step valid]
show valid-pimp $\left(\operatorname{ts}\left[i:=\left(p\right.\right.\right.$, is $\left.\left.\left.^{\prime}, \vartheta^{\prime}, \operatorname{sb}^{\prime}, \mathcal{D}^{\prime}, \mathcal{O}^{\prime}, \mathcal{R}^{\prime}\right)\right]\right)$.
qed
thm PIMP .concurrent-direct-steps-simulates-store-buffer-history-step
thm PIMP.concurrent-direct-steps-simulates-store-buffer-history-steps
thm PIMP.concurrent-direct-steps-simulates-store-buffer-step
We can instantiate PIMP with the various memory models.interpretation direct:
computation direct-memop-step empty-storebuffer-step stmt-step $\lambda \mathrm{p}^{\prime} \mathrm{p}^{\prime}$ is sb . ().
interpretation virtual:
computation virtual-memop-step empty-storebuffer-step stmt-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb . ().
interpretation store-buffer:
computation sb-memop-step store-buffer-step stmt-step $\lambda p p^{\prime}$ is sb. sb .
interpretation store-buffer-history:
computation sbh-memop-step flush-step stmt-step $\lambda \mathrm{p} \mathrm{p}^{\prime}$ is sb . sb @ $\left[\operatorname{Prog}_{\mathrm{sb}} \mathrm{p} \mathrm{p}^{\prime}\right.$ is $]$.
abbreviation direct-pimp-step::
(stmt-config,unit,bool,owns,rels,shared) global-config $\Rightarrow$
(stmt-config,unit,bool,owns,rels,shared) global-config $\Rightarrow$ bool
$\left(-\Rightarrow_{\mathrm{dp}}-[60,60] 100\right)$
where
$\mathrm{c} \Rightarrow_{\mathrm{dp}} \mathrm{d} \equiv$ direct.concurrent-step c d
abbreviation direct-pimp-steps::
(stmt-config,unit,bool,owns,rels,shared) global-config $\Rightarrow$
(stmt-config,unit,bool,owns,rels,shared) global-config $\Rightarrow$ bool
$\left(-\Rightarrow \mathrm{dp}^{*}-[60,60] 100\right)$

## where

direct-pimp-steps $==$ direct-pimp-step ${ }^{〔} * *$
Execution exampleslemma Assign-Const-ex:
$([(($ Assign $\operatorname{True}(\operatorname{Tmp} \quad(\}, \lambda \vartheta . \quad$ a) $) \quad($ Const c$)(\lambda \vartheta . ~ A) \quad(\lambda \vartheta . \mathrm{L}) \quad(\lambda \vartheta . \mathrm{R}) \quad(\lambda \vartheta$ 。
$\mathrm{W}), \mathrm{t}),[], \vartheta,(), \mathcal{D}, \mathcal{O}, \mathcal{R})], \mathrm{m}, \mathcal{S}) \Rightarrow{ }_{\mathrm{dp}}{ }^{*}$
$\left([((\right.$ Skip, t$),[], \vartheta,()$, True, $\mathcal{O} \cup \mathrm{A}-\mathrm{R}$, Map.empty $\left.)], \mathrm{m}(\mathrm{a}:=\mathrm{c}), \mathcal{S} \oplus \mathrm{w}_{\mathrm{w}} \mathrm{R} \ominus_{\mathrm{A}} \mathrm{L}\right)$
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Program [where $\mathrm{i}=0$ ])
apply simp
apply simp
apply (rule Assign)
apply simp
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Memop [where $\mathrm{i}=0$ ])

```
apply simp
apply simp
apply (rule direct-memop-step.WriteVolatile)
apply simp
done
```


## lemma

```
\(([((\) Assign True \((\operatorname{Tmp}(\}, \lambda \vartheta . a))(\) Binop \((+)(\) Mem True \(x)(\) Mem True y) \()(\lambda \vartheta . A)(\lambda \vartheta\).
```

L) $(\lambda \vartheta . \mathrm{R})(\lambda \vartheta . \mathrm{W}), \mathrm{t}),[], \vartheta,(), \mathcal{D}, \mathcal{O}, \mathcal{R})], \mathrm{m}, \mathrm{S})$

```
=>dp*
([((Skip,t + 2),[],\vartheta(t\mapstom x, t + 1\mapstom y),(),True,\mathcal{O}\cup\textrm{A}-\textrm{R},Map.empty)],m(a := m x +
m y),S © }\mp@subsup{\textrm{w}}{\textrm{L}}{R}\mp@subsup{\ominus}{\textrm{A}}{\prime}L
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Program [where i=0])
apply simp
apply simp
apply (rule Assign)
apply simp
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Memop)
apply simp
apply simp
apply (rule direct-memop-step.Read )
apply simp
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Memop)
apply simp
apply simp
apply (rule direct-memop-step.Read)
apply simp
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Memop)
apply simp
apply simp
apply (rule direct-memop-step.WriteVolatile )
apply simp
done
```

```
lemma
```

