Proving the Correctness of Disk Paxos in Isabelle/HOL

Mauro Jaskelioff       Stephan Merz

October 11, 2017

Abstract

Disk Paxos [GL00] is an algorithm for building arbitrary fault-tolerant distributed systems. The specification of Disk Paxos has been proved correct informally and tested using the TLC model checker, but up to now, it has never been fully formally verified. In this work we have formally verified its correctness using the Isabelle theorem prover and the HOL logic system [NPW02], showing that Isabelle is a practical tool for verifying properties of TLA+ specifications.

Contents

1 Introduction 2

2 The Disk Paxos Algorithm 3
   2.1 Informal description of the algorithm 4
   2.2 Disk Paxos and its TLA+ Specification 4

3 Translating from TLA+ to Isabelle/HOL 6
   3.1 Typed vs. Untyped 6
   3.2 Primed Variables 8
   3.3 Restructuring the specification 8

4 Structure of the Correctness Proof 9
   4.1 Going from Informal Proofs to Formal Proofs 10

5 Conclusion 11

A TLA+ correctness specification 12

B Disk Paxos Algorithm Specification 13
1 Introduction

Algorithms for fault-tolerant distributed systems were first introduced to implement critical systems. Nevertheless, what good is such an algorithm if it has a design error? We need some kind of guarantee that the algorithm does not have a faulty design. Formal verification of its specification is one such guarantee.

Disk Paxos [GL00] is an algorithm for building arbitrary fault-tolerant distributed systems, that, due to its complexity, is difficult to reason about. It has been proved correct informally, and tested using the TLC model checker [Lam02]. The informal proof is rigorous but, as it is always the case with large informal proofs, it is easy to overlook details. Thus, one of the motivations of this work is to see if such a rigorous proof can be formalized in a contemporary theorem prover.

In [Pac01] part of the correctness proof (invariance of $HIv1$ and $HIv3$) was verified using the theorem prover ACL2 [KMM00]. An implicit assumption of this formalization is that all sets are finite, thus overlooking the fact that there is a missing conjunct in the Disk Paxos invariant (see Section 4).

We set the goal of formally verifying Disk Paxos correctness using the theorem prover Isabelle/HOL [NPW02]. In this way, we could gain more confidence in the correctness of Disk Paxos design and, at the same time, learn to what extent can Isabelle be a useful tool for proving the correctness of distributed systems using a real world example.
In Section 2 we give a brief description of the algorithm and its specification. In Section 3 we describe the translation from TLA\(^+\) to Isabelle/HOL and the problems that this translation originated. In Section 4 we discuss how our formal proofs relate to the informal ones in [GL00], and in Section 5 we conclude. The entire specification and all formal proofs can be found in the Appendix.

2 The Disk Paxos Algorithm

Disk Paxos is a variant of the classic Paxos algorithm [Lam98] for the implementation of arbitrary fault-tolerant systems with a network of processors and disks. It maintains consistency in the event of any number of non-Byzantine failures. This means that a processor may fail completely or pause for arbitrary long periods and then restart, remembering only that it has failed. A disk may become inaccessible to some or all processors, but it may not be corrupted. We say that a system is stable if all processes are either non-faulty or have failed completely (i.e. there are no new failed processes). Disk Paxos guarantees progress if the system is stable and there is at least one non-faulty processor that can read and write a majority of the disks. Consequently, the fundamental difference between Classic Paxos and Disk Paxos is that the former achieves redundancy by replicating processes while the latter replicates disks. Since disks are usually cheaper than processors, it is possible to obtain more redundancy at a lower cost.

Disk Paxos uses the state machine approach to solve the problem of implementing an arbitrary distributed system. The state machine approach [Sch90] is a general method that reduces this problem to solving a consensus problem. The distributed system is designed as a deterministic state machine that executes a sequence of commands, and a consensus algorithm ensures that, for each \(n\), all processors agree on the \(n^{th}\) command. Hence, each processor \(p\) starts with an input value (a command), and it may output a value (the command to be executed). The problem is solved if all processors eventually output the same value and this value was a value of \(\text{input}[p]\) for some \(p\) (under certain assumptions, in our case that there exists at least one non-faulty processor that can write and read a majority of disks).

Progress of Disk Paxos relies on progress of a leader-election algorithm. It is easy to make a leader-election algorithm if the system stable, but very hard to devise one that works correctly even if the system is unstable. We are requiring stability to ensure progress, but actually only a weaker requirement is needed: progress of the underlying leader-election algorithm. Disk Paxos ensures that all outputs (if any) will be the same even if the leader-election algorithm fails.
2.1 Informal description of the algorithm.

The consensus algorithm of Disk Paxos is called Disk Synod. In it, each processor has an assigned block on each disk. Also it has a local memory that contains its current block (called the dblock), and other state variables (see figure 1). When a process \( p \) starts it contains an input value \( \text{input}[p] \) that will not be modified, except possibly when recovering from a failure.

Disk Synod is structured in two phases, plus one more phase for recovering from failures. In each phase, a processor writes its own block and reads each other processor’s block, on a majority of the disks. The idea is to execute ballots to determine:

**Phase 1:** whether a processor \( p \) can choose its own input value \( \text{input}[p] \) or must choose some other value. When this phase finishes a value \( v \) is chosen.

**Phase 2:** whether it can commit \( v \). When this phase is complete the process has committed value \( v \) and can output it (using variable \( \text{outpt} \)).

In either phase, a processor aborts its ballot if it learns that another processor has begun a higher-numbered ballot. The third phase (Phase 0) is for starting the algorithm or recovering from a failure.

In each block, processors maintain three values:

- \( \text{mbal} \) The current ballot number.
- \( \text{bal} \) The largest ballot number for which the processor entered phase 2.
- \( \text{inp} \) The value the processor tried to commit in ballot number \( \text{bal} \).

For a complete description of the algorithm, see [GL00].

2.2 Disk Paxos and its TLA\(^+\) Specification

The specification of Disk Paxos is written in the TLA\(^+\) specification language [Lam02]. As it is usual with TLA\(^+\), the specification is organized into modules.

The specification of consensus is given in module Synod, which can be found in appendix A. In it there are only two variables: \( \text{input} \) and \( \text{output} \). To formalize the property stating that all processors should choose the same value and that this value should have been an input of a processor, we need variables that represent all past inputs and the value chosen as result. Consequently, an Inner submodule is introduced, which adds two variables: \( \text{allInput} \) and \( \text{chosen} \). Our Synod module will be obtained by existentially quantifying these variables of the Inner module.

The specification of the algorithm is given in the HDiskSynod module. Hence, what we are going to prove is that the (translation to Isabelle/HOL
of the) Inner module is implied by the (translation to Isabelle/HOL of the) algorithm module HDiskSynod.

More concretely we have that the specification of the algorithm is:

\[ \text{HDiskSynodSpec} \triangleq \text{HInit} \land \Box \lbrack \text{HNext} \rbrack_{\langle \text{vars}, \text{chosen}, \text{allInput} \rangle} \]

where \( \text{HInit} \) describes the initial state of the algorithm and \( \text{HNext} \) is the action that models all of its state transitions. The variable \( \text{vars} \) is the tuple of all variables used in the algorithm.

Analogously, we have the specification of the Inner module:

\[ \text{ISpec} \triangleq \text{Init} \land \Box \lbrack \text{INext} \rbrack_{\langle \text{input}, \text{output}, \text{chosen}, \text{allInput} \rangle} \]

We define \( \text{ivars} = \langle \text{input}, \text{output}, \text{chosen}, \text{allInput} \rangle \). In order to prove that HDiskSynodSpec implies ISpec, we follow the structure of the proof given by Gafni and Lamport. We must prove two theorems:

THEOREM R1 \( \text{HInit} \Rightarrow \text{HInit} \)

THEOREM R2 \( \text{HInit} \land \Box \lbrack \text{HNext} \rbrack_{\langle \text{vars}, \text{chosen}, \text{allInput} \rangle} \Rightarrow \Box \lbrack \text{INext} \rbrack_{\text{ivars}} \)

The proof of R1 is trivial. For R2, we use TLA proof rules [Lam02] that show that to prove R2, it suffices to find a state predicate \( \text{HInv} \) for which we can prove:

THEOREM R2a \( \text{HInit} \land \Box \lbrack \text{HNext} \rbrack_{\langle \text{vars}, \text{chosen}, \text{allInput} \rangle} \Rightarrow \Box \lbrack \text{HInv} \rbrack \)

THEOREM R2b \( \text{HInv} \land \text{HInv}' \land \Box \lbrack \text{HNext} \rbrack \Rightarrow \text{INext} \lor (\text{UNCHANGED ivars}) \)

A predicate satisfying \( \text{HInv} \) is said to be an invariant of HDiskSynodSpec. To prove R2a, we make \( \text{HInv} \) strong enough to satisfy:
THEOREM I1 \( HInit \Rightarrow HInv \)

THEOREM I2 \( HInv \land HNext \Rightarrow HInv' \)

Again, we have TLA proof rules that say that I1 and I2 imply R2a. In summary, R2b, I1, and I2 together imply HDiskSynodSpec \( \Rightarrow ISpec \).

Finding a predicate \( HInv \) that is strong enough can be rather difficult. Fortunately, Gafni and Lamport give this predicate. In their paper, they present \( HInv \) as a conjunction of 6 predicates \( HInv_1, \ldots, HInv_6 \), where \( HInv_1 \) is a simple “type invariant” and the higher-numbered predicates add more and more information. The proof is structured such that the preservation of \( HInv_i \) by the algorithm’s next-state relation relies on all \( HInv_j \) (for \( j \leq i \)) being true in the state before the transition. In our proofs we are going to use exactly the same tactic.

Before starting our proofs we have to translate all the specification and theorems above into the formal language of Isabelle/HOL.

### 3 Translating from TLA\(^+\) to Isabelle/HOL

The translation from TLA\(^+\) to Isabelle/HOL is pretty straightforward as Isabelle/HOL has equivalent counterparts for most of the constructs in TLA\(^+\) (some representative examples are shown in table 1). Nevertheless, there are some semantic discrepancies. In the following, we discuss these differences, some of the options that one has when dealing with them, and the reasons for our choices\(^1\).

#### 3.1 Typed vs. Untyped

TLA\(^+\) is an untyped formalism. However, TLA\(^+\) specifications usually have some type information, usually in the form of set membership or set inclusion. When translating these specifications to Isabelle/HOL, which is a typed formalism, one has to invent types that represent these sets of values.

\(^1\)There is no point in using the existing TLA encoding in Isabelle. Since the encoding is also based on HOL and we only prove safety, we would have gained nothing.
TLA⁺:

\[
\begin{align*}
\text{CONSTANT } \text{Inputs} & \\
\text{NotAnInput} & \triangleq \text{CHOOSE } c : c \notin \text{Inputs} \\
\text{DiskBlock} & \triangleq \left\{ \begin{array}{l}
\text{mbal} : (\text{UNION Ballot}(p) : p \in \text{Proc}) \cup \{0\}, \\
\text{bal} : (\text{UNION Ballot}(p) : p \in \text{Proc}) \cup \{0\}, \\
\text{inp} : \text{Inputs} \cup \{\text{NotAnInput}\} 
\end{array} \right\}
\end{align*}
\]

Isabelle/HOL:

\textbf{typedef} \text{InputsOrNi}

\textbf{consts}

\text{Inputs} :: \text{InputsOrNi set}
\text{NotAnInput} :: \text{InputsOrNi}

\textbf{axioms}

\text{NotAnInput}: \text{NotAnInput} \notin \text{Inputs}
\text{InputsOrNi}: (\text{UNIV} :: \text{InputsOrNi set}) = \text{Inputs} \cup \{\text{NotAnInput}\}

\textbf{record}

\text{DiskBlock} =
\text{mbal} :: \text{nat}
\text{bal} :: \text{nat}
\text{inp} :: \text{InputsOrNi}

Figure 2: Untyped TLA⁺ vs. Typed Isabelle/HOL

This process is not automatic and requires some thought to find the right abstractions. Furthermore, simple types may not be expressive enough to represent exactly the set in the specification. In some cases, these sets could be modelled by algebraic datatypes, but this would make the specification more complex and less directly related to the original one. In this work, we have chosen to stick to simple types and add additional axioms to account for their lack of expressiveness.

For example, see figure 2. The type \text{InputsOrNi} models the members of the set \text{Inputs}, and the element \text{NotAnInput}. We record the fact that \text{NotAnInput} is not in \text{Inputs}, with axiom \text{NotAnInput}. Now, looking at the type of the \text{inp} field of the \text{DiskBlock} record in the TLA⁺ specification, we see that its type should be \text{InputsOrNi}. However, this is not the same type as \text{Inputs} \cup \{\text{NotAnInput}\}, as nothing prevents the \text{InputsOrNi} type from having more values. Consequently, we add the axiom \text{InputsOrNi} to establish that the only values of this type are the ones in \text{Inputs} and \text{NotAnInput}.

This example shows the kind of difficulties that can arise when trans-
TLA⁺:

\[ \text{Phase1or2Write}(p, d) \triangleq \]
\[ \land \ phase[p] \in \{1, 2\} \]
\[ \land \ disk'[d] = (\text{disk} \setminus \text{except } !d) \cup \{\text{dblock}[p]\} \]
\[ \land \ disksWritten'[p] = (\text{disksWritten} \setminus \text{except } !p) \cup \{d\} \]
\[ \land \ \text{UNCHANGED}(\text{input}, \text{output}, \text{phase}, \text{dblock}, \text{blocksRead}) \]

Isabelle/HOL:

\[ \text{Phase1or2Write} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool} \]
\[ \text{Phase1or2Write } s \ s' \ p \ d \equiv \]
\[ \land \ \text{disk}' = (\text{disk} \setminus \text{except } d \cup \text{dblock}[p]) \]
\[ \land \ \text{disksWritten}' = (\text{disksWritten} \setminus \text{except } p) \cup \{d\}) \]
\[ \land \ \text{UNCHANGED}(\text{input}, \text{output}, \text{phase}, \text{dblock}, \text{blocksRead}) \]

Figure 3: Translation of an action

3.2 Primed Variables

In TLA⁺, to denote the value of a variable in the state resulting from an action, there exists the concept of primed variables. In Isabelle/HOL, there is no built-in notion of state, so we define the state as a record of the variables involved. Instead of defining a "priming" operator, we just define actions as predicates that take any two states, intended to be the previous state and the next state. In this way, \( P s s' \) will be true iff executing an action \( P \) in the \( s \) state could result in the \( s' \) state. In figure 3 we can see how the action \( \text{Phase1or2Write} \) is expressed in TLA⁺ and in Isabelle/HOL.

3.3 Restructuring the specification

There are some minor changes that can be made to specifications to make the proofs in Isabelle easier.

One such change is the elimination of \texttt{LET} constructs by making them global definitions. This has two benefits, as it makes it possible to:

- Formulate lemmas that use these definitions.
- Selectively unfold these definitions in proofs, instead of adding \texttt{Let-def} to Isabelle’s simplifier, which unfolds all “let” constructs.
Another change that makes proofs easier is to break down actions into simpler actions (e.g. giving separate definitions for subactions corresponding to disjuncts). By making actions smaller, Isabelle has to deal with smaller formulas when we feed the prover with such an action.

For example, \texttt{Phase1or2Read} is mainly a big if-then-else. We break it down into two simpler actions:

\[ \texttt{Phase1or2Read} \triangleq \texttt{Phase1or2ReadThen} \lor \texttt{Phase1or2ReadElse} \]

In \texttt{Phase1or2ReadThen} the condition of the if-then-else is present as a state formula (i.e. it is an enabling condition) while in \texttt{Phase1or2ReadElse} we add the negation of this condition.

Another example is \texttt{HInv2}, which we break down into:

\[ \texttt{HInv2} \triangleq \texttt{Inv2a} \land \texttt{Inv2b} \land \texttt{Inv2c} \]

Not only we break it down into three conjuncts but, since these conjuncts are quantifications over some predicate, we give a separate definition for this predicate. For example for \texttt{Inv2a}, and after translating to Isabelle/HOL, instead of writing:

\[ \texttt{Inv2a} \equiv \forall p. \forall bk \in \text{blocksOf s p} \ldots \]

we write:

\[ \texttt{Inv2a-innermost} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlock} \Rightarrow \text{bool} \]
\[ \texttt{Inv2a-innermost} s p bk \equiv \ldots \]

\[ \texttt{Inv2a-inner} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \]
\[ \texttt{Inv2a-inner} s p \equiv \forall bk \in \text{blocksOf s p}. \text{Inv2a-innermost} s p bk \]

\[ \texttt{Inv2a} :: \text{state} \Rightarrow \text{bool} \]
\[ \texttt{Inv2a} s \equiv \forall p. \text{Inv2a-inner} s p \]

Now we can express that we want to obtain the fact

\[ \texttt{Inv2a-innermost} s q \ (\text{dblock s q}) \]

explicitly stating that we are interested in predicate \texttt{Inv2a}, but only for some process \texttt{q} and block (\texttt{dblock s q}).

\section{Structure of the Correctness Proof}

In [GL00], a specification of correctness and a specification of the algorithm are given. Then, it is proved that the specification of the algorithm implies the specification of correctness. We will do the same for the translated specifications, maintaining the names of theorems and lemmas. It should be noted that only safety properties are given and proved.
4.1 Going from Informal Proofs to Formal Proofs

There are informal proofs for invariants $HInv_3$-$HInv_6$ and for theorem $R2b$ in [GL00]. These informal proofs are written in the structured-proof style that Lamport advocates [Lam95], and are rigorous and quite detailed. We based our formal proofs on these informal proofs, but in many cases a higher level of detail was needed. In some cases, the steps where too big for Isabelle to solve them automatically, and intermediate steps had to be proved first; in other cases, some of the facts relevant to the proofs were omitted in the informal proofs.

As an example of these omissions, the invariant should state that the set $allRdBlks$ is finite. This is needed to choose a block with a maximum ballot number in action $EndPhase_1$. Interestingly, this omission cannot be detected with finite-state model checking. As another example, it was omitted that it is necessary to assume that $HInv_4$ and $HInv_5$ hold in the previous state to prove lemma $I_2$.

Although our proofs were based on the informal ones, the high-level structure was often different. When proving that a predicate $I$ was an invariant of $Next$, we preferred proving the invariance of $I$ for each action, rather than a big theorem proving the invariance of $I$ for the $Next$ action. As a consequence, a proof for some actions was often similar to the proof of some other action. For example, the proof of the invariance of $HInv_3$ for the $EndPhase_0$ and $Fail$ actions is almost the same. This means we could have made only one proof for the two actions, shortening the length of the complete proof. Nevertheless, proving each action separately could be tackled more easily by Isabelle, as it had fewer facts to deal with, and proving a new lemma was often a simple matter of copy, paste and renaming of definitions. Once the invariance of a given predicate has been proved for each simple action, proving it for the $Next$ action is easy since the $Next$ action is a disjunction of all actions.

The informal proofs start working with $Next$, and then do a case split in which each case implies some action. The structure of the formal proof was copied from the informal one, in the parts where the latter focused on a particular action. This structure could be easily maintained since we used Isabelle’s Isar proof language [Wen02, Nip03], a language for writing human-readable structured proofs.

Lamport’s use of a hierarchical scheme for naming subformulas of a formula would have been very useful, as we have to repeatedly extract subfacts from facts. These proofs were always solved automatically by the auto Isabelle tactic, but made the proofs longer and harder to understand. Automatic naming of subformulas of Isabelle definitions would be a very practical feature.
5 Conclusion

We formally verified the correctness of the Disk Paxos specification in Isabelle/HOL. We found some omissions in the informal proofs, including one that could not have been detected with finite-state model checking, but no outright errors. This formal proof gives us a greater confidence in the correctness of the algorithm.

This work was done in little more than two months, including learning Isabelle from scratch. Consequently, this work can be taken as evidence that formal verification of fault-tolerant distributed algorithms may be feasible for software verification in an industrial context.

Isabelle proved to be a very helpful tool for this kind of verification, although it would have been useful to have Lamport’s naming of subfacts to make proofs shorter and easier to write.

References


A TLA\(^{+}\) correctness specification

```
--- MODULE Synod ---

EXTENDS Naturals

CONSTANT N, Inputs

ASSUME (N ∈ Nat) ∧ (N > 0)

Proc ≜ 1..N

NotAnInput ≜ \text{choose } c : c \notin Inputs

VARIABLES inputs, output

--- MODULE Inner ---

VARIABLES allInput, chosen

IInit ≜ \land input ∈ [Proc → Inputs]
\land output = [p ∈ Proc \mapsto NotAnInput]
\land chosen = NotAnInput
\land allInput = input[p] : p ∈ Proc

IChoose(p) ≜ \land output[p] = NotAnInput
\land IF chosen = NotAnInput

\land IF \exists ip ∈ allInput : \land chosen' = ip
\land \land output' = [output \setminus ![p] = ip]
\land \land unchanged \land \land unchanged \langle input, allInput \rangle

IFail(p) ≜ \land output' = [output \setminus ![p] = NotAnInput]
\land \exists ip ∈ Inputs : \land input' = [input \setminus ![p] = ip]
\land \land allInput' = allInput \cup \{ip\}

INext ≜ \exists p ∈ Proc : IChoose(p) ∨ IFail(p)

ISpec ≜ IInit ∧ □[INext](input, output, chosen, allInput)

IS(chosen, allInput) ≜ \text{instance Inner}

SynodSpec ≜ \exists \text{chosen, allInput : } IS(chosen, allInput) \land ISpec
```

12
B Disk Paxos Algorithm Specification

theory DiskPaxos-Model imports Main begin

This is the specification of the Disk Synod algorithm.

typedeccl InputsOrNi

typedeccl Disk

typedeccl Proc

axiomatization
Inputs :: InputsOrNi set and
NotAnInput :: InputsOrNi and
Ballot :: Proc ⇒ nat set and
IsMajority :: Disk set ⇒ bool

where
NotAnInput: NotAnInput /∈ Inputs and
InputsOrNi: (UNIV :: InputsOrNi set) = Inputs ∪ {NotAnInput} and
Ballot-nzero: ∀ p. 0 /∈ Ballot p and
Ballot-disj: ∀ p q. p ≠ q → (Ballot p) ∩ (Ballot q) = {} and
Disk-isMajority: IsMajority(UNIV) and
majorities-intersect:
∀ S T. IsMajority(S) ∧ IsMajority(T) → S ∩ T ≠ {}

lemma ballots-not-zero [simp]:
b ∈ Ballot p =⇒ 0 < b
proof (rule ccontr)
assume b: b ∈ Ballot p
and contr: ¬ (0 < b)
from Ballot-nzero
have 0 /∈ Ballot p ..
with b contr
show False
  by auto
qed

lemma majority-nonempty [simp]: IsMajority(S) ⇒ S ≠ {}
proof(auto)
from majorities-intersect
have IsMajority({}) ∧ IsMajority({}) → {} ∩ {} ≠ {} by auto
thus IsMajority {} → False by auto
qed

definition AllBallots :: nat set
where AllBallots = (UN p. Ballot p)

record DiskBlock =
\textbf{definition} \textit{InitDB} :: DiskBlock \\
\textbf{where} \textit{InitDB} = (\{ mbal = 0, bal = 0, inp = NotAnInput \})

\textbf{record} \\
\textit{BlockProc} = \\
\textit{block} :: DiskBlock \\
\textit{proc} :: Proc

\textbf{record} \\
\textit{state} = \\
\textit{inpt} :: Proc \Rightarrow InputsOrNi \\
\textit{outpt} :: Proc \Rightarrow InputsOrNi \\
\textit{disk} :: Disk \Rightarrow Proc \Rightarrow DiskBlock \\
\textit{dblock} :: Proc \Rightarrow DiskBlock \\
\textit{phase} :: Proc \Rightarrow nat \\
\textit{disksWritten} :: Proc \Rightarrow Disk \\
\textit{blocksRead} :: Proc \Rightarrow Disk \Rightarrow BlockProc set

\textit{allInput} :: InputsOrNi set \\
\textit{chosen} :: InputsOrNi

\textbf{definition} \textit{hasRead} :: state \Rightarrow Proc \Rightarrow Disk \Rightarrow Proc \Rightarrow bool \\
\textbf{where} \textit{hasRead} s p d q = (\exists br \in \textit{blocksRead} s p d. \textit{proc} br = q)

\textbf{definition} \textit{allRdBlks} :: state \Rightarrow Proc \Rightarrow BlockProc set \\
\textbf{where} \textit{allRdBlks} s p = (\textit{UN} d. \textit{blocksRead} s p d)

\textbf{definition} \textit{allBlocksRead} :: state \Rightarrow Proc \Rightarrow DiskBlock set \\
\textbf{where} \textit{allBlocksRead} s p = \textit{block} ' (\textit{allRdBlks} s p)

\textbf{definition} \textit{Init} :: state \Rightarrow bool \\
\textbf{where} \\
\textit{Init} s = \\
\textit{(range \ (\textit{inpt} s) \subseteq \textit{Inputs}} \\
\& \textit{outpt} s = (\lambda p. \textit{NotAnInput}) \\
\& \textit{disk} s = (\lambda d p. \textit{InitDB}) \\
\& \textit{phase} s = (\lambda p. 0) \\
\& \textit{dblock} s = (\lambda p. \textit{InitDB}) \\
\& \textit{disksWritten} s = (\lambda p. \{\}) \\
\& \textit{blocksRead} s = (\lambda p d. \{\})

\textbf{definition} \textit{InitializePhase} :: state \Rightarrow state \Rightarrow Proc \Rightarrow bool \\
\textbf{where} \\
\textit{InitializePhase} s s' p =
(\text{disksWritten } s') = (\text{disksWritten } s)(p := \{\}) \\
\& \text{blocksRead } s' = (\text{blocksRead } s)(p := (\lambda d. \{\}))}

\textbf{definition} \textbf{StartBallot :: state ⇒ state ⇒ Proc ⇒ bool where}
\textbf{StartBallot} s s' p =
\langle \text{phase } s \in \{1, 2\} \rangle \\
\& \text{phase } s' = (\text{phase } s)(p := 1) \\
\& (\exists b \in \text{Ballot } p. \\
\text{mbal}(\text{dblock } s p) < b \\
\& \text{dblock } s' = (\text{dblock } s)(p := (\text{dblock } s p)(\text{mbal} := b \{\})) \\
\& \text{InitializePhase } s s' p \\
\& \text{inpt } s' = \text{inpt } s \& \text{outpt } s' = \text{outpt } s \& \text{disk } s' = \text{disk } s)

\textbf{definition} \textbf{Phase1or2Write :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ bool where}
\textbf{Phase1or2Write} s s' p d =
\langle \text{phase } s \in \{1, 2\} \rangle \\
\& \text{disk } s' = (\text{disk } s)(d := (\text{disk } s d)(p := \text{dblock } s p)) \\
\& \text{disksWritten } s' = (\text{disksWritten } s)(p := (\text{disksWritten } s p) \cup \{d\}) \\
\& \text{inpt } s' = \text{inpt } s \& \text{outpt } s' = \text{outpt } s \\
\& \text{phase } s' = \text{phase } s \& \text{dblock } s' = \text{dblock } s \\
\& \text{blocksRead } s' = \text{blocksRead } s)

\textbf{definition} \textbf{Phase1or2ReadThen :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool where}
\textbf{Phase1or2ReadThen} s s' p d q =
\langle d \in \text{disksWritten } s p \rangle \\
\& \text{mbal}(\text{disk } s d q) < \text{mbal}(\text{dblock } s p) \\
\& \text{blocksRead } s' = (\text{blocksRead } s)(p := (\text{blocksRead } s p)(d := \\
\text{blocksRead } s p d)(p := (\text{blocksRead } s p d) \cup \{d \}| \text{block} = \text{disk } s d q, \\
\text{proc} = q \{\})} \\
\& \text{inpt } s' = \text{inpt } s \& \text{outpt } s' = \text{outpt } s \\
\& \text{disk } s' = \text{disk } s \& \text{phase } s' = \text{phase } s \\
\& \text{dblock } s' = \text{dblock } s \& \text{disksWritten } s' = \text{disksWritten } s)

\textbf{definition} \textbf{Phase1or2ReadElse :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool where}
\textbf{Phase1or2ReadElse} s s' p d q =
\langle d \in \text{disksWritten } s p \rangle \\
\& \text{StartBallot } s s' p)

\textbf{definition} \textbf{Phase1or2Read :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool where}
\textbf{Phase1or2Read} s s' p d q =
\langle \text{Phase1or2ReadThen } s s' p d q \rangle \\
\lor \text{Phase1or2ReadElse } s s' p d q)

\textbf{definition} \textbf{blocksSeen :: state ⇒ Proc ⇒ DiskBlock set}
where \( \text{blocksSeen} \ s \ p = \text{allBlocksRead} \ s \ p \cup \{ \text{dblock} \ s \ p \} \)

definition \( \text{nonInitBlks} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlock set} \)
where \( \text{nonInitBlks} \ s \ p = \{ bs . bs \in \text{blocksSeen} \ s \ p \land \text{inp} \ bs \in \text{Inputs} \} \)

definition \( \text{maxBlk} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{DiskBlock} \)
where \( \text{maxBlk} \ s \ p = (\text{SOME} \ b . b \in \text{nonInitBlks} \ s \ p \land (\forall c \in \text{nonInitBlks} \ s \ p. \text{bal} \ c \leq \text{bal} \ b)) \)

definition \( \text{EndPhase1} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)
where \( \text{EndPhase1} \ s \ s' \ p = \)
\( (\text{IsMajority} \ \{ d . d \in \text{disksWritten} \ s \ p \land (\forall q \in \text{UNIV} - \{ p \}. \text{hasRead} \ s \ p \ d \ q) \}) \)
\( \land \ \text{phase} \ s \ p = 1 \)
\( \land \ \text{dblock} \ s' = (\text{dblock} \ s) \ (p := \text{dblock} \ s \ p) \)
\( (\\{ \ \text{bal} := \text{mbal}(\text{dblock} \ s \ p), \text{inp} := \)
\( (\text{if} \ \text{nonInitBlks} \ s \ p = \{} \)\)
\( \text{then} \ \text{inpt} \ s \ p \)
\( \text{else} \ \text{inp} \ (\text{maxBlk} \ s \ p)) \)
\( \} \)
\( \land \ \text{outpt} \ s' = \text{outpt} \ s \)
\( \land \ \text{phase} \ s' = (\text{phase} \ s \ (p := \text{phase} \ s \ p + 1)) \)
\( \land \ \text{InitializePhase} \ s \ s' \ p \)
\( \land \ \text{inpt} \ s' = \text{inpt} \ s \land \text{disk} \ s' = \text{disk} \ s \)

definition \( \text{EndPhase2} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)
where \( \text{EndPhase2} \ s \ s' \ p = \)
\( (\text{IsMajority} \ \{ d . d \in \text{disksWritten} \ s \ p \land (\forall q \in \text{UNIV} - \{ p \}. \text{hasRead} \ s \ p \ d \ q) \}) \)
\( \land \ \text{phase} \ s \ p = 2 \)
\( \land \ \text{outpt} \ s' = (\text{outpt} \ s) \ (p := \text{inp} \ (\text{dblock} \ s \ p)) \)
\( \land \ \text{dblock} \ s' = \text{dblock} \ s \)
\( \land \ \text{phase} \ s' = (\text{phase} \ s \ (p := \text{phase} \ s \ p + 1)) \)
\( \land \ \text{InitializePhase} \ s \ s' \ p \)
\( \land \ \text{inpt} \ s' = \text{inpt} \ s \land \text{disk} \ s' = \text{disk} \ s \)

definition \( \text{EndPhase1or2} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)
where \( \text{EndPhase1or2} \ s \ s' \ p = (\text{EndPhase1} \ s \ s' \ p \lor \text{EndPhase2} \ s \ s' \ p) \)

definition \( \text{Fail} :: \text{state} \Rightarrow \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool} \)
where \( \text{Fail} \ s \ s' \ p = (\exists \ ip \in \text{Inputs}. \ \text{inpt} \ s' = (\text{inpt} \ s) \ (p := \text{ip}) \)
\( \land \ \text{phase} \ s' = (\text{phase} \ s \ (p := 0)) \)
\( \land \ \text{dblock} \ s' = (\text{dblock} \ s) \ (p := \text{InitDB}) \)
\(\land \text{outpt } s' = (\text{outpt } s) \ (p := \text{NotAnInput})\)
\(\land \text{InitializePhase } s \ s' \ p\)
\(\land \text{disk } s' = \text{disk } s\)

definition Phase0Read :: state \(\Rightarrow\) state \(\Rightarrow\) Proc \(\Rightarrow\) Disk \(\Rightarrow\) bool
where
Phase0Read s s' p d =
(\(\text{phase } s \ p = 0\)
\(\land \text{blocksRead } s' = (\text{blocksRead } s) \ (p := (\text{blocksRead } s \ p) \ (d := \text{blocksRead } s \ p \ d)\)
\(\lor \{\\{\text{block} = \text{disk } s \ d \ p, \ proc = p \}\}\))
\(\land \text{inpt } s' = \text{inpt } s \land \text{outpt } s' = \text{outpt } s\)
\(\land \text{disk } s' = \text{disk } s \land \text{phase } s' = \text{phase } s\)
\(\land \text{dblock } s' = \text{dblock } s \land \text{disksWritten } s' = \text{disksWritten } s\)

definition EndPhase0 :: state \(\Rightarrow\) state \(\Rightarrow\) Proc \(\Rightarrow\) bool
where
EndPhase0 s s' p =
\(\text{phase } s \ p = 0\)
\(\land \text{IsMajority} \ (\{d. \ \text{hasRead } s \ p \ d\})\)
\(\land (\exists b \in \text{Ballot } p.\)
\(\ (\forall r \in \text{allBlocksRead } s \ p. \ \text{mbal } r < b)\)
\(\land \text{dblock } s' = (\text{dblock } s) \ (p := \)
\(\ (\exists \text{Some } r. \ r \in \text{allBlocksRead } s \ p\)
\(\land (\forall s \in \text{allBlocksRead } s \ p. \ \text{bal } s \leq \text{bal } r) \ (\ \text{mbal} := b \})\))
\(\land \text{InitializePhase } s \ s' \ p\)
\(\land \text{inpt } s' = \text{inpt } s \land \text{outpt } s' = \text{outpt } s \land \text{disk } s' = \text{disk } s\)

definition Next :: state \(\Rightarrow\) state \(\Rightarrow\) bool
where
Next s s' = (\(\exists p.\)
\(\ \text{StartBallot } s \ s' \ p\)
\(\lor (\exists d. \ \text{Phase0Read } s \ s' \ p \ d\)
\(\lor \text{Phase1or2Write } s \ s' \ p \ d\)
\(\lor (\exists q. \ q \neq p \land \text{Phase1or2Read } s \ s' \ p \ d \ q))\)
\(\lor \text{EndPhase1or2 } s \ s' \ p\)
\(\lor \text{Fail } s \ s' \ p\)
\(\lor \text{EndPhase0 } s \ s' \ p\)

In the following, for each action or state name we name Hname the corresponding action that includes the history part of the HNext action or state predicate that includes history variables.

definition HInit :: state \(\Rightarrow\) bool
where
HInit s =
(\text{Init } s
\land \text{chosen } s = \text{NotAnInput}
\land \text{allInput } s = \text{range } (\text{inpt } s))
HNextPart is the part of the Next action that is concerned with history variables.

definition HNextPart :: state ⇒ state ⇒> bool where
  HNextPart s s’ = (chosen s’ = (if chosen s ≠ NotAnInput ∨ (∀ p. outpt s’ p = NotAnInput) then chosen s else outpt s’ (SOME p. outpt s’ p ≠ NotAnInput)) ∧ allInput s’ = allInput s ∪ (range (inpt s’)))

definition HNext :: state ⇒ state ⇒ bool where
  HNext s s’ = (Next s s’ ∧ HNextPart s s’)

We add HNextPart to every action (rather than proving that Next maintains the HInv invariant) to make proofs easier.

definition HPhase1or2ReadThen :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool where
  HPhase1or2ReadThen s s’ p d q = (Phase1or2ReadThen s s’ p d q ∧ HNextPart s s’)

definition HEndPhase1 :: state ⇒ state ⇒ Proc ⇒ bool where
  HEndPhase1 s s’ p = (EndPhase1 s s’ p ∧ HNextPart s s’)

definition HStartBallot :: state ⇒ state ⇒ Proc ⇒ bool where
  HStartBallot s s’ p = (StartBallot s s’ p ∧ HNextPart s s’)

definition HPhase1or2Write :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ bool where
  HPhase1or2Write s s’ p d = (Phase1or2Write s s’ p d ∧ HNextPart s s’)

definition HPhase1or2ReadElse :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ Proc ⇒ bool where
  HPhase1or2ReadElse s s’ p d q = (Phase1or2ReadElse s s’ p d q ∧ HNextPart s s’)

definition HEndPhase2 :: state ⇒ state ⇒ Proc ⇒ bool where
  HEndPhase2 s s’ p = (EndPhase2 s s’ p ∧ HNextPart s s’)

definition HFail :: state ⇒ state ⇒ Proc ⇒ bool where
  HFail s s’ p = (Fail s s’ p ∧ HNextPart s s’)

definition
HPhase0Read :: state ⇒ state ⇒ Proc ⇒ Disk ⇒ bool where
HPhase0Read s s' p d = (Phase0Read s s' p d ∧ HNextPart s s')

definition
HEndPhase0 :: state ⇒ state ⇒ Proc ⇒ bool where
HEndPhase0 s s' p = (EndPhase0 s s' p ∧ HNextPart s s')

Since these definitions are the conjunction of two other definitions declaring
them as simplification rules should be harmless.

declare HPhase1or2ReadThen-def [simp]
declare HPhase1or2ReadElse-def [simp]
declare HEndPhase1-def [simp]
declare HStartBallot-def [simp]
declare HPhase1or2Write-def [simp]
declare HEndPhase2-def [simp]
declare HFail-def [simp]
declare HPhase0Read-def [simp]
declare HEndPhase0-def [simp]

end

C Proof of Disk Paxos’ Invariant

theory DiskPaxos-Inv1 imports DiskPaxos-Model begin

C.1 Invariant 1

This is just a type Invariant.

definition Inv1 :: state ⇒ bool where
Inv1 s = (∀ p.
    inpt s p ∈ Inputs
∧ phase s p ≤ 3
∧ finite (allRdBlks s p))

definition HInv1 :: state ⇒ bool where
HInv1 s =
    (Inv1 s
∧ allInput s ⊆ Inputs)

declare HInv1-def [simp]

We added the assertion that the set allRdBlksp is finite for every process p;
one may therefore choose a block with a maximum ballot number in action EndPhase1.
With the following the lemma, it will be enough to prove Inv1 s’ for every action, without taking the history variables in account.