lemma
assumes isTrue: isTrue c
assumes isTrue: isTrue c
shows
shows
([((Cond (Const c) (Assign True (Tmp ({}, \lambda\vartheta. a)) (Const c) (\lambda\vartheta. A) ( \lambda\vartheta. L) (\lambda\vartheta. R) ( \lambda\vartheta.
([((Cond (Const c) (Assign True (Tmp ({}, \lambda\vartheta. a)) (Const c) (\lambda\vartheta. A) ( \lambda\vartheta. L) (\lambda\vartheta. R) ( \lambda\vartheta.
W)) Skip,t) ,[],\vartheta,(),\mathcal{D},\mathcal{O},\mathcal{R})],m,\mathcal{S})}\mp@subsup{=>}{\textrm{dp}}{*
W)) Skip,t) ,[],\vartheta,(),\mathcal{D},\mathcal{O},\mathcal{R})],m,\mathcal{S})}\mp@subsup{=>}{\textrm{dp}}{*
([((Skip,t),[],\vartheta,(),True,\mathcal{O}\cup\textrm{A}-\textrm{R},Map.empty)],m(a := c),\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
([((Skip,t),[],\vartheta,(),True,\mathcal{O}\cup\textrm{A}-\textrm{R},Map.empty)],m(a := c),\mathcal{S}\mp@subsup{\oplus}{\textrm{w}}{}\textrm{R}\mp@subsup{\ominus}{\textrm{A}}{}\textrm{L})
apply (rule converse-rtranclp-into-rtranclp)
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Program [where i=0])

```
apply (rule direct.Program [where i=0])
```

```
apply simp
apply simp
apply (rule Cond)
apply simp
apply simp
apply (rule converse-rtranclp-into-rtranclp)
apply (rule direct.Program [where i=0])
apply simp
apply simp
apply (rule CondTrue)
apply simp
apply (simp add: isTrue)
apply simp
apply (rule Assign-Const-ex)
done
```

end

## References

1. Advanced Micro Devices (AMD), Inc. AMD64 Architecture Programmer's Manual: Volumes 1-3. September 2007. rev. 3.14.
2. Sarita V. Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. IEEE Computer, 29(12):66-76, 1996.
3. David Aspinall and Jaroslav Sevcík. Formalising Java's data race free guarantee. In Klaus Schneider and Jens Brandt, editors, TPHOLs, volume 4732, pages 22-37, 2007.
4. Clemens Ballarin. Locales and locale expressions in Isabelle/Isar. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, Types for Proofs and Programs, International Workshop, TYPES 2003, Torino, Italy, April 30 - May 4, 2003, Revised Selected Papers, volume 3085, pages 34-50. Springer, 2003.
5. Clemens Ballarin. Interpretation of locales in Isabelle: Theories and proof contexts. In Jonathan M. Borwein and William M. Farmer, editors, Mathematical Knowledge Management, 5th International Conference, MKM 2006, Wokingham, UK, August 11-12, 2006, Proceedings, volume 4108, pages 31-43. Springer, 2006.
6. Sebastian Burckhardt and Madanlal Musuvathi. Effective program verification for relaxed memory models. In CAV '08: Proceedings of the 20th international conference on Computer Aided Verification, pages 107-120, Berlin, Heidelberg, 2008. Springer-Verlag.
7. Geng Chen, Ernie Cohen, and Mikhail Kovalev. Store buffer reduction with MMUs. In Dimitra Giannakopoulou and Daniel Kroening, editors, Verified Software: Theories, Tools and Experiments, pages 117-132, Cham, 2014. Springer International Publishing.
8. Ernie Cohen and Bert Schirmer. From total store order to sequential consistency: A practical reduction theorem. In Matt Kaufmann, Lawrence Paulson, and Michael Norrish, editors, Interactive Theorem Proving (ITP 2010), volume 6172 of Lecture Notes in Computer Science, Edinburgh, UK, 2010. Springer.
9. Intel. Intel 64 architecture memory ordering white paper. SKU 318147-001, 2007.
10. Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual: Volumes 1-3b. 2009. rev. 29.
11. Tobias Nipkow. Winskel is (almost) right: Towards a mechanized semantics textbook. In V. Chandru and V. Vinay, editors, Foundations of Software Technology and Theoretical Computer Science, volume 1180, pages 180-192, 1996.
12. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. Isabelle/HOL: A Proof Assistant for Higher-Order Logic, volume 2283. Springer, 2002.
13. Jonas Oberhauser. A simpler reduction theorem for x 86 -tso. In Arie Gurfinkel and Sanjit A. Seshia, editors, Verified Software: Theories, Tools, and Experiments, pages 142-164, Cham, 2016. Springer International Publishing.
14. Scott Owens. Reasoning about the implementation of concurrency abstractions on x86-TSO. Technical report, University of Cambridge, 2009.
15. Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-TSO. In 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009), 2009.
16. Lawrence C. Paulson. Isabelle: A Generic Theorem Prover, volume 828. Springer, 1994.
17. Tom Ridge. Operational reasoning for concurrent Caml programs and weak memory models. In Klaus Schneider and Jens Brandt, editors, Theorem Proving in Higher Order Logics, 20th International Conference, TPHOLs 2007, Kaiserslautern, Germany, September 10-13, 2007, Proceedings, volume 4732, pages 278-293, 2007.
18. Jaroslav Sevcík and David Aspinall. On validity of program transformations in the Java memory model. In Jan Vitek, editor, ECOOP, volume 5142, pages 27-51, 2008.

[^0]:    * Work funded by the German Federal Ministry of Education and Research (BMBF) in the framework of the Verisoft XT project under grant 01 IS 07008.

[^1]:    ${ }^{3}$ Before 2008, Intel [9] and AMD [1] both put forward a weaker memory model in which writes to different memory addresses may be seen in different orders on different processors, but respecting causal ordering. However, current implementations satisfy the stronger conditions described in this report and are also compliant with the latest revisions of the Intel specifications [10]. According to Owens et al. [15] AMD is also planning a similar adaptation of their manuals.
    ${ }^{4}$ This atomicity isn't guaranteed for certain memory types, or for operations that cross a cache line.

[^2]:    ${ }^{5}$ Here we are sloppy with $t s$; strictly we would have to distinguish the thread configurations without the $\mathcal{R}$ component form the ones with the $\mathcal{R}$ component used for delayed releases