**Lemma** \( H\text{NextPart-Inv1} \): 
\[
\begin{align*}
&\text{[ } H\text{Inv1 } s ; H\text{NextPart } s \ s' ; \text{Inv1 } s' \text{]} \implies H\text{Inv1 } s'
\end{align*}
\]
by (auto simp add: HNextPart-def Inv1-def)

**Theorem** \( H\text{Init-HInv1} \): 
\[
H\text{Init } s \implies H\text{Inv1 } s
\]
by (auto simp add: HInit-def Inv1-def Init-def allRdBlks-def)

**Lemma** \( \text{allRdBlks-finite} \):
\[
\begin{align*}
&\text{assumes } \text{inv } : H\text{Inv1 } s \\
&\text{and } \text{asm } : \forall p. \text{allRdBlks } s' p \subseteq \text{insert } bk \ (\text{allRdBlks } s \ p) \\
&\text{shows } \forall p. \text{finite } (\text{allRdBlks } s' p)
\end{align*}
\]
**Proof**
\[
\begin{align*}
&\text{fix } pp \\
&\text{from } \text{inv} \\
&\text{have } \forall p. \text{finite } (\text{allRdBlks } s \ p) \\
&\quad \text{by } (\text{simp add: Inv1-def}) \\
&\quad \text{hence } \text{finite } (\text{allRdBlks } s \ pp) \\
&\quad \text{by blast} \\
&\quad \text{with } \text{asm} \\
&\text{show } \text{finite } (\text{allRdBlks } s' pp) \\
&\quad \text{by } (\text{auto intro: finite-subset})
\end{align*}
\]
**Qed**

**Theorem** \( H\text{Phase1or2ReadThen-HInv1} \):
\[
\begin{align*}
&\text{assumes } \text{inv1 } : H\text{Inv1 } s \\
&\text{and } \text{act } : H\text{Phase1or2ReadThen } s \ s' \ p \ d \ q \\
&\text{shows } H\text{Inv1 } s'
\end{align*}
\]
**Proof** —
\[
\begin{align*}
&\text{— we focus on the last conjunct of Inv1} \\
&\text{from } \text{act} \\
&\text{have } \forall p. \text{allRdBlks } s' p \subseteq \text{allRdBlks } s \ p \cup \{ \text{[block = disk } s \ d \ q, \ \text{proc } = \ q]\} \\
&\quad \text{by } (\text{auto simp add: Phase1or2ReadThen-def allRdBlks-def} \\
&\quad \quad \text{split: if-split-asm}) \\
&\quad \text{with } \text{inv1} \\
&\text{have } \forall p. \text{finite } (\text{allRdBlks } s' p) \\
&\quad \text{by } (\text{blast dest: allRdBlks-finite}) \\
&\quad \text{— the others conjuncts are trivial} \\
&\quad \text{with } \text{inv1 } \text{act} \\
&\text{show } ?\text{thesis} \\
&\quad \text{by } (\text{auto simp add: Inv1-def Phase1or2ReadThen-def HNextPart-def})
\end{align*}
\]
**Qed**

**Theorem** \( H\text{EndPhase1-HInv1} \):
\[
\begin{align*}
&\text{assumes } \text{inv1 } : H\text{Inv1 } s \\
&\text{and } \text{act } : H\text{EndPhase1 } s \ s' \ p \\
&\text{shows } H\text{Inv1 } s'
\end{align*}
\]
**Proof** —
\[
\begin{align*}
&\text{from } \text{inv1 } \text{act}
\end{align*}
\]
have \( \text{Inv1} \ s' \)
by (auto simp add: \( \text{Inv1-def} \ \text{EndPhase1-def} \ \text{InitializePhase-def} \ \text{allRdBlks-def} \))
with \( \text{inv1} \ \text{act} \)
show \( ?\text{thesis} \)
by (auto simp del: \( \text{HInv1-def} \ \text{dest} \) : \( \text{HNextPart-Inv1} \))
\[ \text{qed} \]

\textbf{theorem} \( \text{HStartBallot-HInv1} \):
\textbf{assumes} \( \text{inv1} : \ \text{HInv1} \ s \)
and \( \text{act} : \ \text{HStartBallot} \ s \ s' \ p \)
\textbf{shows} \( \text{HInv1} \ s' \)
\textbf{proof} –
from \( \text{inv1} \ \text{act} \)
have \( \text{Inv1} \ s' \)
by (auto simp add: \( \text{Inv1-def} \ \text{StartBallot-def} \ \text{InitializePhase-def} \ \text{allRdBlks-def} \))
with \( \text{inv1} \ \text{act} \)
show \( ?\text{thesis} \)
by (auto simp del: \( \text{HInv1-def} \ \text{elim} : \text{HNextPart-Inv1} \))
\[ \text{qed} \]

\textbf{theorem} \( \text{HPhase1or2Write-HInv1} \):
\textbf{assumes} \( \text{inv1} : \ \text{HInv1} \ s \)
and \( \text{act} : \ \text{HPhase1or2Write} \ s \ s' \ p \ d \)
\textbf{shows} \( \text{HInv1} \ s' \)
\textbf{proof} –
from \( \text{inv1} \ \text{act} \)
have \( \text{Inv1} \ s' \)
by (auto simp add: \( \text{Inv1-def} \ \text{Phase1or2Write-def} \ \text{allRdBlks-def} \))
with \( \text{inv1} \ \text{act} \)
show \( ?\text{thesis} \)
by (auto simp del: \( \text{HInv1-def} \ \text{elim} : \text{HNextPart-Inv1} \))
\[ \text{qed} \]

\textbf{theorem} \( \text{HPhase1or2ReadElse-HInv1} \):
\textbf{assumes} \( \text{act} : \ \text{HPhase1or2ReadElse} \ s \ s' \ p \ d \ q \)
and \( \text{inv1} : \ \text{HInv1} \ s \)
\textbf{shows} \( \text{HInv1} \ s' \)
\textbf{using} \( \text{HStartBallot-HInv1[OF inv1]} \ \text{act} \)
by (auto simp add: \( \text{Phase1or2ReadElse-def} \))

\textbf{theorem} \( \text{HEndPhase2-HInv1} \):
\textbf{assumes} \( \text{inv1} : \ \text{HInv1} \ s \)
and \( \text{act} : \ \text{HEndPhase2} \ s \ s' \ p \)
\textbf{shows} \( \text{HInv1} \ s' \)
\textbf{proof} –
from \( \text{inv1} \ \text{act} \)
have \( \text{Inv1} \ s' \)
by (auto simp add: \( \text{Inv1-def} \ \text{EndPhase2-def} \ \text{InitializePhase-def} \ \text{allRdBlks-def} \))
with \( \text{inv1} \ \text{act} \)
show \(\text{thesis}\)
  by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HFail-HInv1:
  assumes inv1: HInv1 \(s\)
  and act: HFail \(s, s'\ p\)
  shows HInv1 \(s'\)
proof -
  from inv1 act
  have Inv1 \(s'\)
    by (auto simp add: HInv1-def Fail-def InitializePhase-def allRdBlks-def)
  with inv1 act show \(\text{thesis}\)
    by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HPhase0Read-HInv1:
  assumes inv1: HInv1 \(s\)
  and act: HPhase0Read \(s, s'\ p\ d\)
  shows HInv1 \(s'\)
proof -
  — we focus on the last conjunct of Inv1
  from act
  have \(\forall p.\ \text{allRdBlks} s' p \subseteq \text{allRdBlks} s p \cup \{block = \text{disk} s d p, proc = p\}\)
    by (auto simp add: Phase0Read-def allRdBlks-def
        split: if-split-asm)
  with inv1
  have Inv1 \(s'\)
    by (auto simp add: Inv1-def Phase0Read-def)
  with inv1 act
  show \(\text{thesis}\)
    by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

theorem HEndPhase0-HInv1:
  assumes inv1: HInv1 \(s\)
  and act: HEndPhase0 \(s, s'\ p\)
  shows HInv1 \(s'\)
proof -
  from inv1 act
  have Inv1 \(s'\)
    by (auto simp add: HInv1-def EndPhase0-def allRdBlks-def
        InitializePhase-def)
  with inv1 act
  show \(\text{thesis}\)
    by (auto simp del: HInv1-def elim: HNextPart-Inv1)
qed

declare HInv1-def [simp del]

HInv1 is an invariant of HNext

lemma I2a:
  assumes  nxt: HNext s s'
  and  inv: HInv1 s
  shows  HInv1 s'
  using  assms
  by (auto
    simp add: HNext-def Next-def,
    auto intro: HStartBallot-HInv1,
    auto intro: HPhase0Read-HInv1,
    auto intro: HPhase1or2Write-HInv1,
    auto simp add: Phase1or2Read-def
      intro: HPhase1or2ReadThen-HInv1
            HPhase1or2ReadElse-HInv1,
    auto simp add: EndPhase1or2-def
      intro: HEndPhase1-HInv1
            HEndPhase2-HInv1,
    auto intro: HFail-HInv1,
    auto intro: HEndPhase0-HInv1)

diskpaxos-Inv2 imports DiskPaxos-Inv1 begin

C.2 Invariant 2

The second invariant is split into three main conjuncts called Inv2a, Inv2b, and Inv2c. The main difficulty is in proving the preservation of the first conjunct.

definition rdBy :: state ⇒ Proc ⇒ Proc ⇒ Disk ⇒ BlockProc set
  where
    rdBy s p q d = 
      { br . br ∈ blocksRead s q d ∧ proc br = p }

definition blocksOf :: state ⇒ Proc ⇒ DiskBlock set
  where
    blocksOf s p =
      \{ dblock s p \}
    ∪ \{ disk s d p | d . d ∈ UNIV \}
    ∪ \{ block br | br . br ∈ (UN q d. rdBy s p q d) \}

definition allBlocks :: state ⇒ DiskBlock set
where allBlocks s = (UN p. blocksOf s p)

definition Inv2a-innermost :: state ⇒ Proc ⇒ DiskBlock ⇒ bool
where
Inv2a-innermost s p bk =
  (mbal bk ∈ (Ballot p) ∪ {0})
∧ bal bk ∈ (Ballot p) ∪ {0}
∧ (bal bk = 0) = (inp bk = NotAnInput)
∧ bal bk ≤ mbal bk
∧ inp bk ∈ (allInput s) ∪ {NotAnInput})

definition Inv2a-inner :: state ⇒ Proc ⇒ bool
where Inv2a-inner s p = (∀ bk ∈ blocksOf s p. Inv2a-innermost s p bk)

definition Inv2a :: state ⇒ bool
where Inv2a s = (∀ p. Inv2a-inner s p)

definition Inv2b-inner :: state ⇒ Proc ⇒ Disk ⇒ bool
where
Inv2b-inner s p d =
  (d ∈ disksWritten s p →
    (phase s p ∈ {1,2} ∧ disk s d p = dblock s p))
∧ (phase s p ∈ {1,2} →
  (blocksRead s p d ≠ {} → d ∈ disksWritten s p)
∧ ¬ hasRead s p d))

definition Inv2b :: state ⇒ bool
where Inv2b s = (∀ p d. Inv2b-inner s p d)

definition Inv2c-inner :: state ⇒ Proc ⇒ bool
where
Inv2c-inner s p =
  ((phase s p = 0 →
    (dblock s p = InitDB
∧ disksWritten s p = {})
∧ (∀ d. ∀ br ∈ blocksRead s p d.
  proc br = p ∧ block br = disk s d p)))
∧ (phase s p ≠ 0 →
  (mbal(dblock s p) ∈ Ballot p
∧ bal(dblock s p) ∈ Ballot p ∪ {0}
∧ (∀ d. ∀ br ∈ blocksRead s p d.
  mbal(block br) < mbal(dblock s p))))
∧ (phase s p ∈ {2,3} → bal(dblock s p) = mbal(dblock s p))
∧ outpt s p = (if phase s p = 3 then inp(dblock s p) else NotAnInput)
∧ chosen s ∈ allInput s ∪ {NotAnInput}
∧ (∀ p. inp t p ∈ allInput s
∧ (chosen s = NotAnInput → outpt s p = NotAnInput))

definition Inv2c :: state ⇒ bool
where \( \text{Inv2c} \ s = (\forall p. \text{Inv2c-inner} \ s \ p) \)

definition \( \text{HInv2} :: \text{state} \Rightarrow \text{bool} \)
where \( \text{HInv2} \ s = (\text{Inv2a} \ s \land \text{Inv2b} \ s \land \text{Inv2c} \ s) \)

C.2.1 Proofs of Invariant 2 a

**theorem** \( \text{HInit-Inv2a} : \text{HInit} \ s \rightarrow \text{Inv2a} \ s \)
**by** (auto simp add: \text{HInit-def} \text{Init-def} \text{Inv2a-def} \text{Inv2a-inner-def} \text{Inv2a-innermost-def} \text{rdBy-def} \text{blocksOf-def} \text{InitDB-def})

For every action we define a action-\text{blocksOf} lemma. We have two cases: either the new \text{blocksOf} is included in the old \text{blocksOf}, or the new \text{blocksOf} is included in the old \text{blocksOf} union the new \text{dblock}. In the former case the assumption \text{inv} will imply the thesis. In the latter, we just have to prove the innermost predicate for the particular case of the new \text{dblock}. This particular case is proved in lemma action-Inv2a-dblock.

**lemma** \( \text{HPhase1or2ReadThen-blocksOf} : \)
\[
\begin{align*}
[ & \text{HPhase1or2ReadThen} \ s \ s' p d q ] \implies \text{blocksOf} \ s' r \subseteq \text{blocksOf} \ s r 
\end{align*}
\]
**by** (auto simp add: \text{Phase1or2ReadThen-def} \text{blocksOf-def} \text{rdBy-def})

**theorem** \( \text{HPhase1or2ReadThen-Inv2a} : \)
**assumes** \text{inv2a} \( : \text{Inv2a-innermost} \ s \ p \ (\text{dblock} \ s \ p) \)
**shows** \text{Inv2a-innermost} \( : \text{Inv2a-innermost-def} \ \\ \\ \text{HNextPart-def} \)
**proof** (clarsimp simp add: \text{Inv2a-def} \text{Inv2a-inner-def})
  \begin{align*}
  &\text{fix} \ pp bk \\
  &\text{assume} bk: bk \in \text{blocksOf} \ s' \ pp \\
  &\text{with} \ \text{inv} \ \text{HPhase1or2ReadThen-blocksOf}[OF \ \text{act}] \\
  &\text{have} \ \text{Inv2a-innermost} \ s \ pp bk \\
  &\text{by} (\text{auto simp add: Inv2a-def Inv2a-inner-def}) \\
  &\text{with} \ \text{act} \\
  &\text{show} \ \text{Inv2a-innermost} \ s' \ pp bk \\
  &\text{by} (\text{auto simp add: Inv2a-innermost-def HNextPart-def})
\end{align*}
\text{qed}

**lemma** \( \text{InitializePhase-rdBy} : \)
\[
\begin{align*}
\text{InitializePhase} \ s' \ p \implies \text{rdBy} \ s' \ pp qq dd \subseteq \text{rdBy} \ s \ pp qq dd 
\end{align*}
\]
**by** (auto simp add: \text{InitializePhase-def} \text{rdBy-def})

**lemma** \( \text{HStartBallot-blocksOf} : \)
\[
\begin{align*}
\text{HStartBallot} \ s' \ p \implies \text{blocksOf} \ s' q \subseteq \text{blocksOf} \ s q \cup \{ \text{dblock} \ s' q \} 
\end{align*}
\]
**by** (auto simp add: \text{StartBallot-def} \text{blocksOf-def} \text{dest: subsetD}[OF \ \text{InitializePhase-rdBy}])

**lemma** \( \text{HStartBallot-Inv2a-dblock} : \)
**assumes** \text{act} \( : \text{HStartBallot} \ s' \ p \)
**and** \text{inv2a} \( : \text{Inv2a-innermost} \ s \ p \ (\text{dblock} \ s \ p) \)
shows \( \text{Inv2a-innermost } s' \ p \ (\text{dblock } s' \ p) \)

proof –

from act
have \( \text{mbal}' : \text{mbal} (\text{dblock } s' \ p) \in \text{Ballot } p \)
by (auto simp add: \text{StartBallot-def})

from act
have \( \text{bal}' : \text{bal} (\text{dblock } s' \ p) = \text{bal} (\text{dblock } s \ p) \)
with act
have \( \text{inp}' : \text{inp} (\text{dblock } s' \ p) = \text{inp} (\text{dblock } s \ p) \)
by (auto simp add: \text{StartBallot-def})

from act
have \( \text{mbal} (\text{dblock } s \ p) \leq \text{mbal} (\text{dblock } s' \ p) \)
with bal' inv2a
have \( \text{bal-mbal} : \text{bal} (\text{dblock } s' \ p) \leq \text{mbal} (\text{dblock } s' \ p) \)
by (auto simp add: \text{Inv2a-innermost-def})

from act
have \( \text{allInput } s \subseteq \text{allInput } s' \)
by (auto simp add: \text{HNextPart-def} \text{InitializePhase-def} \text{Inv2a-innermost-def})

with \( \text{mbal}' \ \text{bal}' \ \text{inp}' \ \text{bal-mbal} \ \text{act} \ \text{inv2a} \)
show \( ?\text{thesis} \)
by (auto simp add: \text{Inv2a-innermost-def})

qed

lemma \text{HStartBallot-Inv2a-dblock-q}:
assumes act: \text{HStartBallot } s \ s' \ p
and \( \text{inv2a} : \text{Inv2a-innermost } s \ q \) \( (\text{dblock } s \ q) \)
shows \( \text{Inv2a-innermost } s' \ q \) \( (\text{dblock } s' \ q) \)

proof (cases \( p=q \))

case True
with act inv2a
show \( ?\text{thesis} \)
by (blast dest: \text{HStartBallot-Inv2a-dblock})

next

case False
with act inv2a
show \( ?\text{thesis} \)
by (clarsimp simp add: \text{StartBallot-def} \text{HNextPart-def} \text{InitializePhase-def} \text{Inv2a-innermost-def})

qed

theorem \text{HStartBallot-Inv2a}:
assumes inv: \( \text{Inv2a } s \)
and act: \text{HStartBallot } s \ s' \ p
shows \( \text{Inv2a } s' \)

proof (clarsimp simp add: \text{Inv2a-def} \text{Inv2a-inner-def})
fix \ q \ bk
assume \( \text{bk} : \text{bk} \in \text{blocksOf } s' \ q \)
with inv
have oldBlks: bk ∈ blocksOf s q → Inv2a-innermost s q bk
  by (auto simp add: Inv2a-def Inv2a-inner-def)
from bk HStartBallot-blocksOf[OF act]
have bk ∈ {dblock s' q} ∪ blocksOf s q
  by blast
thus Inv2a-innermost s' q bk
proof
  assume bk-dblock: bk ∈ {dblock s' q}
  from inv
  have inv-q-dblock: Inv2a-innermost s q (dblock s q)
    by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
  with act inv bk-dblock
  show ?thesis
    by (blast dest: HStartBallot-Inv2a-dblock-q)
next
  assume bk-in-blocks: bk ∈ blocksOf s q
  with oldBlks
  have Inv2a-innermost s q bk ..
  with act
  show ?thesis
    by (auto simp add: StartBallot-def HNextPart-def InitializePhase-def Inv2a-innermost-def)
qed
Qed

lemma HPhase1or2Write-blocksOf:
  [ HPhase1or2Write s s' p d ] ⇒ blocksOf s' r ⊆ blocksOf s r
  by (auto simp add: Phase1or2Write-def blocksOf-def rdBy-def)

theorem HPhase1or2Write-Inv2a:
  assumes inv: Inv2a s
  and act: HPhase1or2Write s s' p d
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk ∈ blocksOf s' q
  from inv bk HPhase1or2Write-blocksOf[OF act]
  have inp-q-bk: Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act
  show Inv2a-innermost s' q bk
    by (auto simp add: Inv2a-innermost-def HNextPart-def)
qed

theorem HPhase1or2ReadElse-Inv2a:
  assumes inv: Inv2a s
  and act: HPhase1or2ReadElse s s' p d q
  shows Inv2a s'

proof

from act
have HStartBallot s s' p
  by (simp add: Phase1or2ReadElse-def)
with inv
show ?thesis
  by (auto elim: HStartBallot-Inv2a)
qed

lemma HEndPhase2-blocksOf:
  [ HEndPhase2 s s' p ] \implies blocksOf s' q \subseteq blocksOf s q
by (auto simp add: EndPhase2-def blocksOf-def
  dest: subsetD[OF InitializePhase-rdBy])

theorem HEndPhase2-Inv2a:
  assumes inv: Inv2a s
  and act: HEndPhase2 s s' p
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk \in blocksOf s' q
  from inv bk HEndPhase2-blocksOf[OF act]
  have inp-q-bk: Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act
  show Inv2a-innermost s' q bk
    by (auto simp add: Inv2a-innermost-def HNextPart-def)
qed

lemma HFail-blocksOf:
  HFail s s' p \implies blocksOf s' q \subseteq blocksOf s q \cup \{dblock s' q\}
by (auto simp add: Fail-def blocksOf-def
  dest: subsetD[OF InitializePhase-rdBy])

lemma HFail-Inv2a-dblock-q:
  assumes act: HFail s s' p
  and inv: Inv2a-innermost s q (dblock s q)
  shows Inv2a-innermost s' q (dblock s' q)
proof (cases p=q)
  case True
  with act
  have dblock s' q = InitDB
    by (simp add: Fail-def)
  with True
  show ?thesis
    by (auto simp add: InitDB-def Inv2a-innermost-def)
next
  case False
  with inv act
show ?thesis
  by (auto simp add: Fail-def HNextPart-def
              InitializePhase-def Inv2a-innermost-def)
qed

theorem HFail-Inv2a:
  assumes inv: Inv2a s
  and  act: HFail s s' p
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk ∈ blocksOf s' q
  with HFail-blocksOf[OF act]
  have dblock-blocks: bk ∈ {dblock s' q} ∪ blocksOf s q
      by blast
  thus Inv2a-innermost s' q bk
  proof
    assume bk-dblock: bk ∈ {dblock s' q}
    from inv
    have inv-q-dblock: Inv2a-innermost s q (dblock s q)
      by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
    with act bk-dblock
    show ?thesis
      by (blast dest: HFail-Inv2a-dblock-q)
  next
  assume bk-in-blocks: bk ∈ blocksOf s q
  with inv
  have Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act
  show ?thesis
    by (auto simp add: Fail-def HNextPart-def
                      InitializePhase-def Inv2a-innermost-def)
qed
qed

lemma HPhase0Read-blocksOf:
  HPhase0Read s s' p d ⇒ blocksOf s' q ⊆ blocksOf s q
by (auto simp add: Phase0Read-def InitializePhase-def
                   blocksOf-def rdBy-def)

theorem HPhase0Read-Inv2a:
  assumes inv: Inv2a s
  and  act: HPhase0Read s s' p d
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk ∈ blocksOf s' q
  from inv bk HPhase0Read-blocksOf[OF act]
have inp-q-bk: Inv2a-innermost s q bk
  by (auto simp add: Inv2a-def Inv2a-inner-def)
with act
show Inv2a-innermost s' q bk
  by (auto simp add: Inv2a-innermost-def HNextPart-def)
qed

lemma HEndPhase0-blocksOf:
  HEndPhase0 s s' p \implies blocksOf s' q \subseteq blocksOf s q \cup \{dblock s' q\}
by (auto simp add: EndPhase0-def blocksOf-def
    dest: subsetD[OF InitializePhase-rdBy])

lemma HEndPhase0-blocksRead:
assumes act: HEndPhase0 s s' p
shows \exists d. blocksRead s p d \neq {}
proof -
  from act have IsMajority(\{d. hasRead s p d\})
    by (simp add: EndPhase0-def)
  hence (d. hasRead s p d) \neq {} by (rule majority-nonempty)
  thus \thesis
    by (auto simp add: hasRead-def)
qed

EndPhase0 has the additional difficulty of having a choose expression. We
prove that there exists an x such that the predicate of the choose expression
holds, and then apply someI: \(? ?x \implies \? (Eps ?P)."

lemma HEndPhase0-some:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
shows (SOME b. b \in allBlocksRead s p
  \land (\forall t \in allBlocksRead s p. bal t \leq bal b)) \in allBlocksRead s p
  \land (\forall t \in allBlocksRead s p.
    bal t \leq (SOME b. b \in allBlocksRead s p
    \land (\forall t \in allBlocksRead s p. bal t \leq bal b)))
proof -
  from inv1 have finite (bal \cup allBlocksRead s p) (is finite ?S)
    by (simp add: Inv1-def allBlocksRead-def)
  moreover
  from HEndPhase0-blocksRead[OF act]
  have ?S \neq {}
    by (auto simp add: allBlocksRead-def allRdBlks-def)
  ultimately
  have Max ?S \in ?S and \forall t \in ?S. t \leq Max ?S by auto
  hence \exists r \in ?S. \forall t \in ?S. t \leq r ..
  then obtain mblk
    where mblk \in allBlocksRead s p
      \land (\forall t \in allBlocksRead s p. bal t \leq bal mblk) (is ?P mblk)

30
by auto
thus ?thesis
by (rule someI)
qed

lemma HEndPhase0-dblock-allBlocksRead:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
shows dblock s' p ∈ (λx. x ⟨mbal:= mbal(dblock s' p)⟩) · allBlocksRead s p
using act HEndPhase0-some[OF act inv1]
by(auto simp add: EndPhase0-def)

lemma HNextPart-allInput-or-NotAnInput:
assumes act: HNextPart s s' p
and inv2a: Inv2a-innermost s p (dblock s' p)
shows inpt (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
proof –
from act
have allInput s' = allInput s ∪ (range (inpt s'))
  by(simp add: HNextPart-def)
moreover
from inv2a
have inpt (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
  by(simp add: Inv2a-innermost-def)
ultimately show ?thesis
by blast
qed

lemma HEndPhase0-Inv2a-allBlocksRead:
assumes act: HEndPhase0 s s' p
and inv2a: Inv2a-inner s p
and inv2c: Inv2c-inner s p
shows ∀ t ∈ (λx. x ⟨mbal:= mbal (dblock s' p)⟩) · allBlocksRead s p.
  Inv2a-innermost s p t
proof –
from act
have mbal': mbal (dblock s' p) ∈ Ballot p
  by(auto simp add: EndPhase0-def)
from inv2c act
have allproc-p: ∀ d. ∀ br ∈ blocksRead s p d. proc br = p
  by(simp add: Inv2c-inner-def EndPhase0-def)
with inv2a
have allBlocks-inv2a: ∀ t ∈ allBlocksRead s p. Inv2a-innermost s p t
proof(auto simp add: Inv2a-inner-def allBlocksRead-def
  allRdBlks-def blocksOf-def rdBy-def)
fix d bk
assume bk-in-blocksRead: bk ∈ blocksRead s p d
and inv2a-bk: ∀ x∈ {u. ∃ d. u = disk s d p}
  ∪ {block br | br. (∃ q d. br ∈ blocksRead s q d)
with allproc-p have proc bk = p by auto
with bk-in-blocksRead inv2a-bk
show Inv2a-innermost s p (block bk) by blast
qed
from act
have mbal'gt: ∀ bk ∈ allBlocksRead s p. mbal bk ≤ mbal (dblock s' p)
  by(auto simp add: EndPhase0-def)
with mbal' allBlocks-inv2a
show ?thesis
proof (auto simp add: Inv2a-innermost-def)
fix t
assume t-blocksRead: t ∈ allBlocksRead s p
with allBlocks-inv2a
have bal t ≤ mbal t by (auto simp add: Inv2a-innermost-def)
moreover
from t-blocksRead mbal'gt
have mbal t ≤ mbal (dblock s' p) by blast
ultimately show bal t ≤ mbal (dblock s' p)
  by auto
qed

lemma HEndPhase0-Inv2a-dblock:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and inv2a: Inv2a-inner s p
and inv2c: Inv2c-inner s p
shows Inv2a-innermost s' p (dblock s' p)
proof –
from act inv2a inv2c
have t1: ∀ t ∈ (λx. x (mbal:= mbal (dblock s' p)))) \ allBlocksRead s p.
  Inv2a-innermost s p t
  by(blast dest: HEndPhase0-Inv2a-allBlocksRead)
from act inv1
have dblock s' p ∈ (λx. x (mbal:= mbal(dblock s' p)))) \ allBlocksRead s p
  by(simp, blast dest: HEndPhase0-dblock-allBlocksRead)
with t1
have inv2-dblock: Inv2a-innermost s p (dblock s' p) by auto
with act
have inp (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
  by(auto dest: HNextPart-allInput-or-NotAnInput)
with inv2-dblock
show ?thesis
  by(auto simp add: Inv2a-innermost-def)
qed

lemma HEndPhase0-Inv2a-dblock-q:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and inv2a: Inv2a-inner s q
and inv2c: Inv2c-inner s p
shows Inv2a-innermost s' q (dblock s' q)
proof (cases p=q)
case True
with act inv2a inv2c inv1
show ?thesis
  by (blast dest: HEndPhase0-Inv2a-dblock)
next
case False
from inv2a
have inv-q-dblock: Inv2a-innermost s q (dblock s q)
  by (auto simp add: Inv2a-inner-def blocksOf-def)
with False act
show ?thesis
  by (clarsimp simp add: EndPhase0-def HNextPart-def
    InitializePhase-def Inv2a-innermost-def)
qed

theorem HEndPhase0-Inv2a:
  assumes inv: Inv2a s
  and act: HEndPhase0 s s' p
  and inv1: Inv1 s
  and inv2c: Inv2c-inner s p
  shows Inv2a s'
proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk: bk ∈ blocksOf s' q
  with HEndPhase0-blocksOf[OF act]
  have dblock-blocks: bk ∈ {dblock s' q} ∪ blocksOf s q
    by blast
  thus Inv2a-innermost s' q bk
proof
  from inv
  have inv-q: Inv2a-inner s q
    by (auto simp add: Inv2a-def)
  assume bk ∈ {dblock s' q}
  with act inv1 inv2c inv-q
  show ?thesis
    by (blast dest: HEndPhase0-Inv2a-dblock-q)
next
  assume bk-in-blocks: bk ∈ blocksOf s q
  with inv
  have Inv2a-innermost s q bk
    by (auto simp add: Inv2a-inner-def)
  with act show ?thesis
    by (auto simp add: EndPhase0-def HNextPart-def
      InitializePhase-def Inv2a-innermost-def)
lemma $HEndPhase1$-blocksOf:
$HEndPhase1\ s\ s'\ p \implies \text{blocksOf}\ s'\ q \subseteq \text{blocksOf}\ s\ q \cup \{\text{dblock}\ s'\ q\}$
by (auto simp add: $EndPhase1$-def blocksOf-def
dest: subsetD[OF $InitializePhase$-rdBy])

lemma maxBlk-in-nonInitBlks:
assumes $b\in\text{nonInitBlks}\ s\ p$
and $inv1: Inv1\ s$
shows $\maxBlk\ s\ p\in\text{nonInitBlks}\ s\ p$
\and $(\forall c\in\text{nonInitBlks}\ s\ p.\ \text{bal}\ c\leq\text{bal}\ (\maxBlk\ s\ p))$
proof –
have nibals-finite: finite (bal ' (nonInitBlks s p)) (is finite ?S)
proof (rule finite-image1)
  from inv1
  have finite (allRdBlks s p)
  by (auto simp add: Inv1-def)
  hence finite (allBlocksRead s p)
  by (auto simp add: allBlocksRead-def)
  hence finite (blocksSeen s p)
  by (simp add: blocksSeen-def)
  thus finite (nonInitBlks s p)
  by(auto simp add: nonInitBlks-def intro: finite-subset)
qed
from $b$ have bal ' nonInitBlks s p \neq {}
by auto
with nibals-finite
have $\maxS\in\text{?S}\ \text{and} \ \forall bb\in\text{?S}.\ bb\leq\maxS$ by auto
hence $\exists mb\in\text{?S}.\ \forall bb\in\text{?S}.\ bb\leq\text{mb}.$
then obtain mblk
  where mblk \in nonInitBlks s p
  \and $(\forall c\in\text{nonInitBlks}\ s\ p.\ \text{bal}\ c\leq\text{bal}\ mblk)$
  (is ?P mblk)
by auto
hence ?P (SOME b. ?P b)
by (rule someI)
thus ?thesis
by (simp add: maxBlk-def)
qed

lemma blocksOf-nonInitBlks:
$(\forall p\ bk: bk\in\text{blocksOf}\ s\ p\implies P\ bk)$
$\implies bk\in\text{nonInitBlks}\ s\ p\implies P\ bk$
by (auto simp add: allRdBlks-def blocksOf-def nonInitBlks-deflocksSeen-def allBlocksRead-def rdBy-def, blast)
lemma maxBlk-allInput:
  assumes inv: Inv2a s
  and mblk: maxBlk s p ∈ nonInitBlks s p
  shows inp (maxBlk s p) ∈ allInput s
proof −
  from inv
  have blocks: ∀ p bk. bk ∈ blocksOf s p
       → inp bk ∈ (allInput s) ∪ {NotAnInput}
    by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def)
  from mblk NotAnInput
  have inp (maxBlk s p) ≠ NotAnInput
    by (auto simp add: nonInitBlks-def)
  with mblk blocksOf-nonInitBlks[OF blocks]
  show ?thesis
    by auto
qed

lemma HEndPhase1-dblock-allInput:
  assumes act: HEndPhase1 s s' p
  and inv1: HInv1 s
  and inv2: Inv2a s
  shows inp': inp (dblock s' p) ∈ allInput s'
proof −
  from act
  have inpt: inpt s p ∈ allInput s'
    by (auto simp add: EndPhase1-def)
  have nonInitBlks s p ≠ {} → inp (maxBlk s p) ∈ allInput s
proof
  assume ni: nonInitBlks s p ≠ {}
  with inv1
  have maxBlk s p ∈ nonInitBlks s p
    by (auto simp add: HInv1-def maxBlk-in-nonInitBlks)
  with inv2
  show inp (maxBlk s p) ∈ allInput s
    by (blast dest: maxBlk-allInput)
qed
  with act inpt
  show ?thesis
    by (auto simp add: EndPhase1-def)
qed

lemma HEndPhase1-Inv2a-dblock:
  assumes act: HEndPhase1 s s' p
  and inv1: HInv1 s
  and inv2: Inv2a s
  and inv2c: Inv2c-inner s p
  shows Inv2a-innermost s' p (dblock s' p)
proof −
  from inv1 act have inv1': HInv1 s'
by (blast dest: HEndPhase1-HInv1)
from \( \text{inv2} \)
have \( \text{inv2a}: \text{Inv2a-innermost} s \ p \ (\text{dblock} s \ p) \)
  by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
from \( \text{act} \ \text{inv2c} \)
have \( \text{mbal}' : \text{mbal} \ (\text{dblock} s' \ p) \in \text{Ballot} p \)
  by (auto simp add: EndPhase1-def Inv2c-def Inv2c-inner-def)
moreover
from \( \text{act} \)
have \( \text{bal}' : \text{bal} \ (\text{dblock} s' \ p) = \text{mbal} \ (\text{dblock} s \ p) \)
  by (auto simp add: EndPhase1-def)
moreover
from \( \text{act} \ \text{inv} \ \text{inv2} \)
have \( \text{inp}' : \text{inp} \ (\text{dblock} s' \ p) \in \text{allInput} s' \)
  by (blast dest: HEndPhase1-dblock-allInput)
moreover
with \( \text{inv1}' \ \text{NotAnInput} \)
have \( \text{inp} \ (\text{dblock} s' \ p) \neq \text{NotAnInput} \)
  by (auto simp add: HInv1-def)
ultimately show \(?\text{thesis}\)
  using \( \text{act} \ \text{inv2a} \)
  by (auto simp add: Inv2a-innermost-def EndPhase1-def)
qed

lemma \( \text{HEndPhase1-Inv2a-dblock-q} \):  
assumes \( \text{act} : \text{HEndPhase1} s \ s' \ p \)
and \( \text{inv1} : \text{HInv1} s \)
and \( \text{inv} : \text{Inv2a} s \)
and \( \text{inv2c} : \text{Inv2c-inner} s \ p \)
shows \( \text{Inv2a-innermost} s' \ q \ (\text{dblock} s' \ q) \)
proof (cases \( \text{p} = q \))
  case True
  with \( \text{act} \ \text{inv} \ \text{inv2c} \ \text{inv1} \)
  show \(?\text{thesis}\)
    by (blast dest: HEndPhase1-Inv2a-dblock)
next
  case False
from \( \text{inv} \)
have \( \text{inv-q-dblock} : \text{Inv2a-innermost} s \ q \ (\text{dblock} s \ q) \)
  by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
with \( \text{False} \ \text{act} \)
show \(?\text{thesis}\)
  by (clarsimp simp add: EndPhase1-def HNextPart-def InitializePhase-def Inv2a-innermost-def)
qed

theorem \( \text{HEndPhase1-Inv2a} \):
assumes \( \text{act} : \text{HEndPhase1} s \ s' \ p \)
and \( \text{inv1} : \text{HInv1} s \)

36
and inv: Inv2a s
and inv2c: Inv2c-inner s p
shows Inv2a s'

proof (clarsimp simp add: Inv2a-def Inv2a-inner-def)
  fix q bk
  assume bk-in-bks: bk ∈ blocksOf s' q
  with HEndPhase1-blocksOf[OF act]
  have dblock-blocks: bk ∈ {dblock s' q} ∪ blocksOf s q
    by blast
  thus Inv2a-innermost s' q bk
next
  assume bk-in-blocks: bk ∈ blocksOf s q
  with inv
  have Inv2a-innermost s q bk
    by (auto simp add: Inv2a-def Inv2a-inner-def)
  with act show ?thesis
    by (auto simp add: EndPhase1-def HNextPart-def
      InitializePhase-def Inv2a-innermost-def)
qed

C.2.2 Proofs of Invariant 2b

Invariant 2b is proved automatically, given that we expand the definitions involved.

theorem HInit-Inv2b: HInit s → Inv2b s
by (auto simp add: HInit-def Init-def Inv2b-def Inv2b-inner-def InitDB-def)

theorem HPhase1or2ReadThen-Inv2b:
[ [ Inv2b s; HPhase1or2ReadThen s s' p d q ] ]
⇒ Inv2b s'
by (auto simp add: Phase1or2ReadThen-def Inv2b-def
  Inv2b-inner-def hasRead-def)

theorem HStartBallot-Inv2b:
[ [ Inv2b s; HStartBallot s s' p ] ]
⇒ Inv2b s'
by (auto simp add: StartBallot-def InitializePhase-def
  Inv2b-def Inv2b-inner-def hasRead-def)

theorem HPhase1or2Write-Inv2b:
[ [ Inv2b s; HPhase1or2Write s s' p d ] ]
⇒ Inv2b s'
by (auto simp add: Phase1or2Write-def Inv2b-def Inv2b-inner-def hasRead-def)

theorem HPhase1or2ReadElse-Inv2b:
  [ Inv2b s; HPhase1or2ReadElse s s' p d q ]
  \implies Inv2b s'
by (auto simp add: Phase1or2ReadElse-def StartBallot-def hasRead-def InitializePhase-def Inv2b-def Inv2b-inner-def)

theorem HEndPhase1-Inv2b:
  [ Inv2b s; HEndPhase1 s s' p ] \implies Inv2b s'
by (auto simp add: EndPhase1-def InitializePhase-def Inv2b-def Inv2b-inner-def hasRead-def)

theorem HFail-Inv2b:
  [ Inv2b s; HFail s s' p ]
  \implies Inv2b s'
by (auto simp add: Fail-def InitializePhase-def Inv2b-def Inv2b-inner-def hasRead-def)

theorem HEndPhase2-Inv2b:
  [ Inv2b s; HEndPhase2 s s' p ] \implies Inv2b s'
by (auto simp add: EndPhase2-def InitializePhase-def Inv2b-def Inv2b-inner-def hasRead-def)

theorem HPhase0Read-Inv2b:
  [ Inv2b s; HPhase0Read s s' p d ] \implies Inv2b s'
by (auto simp add: Phase0Read-def Inv2b-def Inv2b-inner-def hasRead-def)

theorem HEndPhase0-Inv2b:
  [ Inv2b s; HEndPhase0 s s' p ] \implies Inv2b s'
by (auto simp add: EndPhase0-def InitializePhase-def Inv2b-def Inv2b-inner-def hasRead-def)

C.2.3 Proofs of Invariant 2 c

theorem HInit-Inv2c:
  HInit s \longrightarrow Inv2c s
by (auto simp add: HInit-def Init-def Inv2c-def Inv2c-inner-def)

lemma HNextPart-Inv2c-chosen:
  assumes hnp: HNextPart s s'
  and inv2c: Inv2c s
  and outpt': \forall p. outpt s' p = (if phase s' p = 3
        then inp(dblock s' p)
        else NotAnInput)
  and inp-dblk: \forall p. inp (dblock s' p) \in allInput s' \cup \{NotAnInput\}
  shows chosen s' \in allInput s' \cup \{NotAnInput\}

38
using hnp outpt' inp-dblk inv2c

proof(auto simp add: HNextPart-def Inv2c-def Inv2c-inner-def
       split: if-split-asm)
qed

lemma HNextPart-chosen:
  assumes hnp: HNextPart s s'
  shows chosen s' = NotAnInput → (∀ p. outpt s' p = NotAnInput)
using hnp
proof(auto simp add: HNextPart-def split: if-split-asm)
fix p pa
assume o1: outpt s' p ≠ NotAnInput
and o2: outpt s' (SOME p. outpt s' p ≠ NotAnInput) = NotAnInput
from o1
have ∃ p. outpt s' p ≠ NotAnInput
  by auto
hence outpt s' (SOME p. outpt s' p ≠ NotAnInput) ≠ NotAnInput
  by (rule someI-ex)
with o2
show outpt s' pa ≠ NotAnInput
  by simp
qed

lemma HNextPart-allInput:
  [ HNextPart s s'; Inv2c s ] ⇒ ∀ p. inp s' p ∈ allInput s'
by (auto simp add: HNextPart-def Inv2c-def Inv2c-inner-def)

theorem HPhase1or2ReadThen-Inv2c:
  assumes inv: Inv2c s
  and act: HPhase1or2ReadThen s s' p d q
  and inv2a: Inv2a s
  shows Inv2c s'
proof -
from inv2a act
  have inv2a': Inv2a s'
    by (blast dest: HPhase1or2ReadThen-Inv2a)
from act inv
  have outpt': ∀ p. outpt s' p = (if phase s' p = 3
    then inp(dblock s' p)
    else NotAnInput)
    by (auto simp add: Phase1or2ReadThen-def Inv2c-def Inv2c-inner-def)
from inv2a'
  have dblk': ∀ p. inp (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
    by (auto simp add: Inv2a-def Inv2a-inner-def
        Inv2a-innermost-def blocksOf-def)
  with act inv outpt'
  have chosen': chosen s' ∈ allInput s' ∪ {NotAnInput}
    by (auto dest: HNextPart-Inv2c-chosen)
from act inv
have \( \forall p. \text{ inp } s' p \in \text{ allInput } s' \)
\( \land (\text{chosen } s' = \text{ NotAnInput} \rightarrow \text{ outpt } s' p = \text{ NotAnInput}) \)
by(auto dest: HNextPart-chosen HNextPart-allInput)

with outpt' chosen' act inv

show \(?\text{thesis} \)
by(auto simp add: Phase1or2ReadThen-def Inv2c-def Inv2c-inner-def)

qed

theorem HStartBallot-Inv2c:

assumes inv: Inv2c s
and act: HStartBallot s s' p
and inv2a: Inv2a s
shows Inv2c s'

proof –

from act
have phase': phase s' p = 1
by(simp add: StartBallot-def)

from act
have phase: phase s p \in \{1,2\}
by(simp add: StartBallot-def)

from act inv
have mbal': mbal(dblock s' p) \in Ballot p
by(auto simp add: StartBallot-def Inv2c-def Inv2c-inner-def)

from inv phase
have bal(dblock s p) \in Ballot p \cup \{0\}
by(auto simp add: Inv2c-def Inv2c-inner-def)

with act
have bal': bal(dblock s' p) \in Ballot p \cup \{0\}
by(auto simp add: StartBallot-def)

from act inv phase phase'
have blks': (\( \forall d. \forall br \in \text{blocksRead } s' p \ d. \) mbal(block br) < mbal(dblock s' p))
by(auto simp add: StartBallot-def InitializePhase-def
Inv2c-def Inv2c-inner-def)

from inv2a act
have inv2a': Inv2a s'
by(blast dest: HStartBallot-Inv2a)

from act inv
have outpt': \( \forall p. \text{ outpt } s' p = (\text{if phase } s' p = 3 \) then inp(dblock s' p) \else NotAnInput\)
by(auto simp add: StartBallot-def Inv2c-def Inv2c-inner-def)

from inv2a'
have dblk: \( \forall p. \text{ inp } (\text{dblock } s' p) \in \text{allInput } s' \cup \{\text{NotAnInput}\} \)
by(auto simp add: Inv2a-def Inv2a-inner-def
Inv2a-innermost-def blocksOf-def)

with act inv outpt'
have chosen': chosen s' \in \text{allInput } s' \cup \{\text{NotAnInput}\}
by(auto dest: HNextPart-Inv2c-chosen)
from act inv
have allinp: \( \forall p. \inpt s' \ p \in \text{allInput} s' \land (\text{chosen} s' = \text{NotAnInput} \rightarrow \text{outpt} s' \ p = \text{NotAnInput}) \)
by(auto dest: HNextPart-chosen HNextPart-allInput)
with phase' mbal' bal' outpt' chosen' act inv blks'
show ?thesis
by(auto simp add: StartBallot-def InitializePhase-def Inv2c-def Inv2c-inner-def)
qed

theorem HPhase1or2Write-Inv2c:
assumes inv: Inv2c s
and act: HPhase1or2Write s s' p d
and inv2a: Inv2a s
shows Inv2c s'
proof -
from inv2a act
have inv2a': Inv2a s'
  by(blast dest: HPhase1or2Write-Inv2a)
from act inv
have outpt': \( \forall p. \text{outpt} s' \ p = (\text{if phase} s' \ p = 3 \text{then inp(dblock s' p) else NotAnInput}) \)
by(auto simp add: Phase1or2Write-def Inv2c-def Inv2c-inner-def)
from inv2a'
have dblk: \( \forall p. \text{inp} (\text{dblock} s' p) \in \text{allInput} s' \cup \{\text{NotAnInput}\} \)
by(auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
with act inv outpt'
have chosen': chosen s' \in \text{allInput} s' \cup \{\text{NotAnInput}\}
  by(auto dest: HNextPart-Inv2c-chosen)
from act inv
have allinp: \( \forall p. \inpt s' \ p \in \text{allInput} s' \land (\text{chosen} s' = \text{NotAnInput} \rightarrow \text{outpt} s' \ p = \text{NotAnInput}) \)
by(auto dest: HNextPart-chosen HNextPart-allInput)
with outpt' chosen' act inv
show ?thesis
by(auto simp add: Phase1or2Write-def Inv2c-def Inv2c-inner-def)
qed

theorem HPhase1or2ReadElse-Inv2c:
[ [ Inv2c s; HPhase1or2ReadElse s s' p d q; Inv2a s ] ] \implies Inv2c s'
by(auto simp add: Phase1or2ReadElse-def dest: HStartBallot-Inv2c)

theorem HEndPhase1-Inv2c:
assumes inv: Inv2c s
and act: HEndPhase1 s s' p
and inv2a: Inv2a s
and inv1: HInv1 s
shows Inv2c s'
proof –
  from inv
  have Inv2c-inner s p by (auto simp add: Inv2c-def)
with inv2a act inv1
  have inv2a': Inv2a s'
    by (blast dest: HEndPhase1-Inv2a)
  from act inv
    have mbal': mbal (dblock s' p) ∈ Ballot p
      by (auto simp add: EndPhase1-def Inv2c-def Inv2c-inner-def)
  from act
    have bal': bal (dblock s' p) = mbal (dblock s' p)
      by (auto simp add: EndPhase1-def)
  from act inv
    have blks': (∀ d. ∀ br ∈ blocksRead s' p d.
      mbal (block br) < mbal (dblock s' p))
      by (auto simp add: EndPhase1-def InitializePhase-def
        Inv2c-def Inv2c-inner-def)
  from act inv
    have outpt': ∀ p. outpt s' p = (if phase s' p = 3
      then inp (dblock s' p)
      else NotAnInput)
      by (auto simp add: EndPhase1-def Inv2c-def Inv2c-inner-def)
  from inv2a'
    have dblk: ∀ p. inp (dblock s' p) ∈ allInput s' ∪ {NotAnInput}
      by (auto simp add: Inv2a-def Inv2a-inner-def
        Inv2a-innermost-def blocksOf-def)
  with act inv outpt'
  have chosen': chosen s' ∈ allInput s' ∪ {NotAnInput}
    by (auto dest: HNextPart-Inv2c-chosen)
  from act inv
    have allinp: ∀ p. inpt s' p ∈ allInput s'
      ∧ (chosen s' = NotAnInput
        → outpt s' p = NotAnInput)
      by (auto dest: HNextPart-chosen HNextPart-allInput)
  with mbal' bal' blks' outpt' chosen' act inv
  show ?thesis
    by (auto simp add: EndPhase1-def InitializePhase-def
      Inv2c-def Inv2c-inner-def)
qed

theorem HEndPhase2-Inv2c:
  assumes inv: Inv2c s
  and act: HEndPhase2 s s' p
  and inv2a: Inv2a s
  shows Inv2c s'
proof –
  from inv2a act

42
have \( \text{inv2a' : Inv2a s'} \)
  by (blast dest: HEndPhase2-Inv2a)
from act inv
have \( \text{outpt' : } \forall p. \text{ outpt s'} p = (\text{if phase s'} p = 3 \) 
  then \( \text{inp (dblock s'} p) \) 
  else \( \text{NotAnInput} \) 
  \)
  by (auto simp add: EndPhase2-def Inv2c-def Inv2c-inner-def 
  Inv2a-innermost-def blocksOf-def)
with act inv outpt'
have \( \text{dblk : } \forall p. \text{ inp (dblock s'} p) \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto simp add: Inv2a-def Inv2a-inner-def 
  Inv2a-innermost-def blocksOf-def)
with act inv outpt'
show \( \text{chosen' : chosen s'} \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto dest: HNextPart-Inv2c-chosen)
from act inv
have \( \text{allinp : } \forall p. \text{ inpt s'} p \in \text{allInput s'} \)
  \( \land \) (chosen s' = NotAnInput 
  \( \rightarrow \) outpt s' p = NotAnInput) 
  by (auto dest: HNextPart-chosen HNextPart-allInput)
show \( ?\text{thesis} \)
  by (auto simp add: EndPhase2-def InitializePhase-def 
  Inv2c-def Inv2c-inner-def)
qed

theorem HFail-Inv2c:
assumes \( \text{inv : Inv2c s} \)
and \( \text{act : HFail s s'} p \)
and \( \text{inv2a : Inv2a s} \)
sows \( \text{Inv2c s'} \)
proof –
from \( \text{inv2a act} \)
have \( \text{inv2a' : Inv2a s'} \)
  by (blast dest: HFail-Inv2a)
from act inv
have \( \text{outpt' : } \forall p. \text{ outpt s'} p = (\text{if phase s'} p = 3 \) 
  then \( \text{inp (dblock s'} p) \) 
  else \( \text{NotAnInput} \) 
  \)
  by (auto simp add: Fail-def Inv2c-def Inv2c-inner-def 
  Inv2a-innermost-def blocksOf-def)
from \( \text{inv2a'} \)
have \( \text{dblk : } \forall p. \text{ inp (dblock s'} p) \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto simp add: Inv2a-def Inv2a-inner-def 
  Inv2a-innermost-def blocksOf-def)
with act inv outpt'
have \( \text{chosen' : chosen s'} \in \text{allInput s'} \cup \{\text{NotAnInput}\} \)
  by (auto dest: HNextPart-Inv2c-chosen)
from act inv
have \( \text{allinp : } \forall p. \text{ inpt s'} p \in \text{allInput s'} \land \) (chosen s' = NotAnInput 
  \( \rightarrow \) outpt s' p = NotAnInput) 
  by (auto dest: HNextPart-Inv2c-chosen)
by (auto dest: HNextPart-chosen HNextPart-allInput)
with outpt' chosen' act inv
show ?thesis
by (auto simp add: Fail-def InitializePhase-def
Inv2c-def Inv2c-inner-def)

qed

theorem HPhase0Read-Inv2c:
assumes inv: Inv2c s
and act: HPhase0Read s s' p d
and inv2a: Inv2a s
shows Inv2c s'
proof
from inv2a
have inv2a': Inv2a s'
  by (blast dest: HPhase0Read-Inv2a)
from act inv
have outpt': \(\forall p. \text{outpt} \ s' \ p = (\text{if} \ phase \ s' \ p = 3 \hspace{1cm} \text{then} \ \text{inp} (\text{dblock} \ s' \ p) \hspace{1cm} \text{else} \ \text{NotAnInput})\)'
  by (auto simp add: Phase0Read-def Inv2c-def Inv2c-inner-def)
from inv2a'
have dblk: \(\forall p. \text{inp} (\text{dblock} \ s' \ p) \in \text{allInput} \ s' \cup \{\text{NotAnInput}\}\)
  by (auto simp add: Inv2a-def Inv2a-innermost-def blocksOf-def)
with act inv outpt'
have chosen': chosen' \in \text{allInput} \ s' \cup \{\text{NotAnInput}\}
  by (auto dest: HNextPart-Inv2c-chosen)
from act inv
have allinp: \(\forall p. \text{inpt} \ s' \ p \in \text{allInput} \ s' \)
  \(\land (\text{chosen} \ s' = \text{NotAnInput} \hspace{1cm} \rightarrow \text{outpt} \ s' \ p = \text{NotAnInput})\)
  by (auto dest: HNextPart-chosen HNextPart-allInput)
with outpt' chosen' act inv
show ?thesis
by (auto simp add: Phase0Read-def
Inv2c-def Inv2c-inner-def)

qed

theorem HEndPhase0-Inv2c:
assumes inv: Inv2c s
and act: HEndPhase0 s s' p
and inv2a: Inv2a s
and inv1: Inv1 s
shows Inv2c s'
proof
from inv
have Inv2c-inner s p by (auto simp add: Inv2c-def)
with inv2a act inv1

44
have inv2a': Inv2a s'
  by (blast dest: HEndPhase0-Inv2a)

hence bal': bal(dblock s' p) ∈ Ballot p ∪ {0}
  by (auto simp add: Inv2a-def Inv2a-inner-def
                                          Inv2a-innermost-def blocksOf-def)

from act inv
have mbal': mbal(dblock s' p) ∈ Ballot p
  by (auto simp add: EndPhase0-def Inv2c-def Inv2c-inner-def)

from act inv
have blk': (∀d. ∀br ∈ blocksRead s' p d.
               mbal(block br) < mbal(dblock s' p))
  by (auto simp add: EndPhase0-def InitializePhase-def
                       Inv2c-def Inv2c-inner-def)

from act inv
have outpt': ∀p. outpt s' p = (if phase s' p = 3
                       then inp(dblock s' p)
                        else NotAnInput)
  by (auto simp add: EndPhase0-def Inv2c-def Inv2c-inner-def)

from inv2a'
have blk: ∀p. inp(dblock s' p) ∈ allInput s' ∪ {NotAnInput}
  by (auto simp add: Inv2a-def Inv2a-inner-def
                     Inv2a-innermost-def blocksOf-def)

with act inv outpt'
have chosen': chosen s' ∈ allInput s' ∪ {NotAnInput}
  by (auto dest: HNextPart-Inv2c-chosen)

from act inv
have allinp: ∀p. inp s' p ∈ allInput s' ∧ (chosen s' = NotAnInput
                                             −→ outpt s' p = NotAnInput)
  by (auto dest: HNextPart-chosen HNextPart-allInput)

with mbal' bal' blk' outpt' chosen' act inv
show ?thesis
  by (auto simp add: EndPhase0-def InitializePhase-def
                     Inv2c-def Inv2c-inner-def)

qed

theorem HInit-HInv2:
  HInit s ⇒ HInv2 s
using HInit-Inv2a HInit-Inv2b HInit-Inv2c
by (auto simp add: HInv2-def)

HInv1 ∧ HInv2 is an invariant of HNext.

lemma I2b:
  assumes nxt: HNext s s'
  and inv: HInv1 s ∧ HInv2 s
  shows HInv2 s'
proof (auto simp add: HInv2-def)
  show Inv2a s' using assms
    by (auto simp add: HInv2-def HNext-def Next-def, auto intro: HStartBallot-Inv2a,
auto intro: HPhase1or2Write-Inv2a,
auto simp add: Phase1or2Read-def
  intro: HPhase1or2ReadThen-Inv2a
  HPhase1or2ReadElse-Inv2a,
auto intro: HPhase0Read-Inv2a,
auto simp add: EndPhase1or2-def Inv2c-def
  intro: HEndPhase1-Inv2a
  HEndPhase2-Inv2a,
auto intro: HFail-Inv2a,
auto simp add: HInv1-def
  intro: HEndPhase0-Inv2a)

show Inv2b s' using assms
  by(auto simp add: HInv2-def HNext-def Next-def,
      auto intro: HStartBallot-Inv2b,
      auto intro: HPhase0Read-Inv2b,
      auto intro: HPhase1or2Write-Inv2b,
      auto simp add: Phase1or2Read-def
        intro: HPhase1or2ReadThen-Inv2b
        HPhase1or2ReadElse-Inv2b,
      auto simp add: EndPhase1or2-def
        intro: HEndPhase1-Inv2b HEndPhase2-Inv2b,
      auto intro: HFail-Inv2b HEndPhase0-Inv2b)

show Inv2c s' using assms
  by(auto simp add: HInv2-def HNext-def Next-def,
      auto intro: HStartBallot-Inv2c,
      auto intro: HPhase0Read-Inv2c,
      auto intro: HPhase1or2Write-Inv2c,
      auto simp add: Phase1or2Read-def
        intro: HPhase1or2ReadThen-Inv2c
        HPhase1or2ReadElse-Inv2c,
      auto simp add: EndPhase1or2-def
        intro: HEndPhase1-Inv2c
        HEndPhase2-Inv2c,
      auto intro: HFail-Inv2c,
      auto simp add: HInv1-def intro: HEndPhase0-Inv2c)

qed

end

theory DiskPaxos-Inv3 imports DiskPaxos-Inv2 begin

C.3 Invariant 3

This invariant says that if two processes have read each other’s block from
disk d during their current phases, then at least one of them has read the
other’s current block.

definition HInv3-L :: state ⇒ Proc ⇒ Proc ⇒ Disk ⇒ bool
  where

 46
\[ H_{inv3-L} s p q d = \left( \text{phase } s p \in \{1, 2\} \land \text{phase } s q \in \{1, 2\} \land \text{hasRead } s p d q \land \text{hasRead } s q d p \right) \]

**definition** \( H_{inv3-R} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool} \)

where

\[ H_{inv3-R} s p q d = (\{\text{block} = \text{dblock } s q, \text{proc} = q\} \in \text{blocksRead } s p d \lor \{\text{block} = \text{dblock } s p, \text{proc} = p\} \in \text{blocksRead } s q d) \]

**definition** \( H_{inv3-inner} :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{Proc} \Rightarrow \text{Disk} \Rightarrow \text{bool} \)

where \( H_{inv3-inner} s p q d = (H_{inv3-L} s p q d \rightarrow H_{inv3-R} s p q d) \)

**definition** \( H_{inv3} :: \text{state} \Rightarrow \text{bool} \)

where \( H_{inv3} s = (\forall p q d. H_{inv3-inner} s p q d) \)

### C.3.1 Proofs of Invariant 3

**theorem** \( H_{init-Hinv3} : H_{init} s \Rightarrow H_{inv3} s \)

by (simp add: \( H_{init-def} \; \text{Init-def} \; H_{inv3-def} \; H_{inv3-inner-def} \; H_{inv3-L-def} \; H_{inv3-R-def} \))

**lemma** \( \text{InitPhase-Hinv3-p} \): \[ [ \text{InitializePhase } s s' p; H_{inv3-L} s' p q d ] \Rightarrow H_{inv3-R} s' p q d \]

by (auto simp add: \( \text{InitializePhase-def} \; H_{inv3-inner-def} \; \text{hasRead-def} \; H_{inv3-L-def} \; H_{inv3-R-def} \))

**lemma** \( \text{InitPhase-Hinv3-q} \): \[ [ \text{InitializePhase } s s' q ; H_{inv3-L} s' p q d ] \Rightarrow H_{inv3-R} s' p q d \]

by (auto simp add: \( \text{InitializePhase-def} \; H_{inv3-inner-def} \; \text{hasRead-def} \; H_{inv3-L-def} \; H_{inv3-R-def} \))

**lemma** \( H_{inv3-L-sym} : H_{inv3-L} s p q d \Rightarrow H_{inv3-L} s q p d \)

by (auto simp add: \( H_{inv3-L-def} \))

**lemma** \( H_{inv3-R-sym} : H_{inv3-R} s p q d \Rightarrow H_{inv3-R} s q p d \)

by (auto simp add: \( H_{inv3-R-def} \))

**lemma** \( \text{Phase1or2ReadThen-Hinv3-pq} \):

assumes \( \text{act} : \text{Phase1or2ReadThen } s s' p d q \)

and \( \text{inv-L'} : H_{inv3-L} s' p q d \)

and \( \text{pq} : p \neq q \)

and \( \text{inv2b} : \text{Inv2b } s \)

shows \( H_{inv3-R} s' p q d \)

**proof** –

from \( \text{inv-L'} \; \text{act } pq \)

have \( \text{phase } s q \in \{1, 2\} \land \text{hasRead } s q d p \)

by (auto simp add: \( \text{Phase1or2ReadThen-def} \; H_{inv3-L-def} \; \text{hasRead-def} \; \text{split: if-split-asm} \))
with inv2b
have disk s d q = dblock s q
  by (auto simp add: Inv2b-def Inv2b-inner-def
      hasRead-def)
with act
show thesis
  by (auto simp add: Phase1or2ReadThen-def HInv3-def
      HInv3-inner-def HInv3-R-def)
qed

lemma Phase1or2ReadThen-HInv3-hasRead:
  [ ¬ hasRead s pp dd qq;
    Phase1or2ReadThen s s′ p d q;
    pp≠p ∨ qq≠q ∨ dd≠d ]
  ⇒ ¬ hasRead s′ pp dd qq
  by (auto simp add: hasRead-def Phase1or2ReadThen-def)

theorem HPhase1or2ReadThen-HInv3:
  assumes act : HPhase1or2ReadThen s s′ p d q
  and inv : HInv3 s
  and pq : p≠q
  and inv2b : Inv2b s
  shows HInv3 s′
proof (clarsimp simp add: HInv3-def HInv3-inner-def)
  fix pp qq dd
  assume h3l' : HInv3-L s′ pp qq dd
  show HInv3-R s′ pp qq dd
  proof (cases HInv3-L s pp qq dd)
    case True
    with inv
    have HInv3-R s pp qq dd
      by (auto simp add: HInv3-def HInv3-inner-def)
    with act h3l'
    show thesis
      by (auto simp add: HInv3-R-def HInv3-L-def
          Phase1or2ReadThen-def)
  next
    case False
    assume nh3l: ¬ HInv3-L s pp qq dd
    show HInv3-R s′ pp qq dd
    proof (cases ((pp=p ∧ qq=q) ∨ (pp=q ∧ qq=p)) ∧ dd≠d)
      case True
      with act pq inv2b h3l' HInv3-L-sym[OF h3l']
      show thesis
        by (auto dest: Phase1or2ReadThen-HInv3-pq HInv3-R-sym)
    next
      case False
      from nh3l h3l' act
      have (¬ hasRead s pp dd ∨ ¬ hasRead s qq dd pp)
\begin{align*}
& \land \text{hasRead} \, s' \, p \, p \, d \, q \, q \land \text{hasRead} \, s' \, q \, q \, d \, p \\
& \text{by (auto simp add: HInv3-L-def Phase1or2ReadThen-def)} \\
& \text{with act False} \\
& \text{show } \neg \text{thesis} \\
& \text{by (auto dest: Phase1or2ReadThen-HInv3-hasRead)} \\
& \text{qed} \\
& \text{qed} \text{qed} \\
\end{align*}

\textbf{lemma StartBallot-HInv3-p:}
\begin{align*}
\left[ \begin{array}{l}
\text{StartBallot } s \, s' \, p; \ HInv3-L \, s' \, p \, q \, d \\
\end{array} \right] \\
\implies HInv3-R \, s' \, p \, q \, d \\
\text{by (auto simp add: StartBallot-def dest: InitPhase-HInv3-p)}
\end{align*}

\textbf{lemma StartBallot-HInv3-q:}
\begin{align*}
\left[ \begin{array}{l}
\text{StartBallot } s \, s' \, q; \ HInv3-L \, s' \, p \, q \, d \\
\end{array} \right] \\
\implies HInv3-R \, s' \, p \, q \, d \\
\text{by (auto simp add: StartBallot-def dest: InitPhase-HInv3-q)}
\end{align*}

\textbf{lemma StartBallot-HInv3-nL:}
\begin{align*}
\left[ \begin{array}{l}
\text{StartBallot } s \, s' \, t; \neg HInv3-L \, s \, p \, q \, d; \ t \neq p; \ t \neq q \\
\end{array} \right] \\
\implies \neg HInv3-L \, s' \, p \, q \, d \\
\text{by (auto simp add: StartBallot-def InitializePhase-def HInv3-L-def hasRead-def)}
\end{align*}

\textbf{lemma StartBallot-HInv3-R:}
\begin{align*}
\left[ \begin{array}{l}
\text{StartBallot } s \, s' \, t; \ HInv3-R \, s \, p \, q \, d; \ t \neq p; \ t \neq q \\
\end{array} \right] \\
\implies HInv3-R \, s' \, p \, q \, d \\
\text{by (auto simp add: StartBallot-def InitializePhase-def HInv3-R-def hasRead-def)}
\end{align*}

\textbf{lemma StartBallot-HInv3-t:}
\begin{align*}
\left[ \begin{array}{l}
\text{StartBallot } s \, s' \, t; \ HInv3-inner \, s \, p \, q \, d; \ t \neq p; \ t \neq q \\
\end{array} \right] \\
\implies HInv3-inner \, s' \, p \, q \, d \\
\text{by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-nL StartBallot-HInv3-R)}
\end{align*}

\textbf{lemma StartBallot-HInv3:}
\begin{align*}
\text{assumes act: StartBallot } s \, s' \, t \\
\text{and inv: HInv3-inner } s \, p \, q \, d \\
\text{shows } HInv3-inner \, s' \, p \, q \, d \\
\text{proof (cases } t = p \lor t = q) \\
\text{case True} \\
\text{with act inv} \\
\text{show } \neg \text{thesis} \\
\text{by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-p StartBallot-HInv3-q)}
\end{align*}

\textbf{next}
\begin{align*}
\text{case False}
\end{align*}
with inv act
show ?thesis
  by (auto simp add: HInv3-inner-def dest: StartBallot-HInv3-t)
qed

theorem HStartBallot-HInv3:
  HStartBallot s s' p; HInv3 s \implies HInv3 s'
  by (auto simp add: HInv3-def dest: StartBallot-HInv3)

theorem HPhase1or2ReadElse-HInv3:
  HPhase1or2ReadElse s s' p d q; HInv3 s \implies HInv3 s'
  by (auto simp add: Phase1or2ReadElse-def HInv3-def dest: StartBallot-HInv3)

theorem HPhase1or2Write-HInv3:
  assumes act: HPhase1or2Write s s' p d
  and inv: HInv3 s
  shows HInv3 s'
proof (auto simp add: HInv3-def)
  fix pp qq dd
  show HInv3-inner s' pp qq dd
  proof (cases HInv3-L s pp qq dd)
    case True
    with inv
    have HInv3-R s pp qq dd
      by (simp add: HInv3-def HInv3-inner-def)
    with act
    show ?thesis
      by (auto simp add: HInv3-inner-def HInv3-R-def Phase1or2Write-def)
  next
    case False
    with act
    have \neg HInv3-L s' pp qq dd
      by (auto simp add: HInv3-L-def hasRead-def Phase1or2Write-def)
    thus ?thesis
      by (simp add: HInv3-inner-def)
  qed
qed

lemma EndPhase1-HInv3-p:
  EndPhase1 s s' p; HInv3-L s' p q d \implies HInv3-R s' p q d
  by (auto simp add: EndPhase1-def dest: InitPhase-HInv3-p)

lemma EndPhase1-HInv3-q:
  EndPhase1 s s' q; HInv3-L s' p q d \implies HInv3-R s' p q d
  by (auto simp add: EndPhase1-def dest: InitPhase-HInv3-q)

lemma EndPhase1-HInv3-nL:
lemma EndPhase1-HInv3-R:
\[
\text{EndPhase1 \ s' s' } t; \ HInv3-R \ s \ p q d; \ t \neq p; \ t \neq q \]
\[\implies HInv3-R \ s' p q d\]
\by(auto simp add: EndPhase1-def InitializePhase-def
\ HInv3-R-def hasRead-def)

lemma EndPhase1-HInv3-t:
\[
\text{EndPhase1 \ s' s' } t; \ HInv3-inner \ s \ p q d; \ t \neq p; \ t \neq q \]
\[\implies HInv3-inner \ s' p q d\]
\by(auto simp add: HInv3-inner-def dest: EndPhase1-HInv3-nL
\ EndPhase1-HInv3-R)

lemma EndPhase1-HInv3:
\[
\text{assumes act: EndPhase1 \ s s' } t
\and inv: HInv3-inner \ s \ p q d
\shows HInv3-inner \ s' p q d\]
\proof(cases \(t=p \lor t=q\))
\quad \text{case True}
\quad \with \ act \ inv
\quad \show \ ?thesis
\quad \by(auto simp add: HInv3-inner-def
\ \ dest: EndPhase1-HInv3-p EndPhase1-HInv3-q)
\next
\quad \text{case False}
\quad \with \ inv \ act
\quad \show \ ?thesis
\quad \by(auto simp add: HInv3-inner-def dest: EndPhase1-HInv3-t)
\qed

theorem HEndPhase1-HInv3:
\[
\text{EndPhase1 \ s s' \ p; HInv3 \ s } \implies HInv3 \ s'
\by(auto simp add: HInv3-def dest: EndPhase1-HInv3)

lemma EndPhase2-HInv3-p:
\[
\text{EndPhase2 \ s s' \ p; HInv3-L \ s' p q d } \implies HInv3-R \ s' p q d
\by(auto simp add: EndPhase2-def dest: InitPhase-HInv3-p)

lemma EndPhase2-HInv3-q:
\[
\text{EndPhase2 \ s s' \ q; HInv3-L \ s' p q d } \implies HInv3-R \ s' p q d
\by(auto simp add: EndPhase2-def dest: InitPhase-HInv3-q)

lemma EndPhase2-HInv3-nL:
\[
\text{EndPhase2 \ s s' \ t; \ HInv3-L \ s p q d; \ t \neq p; \ t \neq q }
\implies \ HInv3-L \ s' p q d
\]
by(auto simp add: EndPhase2-def InitializePhase-def HInv3-L-def hasRead-def)

lemma EndPhase2-HInv3-R:
  \( \begin{align*}
  & [ \text{EndPhase2 } s \ s' \ t; \ HInv3-R s \ p \ q \ d; \ t \neq p; \ t \neq q ] \\
  \Rightarrow & \ HInv3-R s' \ p \ q \ d
  \end{align*} \)
by(auto simp add: EndPhase2-def InitializePhase-def HInv3-R-def hasRead-def)

lemma EndPhase2-HInv3-t:
  \( \begin{align*}
  & [ \text{EndPhase2 } s \ s' \ t; \ HInv3-inner s \ p \ q \ d; \ t \neq p; \ t \neq q ] \\
  \Rightarrow & \ HInv3-inner s' \ p \ q \ d
  \end{align*} \)
by(auto simp add: HInv3-inner-def dest: EndPhase2-HInv3-nL EndPhase2-HInv3-R)

lemma EndPhase2-HInv3:
  assumes act: EndPhase2 s s' t
  and inv: HInv3-inner s p q d
  shows \( HInv3-inner s' \ p \ q \ d \)
proof(cases t=p ∨ t=q)
case True
  with act inv
  show ?thesis
    by(auto simp add: HInv3-inner-def dest: EndPhase2-HInv3-p EndPhase2-HInv3-q)
next
case False
  with inv act
  show ?thesis
    by(auto simp add: HInv3-inner-def dest: EndPhase2-HInv3-t)
qed

theorem HEndPhase2-HInv3:
  \( \begin{align*}
  & [ \text{HEndPhase2 } s \ s' \ p; \ HInv3 s ] \\
  \Rightarrow & \ HInv3 s'
  \end{align*} \)
by(auto simp add: HInv3-def dest: EndPhase2-HInv3)

lemma Fail-HInv3-p:
  \( \begin{align*}
  & [ \text{Fail } s \ s' \ p; \ HInv3-L s' \ p \ q \ d ] \\
  \Rightarrow & \ HInv3-R s' \ p \ q \ d
  \end{align*} \)
by(auto simp add: Fail-def dest: InitPhase-HInv3-p)

lemma Fail-HInv3-q:
  \( \begin{align*}
  & [ \text{Fail } s \ s' \ q; \ HInv3-L s' \ p \ q \ d ] \\
  \Rightarrow & \ HInv3-R s' \ p \ q \ d
  \end{align*} \)
by(auto simp add: Fail-def dest: InitPhase-HInv3-q)

lemma Fail-HInv3-nL:
  \( \begin{align*}
  & [ \text{Fail } s \ s' \ t; \ \neg HInv3-L s \ p \ q \ d; \ t \neq p; \ t \neq q ] \\
  \Rightarrow & \ \neg HInv3-L s' \ p \ q \ d
  \end{align*} \)
by(auto simp add: Fail-def InitializePhase-def HInv3-L-def hasRead-def)

52
lemma Fail-HInv3-R:
\[
\begin{align*}
\text{Fail} & \quad \text{s} \quad s' \quad t; \\
& \quad \text{HInv3-R} \quad \text{s} \quad p \quad q \quad d; \\
& \quad t \neq p; \\
& \quad t \neq q \\
\implies & \quad \text{HInv3-R} \quad s' \quad p \quad q \quad d
\end{align*}
\]
by\((\text{auto simp add: Fail-def InitializePhase-def HInv3-R-def hasRead-def})\)

lemma Fail-HInv3-t:
\[
\begin{align*}
\text{Fail} & \quad \text{s} \quad s' \quad t; \\
& \quad \text{HInv3-inner} \quad \text{s} \quad p \quad q \quad d; \\
& \quad t \neq p; \\
& \quad t \neq q \\
\implies & \quad \text{HInv3-inner} \quad s' \quad p \quad q \quad d
\end{align*}
\]
by\((\text{auto simp add: HInv3-inner-def dest: Fail-HInv3-nL Fail-HInv3-R})\)

lemma Fail-HInv3:
assumes act: \(\text{Fail} \quad \text{s} \quad s' \quad t\)
and inv: \(\text{HInv3-inner} \quad \text{s} \quad p \quad q \quad d\)
shows \(\text{HInv3-inner} \quad s' \quad p \quad q \quad d\)
proof(cases \(t=p \lor t=q\))
case True
with inv act
show \(?\text{thesis}\)
  by\((\text{auto simp add: HInv3-inner-def dest: Fail-HInv3-p Fail-HInv3-q})\)
next
case False
with inv act
show \(?\text{thesis}\)
  by\((\text{auto simp add: HInv3-inner-def dest: Fail-HInv3-t})\)
qed

theorem HFail-HInv3:
\[
\begin{align*}
\text{HFail} & \quad \text{s} \quad s' \quad p; \\
& \quad \text{HInv3} \quad \text{s} \\
\implies & \quad \text{HInv3} \quad s'
\end{align*}
\]
by\((\text{auto simp add: HInv3-def dest: Fail-HInv3})\)

theorem HPhase0Read-HInv3:
assumes act: \(\text{HPhase0Read} \quad \text{s} \quad s' \quad p \quad d\)
and inv: \(\text{HInv3} \quad \text{s}\)
shows \(\text{HInv3} \quad s'\)
proof\((\text{auto simp add: HInv3-def})\)
fix pp qq dd
show \(\text{HInv3-inner} \quad s' \quad pp \quad qq \quad dd\)
proof(cases \(\text{HInv3-L} \quad \text{s} \quad pp \quad qq \quad dd\))
case True
with inv
have \(\text{HInv3-R} \quad \text{s} \quad pp \quad qq \quad dd\)
  by\((\text{simp add: HInv3-def HInv3-inner-def})\)
with act
show \(?\text{thesis}\)
  by\((\text{auto simp add: HInv3-inner-def HInv3-R-def Phase0Read-def})\)

53
next
  case False
  with act
  have ¬HInv3-L s' pp qq dd
    by (auto simp add: HInv3-L-def hasRead-def Phase0Read-def)
  thus ?thesis
    by (simp add: HInv3-inner-def)
  qed

lemma EndPhase0-HInv3-p:
  [ [ EndPhase0 s s' p; HInv3-L s' p q d ] ]
  \implies HInv3-R s' p q d
by (auto simp add: EndPhase0-def dest: InitPhase-HInv3-p)

lemma EndPhase0-HInv3-q:
  [ [ EndPhase0 s s' q; HInv3-L s' p q d ] ]
  \implies HInv3-R s' p q d
by (auto simp add: EndPhase0-def dest: InitPhase-HInv3-q)

lemma EndPhase0-HInv3-nL:
  [ [ EndPhase0 s s' t; ¬HInv3-L s p q d; t\neq p; t\neq q ] ]
  \implies ¬HInv3-L s' p q d
by (auto simp add: EndPhase0-def InitializePhase-def
                      HInv3-L-def hasRead-def)

lemma EndPhase0-HInv3-R:
  [ [ EndPhase0 s s' t; HInv3-R s p q d; t\neq p; t\neq q ] ]
  \implies HInv3-R s' p q d
by (auto simp add: EndPhase0-def InitializePhase-def
                      HInv3-R-def hasRead-def)

lemma EndPhase0-HInv3-t:
  [ [ EndPhase0 s s' t; HInv3-inner s p q d; t\neq p; t\neq q ] ]
  \implies HInv3-inner s' p q d
by (auto simp add: HInv3-inner-def
                      dest: EndPhase0-HInv3-nL EndPhase0-HInv3-R)

lemma EndPhase0-HInv3:
  assumes act: EndPhase0 s s' t
  and inv: HInv3-inner s p q d
  shows HInv3-inner s' p q d
proof (cases t=p \or t=q)
  case True
  with act inv
  show ?thesis
    by (auto simp add: HInv3-inner-def
                      dest: EndPhase0-HInv3-p EndPhase0-HInv3-q)
next
case False
with inv act
show ?thesis
  by (auto simp add: HInv3-inner-def dest: EndPhase0-HInv3-t)
qed

theorem HEndPhase0-HInv3:
  [ HEndPhase0 s s' p; HInv3 s ] \implies HInv3 s'
by (auto simp add: HInv3-def dest: EndPhase0-HInv3)

HInv1 \land HInv2 \land HInv3 is an invariant of HNext.

lemma I2c:
  assumes nxt: HNext s s'
and inv: HInv1 s \land HInv2 s \land HInv3 s
shows HInv3 s' using assms
by (auto simp add: HNext-def Next-def,
  auto intro: HStartBallot-HInv3,
  auto intro: HPhase0Read-HInv3,
  auto intro: HPhase1or2Write-HInv3,
  auto simp add: Phase1or2Read-def HInv2-def
  intro: HPhase1or2ReadThen-HInv3
  HPhase1or2ReadElse-HInv3,
  auto simp add: EndPhase1or2-def
  intro: HEndPhase1-HInv3
  HEndPhase2-HInv3,
  auto intro: HFail-HInv3,
  auto intro: HEndPhase0-HInv3)
end

theory DiskPaxos-Inv4 imports DiskPaxos-Inv2 begin

C.4 Invariant 4

This invariant expresses relations among mbal and bal values of a processor
and of its disk blocks. HInv4a asserts that, when p is not recovering from
a failure, its mbal value is at least as large as the bal field of any of its
blocks, and at least as large as the mbal field of its block on some disk in
any majority set. HInv4b conjunct asserts that, in phase 1, its mbal value
is actually greater than the bal field of any of its blocks. HInv4c asserts that,
in phase 2, its bal value is the mbal field of all its blocks on some majority
set of disks. HInv4d asserts that the bal field of any of its blocks is at most
as large as the mbal field of all its disk blocks on some majority set of disks.

definition MajoritySet :: Disk set set
  where MajoritySet = \{ D. IsMajority(D) \}

definition HInv4a1 :: state \Rightarrow Proc \Rightarrow bool


where $HInv4a1 \ s \ p = (\forall bk \in blocksOf \ s \ p. \ \text{bal} \ bk \leq \text{mbal} \ (\text{dblock} \ s \ p))$

**definition** $HInv4a2 :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$

where $HInv4a2 \ s \ p = (\forall D \in \text{MajoritySet}. (\exists d \in D. \ \text{mbal}(\text{disk} \ s \ d \ p) \leq \text{mbal}(\text{dblock} \ s \ p))$

**definition** $HInv4a :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$

where $HInv4a \ s \ p = (\exists D \in \text{MajoritySet}. (\exists d \in D. \ \text{mbal}(\text{disk} \ s \ d \ p) \leq \text{mbal}(\text{dblock} \ s \ p))$

**definition** $HInv4b :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$

where $HInv4b \ s \ p = (\exists D \in \text{MajoritySet}. (\exists d \in D. \ \text{mbal}(\text{disk} \ s \ d \ p) = \text{bal} \ (\text{dblock} \ s \ p)))$

**definition** $HInv4c :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$

where $HInv4c \ s \ p = (\exists D \in \text{MajoritySet}. (\exists d \in D. \ \text{mbal}(\text{disk} \ s \ d \ p) = \text{bal} \ (\text{dblock} \ s \ p)))$

**definition** $HInv4d :: \text{state} \Rightarrow \text{Proc} \Rightarrow \text{bool}$

where $HInv4d \ s \ p = (\exists D \in \text{MajoritySet}. (\exists d \in D. \ \text{mbal}(\text{disk} \ s \ d \ p) = \text{bal} \ (\text{dblock} \ s \ p)))$

**definition** $HInv4 :: \text{state} \Rightarrow \text{bool}$

where $HInv4 \ s = (\forall p. \ HInv4a \ s \ p \wedge HInv4b \ s \ p \wedge HInv4c \ s \ p \wedge HInv4d \ s \ p)$

The initial state implies Invariant 4.

**theorem** $HInit-HInv4$: $HInit \ s \Rightarrow HInv4 \ s$

**using** Disk-isMajority

**by** (auto simp add: HInit-def Init-def HInv4-def HInv4a-def HInv4a1-def HInv4a2-def HInv4b-def HInv4c-def HInv4d-def MajoritySet-def blocksOf-def InitDB-def rdBy-def)

To prove that the actions preserve $HInv4$, we do it for one conjunct at a time.

For each action $\text{actionss'}q$ and conjunct $x \in a, b, c, d$ of $HInv4x's'p$, we prove two lemmas. The first lemma $\text{action-HInv4x-p}$ proves the case of $p = q$, while lemma $\text{action-HInv4x-q}$ proves the other case.

**C.4.1 Proofs of Invariant 4a**

**lemma** $HStartBallot-HInv4a1$:

**assumes** act: $HStartBallot \ s \ s' \ p$

**and** inv: $HInv4a1 \ s \ p$

**and** inv2a: $\text{Inv2a-inner} \ s' \ p$

**shows** $HInv4a1 \ s' \ p$

**proof** (auto simp add: HInv4a1-def)

fix $bk$
assume $bk \in \text{blocksOf } s' \ p$

with $H\text{StartBallot-blocsOf[OF act]}$

have $bk \in \{\text{dblock } s' \ p\} \cup \text{blocksOf } s \ p$

by blast

thus $\text{bal } bk \leq \text{mbal } (\text{dblock } s' \ p)$

proof

assume $bk \in \{\text{dblock } s' \ p\}$

with $\text{inv2a}$

show $\text{thesis}$

by (auto simp add: $\text{Inv2a-innermost-def Inv2a-inner-def blocksOf-def}$)

next

assume $bk \in \text{blocksOf } s \ p$

with $\text{inv act}$

show $\text{thesis}$

by (auto simp add: $\text{StartBallot-def HInv4a1-def}$)

qed

qed

lemma $H\text{StartBallot-HInv4a2}$:

assumes $\text{act: HStartBallot } s \ s' \ p$

and $\text{inv: HInv4a2 } s \ p$

shows $\text{HInv4a2 } s' \ p$

proof (auto simp add: $\text{HInv4a2-def}$)

fix $D$

assume $D\text{maj: } D \in \text{MajoritySet}$

from $\text{inv Dmaj}$

have $\exists d \in D. \text{ mbal } (\text{disk } s \ d \ p) \leq \text{mbal } (\text{dblock } s \ p)$

$\land \text{bal } (\text{disk } s \ d \ p) \leq \text{bal } (\text{dblock } s \ p)$

by (auto simp add: $\text{HInv4a2-def}$)

then obtain $d$

where $d \in D$

$\land \text{mbal } (\text{disk } s \ d \ p) \leq \text{mbal } (\text{dblock } s \ p)$

$\land \text{bal } (\text{disk } s \ d \ p) \leq \text{bal } (\text{dblock } s \ p)$

by auto

with $\text{act}$

have $d \in D$

$\land \text{mbal } (\text{disk } s' \ d \ p) \leq \text{mbal } (\text{dblock } s' \ p)$

$\land \text{bal } (\text{disk } s' \ d \ p) \leq \text{bal } (\text{dblock } s' \ p)$

by (auto simp add: $\text{StartBallot-def}$)

with $\text{Dmaj}$

show $\exists d \in D. \text{ mbal } (\text{disk } s' \ d \ p) \leq \text{mbal } (\text{dblock } s' \ p)$

$\land \text{bal } (\text{disk } s' \ d \ p) \leq \text{bal } (\text{dblock } s' \ p)$

by auto

qed

lemma $H\text{StartBallot-HInv4a-p}$:

assumes $\text{act: HStartBallot } s \ s' \ p$

and $\text{inv: HInv4a } s \ p$

and $\text{inv2a: Inv2a-inner } s' \ p$
shows $H_{\text{Inv4a}} \ s \ s' \ p$
using act inv inv2a
proof –
  from act
  have phase: $0 < \text{phase} \ s \ p$
    by(auto simp add: StartBallot-def)
  from act inv inv2a
  show ?thesis
    by(auto simp del: HStartBallot-def simp add: HInv4a-def phase
      elim: HStartBallot-HInv4a1 HStartBallot-HInv4a2)
qed

lemma HStartBallot-HInv4a-q:
  assumes act: $H_{\text{StartBallot}} \ s \ s' \ p$
  and inv: $H_{\text{Inv4a}} \ s \ q$
  and pnq: $p \neq q$
  shows $H_{\text{Inv4a}} \ s' \ q$
proof –
  from act pnq
  have blocksOf $s' \ q \subseteq \text{blocksOf} \ s \ q$
    by(auto simp add: StartBallot-def InitializePhase-def
      blocksOf-def rdBy-def)
  with act inv pnq
  show ?thesis
    by(auto simp add: StartBallot-def HInv4a-def
      HInv4a1-def HInv4a2-def)
qed

theorem HStartBallot-HInv4a:
  assumes act: $H_{\text{StartBallot}} \ s \ s' \ p$
  and inv: $H_{\text{Inv4a}} \ s \ q$
  and inv2a: Inv2a $s' \ s$
  shows $H_{\text{Inv4a}} \ s' \ q$
proof(cases $p=q$)
  case True
    from inv2a
    have Inv2a-inner $s' \ p$
      by(auto simp add: Inv2a-def)
    with act inv True
    show ?thesis
      by(blast dest: HStartBallot-HInv4a-p)
  next
  case False
    with act inv
    show ?thesis
      by(blast dest: HStartBallot-HInv4a-q)
qed

lemma Phase1or2Write-HInv4a1:
lemma Phase1or2Write-HInv4a1:
[ Phase1or2Write s s' p d; HInv4a1 s q ] \implies HInv4a1 s' q
by (auto simp add: Phase1or2Write-def HInv4a1-def blocksOf-def rdBy-def)

lemma Phase1or2Write-HInv4a2:
[ Phase1or2Write s s' p d; HInv4a2 s q ] \implies HInv4a2 s' q
by (auto simp add: Phase1or2Write-def HInv4a2-def)

theorem HPhase1or2Write-HInv4a:
assumes act: HPhase1or2Write s s' p d
and inv: HInv4a s q
shows HInv4a s' q
proof
  from act
  have phase': phase s = phase s'
    by (simp add: Phase1or2Write-def)
  show ?thesis
    proof (cases phase s q = 0)
    case True
    with phase' act
    show ?thesis
      by (auto simp add: HInv4a-def)
  next
    case False
    with phase' act inv
    show ?thesis
      by (auto simp add: HInv4a-def
        dest: Phase1or2Write-HInv4a1 Phase1or2Write-HInv4a2)
  qed
qed

lemma HPhase1or2ReadThen-HInv4a1-p:
assumes act: HPhase1or2ReadThen s s' p d q
and inv: HInv4a1 s p
shows HInv4a1 s' p
proof (auto simp: HInv4a1-def)
  fix bk
  assume bk: bk \in blocksOf s' p
  with HPhase1or2ReadThen-blocksOf[OF act]
  have bk \in blocksOf s p by auto
  with inv act
  show bal bk \leq mbal (dblock s' p)
    by (auto simp add: HInv4a1-def Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv4a2:
[ HPhase1or2ReadThen s s' p d r; HInv4a2 s q ] \implies HInv4a2 s' q
by (auto simp add: Phase1or2ReadThen-def HInv4a2-def)
lemma HPhase1or2ReadThen-HInv4a-p:
assumes act: HPhase1or2ReadThen s s' p d r
and inv: HInv4a s p
and inv2b: Inv2b s
shows HInv4a s' p
proof –
  from act inv2b
  have phase s p ∈ {1, 2}
    by (auto simp add: Phase1or2ReadThen-def Inv2b-def Inv2b-inner-def)
  with act inv
  show ?thesis
    by (auto simp add: HPhase1or2ReadThen-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

lemma HPhase1or2ReadThen-HInv4a-q:
assumes act: HPhase1or2ReadThen s s' p d r
and inv: HInv4a s q
and pnq: p ≠ q
shows HInv4a s' q
proof –
  from act pnq
  have blocksOf s' q ⊆ blocksOf s q
    by (auto simp add: Phase1or2ReadThen-def InitializePhase-def blocksOf-def rdBy-def)
  with act inv pnq
  show ?thesis
    by (auto simp add: Phase1or2ReadThen-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

theorem HPhase1or2ReadThen-HInv4a:
[ HPhase1or2ReadThen s s' p d r; HInv4a s q; Inv2b s ] ⇒ HInv4a s' q
by (blast dest: HPhase1or2ReadThen-HInv4a-p HPhase1or2ReadThen-HInv4a-q)

theorem HPhase1or2ReadElse-HInv4a:
assumes act: HPhase1or2ReadElse s s' p d r
and inv: HInv4a s q and inv2a: Inv2a s'
shows HInv4a s' q
proof –
  from act have HStartBallot s s' p
    by (simp add: Phase1or2ReadElse-def)
  with inv inv2a show ?thesis
    by (blast dest: HStartBallot-HInv4a)
qed

lemma HEndPhase1-HInv4a1:
assumes act: HEndPhase1 s s' p
and inv: HInv4a1 s p

shows $H_{Inv4a1} s' p$

proof (auto simp add: $H_{Inv4a1-def}$)
  fix $bk$
  assume $bk$: $bk \in \text{blocksOf } s'$
  from $bk$ $H_{EndPhase1-blocksOf[OF act]}$
  have $bk \in \{\text{dblock } s' p\} \cup \text{blocksOf } s p$
    by blast
  with $act inv$
  show $bal bk \leq mbal (\text{dblock } s' p)$
    by (auto simp add: $H_{Inv4a-def}$ $H_{Inv4a1-def}$ $EndPhase1-def$)
qed

lemma $H_{EndPhase1-HInv4a2}$:
  assumes $act$: $H_{EndPhase1} s s'$
  and $inv$: $H_{Inv4a2} s p$
  and $inv2a$: $Inv2a s$
  shows $H_{Inv4a2} s' p$
proof (auto simp add: $H_{Inv4a2-def}$)
  fix $D$
  assume $Dmaj$: $D \in \text{MajoritySet}$
  from $inv Dmaj$
  have $\exists d \in D. \ mbal (\text{disk } s d p) \leq mbal (\text{dblock } s p)$
    $\land bal (\text{disk } s d p) \leq bal (\text{dblock } s p)$
    by (auto simp add: $H_{Inv4a2-def}$)
  then obtain $d$
    where $d-cond$: $d \in D$
      $\land mbal (\text{disk } s d p) \leq mbal (\text{dblock } s p)$
      $\land bal (\text{disk } s d p) \leq bal (\text{dblock } s p)$
      by auto
  have $\text{disk } s d p \in \text{blocksOf } s p$
    by (auto simp add: $\text{blocksOf-def}$)
  with $inv2a$
  have $\text{bal(disk } s d p) \leq mbal (\text{disk } s d p)$
    by (auto simp add: $\text{Inv2a-def}$ $\text{Inv2a-inner-def}$ $\text{Inv2a-innermost-def}$)
  with $act d-cond$
  have $d \in D$
    $\land mbal (\text{disk } s' d p) \leq mbal (\text{dblock } s' p)$
    $\land bal (\text{disk } s' d p) \leq bal (\text{dblock } s' p)$
    by (auto simp add: $EndPhase1-def$)
  with $Dmaj$
  show $\exists d \in D. \ mbal (\text{disk } s' d p) \leq mbal (\text{dblock } s' p)$
    $\land bal (\text{disk } s' d p) \leq bal (\text{dblock } s' p)$
    by auto
qed

lemma $H_{EndPhase1-HInv4a-p}$:
  assumes $act$: $H_{EndPhase1} s s'$
  and $inv$: $H_{Inv4a} s p$
  and $inv2a$: $Inv2a s$


shows $H_{Inv4a} s' p$

proof –

from act
have phase: $0 < \text{phase}\ s\ p$
  by (auto simp add: EndPhase1-def)
with act inv inv2a
show ?thesis
  by (auto simp del: HEndPhase1-def simp add: HInv4a-def
  elim: HEndPhase1-HInv4a1 HEndPhase1-HInv4a2)
qed

lemma HEndPhase1-HInv4a-q:
  assumes act: $H_{EndPhase1} s\ s'\ p$
  and inv: $H_{Inv4a} s\ q$
  and pnq: $p \neq q$
  shows $H_{Inv4a} s'\ q$

proof –

from act pnq
have dblock $s'\ q = \text{dblock}\ s\ q \land \text{disk}\ s' = \text{disk}\ s$
  by (auto simp add: EndPhase1-def)
moreover
from act pnq
have $\forall\ p\ d. \text{rdBy}\ s'\ q\ p\ d \subseteq \text{rdBy}\ s\ q\ p\ d$
  by (auto simp add: EndPhase1-def InitializePhase-def rdBy-def)
  hence $(\bigcup\ p\ d. \text{rdBy}\ s'\ q\ p\ d) \subseteq (\bigcup\ p\ d. \text{rdBy}\ s\ q\ p\ d)$
  by (auto, blast)
ultimately
have blocksOf $s'\ q \subseteq \text{blocksOf}\ s\ q$
  by (auto simp add: blocksOf-def, blast)
with act inv pnq
show ?thesis
  by (auto simp add: EndPhase1-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

theorem HEndPhase1-HInv4a:
  $[\ [ H_{EndPhase1} s\ s'\ p; H_{Inv4a} s\ q ] ] \Rightarrow H_{Inv4a} s'\ q$
by (blast dest: HEndPhase1-HInv4a-p HEndPhase1-HInv4a-q)

theorem HFail-HInv4a:
  $[\ [ H_{Fail} s\ s'\ p; H_{Inv4a} s\ q ] ] \Rightarrow H_{Inv4a} s'\ q$
by (auto simp add: Fail-def HInv4a1-def HInv4a2-def InitializePhase-def
  blocksOf-def rdBy-def)

theorem HPhase0Read-HInv4a:
  $[\ [ H_{Phase0Read} s\ s'\ p\ d; H_{Inv4a} s\ q ] ] \Rightarrow H_{Inv4a} s'\ q$
by (auto simp add: Phase0Read-def HInv4a1-def HInv4a2-def InitializePhase-def
  blocksOf-def rdBy-def)
**Theorem HEndPhase2-HInv4a:**

\[ \text{HEndPhase2} \ s \ s' \ p \ \text{HInv4a} \ s \ q \implies \text{HInv4a} \ s' \ q \]

by (auto simp add: HEndPhase2-def HInv4a-def HInv4a1-def HInv4a2-def InitializePhase-def blocksOf-def rdBy-def)

**Lemma allSet:**

assumes aPQ: \( \forall \ a. \ \forall \ r \in P \ a. \ Q \ r \) and \( \exists \ rb: \ rb \in P \ d \)

shows \( Q \ rb \)

proof –

from aPQ have \( \forall \ r \in P \ d. \ Q \ r \) by auto

with \( \exists \ rb \)

show \( \text{thesis} \) by auto

qed

**Lemma EndPhase0-44:**

assumes act: \( \text{EndPhase0} \ s \ s' \ p \)
and \( \exists \ bk: \ bk \in \text{blocksOf} \ s \ p \)
and \( \exists \ inv4d: \ \text{HInv4d} \ s \ p \)
and \( \exists \ inv2c: \ \text{Inv2c-inner} \ s \ p \)

shows \( \exists \ d. \ \exists \ rb \in \text{blocksRead} \ s \ p \ d. \ \text{bal} \ bk \leq \text{mbal(block} \ rb) \)

proof –

from \( \exists \ bk \ inv4d \)

have \( \exists \ D1 \in \text{MajoritySet}. \forall \ d \in D1. \ \text{bal} \ bk \leq \text{mbal(disk} \ s \ d \ p) \) — 4.2

by (auto simp add: HInv4d-def)

with \( \text{majorities-intersect} \)

have \( \text{p43: } \forall \ D \in \text{MajoritySet}. \exists \ d \in D. \ \text{bal} \ bk \leq \text{mbal(disk} \ s \ d \ p) \)

by (simp add: MajoritySet-def, blast)

from act

have \( \forall \ d. \ \text{phase} \ s \ p = 0 \) by (simp add: EndPhase0-def)

with \( \exists \ inv2c \)

have \( \forall \ d. \ \exists \ rb \in \text{blocksRead} \ s \ p \ d. \ \text{block} \ rb = \text{disk} \ s \ d \ p \) — 5.1

by (simp add: Inv2c-inner-def)

hence \( \forall \ d. \ \text{hasRead} \ s \ p \ d \)

\[ \implies \exists \ rb \in \text{blocksRead} \ s \ p \ d. \ \text{block} \ rb = \text{disk} \ s \ d \ p \] — 5.2

(is \( \forall \ d. \ ?H \ d \implies \ ?P \ d \)

by (auto simp add: hasRead-def)

with \( \exists \ inv2c \)

have \( \text{p53: } \exists \ D \in \text{MajoritySet}. \ \forall \ d \in D. \ ?P \ d \)

by (auto simp add: MajoritySet-def EndPhase0-def)

show \( \text{thesis} \)

proof –

from \( \text{p43 p53} \)

have \( \exists \ D \in \text{MajoritySet}. \ (\exists \ d \in D. \ \text{bal} \ bk \leq \text{mbal(disk} \ s \ d \ p)) \)

\( \land \ (\forall \ d \in D. \ ?P \ d) \)

by auto

thus \( \text{thesis} \)

by force

qed
lemma $H_{\text{EndPhase0}}$-$H_{\text{Inv}4a1}$-p;
assumes act: $H_{\text{EndPhase0}} s s’ p$
and inv2a’: Inv2a $s’$
and inv2c: Inv2c-inner $s p$
and inv4d: $H_{\text{Inv}4d} s p$
shows $H_{\text{Inv}4a1} s’ p$
proof (auto simp add: $H_{\text{Inv}4a1}$-def)
fix bk
assume bk: $bk \in \text{blocksOf } s’ p$
with $H_{\text{EndPhase0}}$-blocksOf[OF act]
have bk \in \{dblock $s’ p\} \cup \text{blocksOf } s p$ by auto
thus $\text{bal } bk \leq \text{mbal (dblock } s’ p)$
proof
assume bk: $bk \in \{dblock s’ p\}$
with inv2a’
have Inv2a-innermost $s’ p bk$
by(auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
with bk show ?thesis
by(auto simp add: Inv2a-innermost-def)
next
assume bk: $bk \in \text{blocksOf } s p$
from act
have $f1: \forall r \in \text{allBlocksRead } s p. \text{mbal } r < \text{mbal (dblock } s’ p)$
by(auto simp add: $H_{\text{EndPhase0}}$-def)
with act inv4d inv2c bk
have $\exists d. \exists rb \in \text{blocksRead } s p d. \text{bal } bk \leq \text{mbal (block } rb)$
by(auto dest: $H_{\text{EndPhase0}}$-44)
with $f1$
show ?thesis
by(auto simp add: $H_{\text{EndPhase0}}$-def allBlocksRead-def allRdBlks-def dest: allSet)
qed

lemma hasRead-allBlks:
assumes inv2c: Inv2c-inner $s p$
and phase: phase $s p = 0$
shows $(\forall d \in \{d. \text{hasRead } s p d p\}. \text{disk } s d p \in \text{allBlocksRead } s p)$
proof
fix d
assume d: $d \in \{d. \text{hasRead } s p d p\}$ (is $d \in ?D$)
hence br-ne: blocksRead $s p d \neq \{\}$
by (auto simp add: hasRead-def)
from inv2c phase
have $\forall br \in \text{blocksRead } s p d. \text{block } br = \text{disk } s d p$
by(auto simp add: Inv2c-inner-def)
with br-ne
have disk s d p ∈ block \in blocksRead s p d
by force
thus disk s d p ∈ allBlocksRead s p
by (auto simp add: allBlocksRead-def allRdBlks-def)

qed

lemma HEndPhase0-41:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and inv2c: Inv2c-inner s p
shows \exists D\in MajoritySet. \forall d\in D. mbal(disk s d p) ≤ mbal(dblock s' p)
∧ bal(disk s d p) ≤ bal(dblock s' p)

proof –
from act HEndPhase0-some[OF act inv1]
have p51: \forall br\in allBlocksRead s p. mbal br < mbal(dblock s' p)
∧ bal br ≤ bal(dblock s' p)
and a: IsMajority({d. hasRead s p d p})
and phase: phase s p = 0
by (auto simp add: EndPhase0-def)+
from inv2c phase
have (\forall d\in {d. hasRead s p d p}. disk s d p \in allBlocksRead s p)
by (auto dest: hasRead-allBlks)
with p51
have (\forall d\in {d. hasRead s p d p}. mbal(disk s d p) ≤ mbal(dblock s' p)
∧ bal(disk s d p) ≤ bal(dblock s' p))
by force
with a show thesis
by (auto simp add: MajoritySet-def)

qed

lemma Majority-exQ:
assumes asm1: \exists D \in MajoritySet. \forall d \in D. P d
shows \forall D \in MajoritySet. \exists d \in D. P d
using asm1
proof (auto simp add: MajoritySet-def)
fix D1 D2
assume D1: IsMajority D1 and D2: IsMajority D2
and Px: \forall x\in D1. P x
from D1 D2 majorities-intersect
have \exists d\in D1. d\in D2 by auto
with Px
show \exists x\in D2. P x
by auto

qed

lemma HEndPhase0-HInv4a2-p:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and \( \text{inv2c}: \text{Inv2c-inner} \ s \ p \)
shows \( \text{HInv4a2} \ s' \ p \)
proof (simp add: HInv4a2-def)
  from act
  have disk': disk s' = disk s
    by (simp add: EndPhase0-def)
  from act inv1 inv2c
  have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ mbal(\text{disk} \ s \ d \ p) \leq \mbal(\text{dblock} \ s' \ p) \)
    \( \wedge \) \( \text{bal}(\text{disk} \ s \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \)
    by (blast dest: HEndPhase0-41)
  from Majority-exQ[OF this]
  have \( \forall D \in \text{MajoritySet}. \exists d \in D. \ mbal(\text{disk} \ s \ d \ p) \leq \mbal(\text{dblock} \ s' \ p) \)
    \( \wedge \) \( \text{bal}(\text{disk} \ s \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \)
  (is \( ?P (\text{disk} \ s) \) )
  from ssusb[OF disk', of \( ?P \), OF this]
  show \( \forall D \in \text{MajoritySet}. \exists d \in D. \ mbal(\text{disk} \ s' \ d \ p) \leq \mbal(\text{dblock} \ s' \ p) \)
    \( \wedge \) \( \text{bal}(\text{disk} \ s' \ d \ p) \leq \text{bal}(\text{dblock} \ s' \ p) \).
qed

lemma HEndPhase0-HInv4a-p:
  assumes act: HEndPhase0 \ s \ s' \ p
  and \( \text{inv2a}: \text{Inv2a} \ s \)
  and \( \text{inv2}: \text{Inv2c} \ s \)
  and \( \text{inv4d}: \text{HInv4d} \ s \ p \)
  and \( \text{inv1}: \text{Inv1} \ s \)
  and inv: HInv4a s p
  shows HInv4a s' p
proof –
  from inv2
  have inv2c: Inv2c-inner \ s \ p
    by (auto simp add: Inv2c-def)
  with inv1 inv2a act
  have inv2a': Inv2a \ s'
    by (blast dest: HEndPhase0-Inv2a)
  from act
  have phase \ s' \ p = 1
    by (auto simp add: EndPhase0-def)
  with act inv inv2c inv4d inv2a' inv1
  show \ ?thesis
    by (auto simp add: HInv4a-def simp del: HEndPhase0-def
      elim: HEndPhase0-HInv4a1-p HEndPhase0-HInv4a2-p)
quod

lemma HEndPhase0-HInv4a-q:
  assumes act: HEndPhase0 \ s \ s' \ p
  and inv: HInv4a \ s \ q
  and p\neq q
  shows HInv4a \ s' \ q
proof –
from act pnq
have \( \text{dblock}\ s\ s' = \text{dblock}\ s\ q \land \text{disk}\ s' = \text{disk}\ s \)
  by (auto simp add: EndPhase0-def)
moreover
from act pnq
have \( \forall\ p\ d.\ \text{rdBy}\ s\ s' p\ d \subseteq \text{rdBy}\ s\ q\ p\ d \)
  by (auto simp add: EndPhase0-def InitializePhase-def rdBy-def)
hence \( (\forall\ p\ d.\ \text{rdBy}\ s\ s' q\ p\ d) \subseteq (\forall\ p\ d.\ \text{rdBy}\ s\ q\ p\ d) \)
  by (auto, blast)
ultimately
have \( \text{blocksOf}\ s\ s' q \subseteq \text{blocksOf}\ s\ q \)
  by (auto simp add: blocksOf-def, blast)
with act inv pnq
show \( \text{thesis} \)
  by (auto simp add: EndPhase0-def HInv4a-def HInv4a1-def HInv4a2-def)
qed

theorem HEndPhase0-HInv4a:
[ [ HEndPhase0 s s' p; HInv4a s q; HInv4d s p; Inv2a s; Inv1 s; Inv2a s; Inv2c s ]]
\( \Rightarrow \) HInv4a s' q
by (blast dest: HEndPhase0-HInv4a-p HEndPhase0-HInv4a-q)

C.4.2 Proofs of Invariant 4b

lemma blocksRead-allBlocksRead:
\( r b \in \text{blocksRead}\ s\ p\ d \Rightarrow \text{block}\ r b \in \text{allBlocksRead}\ s\ p \)
by (auto simp add: allBlocksRead-def allRdBlks-def)

lemma HEndPhase0-dbloc-mbal:
[ [ HEndPhase0 s s' p ] ]
\( \Rightarrow \forall\ b r \in\ \text{allBlocksRead}\ s\ p.\ \text{mbal}\ b r < \text{mbal}(\text{dblock}\ s' p) \)
by (auto simp add: EndPhase0-def)

lemma HEndPhase0-HInv4b-p-dbloc:
assumes act: HEndPhase0 s s' p
and inv1: Inv1 s
and inv2a: Inv2a s
and inv2c: Inv2c-inner s p
shows \( \text{bal}(\text{dblock}\ s' p) < \text{mbal}(\text{dblock}\ s' p) \)
proof –
from act have phase s p = 0 by (auto simp add: EndPhase0-def)
with inv2c
have \( \forall\ d.\forall\ b r \in\ \text{blocksRead}\ s\ p\ d.\ \text{proc}\ b r = p \land \text{block}\ b r = \text{disk}\ s\ d\ p \)
  by (auto simp add: Inv2c-inner-def)
hence \( \text{allBlks-in-blocksOf}.\ \text{allBlocksRead}\ s\ p \subseteq \text{blocksOf}\ s\ p \)
  by (auto simp add: allBlocksRead-def allRdBlks-def blocksOf-def)
from act HEndPhase0-some[OF act inv1]
have \( p53: \exists \ br \in \text{allBlocksRead} \ s \ p, \ \text{bal}(d\text{block } s' \ p) = \text{bal} \ br \)
by (auto simp add: EndPhase0-def)
from inv2a
have \( i2: \forall \ p. \ \forall \ bk \in \text{blocksOf} \ s \ p, \ \text{bal} \ bk \leq \text{mbal} \ bk \)
by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def)
with allBlks-in-blocksOf
have \( \forall \ bk \in \text{allBlocksRead} \ s \ p, \ \text{bal} \ bk \leq \text{mbal} \ bk \)
by auto
with \( p53 \)
have \( \exists \ br \in \text{allBlocksRead} \ s \ p, \ \text{bal}(d\text{block } s' \ p) \leq \text{mbal} \ br \)
by force
with HEndPhase0-dblock-mbal[OF act]
show \( ?\text{thesis} \)
by auto
qed

lemma HEndPhase0-HInv4b-p-blocksOf:
assumes \( \text{act} : \text{HEndPhase0} \ s \ s' s \)
and \( \text{inv4d} : \text{HInv4d} \ s \ p \)
and \( \text{inv2c} : \text{Inv2c-inner} \ s \ p \)
and \( \text{bk} : \ bk \in \text{blocksOf} \ s \ p \)
shows \( \text{bal} \ bk < \text{mbal}(d\text{block } s' \ p) \)
proof –
from inv4d
have \( \text{majories-intersect} \ bk \)
by (auto simp add: HInv4d-def MajoritySet-def Majority-exQ)
with \( p53 \)
have \( \exists \ br \in \text{allBlocksRead} \ s \ p, \ \text{bal}(d\text{block } s' \ p) \leq \text{mbal} \ br \)
by auto
with act
have \( \text{maj} : \text{IsMajority}(\{d. \ \text{hasRead } s \ p \ d \}) \ (\text{is IsMajority}(\{D\}) \)
and \( \text{phase} : \text{phase } s \ p = 0 \)
by (simp add: EndPhase0-def)+
with \( \text{br-ne} : \forall d \in \{D. \ \text{blocksRead } s \ p \ d \neq \{\} \)
by (auto simp add: hasRead-def)
from phase
have \( \forall d \in \{D. \ \exists \br \in \text{blocksRead } s \ p \ d, \ \text{block } br = \text{disk } s \ d \)
by (auto simp add: Inv2c-inner-def)
with \( \text{br-ne} \)
have \( \forall d \in \{D. \ \exists \br \in \text{allBlocksRead } s \ p, \ \text{br} = \text{disk } s \ d \)
by (blast dest: blocksRead-allBlocksRead)
with \( p53 \)
show \( ?\text{thesis} \)
by (auto simp add: MajoritySet-def)
qed
with HEndPhase0-dblock-mbal[OF act]
show \( ?\text{thesis} \)
by auto
qed
lemma HEndPhase0-HInv4b-p:
  assumes act: HEndPhase0 s s' p
  and inv4d: HInv4d s p
  and inv1: Inv1 s
  and inv2a: Inv2a s
  and inv2c: Inv2c-inner s p
  shows HInv4b s' p
proof(clarsimp simp add: HInv4b-def)
  from act
  have phase: phase s p = 0
    by(auto simp add: EndPhase0-def)
  fix bk
  assume bk: bk∈ blocksOf s' p
  with HEndPhase0-blocksOf[OF act]
  have bk∈{dblock s' p} ∨ bk∈blocksOf s p
    by blast
  thus bal bk < mbal (dblock s' p)
proof
  assume bk: bk∈{dblock s' p}
  with act inv1 inv2a inv2c
  show ?thesis
    by(auto simp del: HEndPhase0-def dest: HEndPhase0-HInv4b-p-dblock)
next
  assume bk: bk ∈ blocksOf s p
  with act inv2c inv4d
  show ?thesis
    by(blast dest: HEndPhase0-HInv4b-p-blocksOf)
qed
qed

lemma HEndPhase0-HInv4b-q:
  assumes act: HEndPhase0 s s' p
  and pnq: p≠q
  and inv: HInv4b s q
  shows HInv4b s' q
proof –
  from act pnq
  have disk': disk s'=disk s
    and dblock': dblock s' q=dblock s q
    and phase': phase s' q =phase s q
    by(auto simp add: EndPhase0-def)
  from act pnq
  have blocksRead': ∃ q. allRdBlks s' q ⊆ allRdBlks s q
    by(auto simp add: EndPhase0-def InitializePhase-def allRdBlks-def)
  with disk' dblock'
  have blocksOf s' q ⊆ blocksOf s q
    by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  with inv phase' dblock'
show \( ?\text{thesis} \)
by (auto simp add: HInv4b-def)
qed

**Theorem** \( H\text{EndPhase0-HInv4b} \):
assumes \( \text{act}: H\text{EndPhase0} \ s \ s' \ p \)
and \( \text{inv}: H\text{Inv4b} \ s \ q \)
and \( \text{inv4d}: H\text{Inv4d} \ s \ p \)
and \( \text{inv1}: \text{Inv1} \ s \)
and \( \text{inv2a}: \text{Inv2a} \ s \)
and \( \text{inv2c}: \text{Inv2c-inner} \ s \ p \)
shows \( H\text{Inv4b} \ s' \ q \)
proof (cases \( p=q \))
case True
with \( H\text{EndPhase0-HInv4b-p} \ [+ \ OF \ act \ inv4d \ inv1 \ inv2a \ inv2c] \)
show \( ?\text{thesis} \) by simp
next
case False
from \( H\text{EndPhase0-HInv4b-q} \ [+ \ OF \ act \ False \ inv] \)
show \( ?\text{thesis} \).
qed

**Lemma** \( H\text{StartBallot-HInv4b-p} \):
assumes \( \text{act}: H\text{StartBallot} \ s \ s' \ p \)
and \( \text{inv2a}: \text{Inv2a-innermost} \ s \ p \ (\text{dblock} \ s \ p) \)
and \( \text{inv4b}: H\text{Inv4b} \ s \ p \)
and \( \text{inv4a}: H\text{Inv4a} \ s \ p \)
shows \( H\text{Inv4b} \ s' \ p \)
proof (clarsimp simp add: HInv4b-def)
fix \( bk \)
assume \( bk: bk \in \text{blocksOf} \ s' \ p \)
from \( \text{act} \)
have \( \text{phase'}: \text{phase} \ s' \ p = 1 \)
and \( \text{phase}: \text{phase} \ s \ p \in \{1,2\} \)
by (auto simp add: StartBallot-def)
from \( \text{act} \)
have \( p42: \ \text{mbal} \ (\text{dblock} \ s \ p) < \text{mbal} \ (\text{dblock} \ s' \ p) \)
\( \land \ \text{bal}((\text{dblock} \ s \ p)) = \text{bal}((\text{dblock} \ s' \ p)) \)
by (auto simp add: StartBallot-def)
from \( H\text{StartBallot-blocksOf}[OF \ act] \ bk \)
have \( bk \in \{\text{dblock} \ s' \ p\} \cup \text{blocksOf} \ s \ p \)
by blast
thus \( \text{bal} \ bk < \text{mbal} \ (\text{dblock} \ s' \ p) \)
proof
assume \( bk: bk \in \{\text{dblock} \ s' \ p\} \)
from \( \text{inv2a} \)
have \( \text{bal} \ (\text{dblock} \ s \ p) \leq \text{mbal} \ (\text{dblock} \ s \ p) \)
by (auto simp add: Inv2a-innermost-def)
with \( p42 \ bk \)
show ?thesis by auto

next
assume bk: bk ∈ blocksOf s p
from phase inv4a
have p41: HInv4a1 s p
  by(auto simp add: HInv4a1-def)
with p42 bk
show ?thesis
  by(auto simp add: HInv4a1-def)
qed

lemma HStartBallot-HInv4b-q:
assumes act: HStartBallot s s' p
and p≠q
and inv: HInv4b s q
shows HInv4b s' q
proof (auto)
from act p≠q
have disk': disk s' = disk s
  by(auto simp add: StartBallot-def)
with act disk' have blocksRead' : \forall q. allRdBlks s' q ⊆ allRdBlks s q
  by(auto simp add: StartBallot-def InitializePhase-def allRdBlks-def)
with inv blocksRead' have blocksOf s' q ⊆ blocksOf s q
  by(auto simp add: blocksOf-def rdBy-def, blast)
show ?thesis
  by(auto simp add: HInv4b-def)
qed

theorem HStartBallot-HInv4b:
assumes act: HStartBallot s s' p
and inv2a: Inv2a s
and inv4b: HInv4b s q
and inv4a: HInv4a s p
shows HInv4b s' q
using act inv2a inv4b inv4a
proof (auto)
case True
from inv2a
have inv2a-innermost s p (dblock s p)
  by(auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def)
with act True inv4b inv4a
show ?thesis
  by(blast dest: HStartBallot-HInv4b)
next
case False
with act inv4b
by (blast dest: HStartBallot-HInv4b-q)
qed

theorem HPhase1or2Write-HInv4b:
[ HPhase1or2Write s s' p d; HInv4b s q ] \implies HInv4b s' q
by (auto simp add: Phase1or2Write-def HInv4b-def
    blocksOf-def rdBy-def)

lemma HPhase1or2ReadThen-HInv4b-p:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv4b s p
  shows HInv4b s' p
proof
  from HPhase1or2ReadThen-blocksOf[OF act] inv act
  show ?thesis
  by (auto simp add: HInv4b-def Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv4b-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4b s q
  and p\neq q
  shows HInv4b s' q
using assms HPhase1or2ReadThen-blocksOf[OF act]
by (auto simp add: Phase1or2ReadThen-def HInv4b-def)

theorem HPhase1or2ReadThen-HInv4b:
[ HPhase1or2ReadThen s s' p d q; HInv4b s r ] \implies HInv4b s' r
by (blast dest: HPhase1or2ReadThen-HInv4b-p
    HPhase1or2ReadThen-HInv4b-q)

theorem HPhase1or2ReadElse-HInv4b:
[ HPhase1or2ReadElse s s' p d q; HInv4b s r;
  Inv2a s; HInv4a s p ]
\implies HInv4b s' r
using HStartBallot-HInv4b
by (auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase1-HInv4b-p:
  HEndPhase1 s s' p \implies HInv4b s' p
by (auto simp add: EndPhase1-def HInv4b-def)

lemma HEndPhase1-HInv4b-q:
  assumes act: HEndPhase1 s s' p
  and p\neq q
and \( \text{inv}: HInv4b \ s \ q \)

shows \( HInv4b \ s' \ q \)

proof —

from act pnq
have disk': disk s'=disk s
    and dblock': dblock s' q=dblock s q
    and phase': phase s' q =phase s q
    by(auto simp add: EndPhase1-def)
from act pnq
have blocksRead': \( \forall \ q. \ \text{allRdBlks} \ s' \ q \subseteq \text{allRdBlks} \ s \ q \)
    by(auto simp add: EndPhase1-def InitializePhase-def allRdBlks-def)
with disk' dblock'
have blocksOf s' q \subseteq blocksOf s q
    by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis
    by(auto simp add: HInv4b-def)
qed

theorem HEndPhase1-HInv4b:
assumes act: HEndPhase1 s s' p
and \( \text{inv}: HInv4b \ s \ q \)
shows HInv4b s' q
proof(cases p=q)
  case True
  with HEndPhase1-HInv4b-p[OF act]
  show ?thesis by simp
next
  case False
  from HEndPhase1-HInv4b-q[OF act False inv]
  show ?thesis .
qed

lemma HEndPhase2-HInv4b-p:
HEndPhase2 s s' p \( \Rightarrow \) HInv4b s' p
by(auto simp add: EndPhase2-def HInv4b-def)

lemma HEndPhase2-HInv4b-q:
assumes act: HEndPhase2 s s' p
and pnq: p\#q
and \( \text{inv}: HInv4b \ s \ q \)
shows HInv4b s' q
proof —
from act pnq
have disk': disk s'=disk s
    and dblock': dblock s' q=dblock s q
    and phase': phase s' q =phase s q
    by(auto simp add: EndPhase2-def)
from act pnq

73
have blocksRead' \( \forall q. \) allRdBlks \( s' q \subseteq \) allRdBlks \( s q \)
by (auto simp add: EndPhase2-def InitializePhase-def allRdBlks-def)
with disk' dblock'
have blocksOf \( s' q \subseteq \) blocksOf \( s q \)
by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis
by (auto simp add: HInv4b-def)
qed

theorem HEndPhase2-HInv4b:
assumes act: HEndPhase2 \( s s' p \)
and inv: HInv4b \( s q \)
shows HInv4b \( s' q \)
proof (cases \( p=q \))
case True
with HEndPhase2-HInv4b-p[OF act]
show ?thesis by simp
next
case False
from act pnq
have disk': disk \( s' = disk \ s \)
and dblock': dblock \( s' q = dblock \ s q \)
and phase': phase \( s' q = phase \ s q \)
by (auto simp add: Fail-def)
from act pnq
have blocksRead': \( \forall q. \) allRdBlks \( s' q \subseteq \) allRdBlks \( s q \)
by (auto simp add: Fail-def InitializePhase-def allRdBlks-def)
with disk' dblock'
have blocksOf \( s' q \subseteq \) blocksOf \( s q \)
by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis
by (auto simp add: HInv4b-def)
qed

lemma HFail-HInv4b-p:
HFail \( s s' p \implies \) HInv4b \( s' p \)
by (auto simp add: Fail-def HInv4b-def)

lemma HFail-HInv4b-q:
assumes act: HFail \( s s' p \)
and pnq: \( p \neq q \)
and inv: HInv4b \( s q \)
shows HInv4b \( s' q \)
proof --
from act pnq
have disk': disk \( s' = disk \ s \)
and dblock': dblock \( s' q = dblock \ s q \)
and phase': phase \( s' q = phase \ s q \)
by (auto simp add: Fail-def)
from act pnq
have blocksRead': \( \forall q. \) allRdBlks \( s' q \subseteq \) allRdBlks \( s q \)
by (auto simp add: Fail-def InitializePhase-def allRdBlks-def)
with disk' dblock'
have blocksOf \( s' q \subseteq \) blocksOf \( s q \)
by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
with inv phase' dblock'
show ?thesis
by (auto simp add: HInv4b-def)
qed
theorem HFail-HInv4b:
assumes act: HFail s s' p
and inv: HInv4b s q
shows HInv4b s' q
proof (cases p=q)
case True
with HFail-HInv4b-p[OF act]
show ?thesis by simp
next
case False
from HFail-HInv4b-q[OF act False inv]
show ?thesis .
qed

lemma HPhase0Read-HInv4b-p:
HPhase0Read s s' p d \Rightarrow HInv4b s' p
by (auto simp add: HPhase0Read-def HInv4b-def)

lemma HPhase0Read-HInv4b-q:
assumes act: HPhase0Read s s' p d
and p=q: p\neq q
and inv: HInv4b s q
shows HInv4b s' q
proof
from act p=q
have disk': disk s'=disk s
and dblock': dblock s' q=dblock s q
and phase': phase s' q =phase s q
by (auto simp add: HPhase0Read-def)
from HPhase0Read-blocksOf[OF act] inv phase' dblock'
show ?thesis
by (auto simp add: HInv4b-def)
qed

theorem HPhase0Read-HInv4b:
assumes act: HPhase0Read s s' p d
and inv: HInv4b s q
shows HInv4b s' q
proof (cases p=q)
case True
with HPhase0Read-HInv4b-p[OF act]
show ?thesis by simp
next
case False
from HPhase0Read-HInv4b-q[OF act False inv]
show ?thesis .
qed
C.4.3 Proofs of Invariant 4c

lemma \( \text{HStartBallot-HInv4c-p} \):
\[
\begin{align*}
[ & \text{HStartBallot} \ s \ s' \ p; \ \text{Hinv4c} \ s \ p ] \implies \text{Hinv4c} \ s' \ p \\
\text{by} & (\text{auto simp add: StartBallot-def HInv4c-def})
\end{align*}
\]

lemma \( \text{HStartBallot-HInv4c-q} \):
\[
\begin{align*}
\text{assumes} & \ \text{act}: \ \text{HStartBallot} \ s \ s' \ p \\
\text{and} & \ \text{inv}: \ \text{Hinv4c} \ s \ q \\
\text{and} & \ \text{pnq}: \ p \neq q \\
\text{shows} & \ \text{Hinv4c} \ s' \ q
\end{align*}
\]
\[
\text{proof} -
\begin{align*}
& \text{from} \ \text{act} \ \text{pnq} \\
& \text{have} \ \text{phase}: \ \text{phase} \ s' \ q = \text{phase} \ s \ q \\
& \text{and} \ \text{dblock}: \ \text{dblock} \ s \ q = \text{dblock} \ s' \ q \\
& \text{and} \ \text{disk}: \ \text{disk} \ s' = \text{disk} \ s \\
& \text{by} (\text{auto simp add: StartBallot-def}) \\
& \text{with} \ \text{inv} \\
& \text{show} \ \text{?thesis} \\
& \text{by} (\text{auto simp add: HInv4c-def})
\end{align*}
\]

qed

theorem \( \text{HStartBallot-HInv4c} \):
\[
\begin{align*}
[ & \text{HStartBallot} \ s \ s' \ p; \ \text{Hinv4c} \ s \ q ] \implies \text{Hinv4c} \ s' \ q \\
\text{by} & (\text{blast dest: HStartBallot-HInv4c-p HStartBallot-HInv4c-q})
\end{align*}
\]

lemma \( \text{HPhase1or2Write-HInv4c-p} \):
\[
\begin{align*}
\text{assumes} & \ \text{act}: \ \text{HPhase1or2Write} \ s \ s' \ p \ d \\
\text{and} & \ \text{inv}: \ \text{Hinv4c} \ s \ p \\
\text{and} & \ \text{inv2c}: \ \text{Inv2c} \ s \\
\text{shows} & \ \text{Hinv4c} \ s' \ p
\end{align*}
\]
\[
\text{proof}(\text{cases phase} \ s' \ p = 2)
\]
\[
\begin{align*}
& \text{assume} \ \text{phase'}: \ \text{phase} \ s' \ p = 2 \\
& \text{show} \ \text{?thesis} \\
& \text{proof}(\text{auto simp add: HInv4c-def phase'-MajoritySet-def}) \\
& \text{from} \ \text{act} \ \text{phase'} \\
& \text{have} \ \text{bal}: \ \text{bal}(\text{dblock} \ s' \ p) = \text{bal}(\text{dblock} \ s \ p) \\
& \text{and} \ \text{phase}: \ \text{phase} \ s \ p = 2 \\
& \text{by} (\text{auto simp add: Phase1or2Write-def}) \\
& \text{from} \ \text{phase'} \ \text{inv2c} \ \text{act} \\
& \text{have} \ \text{mbal}(\text{disk} \ s' \ d \ p) = \text{bal}(\text{dblock} \ s \ p) \\
& \text{by} (\text{auto simp add: Phase1or2Write-def Inv2c-def Inv2c-inner-def}) \\
& \text{with} \ \text{bal} \\
& \text{have} \ \text{bal}(\text{dblock} \ s' \ p) = \text{mbal}(\text{disk} \ s' \ d \ p) \\
& \text{by} \ \text{auto} \\
& \text{with} \ \text{inv} \ \text{phase} \ \text{act} \\
& \text{show} \ \exists D. \ \text{IsMajority} \ D \\
& \ \wedge (\forall d \in D. \ \text{mbal} (\text{disk} \ s' \ d \ p) = \text{bal} (\text{dblock} \ s' \ p)) \\
& \text{by} (\text{auto simp add: HInv4c-def Phase1or2Write-def MajoritySet-def})
\end{align*}
\]

qed
next
  case False
  with act
  show "thesis"
      by (auto simp add: HInv4c-def Phase1or2Write-def)
  qed

lemma HPhase1or2Write-HInv4c-q:
  assumes act: HPhase1or2Write s s' p d
  and inv: HInv4c s q
  and pnq: p\neq q
  shows HInv4c s' q
proof
  from act pnq
  have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: \forall d. disk s' d q = disk s d q
  by (auto simp add: Phase1or2Write-def)
  with inv
  show "thesis"
      by (auto simp add: HInv4c-def)
  qed

theorem HPhase1or2Write-HInv4c:
  [ [ HPhase1or2Write s s' p d; HInv4c s q; Inv2c s ] ]
  \implies HInv4c s' q
by (blast dest: HPhase1or2Write-HInv4c-p
      HPhase1or2Write-HInv4c-q)

lemma HPhase1or2ReadThen-HInv4c-p:
  [ [ HPhase1or2ReadThen s s' p d q; HInv4c s p ] ] \implies HInv4c s' p
by (auto simp add: Phase1or2ReadThen-def HInv4c-def)

lemma HPhase1or2ReadThen-HInv4c-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4c s q
  and pnq: p\neq q
  shows HInv4c s' q
proof
  from act pnq
  have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: disk s' = disk s
  by (auto simp add: Phase1or2ReadThen-def)
  with inv
  show "thesis"
      by (auto simp add: HInv4c-def)
  qed
theorem HPhase1or2ReadThen-HInv4c:
\[
[ \text{HPhase1or2ReadThen } s s' p d r; \text{HInv4c } s q ] \implies \text{HInv4c } s' q
\]
by (blast dest: HPhase1or2ReadThen-HInv4c-p HPhase1or2ReadThen-HInv4c-q)

theorem HPhase1or2ReadElse-HInv4c:
\[
[ \text{HPhase1or2ReadElse } s s' p d r; \text{HInv4c } s q ] \implies \text{HInv4c } s' q
\]
using HStartBallot-HInv4c
by (auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase1-HInv4c-p:
assumes act: \text{HEndPhase1 } s s' p
and inv2b: \text{Inv2b } s
shows \text{HInv4c } s' p
proof
  from act
  have maj: IsMajority \{d. d \in \text{disksWritten } s p \\
  \land (\forall q \in (\text{UNIV} - \{p\}). \text{hasRead } s p d q)\} \\
  is \text{IsMajority } ?M
  by (simp add: HEndPhase1-def)
  from inv2b
  have \forall d \in ?M. \text{disk } d p = \text{dblock } s p
  by (auto simp add: Inv2b-def Inv2b-inner-def)
  with act maj
  show ?thesis
  by (auto simp add: HInv4c-def HEndPhase1-def MajoritySet-def)
qed

lemma HEndPhase1-HInv4c-q:
assumes act: \text{HEndPhase1 } s s' p
and inv: \text{HInv4c } s q
and pnq: p \neq q
shows \text{HInv4c } s' q
proof
  from act pnq
  have phase: phase s' q = phase s q \\
  and dblock: \text{dblock } s q = \text{dblock } s' q \\
  and disk: disk s' = disk s
  by (auto simp add: EndPhase1-def)
  with inv
  show ?thesis
  by (auto simp add: HInv4c-def)
qed

theorem HEndPhase1-HInv4c:
\[
[ \text{HEndPhase1 } s s' p; \text{HInv4c } s q; \text{Inv2b } s ] \implies \text{HInv4c } s' q
\]
by (blast dest: HEndPhase1-HInv4c-p HEndPhase1-HInv4c-q)
lemma HEndPhase2-HInv4c-p:
  \([ H\text{EndPhase2} s s' p; H\text{Inv4c} s p] \Rightarrow H\text{Inv4c} s' p\)
by\((\text{auto simp add: EndPhase2-def HInv4c-def})\)

lemma HEndPhase2-HInv4c-q:
  assumes act: H\text{EndPhase2} s s' p
  and inv: H\text{Inv4c} s q
  and pnq: \(p \neq q\)
  shows H\text{Inv4c} s' q
proof –
from act pnq
have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: disk s' = disk s
by\((\text{auto simp add: EndPhase2-def})\)
with inv
show \?thesis
  by\((\text{auto simp add: HInv4c-def})\)
qed

theorem HEndPhase2-HInv4c:
  \([ H\text{EndPhase2} s s' p; H\text{Inv4c} s q] \Rightarrow H\text{Inv4c} s' q\)
by\((\text{blast dest: HEndPhase2-HInv4c-p HEndPhase2-HInv4c-q})\)

lemma HFail-HInv4c-p:
  \([ H\text{Fail} s s' p; H\text{Inv4c} s p] \Rightarrow H\text{Inv4c} s' p\)
by\((\text{auto simp add: Fail-def HInv4c-def})\)

lemma HFail-HInv4c-q:
  assumes act: H\text{Fail} s s' p
  and inv: H\text{Inv4c} s q
  and pnq: \(p \neq q\)
  shows H\text{Inv4c} s' q
proof –
from act pnq
have phase: phase s' q = phase s q
  and dblock: dblock s q = dblock s' q
  and disk: disk s' = disk s
by\((\text{auto simp add: Fail-def})\)
with inv
show \?thesis
  by\((\text{auto simp add: HInv4c-def})\)
qed

theorem HFail-HInv4c:
  \([ H\text{Fail} s s' p; H\text{Inv4c} s q] \Rightarrow H\text{Inv4c} s' q\)
by\((\text{blast dest: HFail-HInv4c-p HFail-HInv4c-q})\)

lemma HPhase0Read-HInv4c-p:
[ \text{HPhase0Read} s \ s' \ p \ d; \ \text{HInv4c} s \ p] \implies \text{HInv4c} s' \ p

\textbf{by} (\text{auto simp add: HPhase0Read-def HInv4c-def})

\textbf{lemma} \ HPhase0Read-HInv4c-q:
\begin{itemize}
  \item \textbf{assumes} \ act: \ HPhase0Read s \ s' \ p \ d
  \item \textbf{and} \ inv: \ HInv4c s \ q
  \item \textbf{and} \ pnq: \ p \neq q
\end{itemize}
\textbf{shows} \ HInv4c s' \ q
\textbf{proof} –
\begin{itemize}
  \item \textbf{from} \ act \ pnq
  \item \textbf{have} \ phase: \ phase s' q = phase s q
  \item \textbf{and} \ dblock: \ dblock s q = dblock s' q
  \item \textbf{and} \ disk: \ disk s' = disk s
  \item \textbf{by} (\text{auto simp add: HPhase0Read-def})
  \item \textbf{with} \ inv
  \item \textbf{show} \ ?\textbf{thesis}
  \item \textbf{by} (\text{auto simp add: HInv4c-def})
\end{itemize}
\textbf{qed}

\textbf{theorem} \ HPhase0Read-HInv4c:
[ \text{HPhase0Read} s \ s' \ p \ d; \ \text{HInv4c} s \ q] \implies \text{HInv4c} s' \ q
\textbf{by} (\text{blast dest: HPhase0Read-HInv4c-p HPhase0Read-HInv4c-q})

\textbf{lemma} \ HEndPhase0-HInv4c-p:
[ \text{HEndPhase0} s \ s' \ p; \ \text{HInv4c} s \ p] \implies \text{HInv4c} s' \ p
\textbf{by} (\text{auto simp add: HEndPhase0-def HInv4c-def})

\textbf{lemma} \ HEndPhase0-HInv4c-q:
\begin{itemize}
  \item \textbf{assumes} \ act: \ HEndPhase0 s \ s' \ p
  \item \textbf{and} \ inv: \ HInv4c s \ q
  \item \textbf{and} \ pnq: \ p \neq q
\end{itemize}
\textbf{shows} \ HInv4c s' \ q
\textbf{proof} –
\begin{itemize}
  \item \textbf{from} \ act \ pnq
  \item \textbf{have} \ phase: \ phase s' q = phase s q
  \item \textbf{and} \ dblock: \ dblock s q = dblock s' q
  \item \textbf{and} \ disk: \ disk s' = disk s
  \item \textbf{by} (\text{auto simp add: HEndPhase0-def})
  \item \textbf{with} \ inv
  \item \textbf{show} \ ?\textbf{thesis}
  \item \textbf{by} (\text{auto simp add: HInv4c-def})
\end{itemize}
\textbf{qed}

\textbf{theorem} \ HEndPhase0-HInv4c:
[ \text{HEndPhase0} s \ s' \ p; \ \text{HInv4c} s \ q] \implies \text{HInv4c} s' \ q
\textbf{by} (\text{blast dest: HEndPhase0-HInv4c-p HEndPhase0-HInv4c-q})
C.4.4 Proofs of Invariant 4d

lemma HStartBallot-HInv4d-p:
  assumes act: HStartBallot s s′ p
  and inv: HInv4d s p
  shows HInv4d s′ p
proof(clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk ∈ blocksOf s′ p
  from act have bal′: bal (dblock s′ p) = bal (dblock s p)
    by(auto simp add: StartBallot-def)
  from subsetD[OF HStartBallot-blocksOf[OF act] bk]
  have ∃ D∈ MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s d p)
  proof
    assume bk: bk ∈ blocksOf s p
    with inv show ?thesis
      by(auto simp add: HInv4d-def)
  next
    assume bk: bk ∈ {dblock s′ p}
    with bal′ inv
    show ?thesis
      by(auto simp add: HInv4d-def blocksOf-def)
  qed
  with act show ∃ D∈ MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s′ d p)
    by(auto simp add: StartBallot-def)
  qed

lemma HStartBallot-HInv4d-q:
  assumes act: HStartBallot s s′ p
  and inv: HInv4d s q
  and pnq: p̸=q
  shows HInv4d s′ q
proof
  from act pnq have disk′: disk s′=disk s
    and dblock′: dblock s′ q=dblock s q
    by(auto simp add: StartBallot-def)
  from act pnq have blocksRead′: ∀ q. allRdBlks s′ q ⊆ allRdBlks s q
    by(auto simp add: StartBallot-def InitializePhase-def allRdBlks-def)
  with disk′ dblock′ have blocksOf s′ q ⊆ blocksOf s q
    by(auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  from subsetD[OF this] inv
  have ∀ bk∈blocksOf s′ q.
    ∃ D∈ MajoritySet. ∀ d∈D. bal bk ≤ mbal (disk s d q)
    by(auto simp add: HInv4d-def)
with \(\text{disk}'\)

show \(\text{?thesis}\)

by (auto simp add: HInv4d-def)

qed

theorem \(\text{HStartBallot-HInv4d}:\)

\[
\begin{array}{ll}
\text{HStartBallot } s \ s' \ p; \ HInv4d \ s \ q \ \implies \ HInv4d s' q
\end{array}
\]

by (blast dest: HStartBallot-HInv4d-p HStartBallot-HInv4d-q)

lemma \(\text{HPhase1or2Write-HInv4d-p}:\)

assumes act: \(\text{HPhase1or2Write } s \ s' \ p \ d\)

and inv: \(\text{HInv4d } s \ q\)

and inv4a: \(\text{HInv4a } s \ p\)

shows \(\text{HInv4d } s' p\)

proof (clarsimp simp add: HInv4d-def)

fix bk

assume bk: \(bk \in \text{blocksOf } s' \ p\)

from act

have ddisk: \(\forall dd. \ \text{disk } s' dd p = (\text{if } d = dd \ \text{then } \text{dblock } s \ p \ \text{else disk } s \ dd p)\)

and phase: \(\text{phase } s \ p \neq 0\)

by (auto simp add: Phase1or2Write-def)

from inv subsetD[OF \text{HPhase1or2Write-blocksOf}[OF \text{act}] bk]

have asm3: \(\exists D \in \text{MajoritySet}. \ \forall dd \in D. \ \text{bal } bk \leq \text{mbal } (\text{disk } s \ dd p)\)

by (auto simp add: HInv4d-def)

from phase inv4a subsetD[OF \text{HPhase1or2Write-blocksOf}[OF \text{act}] bk] ddisk

have p41: \(\text{bal } bk \leq \text{mbal } (\text{disk } s' \ d p)\)

by (auto simp add: HInv4a-def HInv4a1-def)

with ddisk asm3

show \(\exists D \in \text{MajoritySet}. \ \forall dd \in D. \ \text{bal } bk \leq \text{mbal } (\text{disk } s' \ dd p)\)

by (auto simp add: MajoritySet-def split: if-split-asm)

qed

lemma \(\text{HPhase1or2Write-HInv4d-q}:\)

assumes act: \(\text{HPhase1or2Write } s \ s' \ p \ d\)

and inv: \(\text{HInv4d } s \ q\)

and pnq: \(p \neq q\)

shows \(\text{HInv4d } s' q\)

proof –

from act pnq

have disk': \(\forall d. \ \text{disk } s' d q = \text{disk } s \ d q\)

by (auto simp add: Phase1or2Write-def)

from act pnq

have blocksRead': \(\forall q. \ \text{allRdBlks } s' q \subseteq \text{allRdBlks } s \ q\)

by (auto simp add: Phase1or2Write-def

InitializePhase-def allRdBlks-def)

with act pnq

have blocksOf s' q \(\subseteq\) blocksOf s q

82
by (auto simp add: Phase1or2Write-def allRdBlks-def blocksOf-def rdBy-def)

from subsetD[OF this] inv
have \( \forall bk \in \text{blocksOf } s' q. \exists D \in \text{MajoritySet}. \forall d \in D. \text{bal } bk \leq \text{mbal}(\text{disk } s d q) \)
  by (auto simp add: HInv4d-def)

with disk'
show ?thesis
by (auto simp add: HInv4d-def)
qed

theorem HPhase1or2Write-HInv4d:
  \[ \text{HPhase1or2Write } s s' p d; \text{HInv4d } s q; \text{HInv4a } s p \] \( \Rightarrow \) \text{HInv4d } s' q
by (blast dest: HPhase1or2Write-HInv4d-p HPhase1or2Write-HInv4d-q)

lemma HPhase1or2ReadThen-HInv4d-p:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: HInv4d s p
  shows HInv4d s' p
proof (clarsimp simp add: HInv4d-def)
fix bk
assume bk: bk \( \in \) blocksOf s' p
from act
have bal': bal (\text{dblock } s' p) = bal (\text{dblock } s p)
  by (auto simp add: Phase1or2ReadThen-def)
from subsetD[OF HPhase1or2ReadThen-blocksOf[OF act] bk] inv
have \( \exists D \in \text{MajoritySet}. \forall d \in D. \text{bal } bk \leq \text{mbal}(\text{disk } s d p) \)
  by (auto simp add: HInv4d-def)
with act
show \( \exists D \in \text{MajoritySet}. \forall d \in D. \text{bal } bk \leq \text{mbal}(\text{disk } s' d p) \)
  by (auto simp add: Phase1or2ReadThen-def)
qed

lemma HPhase1or2ReadThen-HInv4d-q:
  assumes act: HPhase1or2ReadThen s s' p d r
  and inv: HInv4d s q
  and pnq: p \( \neq \) q
  shows HInv4d s' q
proof
from act pnq
have disk': disk s' = disk s
  by (auto simp add: Phase1or2ReadThen-def)
from act pnq
have blocksOf s' q \( \subseteq \) blocksOf s q
  by (auto simp add: Phase1or2ReadThen-def allRdBlks-def blocksOf-def rdBy-def)
from subsetD[OF this] inv
have \( \forall bk \in \text{blocksOf } s' q. \exists D \in \text{MajoritySet}. \forall d \in D. \text{bal } bk \leq \text{mbal}(\text{disk } s d q) \)

83
theorem HPhase1or2ReadThen-HInv4d:
  \[ HPhase1or2ReadThen \ s' \ p \ d \ r; \ HInv4d \ s \ q \ \Rightarrow \ HInv4d \ s' \ q \]
by (blast dest: HPhase1or2ReadThen-HInv4d-p HPhase1or2ReadThen-HInv4d-q)

theorem HPhase1or2ReadElse-HInv4d:
  \[ HPhase1or2ReadElse \ s' \ p \ d \ r; \ HInv4d \ s \ q \ \Rightarrow \ HInv4d \ s' \ q \]
using HStartBallot-HInv4d
by (auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase1-HInv4d-p:
  assumes act: HEndPhase1 \ s' \ p
  and inv: HInv4d \ s' \ p
  and inv2b: Inv2b \ s
  and inv4c: HInv4c \ s' \ p
  shows HInv4d \ s' \ p
proof (clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk \in blocksOf \ s' \ p
  from HEndPhase1-HInv4c (OF act inv4c inv2b)
  have HInv4c \ s' \ p .
  with act
  have p31: \exists D \in MajoritySet.
      \forall d \in D. \ mbal (disk' \ d \ p) = bal (dblock' \ d \ p)
    and disk': disk' = disk
  by (auto simp add: EndPhase1-def HInv4c-def)
  from subsetD (OF HEndPhase1-blocksOf (OF act) bk)
  show \exists D \in MajoritySet. \forall d \in D. \ bal \ bk \leq \ mbal (disk' \ d \ p)
proof
  assume bk: bk \in blocksOf \ s \ p
  with inv disk'
  show ?thesis
  by (auto simp add: HInv4d-def)
next
  assume bk: bk \in \{dblock \ s' \ p\}
  with p31
  show ?thesis
  by force
qed

lemma HEndPhase1-HInv4d-q:
  assumes act: HEndPhase1 \ s \ s' \ p
and inv: $H_{Inv4d} s q$
and pnq: $p \neq q$
shows $H_{Inv4d} s' q$

proof –
from act pnq
have $\text{disk'}$: disk $s' = \text{disk } s$ and $\text{dblock'}$: dblock $s' q = \text{dblock } s q$
  by (auto simp add: EndPhase1-def)
from act pnq
have $\text{blocksRead'}$: $\forall q. \text{allRdBlks } s' q \subseteq \text{allRdBlks } s q$
  by (auto simp add: EndPhase1-def InitializePhase-def allRdBlks-def)
with $\text{disk'}$ $\text{dblock'}$
have $\text{blocksOf'}$: $\forall q. \text{blocksOf } s' q \subseteq \text{blocksOf } s q$
  by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
from subsetD[OF this] inv
have $\forall bk \in \text{blocksOf } s' q$. $\exists D \in \text{MajoritySet. } \forall d \in D. \text{bal } bk \leq \text{mbal( disk } s d q)$
  by (auto simp add: HInv4d-def)
with $\text{disk'}$
show $?$thesis
by (auto simp add: HInv4d-def)
qed

theorem $\text{HEndPhase1-HInv4d}$:
[ $\text{HEndPhase1 } s s' p; \text{HInv4d } s q; \text{Inv2b } s s' p; \text{HInv4c } s p$] $\implies$ $\text{HInv4d } s' q$
by (blast dest: HEndPhase1-HInv4d-p HEndPhase1-HInv4d-q)

lemma $\text{HEndPhase2-HInv4d-p}$:
assumes act: $\text{HEndPhase2 } s s' p$
and inv: $\text{HInv4d } s p$
shows $\text{HInv4d } s' p$
proof (clar simp simp add: HInv4d-def)
fix bk
assume bk: $bk \in \text{blocksOf } s' p$
from act
have $$bal'$$: bal (dblock $s' p) = bal (dblock s p)$
  by (auto simp add: EndPhase2-def)
from subsetD[OF HEndPhase2-blocksOf[OF act] bk] inv
have $\exists D \in \text{MajoritySet. } \forall d \in D. \text{bal } bk \leq \text{mbal( disk } s d p)$
  by (auto simp add: HInv4d-def)
with act
show $\exists D \in \text{MajoritySet. } \forall d \in D. \text{bal } bk \leq \text{mbal( disk } s' d p)$
  by (auto simp add: EndPhase2-def)
qed

lemma $\text{HEndPhase2-HInv4d-q}$:
assumes act: $\text{HEndPhase2 } s s' p$
and \( inv: HInv4d s q \)
and \( pq: p \neq q \)
shows \( HInv4d s' q \)

proof –

from \( act\ pq\)
have \( disk': disk s' = disk s \)
  by (auto simp add: EndPhase2-def)
from \( act\ pq\)
have \( blocksOf s' q \subseteq blocksOf s q \)
  by (auto simp add: EndPhase2-def InitializePhase-def
       allRdBlks-def blocksOf-def rdBy-def)
from \( subsetD[OF this]\) \( inv\)
have \( \forall bk \in blocksOf s' q. \exists D \in \text{MajoritySet}. \forall d \in D. \ bal bk \leq mbal(disk s d q) \)
  by (auto simp add: HInv4d-def)
with \( disk'\)
show \(?thesis\)
by (auto simp add: HInv4d-def)

qed

theorem \( HEndPhase2-HInv4d \):
\[
[ HEndPhase2 s s' p; HInv4d s q ] \implies HInv4d s' q
\]
by (blast dest: HEndPhase2-HInv4d-p HEndPhase2-HInv4d-q)

lemma \( HFail-HInv4d-p\):
assumes \( act: HFail s s' p\)
and \( inv: HInv4d s p\)
shows \( HInv4d s' p\)
proof (clarsimp simp add: HInv4d-def)
fix \( bk\)
assume \( bk: bk \in blocksOf s' p\)
from \( act\)
have \( disk': disk s' = disk s \)
  by (auto simp add: Fail-def)
from \( subsetD[OF HFail-blocksOf[OF \ act] bk]\)
show \( \exists D \in \text{MajoritySet}. \forall d \in D. \ bal bk \leq mbal(disk s d p) \)
proof
assume \( bk: bk \in \{dblock s' p\}\)
with \( inv\ disk'\)
show \(?thesis\)
  by (auto simp add: HInv4d-def)
next
assume \( bk: bk \in \{dblock s' p\}\)
with \( act\)
have \( bal bk = 0\)
  by (auto simp add: Fail-def InitDB-def)
with \( Disk-isMajority\)
show \(?thesis\)
  by (auto simp add: MajoritySet-def)
lemma HFail-HInv4d-q:
  assumes act: HFail s s' p
  and inv: HInv4d s q
  and pnq: p ≠ q
  shows HInv4d s' q
proof –
  from act pnq
  have disk': disk s' = disk s
    and dblock': dblock s' q = dblock s q
    by (auto simp add: Fail-def)
  from act pnq
  have blocksRead': ∀ q. allRdBlks s' q ⊆ allRdBlks s q
    by (auto simp add: Fail-def InitializePhase-def allRdBlks-def)
  with disk' dblock'
  have blocksOf s' q ⊆ blocksOf s q
    by (auto simp add: allRdBlks-def blocksOf-def rdBy-def, blast)
  from subsetD[OF this] inv
  have ∀ bk ∈ blocksOf s' q.
    ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s d q)
    by (auto simp add: HInv4d-def)
  with disk'
  show ?thesis
  by (auto simp add: HInv4d-def)
qed

theorem HFail-HInv4d:
[ HFail s s' p; HInv4d s q ] ⇒ HInv4d s' q
by (blast dest: HFail-HInv4d-p HFail-HInv4d-q)

lemma HPhase0Read-HInv4d-p:
  assumes act: HPhase0Read s s' p d
  and inv: HInv4d s p
  shows HInv4d s' p
proof (clarsimp simp add: HInv4d-def)
  fix bk
  assume bk: bk ∈ blocksOf s' p
  from act
  have bal': bal (dblock s' p) = bal (dblock s p)
    by (auto simp add: Phase0Read-def)
  from subsetD[OF HPhase0Read-blocksOf[OF act] bk] inv
  have ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s d p)
    by (auto simp add: HInv4d-def)
  with act
  show ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal (disk s' d p)
    by (auto simp add: Phase0Read-def)
qed
lemma HPhase0Read-HInv4d-q:
assumes act: HPhase0Read s s' p d
and inv: HInv4d s q
and pnq: p ≠ q
shows HInv4d s' q
proof –
  from act pnq
  have disk': disk s' = disk s
    by (auto simp add: Phase0Read-def)
  from act pnq
  have blocksOf s' q ⊆ blocksOf s q
    by (auto simp add: Phase0Read-def allRdBlks-def blocksOf-def rdBy-def)
  from subsetD[OF this] inv
  have ∀ bk ∈ blocksOf s' q.
    ∃ D ∈ MajoritySet. ∀ d ∈ D. bal bk ≤ mbal(disk s d q)
    by (auto simp add: HInv4d-def)
  with disk'
  show ?thesis
  by (auto simp add: HInv4d-def)
qed

theorem HPhase0Read-HInv4d:
[ HPhase0Read s s' p d; HInv4d s q ] ⇒ HInv4d s' q
by (blast dest: HPhase0Read-HInv4d-p HPhase0Read-HInv4d-q)

lemma HEndPhase0-blocksOf2:
assumes act: HEndPhase0 s s' p
and inv2c: Inv2c-inner s p
shows allBlocksRead s p ⊆ blocksOf s p
proof –
  from act inv2c
  have ∀ d.∀ br ∈ blocksRead s p d. proc br = p
    ∧ block br = disk s d p
    by (auto simp add: EndPhase0-def Inv2c-inner-def)
  thus ?thesis
    by (auto simp add: allBlocksRead-def allRdBlks-def blocksOf-def)
qed

lemma HEndPhase0-HInv4d-p:
assumes act: HEndPhase0 s s' p
and inv: HInv4d s p
and inv2c: Inv2c s
and inv1: Inv1 s
shows HInv4d s' p
proof (clarsimp simp add: HInv4d-def)
  fix bk

assume \( bk : bk \in \text{blocksOf } s' \ p \)
from \( \text{subsetD} \[ \text{OF } H\text{EndPhase0}\text{-blocksOf} \[ \text{OF } \text{act} \] bk \] 
have \( \exists D \in \text{MajoritySet}. \ \forall d \in D. \ \text{bal } bk \leq \text{mbal } (\text{disk } s \ d \ p) \)

proof
assume \( bk : bk \in \text{blocksOf } s \ p \)
with \( \text{inv} \)
show \( \text{?thesis} \)
by\( (\text{auto simp add: } H\text{Inv4d-def}) \)

next
assume \( bk : bk \in \{ \text{dblock } s' \ p \} \)
from \( \text{inv2c} \)
have \( \text{inv2c-inner: Inv2c-inner } s \ p \)
by\( (\text{auto simp add: Inv2c-def}) \)
from \( bk \ H\text{EndPhase0-some}[\text{OF } \text{act } \text{inv1}] \)
\( H\text{EndPhase0}\text{-blocksOf}2[\text{OF } \text{act } \text{inv2c-inner}] \text{ act} \)
have \( \text{bal } bk \in \text{bal '}(\text{blocksOf } s \ p) \)
with \( \text{inv} \)
show \( \text{?thesis} \)
by\( (\text{auto simp add: EndPhase0-def}) \)
qed

with \( \text{act} \)
show \( \exists D \in \text{MajoritySet}. \ \forall d \in D. \ \text{bal } bk \leq \text{mbal } (\text{disk } s' \ d \ p) \)
by\( (\text{auto simp add: EndPhase0-def}) \)
qed

lemma \( H\text{EndPhase0-HInv4d-q} : \)
assumes \( \text{act}: H\text{EndPhase0 } s \ s' \ p \)
and \( \text{inv}: H\text{Inv4d } s \ q \)
and \( \text{pnq}: p \neq q \)
shows \( H\text{Inv4d} s' \ q \)

proof
−
from \( \text{act } \text{pnq} \)
have \( \text{dblock } s' \ q = \text{dblock } s \ q \land \text{disk } s' = \text{disk } s \)
by\( (\text{auto simp add: EndPhase0-def}) \)

moreover
from \( \text{act } \text{pnq} \)
have \( \forall p \ d. \ \text{rdBy } s' \ q \ p \ d \subseteq \text{rdBy } s \ q \ p \ d \)
by\( (\text{auto simp add: EndPhase0-def InitializePhase-def rdBy-def}) \)

hence \( (\text{UN } p \ d. \ \text{rdBy } s' \ q \ p \ d) \subseteq (\text{UN } p \ d. \ \text{rdBy } s \ q \ p \ d) \)
by\( (\text{auto, blast}) \)

ultimately
have \( \text{blocksOf } s' \ q \subseteq \text{blocksOf } s \ q \)
by\( (\text{auto simp add: blocksOf-def, blast}) \)
from \( \text{subsetD}[\text{OF } \text{this}] \text{ inv} \)
have \( \forall bk \in \text{blocksOf } s' \ q. \)
\[ \exists D \in \text{MajoritySet}. \ \forall d \in D. \ \text{bal } bk \leq \text{mbal } (\text{disk } s \ d \ q) \]
by\( (\text{auto simp add: } H\text{Inv4d-def}) \)
with \( \text{act} \)
show ?thesis
by (auto simp add: EndPhase0-def HInv4d-def)
qed

theorem HEndPhase0-HInv4d:
[ HEndPhase0 s s' p; HInv4d s q; Inv2c s; Inv1 s' \Longrightarrow HInv4d s' q ]
by (blast dest: HEndPhase0-HInv4d-p HEndPhase0-HInv4d-q)

Since we have already proved HInv2 is an invariant of HNext, HInv1 \land
HInv2 \land HInv4 is also an invariant of HNext.

lemma I2d:
assumes nxt: HNext s s'
and inv: HInv1 s \land HInv2 s \land HInv2 s' \land HInv4 s
shows HInv4 s'
proof (auto simp add: HInv4-def)
fix p
show HInv4a s' p using assms
by (auto simp add: HInv4-def HNext-def Next-def
auto simp add: HInv2-def intro: HStartBallot-HInv4a,
auto intro: HPhase0Read-HInv4a,
auto intro: HPhase1or2Write-HInv4a,
auto simp add: Phase1or2Read-def
intro: HPhase1or2ReadThen-HInv4a
Phase1or2ReadElse-HInv4a,
auto simp add: EndPhase1or2-def
intro: HEndPhase1-HInv4a
HEndPhase2-HInv4a,
auto intro: HFail-HInv4a,
auto intro: HEndPhase0-HInv4a simp add: HInv1-def)

show HInv4b s' p using assms
by (auto simp add: HInv4-def HNext-def Next-def
auto simp add: HInv2-def
intro: HStartBallot-HInv4b,
auto intro: HPhase0Read-HInv4b,
auto intro: HPhase1or2Write-HInv4b,
auto simp add: Phase1or2Read-def
intro: HPhase1or2ReadThen-HInv4b
Phase1or2ReadElse-HInv4b,
auto simp add: EndPhase1or2-def
intro: HEndPhase1-HInv4b
HEndPhase2-HInv4b,
auto intro: HFail-HInv4b,
auto intro: HEndPhase0-HInv4b simp add: HInv1-def Inv2c-def)

show HInv4c s' p using assms
by (auto simp add: HInv4-def HNext-def Next-def
auto simp add: HInv2-def
intro: HStartBallot-HInv4c,
auto intro: HPhase0Read-HInv4c,
C.5 Invariant 5

This invariant asserts that, if a processor $p$ is in phase 2, then either its $bal$ and $inp$ values satisfy $maxBalInp$, or else $p$ must eventually abort its current ballot. Processor $p$ will eventually abort its ballot if there is some processor $q$ and majority set $D$ such that $p$ has not read $q$’s block on any disk $D$, and all of those blocks have $mbal$ values greater than $bal(dblocksp)$.

**definition** $maxBalInp ::$ state $\Rightarrow$ nat $\Rightarrow$ InputsOrNi $\Rightarrow$ bool

**where** $maxBalInp s b v = (\forall bk \in allBlocks s. b \leq bal bk \quad \rightarrow \quad inp bk = v)$

**definition** $HInv5-inner-R ::$ state $\Rightarrow$ Proc $\Rightarrow$ bool

**where**

$HInv5-inner-R s p =$

$(maxBalInp s (bal(dblock s p)) (inp(dblock s p))$

$\vee (\exists D \in MajoritySet. \exists q. (\forall d \in D. bal(dblock s p) < mbal(disk s d q) \wedge \neg hasRead s p d q)))$
where \( HInv5-\text{inner } s \ p = (\text{phase } s \ p = 2 \implies HInv5-\text{inner-R } s \ p) \)

**Definition**

\( HInv5 :: \text{state} \Rightarrow \text{bool} \)

where \( HInv5 \ s = (\forall p. \ HInv5-\text{inner } s \ p) \)

### C.5.1 Proof of Invariant 5

The initial state implies Invariant 5.

**Theorem** \( HInit-HInv5: HInit \ s \implies HInv5 \ s \)

**Using** Disk-isMajority

**By** (auto simp add: HInit-def Init-def HInv5-def HInv5-inner-def)

We will use the notation used in the proofs of invariant 4, and prove the lemma action-\(HInv5\)-p and action-\(HInv5\)-q for each action, for the cases \( p = q \) and \( p \neq q \) respectively.

Also, for each action we will define an action-allBlocks lemma in the same way that we defined -blocksOf lemmas in the proofs of \( HInv2 \). Now we prove that for each action the new allBlocks are included in the old allBlocks or, in some cases, included in the old allBlocks union the new dblock.

**Lemma** \( HStartBallot-HInv5\)-p:

**Assumes** act: \( HStartBallot \ s \ s' \ p \)

**And** inv: \( HInv5-\text{inner } s \ p \)

**Shows** \( HInv5-\text{inner } s' \ p \)

**Using** assms

**By** (auto simp add: StartBallot-def HInv5-inner-def)

**Lemma** \( HStartBallot\)-blocksOf-q:

**Assumes** act: \( HStartBallot \ s \ s' \ p \)

**And** \( p_n q : p \neq q \)

**Shows** \( \text{blocksOf } s' \ q \subseteq \text{blocksOf } s \ q \)

**Using** assms

**By** (auto simp add: StartBallot-def InitializePhase-def blocksOf-def rdBy-def)

**Lemma** \( HStartBallot\)-allBlocks:

**Assumes** act: \( HStartBallot \ s \ s' \ p \)

**Shows** \( \text{allBlocks } s' \subseteq \text{allBlocks } s \cup \{ \text{dblock } s' \ p \} \)

**Proof** (auto simp del: HStartBallot-def simp add: allBlocks-def

dest: HStartBallot-blocksOf-q[OF act])

**Fix** \( x \ p a \)

**Assume** \( x-pa: x \in \text{blocksOf } s' \ p \) and

\( x-nblks: \forall x a. x \notin \text{blocksOf } s x a \)

**Show** \( x=\text{dblock } s' \ p \)

**Proof** (cases \( p=pa \))

**Case** True

**From** \( x-nblks \)

**Have** \( x \notin \text{blocksOf } s \ p \)

**By** auto

**With** True subsetD[OF HStartBallot-blocksOf[OF act] x-pa]
show \( ?\text{thesis} \)
by auto
next
case False
from \( x\)-nblks subsetD[OF HStartBallot-blocksOf-q[OF act False] \ x-pa]
show \( ?\text{thesis} \)
by auto
qed
qed

lemma HStartBallot-HInv5-q1:
assumes act: HStartBallot \( s s' p \)
and pnq: \( p \neq q \)
and inv5-1: maxBalInp \( s (bal(dblock s q)) (\text{inp}(dblock s q)) \)
shows maxBalInp \( s' (bal(dblock s' q)) (\text{inp}(dblock s' q)) \)
proof (auto simp add: maxBalInp-def)
fix \( bk \)
assume bk: \( bk \in \text{allBlocks} s' \)
and bal: \( bal (dblock s' q) \leq bal bk \)
from act pnq
have dblock': dblock \( s' q = dblock s q \) by(auto simp add: StartBallot-def)
from subsetD[OF HStartBallot-allBlocks[OF act] \ bk]
show \( \text{inp} bk = \text{inp} (dblock s' q) \)
proof
assume bk: \( bk \in \text{allBlocks} s \)
with inv5-1 dblock' bal
show \( ?\text{thesis} \)
by(auto simp add: maxBalInp-def)
next
assume bk: \( bk \in \{dblock s' p\} \)
have dblock s p \( \in \text{allBlocks} s \)
by(auto simp add: allBlocks-def blocksOf-def)
with bal act bk dblock' inv5-1
show \( ?\text{thesis} \)
by(auto simp add: maxBalInp-def StartBallot-def)
qed
qed

lemma HStartBallot-HInv5-q2:
assumes act: HStartBallot \( s s' p \)
and pnq: \( p \neq q \)
and inv5-2: \( \exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \ \\text{bal}(dblock s q) < \text{mbal}(disk s d qq) \wedge \neg \text{hasRead} s q d qq) \)
shows \( \exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \ \text{bal}(dblock s' q) < \text{mbal}(disk s' d qq) \wedge \neg \text{hasRead} s' q d qq) \)
proof
from act pnq
have disk: disk \( s' = disk s \)
and \( \text{blocksRead} : \forall d, \text{blocksRead} s' q d = \text{blocksRead} s q d \)
and \( \text{dblock} : \text{dblock} s' q = \text{dblock} s q \)
by (auto simp add: StartBallot-def InitializePhase-def)

with inv5-2
show ?thesis
by (auto simp add: hasRead-def)
qed

lemma HStartBallot-HInv5-q:
assumes act: HStartBallot s s' p
and inv: HInv5-inner s q
and pnq: p \neq q
shows HInv5-inner s' q
using assms and HStartBallot-HInv5-q1[OF act pnq] HStartBallot-HInv5-q2[OF act pnq]
by (auto simp add: HInv5-inner-def HInv5-inner-R-def StartBallot-def)

theorem HStartBallot-HInv5:
[ HStartBallot s s' p; HInv5-inner s q ] \implies HInv5-inner s' q
by (blast dest: HStartBallot-HInv5-q HStartBallot-HInv5-p)

lemma HPhase1or2Write-HInv5-1:
assumes act: HPhase1or2Write s s' p d
and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q))
shows maxBalInp s' (bal(dblock s' q)) (inp(dblock s' q))
using assms and HPhase1or2Write-blocksOf[OF act]
by (auto simp add: Phase1or2Write-def maxBalInp-def allBlocks-def)

lemma HPhase1or2Write-HInv5-p2:
assumes act: HPhase1or2Write s s' p d
and inv4c: HInv4c s p
and phase: phase s p = 2
and inv5-2: \( \exists D \in \text{MajoritySet}. \forall q, (\forall d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q)) \)
\( \land \neg\text{hasRead} s p d q \)
shows \( \exists D \in \text{MajoritySet}. \forall q, (\forall d \in D. \text{bal}(\text{dblock} s' p) < \text{mbal}(\text{disk} s' d q)) \)
\( \land \neg\text{hasRead} s' p d q \)
proof –
from inv5-2
obtain D q
where i1: IsMajority D
\( \land \text{i2}: \forall d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q) \)
\( \land \text{i3}: \forall d \in D. \neg\text{hasRead} s p d q \)
by (auto simp add: MajoritySet-def)
have pnq: p \neq q
proof –
from inv4c phase
obtain D1 where r1: IsMajority D1 \( \land (\forall d \in D1. \text{mbal}(\text{disk} s d p) = \text{bal}(\text{dblock} s p)) \)
by (auto simp add: HInv4c-def MajoritySet-def) with i1 majorities-intersect have \( D \cap D \neq \{\} \) by auto then obtain \( dd \) where \( dd \in D \cap D \) by auto with i1 i2 r1 have \( bal(dblock s p) < mbal(disk s dd q) \land mbal(disk s dd p) = bal(dblock s p) \) by auto thus \(?thesis\) by auto qed from act pnq — dblock and hasRead do not change have \( dblock s' = dblock s \) and \( \forall d. \ hasRead s' p d q = hasRead s p d q \) — In all disks q blocks don’t change and \( \forall d. disk s' d q = disk s d q \) by (auto simp add: Phase1or2Write-def hasRead-def) with i2 i1 i3 majority-nonempty have \( \forall d \in D. bal(dblock s' p) < mbal(disk s' d q) \land \neg hasRead s' p d q \) by auto with i1 show \(?thesis\) by (auto simp add: MajoritySet-def) qed

lemma HPhase1or2Write-HInv5-p: assumes act: HPhase1or2Write s s' p d and inv: HInv5-inner s p and inv4: HInv4c s p shows HInv5-inner s' p proof (auto simp add: HInv5-inner-def HInv5-inner-R-def) assume phase': phase s' p = 2 and i2: \( \forall D \in MajoritySet. \forall q. \exists d \in D. bal(dblock s' p) < mbal(disk s' d q) \rightarrow hasRead s' p d q \) with act have phase: phase s p = 2 by (auto simp add: Phase1or2Write-def) show maxBalIrp s' (bal (dblock s' p)) (inp (dblock s' p)) proof (rule HPhase1or2Write-HInv5-1[OF act, of p]) from HPhase1or2Write-HInv5-p2[OF act inv4 phase] inv i2 phase show maxBalIrp s (bal (dblock s p)) (inp (dblock s p)) by (auto simp add: HInv5-inner-def HInv5-inner-R-def, blast) qed

lemma HPhase1or2Write-allBlocks: assumes act: HPhase1or2Write s s' p d shows allBlocks s' \( \subseteq \) allBlocks s using HPhase1or2Write-blocksOf[OF act] by (auto simp add: allBlocks-def)
lemma \texttt{HPhase1or2Write-HInv5-q2}: 
\textbf{assumes} \texttt{act: HPhase1or2Write s s' p d} 
\text{and} \texttt{pnq: p \neq q} 
\text{and} \texttt{inv4a: HInv4a s p} 
\text{and} \texttt{inv5-2: \exists D \in MajoritySet. \exists qq. (\forall d \in D. bal(dblock s q) < mbal(disk s d qq)} 
\quad \land \lnot \text{hasRead s q d qq)} 
\text{shows} \exists D \in MajoritySet. \exists qq. (\forall d \in D. bal(dblock s' q) < mbal(disk s' d qq)} 
\quad \land \lnot \text{hasRead s' q d qq)} 

\textbf{proof} – 
\texttt{from inv5-2} 
\texttt{obtain D qq} 
\texttt{where i1: IsMajority D} 
\texttt{and i2: \forall d \in D. bal(dblock s q) < mbal(disk s d qq)} 
\texttt{and i3: \forall d \in D. \lnot \text{hasRead s q d qq}} 
\texttt{by (auto simp add: MajoritySet-def)} 
\texttt{from act pnq} 
\texttt{— dblock and hasRead do not change} 
\texttt{have dblock': dblock s' = dblock s} 
\texttt{and hasread: \forall d. hasRead s' q d qq = hasRead s q d qq} 
\texttt{by (auto simp add: Phase1or2Write-def hasRead-def)} 
\texttt{have \forall d \in D. bal(dblock s' q) < mbal(disk s' d qq) \land \lnot \text{hasRead s' q d qq}} 
\texttt{proof (cases qq=p)} 
\texttt{case True} 
\texttt{have bal(dblock s q) < mbal(dblock s p)} 
\texttt{proof –} 
\texttt{from inv4a act i1} 
\texttt{have \exists d \in D. mbal(disk s d p) \leq mbal(dblock s p)} 
\texttt{by (auto simp add: MajoritySet-def HInv4a-def HInv4a2-def Phase1or2Write-def)} 
\texttt{with True i2} 
\texttt{show bal(dblock s q) < mbal(dblock s p)} 
\texttt{by auto} 
\texttt{qed} 
\texttt{with hasread dblock' True i1 i2 i3 act} 
\texttt{show \texttt{?thesis}} 
\texttt{by (auto simp add: Phase1or2Write-def)} 
\texttt{next} 
\texttt{case False} 
\texttt{with act i2 i3} 
\texttt{show \texttt{?thesis}} 
\texttt{by (auto simp add: Phase1or2Write-def hasRead-def)} 
\texttt{qed} 
\texttt{with i1} 
\texttt{show \texttt{?thesis}} 
\texttt{by (auto simp add: MajoritySet-def)} 
\texttt{qed}
lemma HPhase1or2Write-HInv5-q:
assumes act: HPhase1or2Write s s' p d
and inv: HInv5-inner s q
and inv4a: HInv4a s p
and pnq: p\#q
shows HInv5-inner s' q
proof (auto simp add: HInv5-inner-def HInv5-inner-R-def)
  assume phase': phase s' q = 2
  and i2: \( \forall D \in \text{MajoritySet}. \forall q. \exists d \in D. \text{bal}(\text{dblock } s' q) < \text{mbal}(\text{disk } s' d q) \)
  \( \rightarrow \) hasRead s' q d qa
  from phase' act have phase: phase s q = 2
    by (auto simp add: Phase1or2Write-def)
  show maxBalInp s' (bal (dblock s' q)) (inp (dblock s' q))
    proof (rule HPhase1or2Write-HInv5-1 [OF act, of q])
      from HPhase1or2Write-HInv5-q2 [OF act pnq inv4a]
      inv i2 phase
      show maxBalInp s (bal (dblock s q)) (inp (dblock s q))
        by (auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
    qed
  qed

theorem HPhase1or2Write-HInv5:
  [ HPhase1or2Write s s' p d; HInv5-inner s q; HInv4c s p; HInv4a s p ] \( \rightarrow \) HInv5-inner s' q
by (blast dest: HPhase1or2Write-HInv5-q HPhase1or2Write-HInv5-p)

lemma HPhase1or2ReadThen-HInv5-1:
assumes act: HPhase1or2ReadThen s s' p d r
and inv5-1: maxBalInp s (bal (dblock s q)) (inp (dblock s q))
shows maxBalInp s' (bal (dblock s' q)) (inp (dblock s' q))
using assms and HPhase1or2ReadThen-blocksOf [OF act]
by (auto simp add: Phase1or2ReadThen-def maxBalInp-def allBlocks-def)

lemma HPhase1or2ReadThen-HInv5-p2:
assumes act: HPhase1or2ReadThen s s' p d r
and inv4c: HInv4c s p
and inv2c: Inv2c-inner s p
and phase: phase s p = 2
and inv5-2: \( \exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock } s p) < \text{mbal}(\text{disk } s d q) \wedge \neg\text{hasRead } s p d q) \)
shows \( \exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock } s' p) < \text{mbal}(\text{disk } s' d q) \wedge \neg\text{hasRead } s' p d q) \)
proof
  from inv5-2 obtain D q
    where i1: IsMajority D
      and i2: \( \forall d \in D. \text{bal}(\text{dblock } s p) < \text{mbal}(\text{disk } s d q) \)
      and i3: \( \forall d \in D. \neg\text{hasRead } s p d q \)
    by (auto simp add: MajoritySet-def)
  from inv2c phase

97
have \( \text{bal}(\text{dblock } s \, p) = \text{mbal}(\text{dblock } s \, p) \)
by (auto simp add: \text{Inv2c-inner-def})
moreover
from \text{act} have \( \text{mbal}(\text{disk } s \, d \, r) < \text{mbal}(\text{dblock } s \, p) \)
by (auto simp add: \text{Phase1or2ReadThen-def})
moreover
from \text{i2} have \( d \in D \rightarrow \text{bal}(\text{dblock } s \, p) < \text{mbal}(\text{disk } s \, d \, q) \) by auto
ultimately have \( \text{pnr}: d \in D \rightarrow q \neq r \) by auto
have \( \text{pnr}: p \neq q \)
proof
  from \text{inv4c phase}
obtain \( D1 \) where \( r1: \text{IsMajority } D1 \wedge (\forall d \in D1, \text{mbal}(\text{disk } s \, d \, p) = \text{bal}(\text{dblock } s \, p)) \)
  by (auto simp add: \text{HInv4c-def MajoritySet-def})
  with \text{i1} \text{majorities-intersect}
  have \( D \cap D1 \neq \{} \) by auto
  then obtain \( dd \) where \( dd \in D \cap D1 \)
  by auto
  with \text{i1} \text{i2} \text{r1}
  have \( \text{bal}(\text{dblock } s \, p) < \text{mbal}(\text{disk } s \, dd \, q) \wedge \text{mbal}(\text{disk } s \, dd \, p) = \text{bal}(\text{dblock } s \, p) \)
  by auto
  thus \( \text{?thesis} \) by auto
qed
from \text{pnr} \text{act}
have \( \text{hasRead}': \forall d \in D. \text{hasRead } s' \, p \, d \, q = \text{hasRead } s \, p \, d \, q \)
by (auto simp add: \text{Phase1or2ReadThen-def hasRead-def})
from \text{act} \text{pnr}
  — \text{dblock and disk do not change}
have \( \text{dblock } s' = \text{dblock } s \)
  and \( \forall d. \text{disk } s' = \text{disk } s \)
  by (auto simp add: \text{Phase1or2ReadThen-def})
  with \text{i2} \text{hasRead'} \text{i3}
  have \( \forall d \in D. \text{bal}(\text{dblock } s' \, p) < \text{mbal}(\text{disk } s' \, d \, q) \wedge \neg \text{hasRead } s' \, p \, d \, q \)
  by auto
  with \text{i1}
  show \( \text{?thesis} \)
  by (auto simp add: \text{MajoritySet-def})
qed

lemma \text{HPhase1or2ReadThen-HInv5-p}:
  assumes \text{act}: \text{HPhase1or2ReadThen } s \, s' \, p \, d \, r
  and \text{inv}: \text{HInv5-inner } s \, p
  and \text{inv4}: \text{HInv4c } s \, p
  and \text{inv2c}: \text{Inv2c } s
  shows \( \text{HInv5-inner } s' \)
proof (auto simp add: \text{HInv5-inner-def HInv5-inner-R-def})
  assume \text{phase'`: phase } s' \, p = 2
  and \text{i2: } \forall D \in \text{MajoritySet}. \forall q. \exists d \in D. \text{bal}(\text{dblock } s' \, p) < \text{mbal}(\text{disk } s' \, d \, q)
  \rightarrow \text{hasRead } s' \, p \, d \, q

  98
with act have phase: phase s p = 2
  by(auto simp add: Phase1or2ReadThen-def)
show maxBalIns s' (bal (dblock s' p)) (inp (dblock s' p))
proof(rule HPhase1or2ReadThen-HInv5-1[OF act, of p])
  from Inv2c
  have Inv2c-inner s p by(auto simp add: Inv2c-def)
  from HPhase1or2ReadThen-HInv5-p2[OF act Inv4 this phase]
  show maxBalIns s (bal (dblock s p)) (inp (dblock s p))
    by(auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
qed

lemma HPhase1or2ReadThen-allBlocks:
  assumes act: HPhase1or2ReadThen s s' p d r
  shows allBlocks s' ⊆ allBlocks s
  using HPhase1or2ReadThen-blocksOf[OF act]
  by(auto simp add: allBlocks-def)

lemma HPhase1or2ReadThen-HInv5-q2:
  assumes act: HPhase1or2ReadThen s s' p d r
  and pnq: p ≠ q
  and Inv4a: HInv4a s p
  and Inv5-2: ∃D∈MajoritySet. ∃qq. (∀ d∈D. bal(dblock s q) < mbal(disk s d qq)
    ∧ ¬hasRead s q d qq)
  shows ∃D∈MajoritySet. ∃qq. (∀ d∈D. bal(dblock s' q) < mbal(disk s' d qq)
    ∧ ¬hasRead s' q d qq)
proof -
  from Inv5-2
  obtain D qq
    where i1: IsMajority D
      and i2: ∀ d∈D. bal(dblock s q) < mbal(disk s d qq)
      and i3: ∀ d∈D. ¬hasRead s q d qq
    by(auto simp add: MajoritySet-def)
  from act pnq
    — dblock and hasRead do not change
  have dblock': dblock s' = dblock s
    and disk': disk s' = disk s
    and hasread: ∀ d. hasRead s' q d qq = hasRead s q d qq
    by(auto simp add: Phase1or2ReadThen-def hasRead-def)
  with i2 i3
  have ∀ d∈D. bal (dblock s' q) < mbal (disk s' d qq) ∧ ¬hasRead s' q d qq
    by auto
  with i1
  show ?thesis
    by(auto simp add: MajoritySet-def)
qed

lemma HPhase1or2ReadThen-HInv5-q:
assumes \( \text{act}: \text{HPhase1or2ReadThen} \ s \ s' \ p \ d \ r \)
and \( \text{inv}: \text{HInv5-inner} \ s \ q \)
and \( \text{inv4a}: \text{HInv4a} \ s \ p \)
and \( \text{pnq}: p \neq q \)
shows \( \text{HInv5-inner} \ s' \ q \)

proof(auto simp add: HInv5-inner-def HInv5-inner-R-def)
assume \( \text{phase'}: \text{phase} \ s' \ q = 2 \)
and \( \text{i2}: \forall D \in \text{MajoritySet}. \forall qa. \exists d \in D. \ \text{bal} (\text{dblock} \ s' \ q) < m\text{bal} (\text{disk} \ s' \ d \ qa) \)
\( \longrightarrow \text{hasRead} \ s' \ q \ d \ qa \)
from phase' act have \( \text{phase} \ s \ q = 2 \)
by(auto simp add: Phase1or2ReadThen-def)
show \( \text{maxBalInp} \ s' \ (\text{bal} (\text{dblock} \ s' \ q)) \ (\text{inp} (\text{dblock} \ s' \ q)) \)
proof(rule HPhase1or2ReadThen-HInv5-1[OF act, of q])
from HPhase1or2ReadThen-HInv5-q2[OF act pnq inv4a] inv i2 phase
show \( \text{maxBalInp} \ s \ (\text{bal} (\text{dblock} \ s \ q)) \ (\text{inp} (\text{dblock} \ s \ q)) \)
by(auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
qed

theorem HPhase1or2ReadThen-HInv5:
[ \[ \text{HPhase1or2ReadThen} \ s \ s' \ p \ d \ r; \ \text{HInv5-inner} \ s \ q; \text{Inv2c} \ s; \text{HInv4c} \ s \ p; \text{HInv4a} \ s \ p \] ] \( \Rightarrow \) \( \text{HInv5-inner} \ s' \ q \)
by(blasm dest: HPhase1or2ReadThen-HInv5-p)

theorem HPhase1or2ReadElse-HInv5:
[ \[ \text{HPhase1or2ReadElse} \ s \ s' \ p \ d \ r; \ \text{HInv5-inner} \ s \ q \] ] \( \Rightarrow \) \( \text{HInv5-inner} \ s' \ q \)
using HStartBallot-HInv5
by(auto simp add: Phase1or2ReadElse-def)

lemma HEndPhase2-HInv5-p:
\( \text{HEndPhase2} \ s \ s' \ p \Rightarrow \text{HInv5-inner} \ s' \ p \)
by(auto simp add: EndPhase2-def HInv5-inner-def)

lemma HEndPhase2-allBlocks:
assumes \( \text{act}: \text{HEndPhase2} \ s \ s' \ p \)
shows \( \text{allBlocks} \ s' \subseteq \text{allBlocks} \ s \)
using HEndPhase2-blocksOf[OF act]
by(auto simp add: allBlocks-def)

lemma HEndPhase2-HInv5-q1:
assumes \( \text{act}: \text{HEndPhase2} \ s \ s' \ p \)
and \( \text{pnq}: p \neq q \)
and \( \text{inv5-1}: \text{maxBalInp} \ s \ (\text{bal} (\text{dblock} \ s \ q)) \ (\text{inp} (\text{dblock} \ s \ q)) \)
shows \( \text{maxBalInp} \ s' \ (\text{bal} (\text{dblock} \ s' \ q)) \ (\text{inp} (\text{dblock} \ s' \ q)) \)
proof(auto simp add: maxBalInp-def)
fix \( \text{bk} \)
assume \( \text{bk}: \text{bk} \in \text{allBlocks} \ s' \)
and \( \text{bal}: \text{bal} (\text{dblock} \ s' \ q) \leq \text{bal} \text{bk} \)

qed
from act pnq
have dblock': dblock s' q = dblock s q by (auto simp add: EndPhase2-def)
from subsetD[OF HEndPhase2-allBlocks[OF act] bk] inv5-1 dblock' bal
show inp bk = inp (dblock s q)
  by (auto simp add: maxBalInp-def)
qed

lemma HEndPhase2-HInv5-q2:
  assumes act: HEndPhase2 s s' p
  and pnq: p≠q
  and inv5-2: ∃D∈MajoritySet. ∃qq. (∀d∈D. bal(dblock s q) < mbal(disk s d qq)
    ∧ ¬hasRead s q d qq)
  shows ∃D∈MajoritySet. ∃qq. (∀d∈D. bal(dblock s' q) < mbal(disk s' d qq)
    ∧ ¬hasRead s' q d qq)
proof —
  from act pnq
  have disk: disk s' = disk s
    and blocksRead: ∀d. blocksRead s' q d = blocksRead s q d
    and dblock: dblock s' q = dblock s q
    by (auto simp add: EndPhase2-def InitializePhase-def)
  with inv5-2
  show ?thesis
    by (auto simp add: hasRead-def)
qed

lemma HEndPhase2-HInv5-q:
  assumes act: HEndPhase2 s s' p
  and inv: HInv5-inner s q
  and pnq: p≠q
  shows HInv5-inner s' q
using assms and HEndPhase2-HInv5-q1[OF act pnq] HEndPhase2-HInv5-q2[OF act pnq]
by (auto simp add: HInv5-inner-def HInv5-inner-R-def EndPhase2-def)

theorem HEndPhase2-HInv5:
[ HEndPhase2 s s' p; HInv5-inner s q ] ⟹ HInv5-inner s' q
by (blast dest: HEndPhase2-HInv5-q HEndPhase2-HInv5-q HEndPhase2-HInv5-p)

lemma HEndPhase1-HInv5-p:
  assumes act: HEndPhase1 s s' p
  and inv4: Hinv4 s
  and inv2a: Inv2a s
  and inv2a': Inv2a' s'
  and inv2c: Inv2c s
  and asm4: ¬maxBalInp s' (bal(dblock s' p)) (inp(dblock s' p))
  shows (∃D∈MajoritySet. ∃q. (∀d∈D. bal(dblock s' q) < mbal(disk s' d q)
    ∧ ¬hasRead s' q d q))
proof —
have $\exists bk \in \text{allBlocks } s. \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p$
proof
  from asm4
  obtain bk
    where $p31: bk \in \text{allBlocks } s' \land \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p$
    by(auto simp add: maxBalImp-def)
  then obtain q where $p32: bk \in \text{blocksOf } s' q$
    by(auto simp add: allBlocks-def)
  from act
  have $\text{dblock}: p \neq q \Longrightarrow \text{dblock } s' q = \text{dblock } s q$
    by(auto simp add: EndPhase1-def)
  have $bk \in \text{blocksOf } s q$
proof(cases $p = q$)
  case True
    with $p32 \ p31 \ HEndPhase1-blocksOf[\text{OF } act]$
    show $?thesis$
      by auto
next
  case False
    from $\text{dblock}[\text{OF false}] \ subsetD[\text{OF } HEndPhase1-blocksOf[\text{OF } act, \ of } q \ p32]$
    show $?thesis$
      by(auto simp add: blocksOf-def)
  qed
  with $p31$
  show $?thesis$
    by(auto simp add: allBlocks-def)
  qed
then obtain bk where $p22: bk \in \text{allBlocks } s \land \text{bal}(\text{dblock } s' p) \leq \text{bal } bk \land bk \neq \text{dblock } s' p$
  by(auto)
have $\exists q \in \text{UNIV} - \{p\}. bk \in \text{blocksOf } s q$
proof
  from $p22$
  obtain q where $bk: bk \in \text{blocksOf } s q$
    by(auto simp add: allBlocks-def)
  from act $p22$
  have $\text{mbal}(\text{dblock } s p) \leq \text{bal } bk$
    by(auto simp add: EndPhase1-def)
moreover
  from act
  have $\text{phase } s p = 1$
    by(auto simp add: EndPhase1-def)
moreover
  from inv4
  have $H\text{Inv4b } s p$ by(auto simp add: HInv4-def)
ultimately
  have $p \neq q$
    using $bk$
    by(auto simp add: HInv4-def HInv4b-def)
  with $bk$

102
show \( ?\text{thesis} \)
by auto

qed

then obtain \( q \) where \( p23 \): \( q \in \text{UNIV} - \{ p \} \land bk \in \text{blocksOf} \ s \ q \)
by auto

have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} (\text{dblock} \ s' p) \leq \text{mbal} (\text{disk} \ s \ d \ q) \)
proof –

from \( p23 \) inv4
have \( \exists D \in \text{MajoritySet}. \forall d \in D. \ \text{bal} \ bk \leq \text{mbal} (\text{disk} \ s \ d \ q) \)
by (auto simp add: HInv4-def HInv4d-def)

from \( \exists d \ p22 \)
show \( ?\text{thesis} \)
by force

qed

then obtain \( D \) where \( D\text{maj} \): \( D \in \text{MajoritySet} \) and \( p24 \): \( (\forall d \in D. \ \text{bal} (\text{dblock} \ s' p) \leq \text{mbal} (\text{disk} \ s \ d \ q) \)
proof –

from \( inv2c \)
have \( \text{Inv2c-inner} \ s \ p \)
by (auto simp add: Inv2c-def)

with \( \text{act} \)
have \( \text{bal-pos}: \ 0 < \text{bal} (\text{dblock} \ s' p) \)
by (auto simp add: Inv2c-inner-def EndPhase1-def)

with \( \text{inv2a} \)
have \( \text{bal} (\text{dblock} \ s' p) \in \text{Ballot} \ p \cup \{ 0 \} \)
by (auto simp add: Inv2a-def Inv2a-inner-def Inv2a-innermost-def blocksOf-def)

with \( \text{bal-pos} \) have \( \text{bal-in-p}: \ \text{bal} (\text{dblock} \ s' p) \in \text{Ballot} \ p \)
by auto

from \( \text{inv2a} \) have \( \text{Inv2a-inner} \ s \ q \)
by (auto simp add: Inv2a-def)

hence \( \forall d \in D. \ \text{mbal} (\text{disk} \ s \ d \ q) \in \text{Ballot} \ q \cup \{ 0 \} \)
by (auto simp add: Inv2a-inner-def Inv2a-innermost-def blocksOf-def)

with \( p24 \) bal-pos
have \( \forall d \in D. \ \text{mbal} (\text{disk} \ s \ d \ q) \in \text{Ballot} \ q \)
by force

with \( \text{Ballot-disj} p23 \) bal-in-p
have \( \forall d \in D. \ \text{mbal} (\text{disk} \ s \ d \ q) \neq \text{bal} (\text{dblock} \ s' p) \)
by force

with \( p23 \ p24 \)
show \( ?\text{thesis} \)
by force

qed

with \( p23 \) \( \text{act} \)
have \( \forall d \in D. \ \text{bal} (\text{dblock} \ s' p) < \text{mbal} (\text{disk} \ s' d \ q) \land \neg \text{hasRead} \ s' p \ d \ q \)
by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def)

with \( D\text{maj} \)
```plaintext
show ?thesis
  by blast
qed

lemma union-inclusion:
  \[ A \subseteq A' ; B \subseteq B' \] \implies A \cup B \subseteq A' \cup B'
by blast

lemma HEndPhase1-blocksOf-q:
  assumes act: HEndPhase1 s s' p
  and png: p \neq q
  shows blocksOf s' q \subseteq blocksOf s q
proof –
from act png
have dblock: \{ dblock s' q \} \subseteq \{ dblock s q \}
  and disk: disk s' = disk s
  and blk: blocksRead s' q = blocksRead s q
  by(auto simp add: EndPhase1-def InitializePhase-def)
from disk
have disk': \{ disk s' d q | d . d \in UNIV \} \subseteq \{ disk s d q | d . d \in UNIV \} (is ?D)
  by auto
from png act
have (UN qq d. rdBy s' q qq d) \subseteq (UN qq d. rdBy s q qq d)
  by(auto simp add: EndPhase1-def InitializePhase-def rdBy-def split: if-split-asm, blast)
  hence \{ block br | br . br \in (UN qq d. rdBy s' q qq d) \} \subseteq \{ block br | br . br \in (UN qq d. rdBy s q qq d) \} (is ?R)
  by auto blast
from union-inclusion[OF dblock union-inclusion[OF disk' this]]
show ?thesis
  by(auto simp add: blocksOf-def)
qed

lemma HEndPhase1-allBlocks:
  assumes act: HEndPhase1 s s' p
  shows allBlocks s' \subseteq allBlocks s \cup \{ dblock s' p \}
proof(auto simp del: HEndPhase1-def simp add: allBlocks-def
  dest: HEndPhase1-blocksOf-q[OF act])
fix x pa
assume x-pa: x \in blocksOf s' pa and
  x-nblks: \forall xa. x \notin blocksOf s xa
show x=dblock s' p
proof(cases p=pa)
case True
  from x-nblks
  have x \notin blocksOf s p
  by auto
  with True subsetD[OF HEndPhase1-blocksOf[OF act] x-pa]

104
```
show \( ?thesis \)
by auto
next
  case False
from \( x\text{-nblks} \ subset D[0 \ F H\text{EndPhase1}\text{-blocks}\text{Of}-q[0 \ F \text{act} \ False] \ x-pa] \)
show \( ?thesis \)
by auto
qed
qed

lemma \( H\text{EndPhase1}\text{-HInv5}\text{-q} \)
assumes \( \text{act} \): \( H\text{EndPhase1} \ \text{s s'} p \)
and \( \text{inv} \): \( H\text{Inv5} \ \text{s} \)
and \( \text{inv1} \): \( \text{Inv1} \ \text{s} \)
and \( \text{inv2a} \): \( \text{Inv2a} \ \text{s'} \)
and \( \text{inv2a-q} \): \( \text{Inv2a} \ \text{s} \)
and \( \text{inv2b} \): \( \text{Inv2b} \ \text{s} \)
and \( \text{inv2c} \): \( \text{Inv2c} \ \text{s} \)
and \( \text{inv3} \): \( H\text{Inv3} \ \text{s} \)
and \( \text{phase'} \): \( \text{phase} \ \text{s'} q = 2 \)
and \( \text{pnq} \): \( p \neq q \)
and \( \text{asm4':} \ \neg \text{maxBalInp} \ \text{s'} \ (\text{bal}(\text{dblock} \ \text{s'} \ q)) \) \( (\text{inp}(\text{dblock} \ \text{s'} \ q)) \)
shows \( \exists \ D \in \text{MajoritySet}. \ \exists \ \text{qq}. \ (\forall d \in D. \ \text{bal}(\text{dblock} \ \text{s'} \ q) < \text{mbal}(\text{disk} \ \text{d} \ \text{qq}) \)
\( \land \ \neg \text{hasRead} \ \text{s' q d q}) \)

proof –
from \( \text{act} \ \text{pnq} \)
have \( \text{phase s'} q = \text{phase s} q \)
  and \( \text{phase-p} \): \( \text{phase s p} = 1 \)
  and \( \text{disk} \): \( \text{disk} s' = \text{disk} s \)
  and \( \text{dblock} \): \( \text{dblock} s' q = \text{dblock} s q \)
  and \( \text{bal} \): \( \text{bal}(\text{dblock} s' p) = \text{mbal}(\text{dblock} s p) \)
by (auto simp add: \( \text{EndPhase1-def InitializePhase-def} \))
with \( \text{phase'} \)
have \( \text{phase} \): \( \text{phase s q} = 2 \) by auto
from \( \text{phase inv2c} \)
have \( \text{bal-dblk-q} \): \( \text{bal}(\text{dblock} s q) \in \text{Ballot} q \)
  by (auto simp add: \( \text{Inv2c-def Inv2c-inner-def} \))
have \( \exists D \in \text{MajoritySet.} \ \exists \ \text{qq}. \ (\forall d \in D. \ \text{bal}(\text{dblock} \ \text{s} \ q) < \text{mbal}(\text{disk} \ \text{d} \ \text{qq}) \)
\( \land \ \neg \text{hasRead} \ \text{s q d q}) \)
proof (cases \( \text{maxBalInp} \ s \ (\text{bal}(\text{dblock} s q)) \) \( (\text{inp}(\text{dblock} s q)) \))
case True
  have \( p21 \): \( \text{bal}(\text{dblock} s q) < \text{bal}(\text{dblock} s' p) \) \( \land \ \text{inp}(\text{dblock} s q) \neq \text{inp}(\text{dblock} s' p) \)
by (auto simp add: \( \text{maxBalInp-def} \))
from \( \text{inv2a} \)
have \( \text{bal}(\text{dblock } s' p) \in \text{Ballot } p \cup \{ \theta \} \)
by (auto simp add: Inv2a-def Inv2a-inner-def
    Inv2a-innermost-def blocksOf-def)

moreover
from Ballot-disj Ballot-nzero pnq
have \( \text{Ballot } q \cap (\text{Ballot } p \cup \{ \theta \}) = \{ \} \)
by auto
ultimately
have \( \text{bal}(\text{dblock } s' p) \neq \text{bal}(\text{dblock } s q) \)
using bal-dblk-q
by auto
with p32
show \(?\text{thesis}\)
by auto

have \( \exists D \in \text{MajoritySet}. \forall d \in D. \text{bal}(\text{dblock } s q) < \text{mbal}(\text{disk } s d p) \land \text{hasRead } s p d q \)
proof
  from act
  have \( \exists D \in \text{MajoritySet}. \forall d \in D. d \in \text{disksWritten } s p \land (\forall q \in \text{UNIV} - \{ p \}. \text{hasRead } s p d q) \)
  by (auto simp add: EndPhase1-def MajoritySet-def)
  then obtain \( D \)
  where \( \text{act1}: \forall d \in D. d \in \text{disksWritten } s p \land (\forall q \in \text{UNIV} - \{ p \}. \text{hasRead } s p d q) \)
      and \( \text{Dmaj}: D \in \text{MajoritySet} \)
  by auto
from inv2b
have \( \forall d. \text{Inv2b-inner } s p d \) by (auto simp add: Inv2b-def)
with act1 pnq phase-p bal
have \( \forall d \in D. \text{bal}(\text{dblock } s' p) = \text{mbal}(\text{disk } s d p) \land \text{hasRead } s p d q \)
  by (auto simp add: Inv2b-def Inv2b-inner-def)
with p21 Dmaj
have \( \forall d \in D. \text{bal}(\text{dblock } s q) < \text{mbal}(\text{disk } s d p) \land \text{hasRead } s p d q \)
  by auto
with Dmaj
show \(?\text{thesis}\)
  by auto
qed
then obtain \( D \)
  where \( \text{p22}: D \in \text{MajoritySet} \land (\forall d \in D. \text{bal}(\text{dblock } s q) < \text{mbal}(\text{disk } s d p) \land \text{hasRead } s p d q) \)
  by auto
have \( \text{p23}: \forall d \in D. \{ \text{block} = \text{dblock } s q, \ \text{proc} = q \} \notin \text{blocksRead } s p d \)
proof
  have \( \text{dblock } s q \in \text{allBlocksRead } s p \longrightarrow \text{inp}(\text{dblock } s' p) = \text{inp}(\text{dblock } s q) \)
  proof
    assume \( \text{dblock-q}: \text{dblock } s q \in \text{allBlocksRead } s p \)
    from inv2a-q

106
have \((\text{bal}(\text{dblock } s \ q) = 0) = (\text{inp}(\text{dblock } s \ q) = \text{NotAnInput})\)
  by (auto simp add: Inv2a-def Inv2a-inner-def blocksOf-def Inv2a-innermost-def)
with bal-dblk-q Ballot-nzero dblock-q InputsOrNi
have dblock-q-nib: dblock s q ∈ nonInitBlks s p
  by (auto simp add: nonInitBlks-def blocksSeen-def)
with act
have dblock-max: \(\text{inp}(\text{dblock } s' p) = \text{inp}(\text{maxBlk } s p)\)
  by (auto simp add: EndPhase1-def)
from maxBlk-in-nonInitBlks[OF dblock-q-nib inv1]
have max-in-nib: maxBlk s p ∈ nonInitBlks s p ..
hence nonInitBlks s p ⊆ allBlks
  by (auto simp add: allBlks-def nonInitBlks-def blocksSeen-def blocksOf-def rdBy-def allBlocksRead-def allRdBlks-def)
with True subsetD[OF this max-in-nib]
have bal (dblock s q) ≤ bal (maxBlk s p) → inp (maxBlk s p) = inp (dblock s q)
  by (auto simp add: maxBalInp-def)
with maxBlk-in-nonInitBlks[OF dblock-q-nib inv1]
dblock-q-nib dblock-max
show inp(dblock s' p) = inp(dblock s q)
  by auto
qed
with p21
have dblock s q ∉ block ' allRdBlks s p
  by (auto simp add: allBlocksRead-def)
hence ∃ d. dblock s q ∉ block ' blocksRead s p d
  by (auto simp add: allRdBlks-def)
thus ?thesis
  by force
qed
have p24: \(\forall d ∈ D. \neg(∃ br ∈ \text{blocksRead } s q d. \text{bal}(\text{dblock } s q) ≤ \text{mbal}(\text{block } br))\)
proof
  from inv2c phase
  have \(\forall d. \exists \text{ br } ∈ \text{blocksRead } s q d. \text{mbal}(\text{block } br) < \text{mbal}(\text{dblock } s q)\)
    and \(\text{bal}(\text{dblock } s q) = \text{mbal}(\text{dblock } s q)\)
    by (auto simp add: Inv2c-def Inv2c-inner-def)
  thus ?thesis
    by force
qed
have p25: \(\forall d ∈ D. \neg\text{hasRead } s q d p\)
proof auto
  fix d
  assume d-in-D: \(d ∈ D\)
  and hasRead-gdp: \(\text{hasRead } s q d p\)
  have p31: \(∃ \text{ br } = \text{dblock } s p, \text{proc} = p) ∈ \text{blocksRead } s q d\)
  proof
    from d-in-D p22
  qed
have hasRead-pdq; hasRead s p d q by auto
with hasRead-qdp phase phase-p inv3
have HInv3-R s q p d
  by(auto simp add: HInv3-def HInv3-inner-def HInv3-L-def)
with p23 d-in-D
show ?thesis
  by(auto simp add: HInv3-R-def)
qed
from p21 act
have p32: bal(dblock s q) < mbal(dblock s p)
  by(auto simp add: EndPhase1-def)
with p31 d-in-D hasRead-qdp p24
show False
  by(force)
qed
with p22
show ?thesis
  by auto
next
case False
with inv phase
show ?thesis
  by(auto simp add: HInv5-def HInv5-inner-def HInv5-inner-R-def)
qed
then obtain D qq
  where D∈ MajoritySet ∧ (∀d∈D. bal(dblock s q) < mbal(disk s d qq)
    ∧ ¬hasRead s q d qq)
  by auto
moreover from act pnq
have ∀d. hasRead s′ q d qq = hasRead s q d qq
  by(auto simp add: EndPhase1-def InitializePhase-def hasRead-def)
ultimately show ?thesis
  using disk dblock
  by auto
qed

theorem HEndPhase1-HInv5:
assumes act: HEndPhase1 s s′ p
and inv: HInv5 s
and inv1: Inv1 s
and inv2a: Inv2a s
and inv2a′: Inv2a′ s′
and inv2b: Inv2b s
and inv2c: Inv2c s
and inv3: HInv3 s
and inv4: HInv4 s
shows HInv5-inner s′ q
using HEndPhase1-HInv5-p[OF act inv4 inv2a inv2a′ inv2c]
lemma HFail-HInv5-p:
HFail s s' p \implies HInv5-inner s' p
by (auto simp add: Fail-def HInv5-inner-def HInv5-inner-R-def)

lemma HFail-blocksOf-q:
assumes act: HFail s s' p
and pnq: p \neq q
shows blocksOf s' q \subseteq blocksOf s q
using assms
by (auto simp add: Fail-def InitializePhase-def blocksOf-def rdBy-def)

lemma HFail-allBlocks:
assumes act: HFail s s' p
shows allBlocks s' \subseteq allBlocks s \cup \{ dblock s' p \}
proof (auto simp del: HFail-def simp add: allBlocks-def
dest: HFail-blocksOf-q (OF act])
fix x pa
assume x-pa: x \in blocksOf s' pa and
x-nblks: \( \forall xa. x \notin blocksOf s xa \)
show x=dblock s' p
proof (cases p=pa)
case True
from x-nblks
have x \notin blocksOf s p
by auto
with True subsetD [OF HFail-blocksOf-q (OF act x-pa]
show \?thesis
by auto
next
case False
from x-nblks subsetD [OF HFail-blocksOf-q (OF act False x-pa]
show \?thesis
by auto
qed
qed

lemma HFail-HInv5-q1:
assumes act: HFail s s' p
and pnq: p \neq q
and inv2a: Inv2a-inner s' q
and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q))
shows maxBalInp s' (bal(dblock s' q)) (inp(dblock s' q))
proof (auto simp add: maxBalInp-def)
fix bk
assume bk: bk \in allBlocks s'
and bal: bal (dblock s' q) \leq bal bk
from act pnq
have dblock': dblock' s' q = dblock s q by (auto simp add: Fail-def)
from subsetD[OF HFail-allBlocks[OF act] bk]
show inp bk = inp (dblock s' q)
proof
  assume bk: bk ∈ allBlocks s
  with inv5-1 dblock' bal
  show ?thesis
    by (auto simp add: Fail-def)
next
  assume bk: bk ∈ {dblock s' p}
  with act have bk-init: bk = InitDB
  with bal
  have bal (dblock s' q)=0
    by (auto simp add: InitDB-def)
  with inv2a
  have inp (dblock s' q)= NotAnInput
    by (auto simp add: Inv2a-inner-def Inv2a-innermost-def blocksOf-def)
  with bk-init
  show ?thesis
    by (auto simp add: InitDB-def)
qed

lemma HFail-HInv5-q2:
  assumes act: HFail s s' p
  and pnq: p≠q
  and inv5-2: ∃ D∈MajoritySet. ∃ qq. (∀ d∈D. bal(dblock s q) < mbal(disk s d qq)
    ∧ ¬hasRead s q d qq)
  shows ∃ D∈MajoritySet. ∃ qq. (∀ d∈D. bal(dblock s' q) < mbal(disk s' d qq)
    ∧ ¬hasRead s' q d qq)
proof –
  from act pnq
  have disk: disk s' = disk s
    and blocksRead: ∀ d. blocksRead s' q d = blocksRead s q d
    and dblock: dblock s' q = dblock s q
    by (auto simp add: Fail-def InitializePhase-def)
  with inv5-2
  show ?thesis
    by (auto simp add: hasRead-def)
qed

lemma HFail-HInv5-q:
  assumes act: HFail s s' p
  and inv: HInv5-inner s q
  and pnq: p≠q
  and inv2a: Inv2a s'
shows $HInv5$-inner $s' q$
proof(auto simp add: $HInv5$-inner-def $HInv5$-inner-R-def)
  assume phase': phase $s' q = 2$
  and nR2: $\forall D \in \text{MajoritySet}$. $\forall qa. \exists d \in D. \text{bal (dblock } s' q) < \text{mbal (disk } d \text{ qa) } \longrightarrow\text{ hasRead } s' q d qa (\text{is } ?P s')$
from $HFail-HInv5$-q2[OF act pnq]
  have $\neg (\exists P s) \Longrightarrow \neg (\exists P s')$
    by auto
with nR2
  have P: $\exists P s$
    by blast
from inv2a
  have inv2a': $\exists s' q (\text{Inv2a-inner } s' q)$ by (auto simp add: Inv2a-def)
from act pnq phase'
  have phase $s q = 2$
    by (auto simp add: Fail-def split: if-split-asm)
with inv $HFail-HInv5$-q1[OF act pnq inv2a'] P
  show maxBalInp $s' (\text{bal (dblock } s' q)) (\text{inp (dblock } s' q))$
    by (auto simp add: $HInv5$-inner-def $HInv5$-inner-R-def)
qed

theorem $HFail-HInv5$:
[ $HFail s s' p; HInv5$-inner $s q; \text{Inv2a } s' q$ ] $\Longrightarrow HInv5$-inner $s' q$
by(blast dest: $HFail-HInv5$-q $HFail-HInv5$-p)

lemma $HPhase0Read-HInv5$-p:
  $HPhase0Read s s' p d \Longrightarrow HInv5$-inner $s' p$
by(auto simp add: $HPhase0Read$-def $HInv5$-inner-def)

lemma $HPhase0Read$-allBlocks:
  assumes act: $HPhase0Read$ $s s' p d$
  shows allBlocks $s' \subseteq$ allBlocks $s$
  using $HPhase0Read$-blocksOf[OF act]
by(auto simp add: allBlocks-def)

lemma $HPhase0Read-HInv5$-1:
  assumes act: $HPhase0Read$ $s s' p d$
  and inv5-1: maxBalInp $s (\text{bal (dblock } s q)) (\text{inp (dblock } s q))$
  shows maxBalInp $s' (\text{bal (dblock } s' q)) (\text{inp (dblock } s' q))$
  using assms and $HPhase0Read$-blocksOf[OF act]
by(auto simp add: $HPhase0Read$-def maxBalInp-def allBlocks-def)

lemma $HPhase0Read-HInv5$-q2:
  assumes act: $HPhase0Read$ $s s' p d$
  and pnq: $p \neq q$
  and inv5-2: $\exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \text{bal (dblock } s q) < \text{mbal (disk } d \text{ qq) } \land \neg \text{hasRead } s q d qq)$
  shows $HPhase0Read$ $s s' p d$
by(auto simp add: $HPhase0Read$-def maxBalInp-def allBlocks-def)
shows $\exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \text{bal}(\text{dblock } s' q) < \text{mbal}(\text{disk } s' d qq) \\
\land \neg \text{hasRead } s' q d qq)$

proof –
from act pnq
have disk: disk $s' = disk s$
and blocksRead: $\forall d. \text{blocksRead } s' q d = \text{blocksRead } s q d$
and dblock: $\text{dblock } s' q = \text{dblock } s q$
by(auto simp add: Phase0Read-def InitializePhase-def)
with inv5-2
show ?thesis
by(auto simp add: hasRead-def)
qed

lemma HPhase0Read-HInv5-q:
assumes act: HPhase0Read $s s' p d$
and inv: HInv5-inner $s q$
and pnq: $p \neq q$
shows HInv5-inner $s' q$
proof(auto simp add: HInv5-inner-def HInv5-inner-R-def)
assume phase': phase $s' q = 2$
and i2: $\forall D \in \text{MajoritySet}. \forall qa. \exists d \in D. \text{bal}(\text{dblock } s' q d qa) \\
\rightarrow \text{hasRead } s' q d qa$ from phase' act have phase: phase $s q = 2$
by(auto simp add: Phase0Read-def)
show maxBalInp $s' (\text{bal}(\text{dblock } s' q)) (\text{inp}(\text{dblock } s' q))$
proof(rule HPhase0Read-HInv5-1[OF act, of $q$])
from HPhase0Read-HInv5-2[OF act pnq] inv i2 phase
show maxBalInp $s (\text{bal}(\text{dblock } s q)) (\text{inp}(\text{dblock } s q))$
by(auto simp add: HInv5-inner-def HInv5-inner-R-def, blast)
qed

lemma HPhase0Read-HInv5:
[ HPhase0Read $s s' p d; \text{HInv5-inner } s q \Rightarrow \text{HInv5-inner } s' q$ by(blast dest: HPhase0Read-HInv5-q HPhase0Read-HInv5-p) ]

lemma HEndPhase0-HInv5-p:
HEndPhase0 $s s' p \Rightarrow \text{HInv5-inner } s' p$
by(auto simp add: EndPhase0-def HInv5-inner-def)

lemma HEndPhase0-blocksOf-q:
assumes act: HEndPhase0 $s s' p$
and pnq: $p \neq q$
shows blocksOf $s' q \subseteq \text{blocksOf } s q$
proof –
from act pnq
have dblock: $\{\text{dblock } s' q\} \subseteq \{\text{dblock } s q\}$
and disk: disk $s' = disk s$
and blks: blocksRead s' q = blocksRead s q
by (auto simp add: EndPhase0-def InitializePhase-def)
from disk
have disk': \{ disk s' d q \mid d : d \in \text{UNIV} \} \subseteq \{ disk s d q \mid d : d \in \text{UNIV} \} (\text{is } ?D' \subseteq ?D)
  by auto
from pnq act
have (UN qq d. rdBy s' q qq d) \subseteq (UN qq d. rdBy s q qq d)
  by (auto simp add: EndPhase0-def InitializePhase-def rdBy-def split: if-split-asm, blast)
hence \{ block br \mid br : br \in (UN qq d. rdBy s' q qq d) \} \subseteq
  \{ block br \mid br : br \in (UN qq d. rdBy s q qq d) \}
  (\text{is } ?R' \subseteq ?R)
  by auto blast
from union-inclusion[OF dblock union-inclusion[OF disk' this]]
show ?thesis
  by (auto simp add: blocksOf-def)
qed

lemma HEndPhase0-allBlocks:
assumes act: HEndPhase0 s s' p
shows allBlocks s' \subseteq allBlocks s \cup \{ dblock s' p \}
proof (auto simp del: HEndPhase0-def simp add: allBlocks-def
dest: HEndPhase0-blocksOf-q[OF act])
fix x pa
assume x-pa: x \in blocksOf s' pa and
  x-nblks: \forall xa. x \notin blocksOf s xa
show x=dblock s' p
proof (cases p=pa)
case True
  from x-nblks
  have x \notin blocksOf s p
    by auto
  with True subsetD[OF HEndPhase0-blocksOf[OF act] x-pa]
  show ?thesis
    by auto
next
case False
  from x-nblks subsetD[OF HEndPhase0-blocksOf-q[OF act False] x-pa]
  show ?thesis
    by auto
qed

lemma HEndPhase0-Hinv5-q1:
assumes act: HEndPhase0 s s' p
and pnq: p\neq q
and inv1: Inv1 s
and inv5-1: maxBalInp s (bal(dblock s q)) (inp(dblock s q))
shows \( \text{maxBalInp} s' \ (\text{bal}(\text{dblock} s' q)) \ (\text{inp}(\text{dblock} s' q)) \)

**proof** (auto simp add: maxBalInp-def)

fix \( bk \)

assume \( bk \in \text{allBlocks} s' \)

and \( \text{bal}: \text{bal} (\text{dblock} s' q) \leq \text{bal} bk \)

from act pnq
have \( \text{dblock}' s q = \text{dblock} s q \) by (auto simp add: EndPhase0-def)

from subsetD[OF HEndPhase0-allBlocks[OF act] bk]
show \( \text{inp} bk = \text{inp} (\text{dblock} s q) \)

next
assume \( bk \in \{\text{dblock} s' p\} \)

with HEndPhase0-some[OF act inv1]
have \( \exists ba \in \text{allBlocksRead} s p. \text{bal} ba = \text{bal} (\text{dblock} s' p) \wedge \text{inp} ba = \text{inp} (\text{dblock} s' p) \)

by (auto simp add: EndPhase0-def)

then obtain \( ba \)

where \( \text{ba-blksread}: ba \in \text{allBlocksRead} s p \)

and \( \text{ba-balinp}: \text{bal} ba = \text{bal} (\text{dblock} s' p) \wedge \text{inp} ba = \text{inp} (\text{dblock} s' p) \)

by auto

have allBlocksRead s p \( \subseteq \) allBlocks s

by (auto simp add: allBlocksRead-def allRdBlks-def allBlocks-def blocksOf-def rdBy-def)

from subsetD[OF OF this ba-blksread] ba-balinp bal bk dblock' inv5-1
show \( \text{thesis} \)

by (auto simp add: maxBalInp-def)

qed

**lemma** HEndPhase0-HInv5-q2:

assumes act: HEndPhase0 s s' p

and pnq: \( p \neq q \)

and inv5-2: \( \exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \text{bal}(\text{dblock} s q) < \text{mbal}(\text{disk} s d qq) \wedge \neg \text{hasRead} s q d qq) \)

shows \( \exists D \in \text{MajoritySet}. \exists qq. (\forall d \in D. \text{bal}(\text{dblock} s' q) < \text{mbal}(\text{disk} s' d qq) \wedge \neg \text{hasRead} s' q d qq) \)

**proof** –
from act pnq
have \( \text{disk}: \text{disk} s' = \text{disk} s \)

and \( \text{blocksRead}: \forall d. \text{blocksRead} s' q d = \text{blocksRead} s q d \)

and \( \text{dblock}: \text{dblock} s' q = \text{dblock} s q \)

by (auto simp add: EndPhase0-def InitializePhase-def)

with inv5-2
show \( \text{thesis} \)
by (auto simp add: hasRead-def)

qed

lemma HEndPhase0-HInv5-q:
  assumes act: HEndPhase0 s s' p
  and inv: HInv5-inner s q
  and inv1: Inv1 s
  and pnq: p ≠ q
  shows HInv5-inner s' q
  using assms and
  HEndPhase0-HInv5-q1 [OF act pnq inv1]
  HEndPhase0-HInv5-q2 [OF act pnq]
  by (auto simp add: HInv5-inner-def HInv5-inner-R-def EndPhase0-def)

theorem HEndPhase0-HInv5:
  [ HEndPhase0 s s' p; HInv5-inner s q; Inv1 s ] ⇒ HInv5-inner s' q
  by (blast dest: HEndPhase0-HInv5-q HEndPhase0-HInv5-p)

HInv1 ∧ HInv2 ∧ HInv3 ∧ HInv4 ∧ HInv5 is an invariant of HNext.

lemma I2c:
  assumes nxt: HNext s s'
  and inv: HInv1 s ∧ HInv2 s ∧ HInv2 s' ∧ HInv3 s ∧ HInv4 s ∧ HInv5 s
  shows HInv5 s'
  using assms
  by (auto simp add: HInv5-def HNext-def Next-def,
    auto simp add: HInv2-def intro: HStartBallot-HInv5,
    auto intro: HPhase0Read-HInv5,
    auto simp add: HInv4-def intro: HPhase1or2Write-HInv5,
    auto simp add: Phase1or2Read-def
      intro: HPhase1or2ReadThen-HInv5
      HPhase1or2ReadElse-HInv5,
    auto simp add: EndPhase1or2-def HInv1-def HInv4-def HInv5-def
      intro: HEndPhase1-HInv5
    HEndPhase2-HInv5,
    auto intro: HFail-HInv5,
    auto intro: HEndPhase0-HInv5 simp add: HInv1-def)

end

theory DiskPaxos-Chosen imports DiskPaxos-Inv5 begin

C.6 Lemma I2f

To prove the final conjunct we will use the predicate valueChosen(v). This
predicate is true if v is the only possible value that can be chosen as output.
It also asserts that, for every disk d in D, if q has already read disksdp, then
it has read a block with bal field at least b.
**definition** valueChosen :: state ⇒ InputsOrNi ⇒ bool

where

\[
\text{valueChosen } s \ v = \\
(\exists b \in (\text{UN } p \ . \ \text{Ballot } p) . \\
\text{maxBalInp } s \ b \ v \\
\land (\exists p. \exists D \in \text{MajoritySet}(\forall d \in D. \ b \leq \text{bal}(\text{disk } s \ d \ p) \\
\land (\forall q. (\text{phase } s \ q = 1 \\
\land b \leq \text{mbal}(\text{dblock } s \ q) \\
\land \text{hasRead } s q d p \\
\implies (\exists \text{br} \in \text{blocksRead } s q d. \ b \leq \text{bal}(\text{block } \text{br})))))
\]

**lemma** HEndPhase1-valueChosen-inp:

**assumes** act: HEndPhase1 s s' q

and \text{inv2a}: Inv2a s

and \text{asm1}: b \in (\text{UN } p \ . \ \text{Ballot } p)

and \text{bk-blocksOf}: bk \in \text{blocksOf } s \ r

and \text{bk}: bk \in \text{blocksSeen } s q

and \text{b-bal}: b \leq \text{bal } bk

and \text{asm3}: \text{maxBalInp } s \ b \ v

and \text{inv1}: Inv1 s

shows \text{inp}(\text{dblock } s' q) = v

**proof** –

from \text{bk-blocksOf} \text{inv2a}

have \text{inv2a-bk}: Inv2a-innermost s r bk

by (auto simp add: Inv2a-def Inv2a-inner-def)

from \text{Ballot-nzero} \text{asm1}

have \text{0 }< \text{b} by auto

with \text{b-bal}

have \text{0 }< \text{bal } bk by auto

with \text{inv2a-bk}

have \text{inp } bk \neq \text{NotAnInput}

by (auto simp add: Inv2a-innermost-def)

with \text{bk} \text{InputsOrNi}

have \text{bk-noninit}: bk \in \text{nonInitBlks } s q

by (auto simp add: nonInitBlks-def blocksSeen-def

blocksRead-def allRdBlks-def)

with \text{maxBlk} \text{-nonInitBlks}[OF \ this \ \text{inv1}] \text{b-bal}

have \text{maxBlk-b}: b \leq \text{bal}(\text{maxBlk } s q)

by auto

from \text{maxBlk} \text{-nonInitBlks}[OF \ \text{bk-noninit} \text{inv1}]

have \text{3 p d. maxBlk } s q \in \text{blocksSeen } s p

by (auto simp add: nonInitBlks-def blocksSeen-def

blocksRead-def allRdBlks-def rdBy-def, force)

with \text{maxBlk-b} \text{asm3}

have \text{inp}(\text{maxBlk } s q) = v

by (auto simp add: maxBalInp-def allBlocks-def)
with bk-noninit act
show ?thesis
by (auto simp add: EndPhase1-def)
qed

lemma HEndPhase1-maxBalInp:
assumes act: HEndPhase1 s s' q
and asm1: b ∈ (UN p. Ballot p)
and asm2: D ∈ MajoritySet
and asm3: maxBalInp s b v
and asm4: ∀ d ∈ D. b ≤ bal(dblock s d p)
∧ (∀ q. (phase s q = 1
∧ b ≤ mbal(dblock s q)
∧ hasRead s q d p
) −→ (∃ br ∈ blocksRead s q d. b ≤ bal(block br)))

and inv1: Inv1 s
and inv2a: Inv2a s
and inv2b: Inv2b s
shows maxBalInp s' b v
proof(cases b ≤ mbal(dblock s q))
case True
show ?thesis
proof -
from act
have IsMajority({d. d ∈ disksWritten s q ∧ (∀ r ∈ UNIV − {q}. hasRead s q d r)}) (is IsMajority(?M))
  by (auto simp add: EndPhase1-def)
with majorities-intersect asm2
have D ∩ ?M ≠ {}
  by (auto simp add: MajoritySet-def)
hence ∃ d ∈ D. (∀ r ∈ UNIV − {q}. hasRead s q d r)
  by auto
with pnq
show ?thesis
  by auto
qed
then obtain d where p41: d ∈ D ∧ hasRead s q d p by auto
with asm4 asm3 act True
have p42: ∃ br ∈ blocksRead s q d. b ≤ bal(block br)
  by (auto simp add: EndPhase1-def)
from True act
have thesis-L: b ≤ bal (dblock s' q)
  by (auto simp add: EndPhase1-def)
from p42
have inp(dblock s' q) = v
proof auto
  fix br
  assume br: br ∈ blocksRead s q d
  and b-bal: b ≤ bal (block br)
hence br-rdBy: br ∈ (UN q d. rdBy s (proc br) q d)
  by(auto simp add: rdBy-def)
hence br-blksof: block br ∈ blocksOf s (proc br)
  by(auto simp add: blocksOf-def)
from br have br-bseen: block br ∈ blocksSeen s q
  by(auto simp add: blocksSeen-def allRdBlks-def)
from HEndPhase1-valueChosen-imp[OF act inv2a asm1 br-blksof br-bseen b-bal
asm3 inv1]
  show ?thesis .
  qed
next
  case False
  from asm4
  have p41: ∀ d∈D. b ≤ bal(disk s d p)
    by auto
  have p42: ∃ d∈D. disk s d p = dblock s p
  proof –
    from act
    have IsMajority {d. d∈disksWritten s q ∧ (∀ p∈UNIV − {q}. hasRead s q d p)} (is IsMajority ?S)
      by(auto simp add: EndPhase1-def)
    with majorities-intersect asm2
    have D ∩ ?S ≠ {}
      by(auto simp add: MajoritySet-def)
    hence ∃ d∈D. d∈disksWritten s q
      by auto
    with inv2b False
    show ?thesis
      by(auto simp add: Inv2b-def Inv2b-inner-def)
  qed
  have inp(dblock s' q) = v
  proof –
    from p42 p41 False
    have b-bal: b ≤ bal(dblock s q) by auto
    have db-blksof: (dblock s q) ∈ blocksOf s q
      by(auto simp add: blocksOf-def)
    have db-bseen: (dblock s q) ∈ blocksSeen s q
      by(auto simp add: blocksSeen-def)
    from HEndPhase1-valueChosen-imp[OF act inv2a asm1 db-blksof db-bseen
b-bal asm3 inv1]
    show ?thesis .
  qed
with \(\text{asm3 HEndPhase1-allBlocks[OF act]}\)

show \(?\text{thesis}\)
  by(auto simp add: maxBalInp-def)
qed

next
case False
have \(\text{dblock s' q \in allBlocks s'}\)
  by(auto simp add: allBlocks-def blocksOf-def)
show \(?\text{thesis}\)
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: \(bk \in \text{allBlocks s}\)
  and b-bal: \(b \leq \text{bal bk}\)
  from \(\text{subsetD[OF HEndPhase1-allBlocks[OF act] bk]}\)
  show \(\text{inp bk = v}\)
    proof
      assume bk: \(bk \in \text{allBlocks s}\)
      with \(\text{asm3 b-bal}\)
      show \(?\text{thesis}\)
        by(auto simp add: maxBalInp-def)
    next
      assume bk: \(bk \in \{\text{dblock s' q}\}\)
      from \(\text{act False}\)
      have \(\neg b \leq \text{bal (dblock s' q)}\)
        by(auto simp add: EndPhase1-def)
      with bk b-bal
      show \(?\text{thesis}\)
        by(auto)
    qed
  qed

qed

lemma \(\text{HEndPhase1-valueChosen2}:\)
assumes act: \(\text{HEndPhase1 s s' q}\)
and asm4: \(\forall d \in D. b \leq \text{bal(disk s d p)}\)
\(\land (\forall q. (\text{phase s q = 1})\)
\(\land b \leq \text{mbal(dblock s q)}\)
\(\land \text{hasRead s q d p}\)
\(\rightarrow (\exists br \in \text{blocksRead s q d. b \leq \text{bal(block br)}))\) (is \(?P s)\)
shows \(?P s'\)
proof(auto)
  fix d
  assume d: \(d \in D\)
  with act asm4
  show \(b \leq \text{bal (disk s' d p)}\)
    by(auto simp add: EndPhase1-def)
  fix d q
  assume d: \(d \in D\)
  and phase': \(\text{phase s' q = Suc 0}\)
and blk-mbal: b ≤ mbal (blk s' q)
with act
have p31: phase s q = 1
and p32: blk s' q = blk s q
by (auto simp add: EndPhase1-def split: if-split-asm)
with blk-mbal
have b ≤ mbal (blk s q) by auto
moreover
assume hasRead: hasRead s' q d p
with act
have hasRead s q d p
by (auto simp add: EndPhase1-def InitializePhase-def
hasRead-def split: if-split-asm)
ultimately
have ∃ br ∈ blocksRead s q d. b ≤ bal (blk br)
using p31 asm4 d
by blast
with act hasRead
show ∃ br ∈ blocksRead s' q d. b ≤ bal (blk br)
by (auto simp add: EndPhase1-def InitializePhase-def hasRead-def)
qed

theorem HEndPhase1-valueChosen:
assumes act: HEndPhase1 s s' q
and vc: valueChosen s v
and inv1: Inv1 s
and inv2a: Inv2a s
and inv2b: Inv2b s
and v-input: v ∈ Inputs
shows valueChosen s' v
proof (−)
from vc
obtain b p D where
asm1: b ∈ (UN p. Ballot p)
and asm2: D ∈ MajoritySet
and asm3: maxBalInp s b v
and asm4: ∀ d ∈ D. b ≤ bal (blk s d p)
∀ q. (phase s q = 1
∧ b ≤ mbal (blk s q)
∧ hasRead s q d p)
→ (∃ br ∈ blocksRead s q d. b ≤ bal (blk br)))
by (auto simp add: valueChosen-def)
from HEndPhase1-maxBalInp[OF act asm1 asm2 asm3 asm4 inv1 inv2a inv2b]
have maxBalInp s' b v .
with HEndPhase1-valueChosen2[OF act asm4] asm1 asm2
show ?thesis
by (auto simp add: valueChosen-def)
qed
lemma HStartBallot-maxBalInp:
assumes act: HStartBallot s s' q
  and asm3: maxBalInp s b v
shows maxBalInp s' b v
proof (auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s
    and b-bal: b ≤ bal bk
  from subsetD[OF HStartBallot-allBlocks[OF act] bk]
  show inp bk = v
  proof
    assume bk: bk ∈ allBlocks s
    with asm3 b-bal
    show ?thesis
      by (auto simp add: maxBalInp-def)
  next
    assume bk: bk ∈ {dblock s' q}
    from asm3
    have b ≤ bal (dblock s q) −→ inp (dblock s q) = v
      by (auto simp add: maxBalInp-def allBlocks-def blocksOf-def)
    with act bk b-bal
    show ?thesis
      by (auto simp add: StartBallot-def)
  qed
  qed

lemma HStartBallot-valueChosen2:
assumes act: HStartBallot s s' q
  and asm4: ∀ d ∈ D. b ≤ bal (disk s d p)
    ∧ (∀ q. (phase s q = 1
      ∧ b ≤ mbal (dblock s q)
      ∧ hasRead s q d p
    ) −→ (∃ br ∈ blocksRead s q d. b ≤ bal (block br))) (is ?P s)
shows ?P s'
proof (auto)
  fix d
  assume d: d ∈ D
  with act asm4
  show b ≤ bal (disk s' d p)
    by (auto simp add: StartBallot-def)
  fix d q
  assume d: d ∈ D
    and phase': phase s' q = Suc 0
    and dblk-mbal: b ≤ mbal (dblock s' q)
    and hasRead: hasRead s' q d p
  from phase' act hasRead
  have p31: phase s q = 1
    and p32: dblock s' q = dblock s q
    by (auto simp add: StartBallot-def InitializePhase-def)
hasRead-def split : if-split-asm)
with dblk-mbal
have b≤mbal(dblock s q) by auto
moreover
from act hasRead
have hasRead s q d p
  by(auto simp add: StartBallot-def InitializePhase-def
      hasRead-def split: if-split-asm)
ultimately
have ∃ br∈blocksRead s q d. b≤bal(block br)
  using p31 asm4 d
  by blast
with act hasRead
show ∃ br∈blocksRead s' q d. b≤bal(block br)
  by(auto simp add: StartBallot-def InitializePhase-def
       hasRead-def)
qed

theorem HStartBallot-valueChosen:
  assumes act: HStartBallot s s' q
  and vc: valueChosen s v
  and v-input: v ∈ Inputs
  shows valueChosen s' v
proof
  from vc
  obtain b p D where
    asm1: b ∈ (UN p. Ballot p)
    and asm2: D∈MajoritySet
    and asm3: maxBalInp s b v
    and asm4: ∀ d∈D. b ≤ bal(disk s d p)
      ∧ (∀ q. ($phase s q = 1$
        ∧ b ≤ mbal(dblock s q)
        ∧ hasRead s q d p)
      ) → (∃ br∈blocksRead s q d. b ≤ bal(block br)))
  by(auto simp add: valueChosen-def)
  from HStartBallot-maxBalInp[OF act asm3]
  have maxBalInp s' b v .
  with HStartBallot-valueChosen2[OF act asm4] asm1 asm2
  show ?thesis
  by(auto simp add: valueChosen-def)
qed

lemma HPhase1or2Write-maxBalInp:
  assumes act: HPhase1or2Write s s' q d
  and asm3: maxBalInp s b v
  shows maxBalInp s' b v
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s'
and \( b \)-bal: \( b \leq \text{bal}_k \)

from \( \text{subsetD[} \text{OF HPhase1or2Write-allBlocks[} \text{OF act} \text{] bk] asm3 b\)-bal

show \( \text{inp bk = v} \)

by (auto simp add: \text{maxBalInp-def})

qed

**Lemma HPhase1or2Write-valueChosen2:**

assumes act: \( \text{HPhase1or2Write s s'} pp d \)

and \( \text{asm2: } D \in \text{MajoritySet} \)

and \( \text{asm4: } \forall d \in D. \ b \leq \text{bal(disk s d p)} \)

\( \wedge (\forall q.(\text{phase s q = 1}) \wedge b \leq \text{mbal(dblock s q)}) \wedge \text{hasRead s q d p} \)  \( \rightarrow (\exists br \in \text{blocksRead s q d}. b \leq \text{bal(block br)}) \) (is \( ?P s \))

and \( \text{inv4: } \text{HInv4a s pp} \)

shows \( ?P s' \)

**Proof (auto)**

fix \( d1 \)

assume \( d: d1 \in D \)

show \( b \leq \text{bal(disk s' d1 p)} \)

**Proof (cases \( d1 = d \wedge pp = p \))**

**Case True**

with \( \text{inv4 act} \)

have \( \text{HInv4a2 s p} \)

by (auto simp add: \text{Phase1or2Write-def HInv4a-def})

with \( \text{asm2 majority-intersect} \)

have \( \exists dd \in D. \ \text{bal(disk s dd p)} \leq \text{bal(dblock s p)} \)

by (auto simp add: \text{HInv4a2-def MajoritySet-def})

then obtain \( dd \) where \( p41: dd \in D \wedge \text{bal(disk s dd p)} \leq \text{bal(dblock s p)} \)

by auto

from \( \text{asm4 p41} \)

have \( b \leq \text{bal(disk s dd p)} \)

by auto

with \( p41 \)

have \( p42: b \leq \text{bal(dblock s p)} \)

by auto

from \( \text{act True} \)

have \( \text{dblock s p = disk s' d p} \)

by (auto simp add: \text{Phase1or2Write-def})

with \( p42 \) \( \text{True} \)

show \( ?\text{thesis} \)

by auto

next

**Case False**

with \( \text{act asm4 d} \)

show \( ?\text{thesis} \)

by (auto simp add: \text{Phase1or2Write-def})

qed

next
fix \( d \ q \)

**assume** \( d: d \in D \)
- and \( \text{phase' phase s' q = Suc 0} \)
- and \( \text{dblk-mbal: b \leq mbal (dblock s' q)} \)
- and \( \text{hasRead: hasRead s' q d p} \)

**from** \( \text{phase' act hasRead} \)

**have** \( p31: \text{phase s q = 1} \)
- and \( p32: \text{dblock s' q = dblock s q} \)
  - **by** \((\text{auto simp add: Phase1or2Write-def InitializePhase-def hasRead-def split: if-split-asm})\)

**with** \( \text{dblk-mbal} \)

**have** \( b \leq mbal(dblock s q) \) **by** \( \text{auto} \)

**moreover**

**from** \( \text{act hasRead} \)

**have** \( \text{hasRead s q d p} \)
  - **by** \((\text{auto simp add: Phase1or2Write-def InitializePhase-def hasRead-def split: if-split-asm})\)

**ultimately**

**have** \( \exists br \in \text{blocksRead s q d}. b \leq \text{bal(block br)} \)
  - **using** \( p31 \) \( \text{asm4 d} \)
  - **by** \( \text{blast} \)

**with** \( \text{act hasRead} \)

**show** \( \exists br \in \text{blocksRead s' q d}. b \leq \text{bal(block br)} \)
  - **by** \((\text{auto simp add: Phase1or2Write-def InitializePhase-def hasRead-def})\)

**qed**

**theorem** \( HPhase1or2Write-valueChosen: \)

**assumes** \( \text{act: HPhase1or2Write s s' q d} \)
- and \( \text{vc: valueChosen s v} \)
- and \( \text{v-input: v \in Inputs} \)
- and \( \text{inv4: HInv4a s q} \)

**shows** \( \text{valueChosen s' v} \)

**proof**

- **from** \( \text{vc} \)
  **obtain** \( b p D \) **where**
    - \( \text{asm1: b \in (UN p. Ballot p)} \)
    - \( \text{asm2: D \in MajoritySet} \)
    - \( \text{asm3: maxBallInp s b v} \)
    - \( \text{asm4: \forall d \in D. b \leq bal(disk s d p)} \)
      

- **by** \((\text{auto simp add: valueChosen-def})\)

**from** \( HPhase1or2Write-maxBallInp[OF act asm3] \)

**have** \( \text{maxBallInp s' b v} \)

**with** \( HPhase1or2Write-valueChosen2[OF act asm2 asm4 inv4] \) \( \text{asm1 asm2} \)

**show** \( ?\text{thesis} \)

124
by(auto simp add: valueChosen-def)
qed

lemma HPhase1or2ReadThen-maxBalInp:
  assumes act: HPhase1or2ReadThen s s' q d p
  and asm3: maxBalInp s b v
  shows maxBalInp s' b v
proof(auto simp add: maxBalInp-def)
  fix bk
  assume: bk: bk ∈ allBlocks s'
  and b-bal: b ≤ bal bk
  from subsetD[OF HPhase1or2ReadThen-allBlocks[OF act] bk] asm3 b-bal
  show inp bk = v
    by(auto simp add: maxBalInp-def)
qed

lemma HPhase1or2ReadThen-valueChosen2:
  assumes act: HPhase1or2ReadThen s s' q d pp
  and asm4: ∀ d∈D. b ≤ bal(disk s d p)
  ∧ (∀ q.( phase s q = 1
  ∧ b ≤ mbal(dblock s q)
    ∧ hasRead s q d p
      ) → (∃ br∈blocksRead s q d. b ≤ bal(block br))) (is ?P s)
  shows ?P s'
proof(auto)
  fix dd
  assume d: dd∈D
  with act asm4
  show b ≤ bal (disk s' dd p)
    by(auto simp add: Phase1or2ReadThen-def)
  fix dd qq
  assume d: dd∈D
  and phase': phase s' qq = Suc 0
  and dblk-mbal: b ≤ mbal(dblock s' qq)
  and hasRead: hasRead s' qq dd p
  show ∃ br∈blocksRead s' qq dd d. b ≤ bal(block br)
proof(cases d=dd ∧ qq=q ∧ pp=p)
  case True
  from d asm4
  have b ≤ bal(disk s dd p)
    by auto
  with act True
  show thesis
    by(auto simp add: Phase1or2ReadThen-def)
next
  case False
  with phase' act
  have p31: phase s qq = 1
and \( p32 \): \( \text{dblock } s' \text{ } qq = \text{dblock } s \text{ } qq \)

by\((\text{auto simp add: Phase1or2ReadThen-def})\)

with \( \text{dblk-mbal} \)

have \( b \leq \text{mbal}(\text{dblock } s \text{ } qq) \) by auto

moreover

from \( \text{act hasRead False} \)

have \( \text{hasRead } s \text{ } qq \text{ } dd \text{ } p \)

by\((\text{auto simp add: Phase1or2ReadThen-def})\)

ultimately

have \( \exists br \in \text{blocksRead } s \text{ } qq \text{ } dd \text{ } . \text{ } b \leq \text{bal}(\text{block } br) \)

using \( p31 \text{ } asm4 \text{ } d \)

by blast

with \( \text{act hasRead} \)

show \( \exists br \in \text{blocksRead } s' \text{ } qq \text{ } dd \text{ } . \text{ } b \leq \text{bal}(\text{block } br) \)

by\((\text{auto simp add: Phase1or2ReadThen-def hasRead-def})\)

qed

theorem HPhase1or2ReadThen-valueChosen:

assumes \( \text{act} \text{ } : \text{HPhase1or2ReadThen } s \text{ } s' \text{ } q \text{ } d \text{ } p \)

and \( \text{vc} \text{ } : \text{valueChosen } s \text{ } v \)

and \( \text{v-input} \text{ } : \text{v} \in \text{Inputs} \)

shows \( \text{valueChosen } s \text{ } v \)

proof

from \( \text{vc} \)

obtain \( b \text{ } p \text{ } D \text{ } where \)

\( \text{asm1} \text{ } : \text{ } b \in (\text{UN } p \text{. } \text{Ballot } p) \)

\( \text{asm2} \text{ } : \text{ } D \in \text{MajoritySet} \)

\( \text{asm3} \text{ } : \text{ } \text{maxBalInp } s \text{ } b \text{ } v \)

\( \text{asm4} \text{ } : \text{ } \forall \text{ } d \text{ } \in \text{ } D \text{ } . \text{ } \text{ } b \leq \text{bal}(\text{disk } s \text{ } d \text{ } p) \)

\( \land (\forall \text{ } q \text{ } . \text{ } \text{phase } s \text{ } q = 1 \)

\( \land b \leq \text{mbal}(\text{dblock } s \text{ } q) \)

\( \land \text{hasRead } s \text{ } q \text{ } d \text{ } p \)

\( ) \text{ } \rightarrow (\exists br \in \text{blocksRead } s \text{ } q \text{ } d \text{ } . \text{ } b \leq \text{bal}(\text{block } br)) \)

by\((\text{auto simp add: valueChosen-def})\)

from \( \text{HPhase1or2ReadThen-maxBalInp[OF } \text{act } \text{asm3]} \)

have \( \text{maxBalInp } s' \text{ } b \text{ } v \).

with \( \text{HPhase1or2ReadThen-valueChosen2[OF } \text{act } \text{asm4]} \text{ } \text{asm1 } \text{asm2} \)

show \?thesis

by\((\text{auto simp add: valueChosen-def})\)

qed

theorem HPhase1or2ReadElse-valueChosen:

\[ \text{HPhase1or2ReadElse } s \text{ } s' \text{ } p \text{ } d \text{ } r \text{ } ; \text{valueChosen } s \text{ } v \text{ } ; \text{v} \in \text{Inputs} \]

\( \Rightarrow \text{valueChosen } s' \text{ } v \)

using \( \text{HStartBallot-valueChosen} \)

by\((\text{auto simp add: Phase1or2ReadElse-def})\)

126
lemma HEndPhase2-maxBalInp:
assumes act: HEndPhase2 s s' q
  and asm3: maxBalInp s b v
shows maxBalInp s' b v
proof(auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s'
  and b-bal: b ≤ bal bk
  from subsetD[OF HEndPhase2-allBlocks[OF act] bk] asm3 b-bal
  show inp bk = v
    by(auto simp add: maxBalInp-def)
qed

lemma HEndPhase2-valueChosen2:
assumes act: HEndPhase2 s s' q
  and asm4: ∀ d ∈ D. b ≤ bal(disk s d p)
  ∧ (∀ q.( phase s q = 1
                ∧ b ≤ mbal(dblock s q)
                ∧ hasRead s q d p
          ) → (∃ br ∈ blocksRead s q d. b ≤ bal(block br))) (is ?P s)
shows ?P s'
proof(auto)
  fix d
  assume d: d ∈ D
  with act asm4
  show b ≤ bal(disk s' d p)
    by(auto simp add: EndPhase2-def)
  fix d q
  assume d: d ∈ D
  and phase': phase s' q = Suc 0
  and dblk-mbal: b ≤ mbal(dblock s' q)
  and hasRead: hasRead s' q d p
  from phase' act hasRead
  have p31: phase s q = 1
    and p32: dblock s' q = dblock s q
    by(auto simp add: EndPhase2-def InitializePhase-def
                           hasRead-def split : if-split-asm)
  with dblk-mbal
  have b ≤ mbal(dblock s q) by auto
  moreover
    from act hasRead
    have hasRead s q d p
      by(auto simp add: EndPhase2-def InitializePhase-def
                           hasRead-def split: if-split-asm)
  ultimately
    have ∃ br ∈ blocksRead s q d. b ≤ bal(block br)
      using p31 asm4 d
      by blast
    with act hasRead
\begin{center}
\begin{verbatim}
  show \( \exists br \in \text{blocksRead}\ s\ q\ d.\ b \leq \text{bal}(\text{block}\ br) \)
  by (auto simp add: EndPhase2-def InitializePhase-def
                   hasRead-def)

  qed

  theorem HEndPhase2-valueChosen:
    assumes act: HEndPhase2 s s' q
    and vc: valueChosen s v
    and v-input: v \in \text{Inputs}
    shows valueChosen s' v
  proof
    from vc
    obtain b p D where
      asm1: b \in (UN p. \text{Ballot}\ p)
    and asm2: D \in \text{MajoritySet}
    and asm3: maxBalInp s b v
    and asm4: \( \forall d \in D.\ b \leq \text{bal}(\text{disk}\ s\ d\ p) \)
      \wedge (\forall q. (\text{phase}\ s\ q = 1
                     \wedge b \leq \text{mbal}(\text{dblock}\ s\ q)
                     \wedge \text{hasRead}\ s\ q\ d\ p
                     ) \rightarrow (\exists br \in \text{blocksRead}\ s\ q\ d.\ b \leq \text{bal}(\text{block}\ br)))
    by (auto simp add: valueChosen-def)
    from HEndPhase2-maxBalInp[OF act asm3]
    have maxBalInp s' b v .
    with HEndPhase2-valueChosen2[OF act asm4] asm1 asm2
    show \?thesis
      by (auto simp add: valueChosen-def)
  qed

  lemma HFail-maxBalInp:
    assumes act: HFail s s' q
    and asm1: b \in (UN p. \text{Ballot}\ p)
    and asm3: maxBalInp s b v
    shows maxBalInp s' b v
  proof (auto simp add: maxBalInp-def)
    fix bk
    assume bk: bk \in \text{allBlocks}\ s'
    and b-bal: b \leq \text{bal}\ bk
    from subsetD[OF HFail-allBlocks[OF act] bk]
    show inp bk = v
  proof
    assume bk: bk \in \text{allBlocks}\ s
    with asm3 b-bal
    show \?thesis
      by (auto simp add: maxBalInp-def)
  next
    assume bk: bk \in \{\text{dblock}\ s' q\}
    with act
    have bal bk = 0
  qed
\end{verbatim}
\end{center}
by (auto simp add: Fail-def InitDB-def)
moreover
from Ballot-nzero asm1
have 0 < b
  by auto
ultimately
show ?thesis
  using b-bal
  by auto
qed
qed

lemma HFail-valueChosen2:
assumes act: HFail s s' q
  and asm4: \( \forall d \in D. \quad b \leq \text{bal}(\text{disk } s d p) \)
  \( \wedge (\forall q.(\quad \text{phase } s q = 1 \)
     \wedge b \leq \text{mbal}(\text{dblock } s q) \)
     \wedge \text{hasRead } s q d p \)
  \( \rightarrow (\exists \text{br } \in \text{blocksRead } s q d. \quad b \leq \text{bal}(\text{block br})) \) (is \( ?P s \))
shows ?P s'
proof (auto)
fix d
assume d: d \in D
with act asm4
show b \leq bal (disk s d p)
  by (auto simp add: Fail-def)
fix d q
assume d: d \in D
and phase': phase s' q = Suc 0
and dblk-mbal: b \leq mbal (dblock s' q)
and hasRead: \exists \text{hasRead } s' q d p
from phase' act hasRead
have p31: phase s q = 1
  and p32: dblock s' q = dblock s q
  by (auto simp add: Fail-def InitializePhase-def
          hasRead-def split : if-split-asm)
with dblk-mbal
have b \leq mbal (dblock s q) by auto
moreover
from act hasRead
have hasRead s q d p
  by (auto simp add: Fail-def InitializePhase-def
          hasRead-def split : if-split-asm)
ultimately
have \( \exists \text{br } \in \text{blocksRead } s q d. \quad b \leq \text{bal}(\text{block br}) \)
  using p31 asm4 d
  by blast
with act hasRead
show \( \exists \text{br } \in \text{blocksRead } s' q d. \quad b \leq \text{bal}(\text{block br}) \)
by (auto simp add: Fail-def InitializePhase-def hasRead-def)

qed

theorem HFail-valueChosen:
  assumes act: HFail s s' q
  and vc: valueChosen s v
  and v-input: v ∈ Inputs
  shows valueChosen s' v
proof –
  from vc
  obtain b p D where
    asm1: b ∈ (UN p. Ballot p)
    and asm2: D ∈ MajoritySet
    and asm3: maxBalInp s b v
    and asm4: ∀ d ∈ D. b ≤ bal (disk s d p)
      ∧ (∀ q. (phase s q = 1
            ∧ b ≤ mbal (dblock s q)
            ∧ hasRead s q d p)
      ) → (∃ br ∈ blocksRead s q d. b ≤ bal (block br)))
  by (auto simp add: valueChosen-def)
  from HFail-maxBalInp[OF act asm1 asm3]
  have maxBalInp s' b v.
  with HFail-valueChosen2[OF asm4] asm1 asm2
  show ?thesis
  by (auto simp add: valueChosen-def)
qed

lemma HPhase0Read-maxBalInp:
  assumes act: HPhase0Read s s' q d
  and asm3: maxBalInp s b v
  shows maxBalInp s' b v
proof (auto simp add: maxBalInp-def)
  fix bk
  assume bk: bk ∈ allBlocks s'
  and b-bal: b ≤ bal bk
  from subsetD[OF HPhase0Read-allBlocks[OF act] bk] asm3 b-bal
  show inp bk = v
  by (auto simp add: maxBalInp-def)
qed

lemma HPhase0Read-valueChosen2:
  assumes act: HPhase0Read s s' qq dd
  and asm4: ∀ d ∈ D. b ≤ bal (disk s d p)
    ∧ (∀ q. (phase s q = 1
            ∧ b ≤ mbal (dblock s q)
            ∧ hasRead s q d p)
      ) → (∃ br ∈ blocksRead s q d. b ≤ bal (block br))) (is ?P s)
  shows ?P s'
proof (auto)
fix \( d \)
assume \( d: d \in D \)
with \( \text{act asm4} \)
show \( b \leq \text{bal (disk } s’ d \ p) \)
by(\(\text{auto simp add: Phase0Read-def} \))

next
fix \( d q \)
assume \( d: d \in D \)
and \( \text{phase‘: phase } s’ q = \text{Suc } 0 \)
and \( \text{dblk-mbal: } b \leq \text{mbal } (\text{dblock } s’ q) \)
and \( \text{hasRead: hasRead } s’ q d p \)
from \( \text{phase‘ act} \)
have \( \text{qqnq: } q \neq q \)
by(\(\text{auto simp add: Phase0Read-def} \))
show \( \exists br \in \text{blocksRead } s’ q d. b \leq \text{bal } (\text{block } br) \)
proof –
from \( \text{phase‘ act hasRead} \)
have \( p31: \text{phase } s q = 1 \)
and \( p32: \text{dblock } s’ q = \text{dblock } s q \)
by(\(\text{auto simp add: Phase0Read-def hasRead-def} \))
with \( \text{dblk-mbal} \)
have \( b \leq \text{mbal } (\text{dblock } s q) \) by auto
moreover
from \( \text{act hasRead qqnq} \)
have \( \text{hasRead } s q d p \)
by(\(\text{auto simp add: Phase0Read-def hasRead-def} \)
\(\text{split: if-split-asn} \))
ultimately
have \( \exists br \in \text{blocksRead } s q d. b \leq \text{bal } (\text{block } br) \)
using \( p31 \text{ asm4 } d \)
by blast
with \( \text{act hasRead} \)
show \( \exists br \in \text{blocksRead } s’ q d. b \leq \text{bal } (\text{block } br) \)
by(\(\text{auto simp add: Phase0Read-def InitializePhase-def hasRead-def} \))
qed

ded

\(\text{theorem HPhase0Read-valueChosen:} \)
assumes \( \text{act: HPhase0Read } s s’ q d \)
and \( \text{vc: valueChosen } s v \)
and \( \text{v-input: } v \in \text{Inputs} \)
shows \( \text{valueChosen } s’ v \)
proof –
from \( \text{vc} \)
obtain \( b p D \) where
\( \text{asm1: } b \in (\text{UN } p. \text{Ballot } p) \)
and \( \text{asm2: } D \in \text{MajoritySet} \)
and \( \text{asm3: maxBalInp } s b v \)
and asm4: \( \forall d \in D. \ b \leq bal(disk s d p) \)
\( \land \forall q. (\ 
\land b \leq mbal(dblock s q) \\
\land hasRead s q d p \\
) \rightarrow (\exists br \in \text{blocksRead} s q d. \ b \leq bal(block br)) \)

by (auto simp add: valueChosen-def)
from HPhase0Read-maxBalInp[OF act asm3]
have maxBalInp s' b v.
with HPhase0Read-valueChosen2[OF act asm4] asm1 asm2
show \(?thesis \)
by (auto simp add: valueChosen-def)

qed

lemma HEndPhase0-maxBalInp:
assumes act: HEndPhase0 s s' q
and asm3: maxBalInp s b v
and inv1: Inv1 s
shows maxBalInp s' b v
proof (auto simp add: maxBalInp-def)
fix bk
assume bk: bk \in allBlocks s'
and b-bal: b \leq bal bk
from subsetD[OF HEndPhase0-allBlocks[OF act] bk]
show inp bk = v
proof
assume bk: bk \in allBlocks s
with asm3 b-bal
show \(?thesis \)
by (auto simp add: maxBalInp-def)
next
assume bk: bk \in \{dblock s' q\}
with HEndPhase0-some[OF act inv1] act
have \(\exists ba \in \text{allBlocksRead} s q. \ bal ba = bal(dblock s' q) \land inp ba = inp(dblock s' q) \)
by (auto simp add: EndPhase0-def)
then obtain ba
where ba-blksread: ba \in allBlocksRead s q
and ba-balinp: bal ba = bal(dblock s' q) \land inp ba = inp(dblock s' q)
by auto
have allBlocksRead s q \subseteq allBlocks s
by (auto simp add: allBlocksRead-def allRdBlks-def
allBlocks-def blocksOf-def rdBy-def)
from subsetD[OF this ba-blksread] ba-balinp bk b-bal asm3
show \(?thesis \)
by (auto simp add: maxBalInp-def)
qed
qed
lemma HEndPhase0-valueChosen2:
assumes act: HEndPhase0 s s' q
  and asm4: \( \forall d \in D. \ b \leq bal(disk\ s\ d\ p) \)
  \( \land (\forall q. (phase\ s\ q = 1 \land b \leq mbal(dblock\ s\ q)) \land hasRead\ s\ q\ d\ p) \)
  \( \rightarrow (\exists b' \in blocksRead\ s\ q\ d. b \leq bal(block\ b')) \) (is \( ?P\ s' \))
shows \( ?P\ s' \)
proof (auto)
fix d
assume d: d \in D
with act asm4
show \( b \leq bal(disk\ s'\ d\ p) \)
  by (auto simp add: EndPhase0-def)
fix d q
assume d: d \in D
  and phase': phase\ s'\ q = Suc\ 0
  and dblk-mbal: b \leq mbal(dblock\ s'\ q)
  and hasRead: hasRead\ s'\ q\ d\ p
from phase' act hasRead
have p\\$1\$: phase\ s\ q = 1
  and p\\$2\$: dblock\ s'\ q = dblock\ s\ q
  by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def split : if-split-asm)
with dblk-mbal
have \( b \leq mbal(dblock\ s\ q) \) by auto
moreover
from act hasRead
have hasRead\ s\ q\ d\ p
  by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def split: if-split-asm)
ultimately
have \( \exists b' \in blocksRead\ s\ q\ d. b \leq bal(block\ b') \)
  using p\\$1\$ asm4 d
  by blast
with act hasRead
show \( \exists b' \in blocksRead\ s'\ q\ d. b \leq bal(block\ b') \)
  by (auto simp add: EndPhase0-def InitializePhase-def hasRead-def)
qed

theorem HEndPhase0-valueChosen:
assumes act: HEndPhase0 s s' q
and vc: valueChosen s v
and v-input: v \in Inputs
and inv1: Inv1 s
shows valueChosen s' v
proof
  from vc

obtain \( b \) \( p \) \( D \) where

\begin{align*}
\text{asm1: } & b \in (\text{UN} p. \text{Ballot} \ p) \\
\text{and } & \text{asm2: } D \in \text{MajoritySet} \\
\text{and } & \text{asm3: } \text{maxBalInp} \ s \ b \ v \\
\text{and } & \text{asm4: } \forall \ d \in D. \ b \leq \text{bal}(\text{disk} \ s \ d \ p) \\
& \land (\forall q. (\text{phase} \ s \ q = 1 \\
& \land b \leq \text{mbal}(\text{dblock} \ s \ q) \\
& \land \text{hasRead} \ s \ q \ d \ p) \\
& \rightarrow (\exists b_\in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal}(\text{block} \ b)))
\end{align*}

by (auto simp add: valueChosen-def) from HEndPhase0-maxBalInp[OF act asm3 inv1]

have maxBalInp s' \ b \ v .

with HEndPhase0-valueChosen2[OF act asm4] asm1 asm2

show ?thesis by (auto simp add: valueChosen-def)

qed

end

theory DiskPaxos-Inv6 imports DiskPaxos-Chosen begin

C.7 Invariant 6

The final conjunct of \( HInv \) asserts that, once an output has been chosen, \( \text{valueChosen}(\text{chosen}) \) holds, and each processor’s output equals either \( \text{chosen} \) or \( \text{NotAnInput} \).

definition HInv6 :: state \Rightarrow bool
where
\[ HInv6 \ s = ((\text{chosen} \ s \neq \text{NotAnInput} \rightarrow \text{valueChosen} \ s \ (\text{chosen} \ s)) \land (\forall \ p. \ \text{outpt} \ s \ p \in \{\text{chosen} \ s, \text{NotAnInput}\})). \]

theorem HInit-HInv6: HInit s \implies HInv6 s

by (auto simp add: HInit-def Init-def InitDB-def HInv6-def)

lemma HEndPhase2-Inv6-1:
assumes act: \( H\text{EndPhase2} \ 2 \ s \ s' \ p \)
and inv: HInv6 s
and inv2b: Inv2b s
and inv2c: Inv2c s
and inv3: HInv3 s
and inv5: HInv5-inner s p
and chosen': chosen s' \neq \text{NotAnInput}

shows valueChosen s' (chosen s')

proof (cases chosen s = \text{NotAnInput})

from inv5 act

have inv5R: HInv5-inner-R s p
and phase: phase s p = 2
and ep2-maj: IsMajority \{d . \ d \in \text{disksWritten} \ s \ p\}

134
\(\forall q \in \text{UNIV} - \{p\}. \text{hasRead } s \ p \ d \ q\)}

by(auto simp add: EndPhase2-def HInv5-inner-def)

case True

have \(\text{p32}: \text{maxBalInp } s \ (\text{bal}(\text{dblock } s \ p)) \ (\text{inp}(\text{dblock } s \ p))\)

proof

have \(\neg(\exists D \in \text{MajoritySet.} \exists q. (\forall d \in D. \text{bal}(\text{dblock } s \ p) < \text{mbal}(\text{disk } s \ d \ q) \land \neg \text{hasRead } s \ p \ d \ q))\)

proof auto

fix \(D q\)

assume Dmaj: \(D \in \text{MajoritySet}\)

from ep2-maj Dmaj majorities-intersect

have \(\exists d \in D. d \in \text{disksWritten } s \ p\)

\(\land (\forall q \in \text{UNIV} - \{p\}. \text{hasRead } s \ p \ d \ q)\)

by(auto simp add: MajoritySet-def, blast)

then obtain \(d\)

where dinD: \(d \in D\)

and ddisk: \(d \in \text{disksWritten } s \ p\)

and dhasR: \(\forall q \in \text{UNIV} - \{p\}. \text{hasRead } s \ p \ d \ q\)

by auto

from inv2b

have \(\text{Inv2b-inner } s \ p \ d\)

by(auto simp add: Inv2b-def)

with ddisk

have \(\text{disk } s \ d \ p = \text{dblock } s \ p\)

by(auto simp add: Inv2b-inner-def)

with inv2c phase

have \(\text{bal}(\text{dblock } s \ p) = \text{mbal}(\text{disk } s \ d \ p)\)

by(auto simp add: Inv2c-def Inv2c-inner-def)

with dhasR dinD

show \(\exists d \in D. \text{bal}(\text{dblock } s \ p) < \text{mbal}(\text{disk } s \ d \ q) \rightarrow \text{hasRead } s \ p \ d \ q\)

by auto

qed

with inv5R

show \(?\text{thesis}\)

by(auto simp add: HInv5-inner-R-def)

qed

have \(\text{p33}: \text{maxBalInp } s' \ (\text{bal}(\text{dblock } s' \ p)) \ (\text{chosen } s')\)

proof

from act

have \(\text{outpt}'\): \(\text{outpt } s' = (\text{outpt } s) \ (p:= \text{inp}(\text{dblock } s \ p))\)

by(auto simp add: EndPhase2-def)

have \(\text{outpt}'-q' \ (\forall q. \ p\neq q \rightarrow \text{outpt } s' q = \text{NotAnInput}\)

proof auto

fix \(q\)

assume \(\text{pnq: } p\neq q\)

from \(\text{outpt}' \ \text{pnq}\)

have \(\text{outpt } s' q = \text{outpt } s q\)

by(auto simp add: EndPhase2-def)

with True inv2c
show \textit{outpt}\ ' \textit{s}\ ' = \textit{NotAnInput}
by (auto simp add: \textit{Inv2c-def} \textit{Inv2c-inner-def})

qed
from \textit{True act chosen'}
have \textit{chosen}\ ' \textit{s}\ ' = \textit{inp}\ (\textit{dblock}\ ' \textit{s}\ ' \textit{p})
proof (auto simp add: \textit{HNextPart-def} split: if-split-asm)

fix \textit{pa}
assume \textit{outpt}\ '-\textit{pa}: \textit{outpt}\ ' \textit{s}\ ' \textit{pa} \neq \textit{NotAnInput}
from \textit{outpt}\ '-\textit{q}
have \textit{someeq2}: \land \textit{pa}. \textit{outpt}\ ' \textit{s}\ ' \textit{pa} \neq \textit{NotAnInput} \implies \textit{pa}=\textit{p}

by auto
with \textit{outpt\ '-\textit{pa}}
have \textit{outpt}\ ' \textit{s}\ ' \textit{p} \neq \textit{NotAnInput}

by auto
from \textit{some-equality[of \lambda p. \textit{outpt}\ ' \textit{s}\ ' \textit{p} \neq \textit{NotAnInput}, OF this \textit{someeq2}]}
have \textit{(SOME p. \textit{outpt}\ ' \textit{s}\ ' \textit{p} \neq \textit{NotAnInput}) = \textit{p}}

with \textit{outpt}'
show \textit{outpt}\ ' (\textit{SOME p. \textit{outpt}\ ' \textit{s}\ ' \textit{p} \neq \textit{NotAnInput}}) = \textit{inp}\ (\textit{dblock}\ ' \textit{s}\ ' \textit{p})

by auto
qed
moreover
from \textit{act}
have \textit{bal}\ (\textit{dblock}\ ' \textit{s}\ ' \textit{p}) = \textit{bal}\ (\textit{dblock}\ ' \textit{s}\ ' \textit{p})
by (auto simp add: \textit{EndPhase2-def})
ultimately
have \textit{maxBalInp}\ ' \textit{s}\ (\textit{bal}\ (\textit{dblock}\ ' \textit{s}\ ' \textit{p})) (\textit{chosen}\ ' \textit{s})

using \textit{p32}

by auto
with \textit{HEndPhase2-allBlocks[OF \textit{act}]}
show ?thesis

by (auto simp add: \textit{maxBalInp-def})
qed
from \textit{ep2-maj inv2b majorities-intersect}
have \textit{\exists D\ \in MajoritySet. (\forall d\ \in D. disk\ d\ s\ d\ p = \textit{dblock}\ ' \textit{s}\ ' \textit{p}}

\land (\forall q \in \textit{UNIV} - \{p\}. \textit{hasRead}\ s\ q\ d\ p))

by (auto simp add: \textit{Inv2b-def} \textit{Inv2b-inner-def} \textit{MajoritySet-def})
then obtain \textit{D}
where \textit{Dmaj}: \textit{D\ \in MajoritySet}
and \textit{p34}: \forall d\ \in D. disk\ d\ s\ p = \textit{dblock}\ ' \textit{s}\ ' \textit{p}

\land (\forall q \in \textit{UNIV} - \{p\}. \textit{hasRead}\ s\ q\ d\ p)

by auto
have \textit{p35}: \forall q. \forall d\ \in D. (\textit{phase}\ s\ q=1 \land \textit{bal}(\textit{dblock}\ s\ p) \leq \textit{mbal}(\textit{dblock}\ s\ q) \land \textit{hasRead}\ s\ q\ d\ p)

\longrightarrow (\exists block=\textit{dblock}\ s\ p, proc=p) \in \textit{blocksRead}\ s\ q\ d\ p)

proof auto
fix \textit{q}\ \textit{d}
assume \textit{dD}: \textit{d}\ \in \textit{D} and \textit{phase-q}: \textit{phase}\ s\ q= Suc 0
and \textit{bal-mbal}: \textit{bal}(\textit{dblock}\ s\ p) \leq \textit{mbal}(\textit{dblock}\ s\ q) and \textit{hasRead}: \textit{hasRead}\ s\ q\ d\ p
from \textit{phase inv2c}
have \( \text{bal}(\text{dblock } s \ p) = \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: \text{Inv2c-def} \text{Inv2c-inner-def})

moreover
from \text{inv2c phase}
have \( \forall \text{br} \in \text{blocksRead } s \ p \ d. \ \text{mbal}(\text{block } \text{br}) < \text{mbal}(\text{dblock } s \ p) \)
by (auto simp add: \text{Inv2c-def} \text{Inv2c-inner-def})

ultimately
have \text{p41}: \{ \text{block}=\text{dblock } s \ q, \ \text{proc}=\text{q}\} \in \text{blocksRead } s \ p \ d
using \text{bal-mbal}
by auto

from \text{phase phase-q}
have \( \exists q \) by auto
with \text{p34} \text{dD}
have \( \text{hasRead } s \ p \ d \ q \)
by auto
with \text{phase phase-q} \text{hasRead inv3 p41}
show
proof (auto simp add: \text{HInv3-def} \text{HInv3-inner-def} \text{HInv3-L-def} \text{HInv3-R-def})
qed

have \( \forall q. \forall d \in D. \ \text{phase } s' \ q = 1 \land \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s' \ q) \land \text{hasRead } s' \ q \ d \ p \)
\( \longrightarrow (\exists \text{br} \in \text{blocksRead } s' \ q \ d. \ \text{bal}(\text{block } \text{br}) = \text{bal}(\text{dblock } s \ p)) \)

proof (auto)
fix \ q \ d
assume \text{dD}: \( d \in D \) and \text{phase-q}: \text{phase } s' \ q = \text{Suc 0}
and \( \text{bal}: \text{bal } (\text{dblock } s \ p) \leq \text{mbal } (\text{dblock } s' \ q) \)
and \( \text{hasRead}: \text{hasRead } s' \ q \ d \ p \)
from \text{phase-q act}
have \( \text{phase } s' \ q = \text{phase } s \ q \land \text{dblock } s' \ q = \text{dblock } s \ q \land \text{hasRead } s' \ q \ d \ p = \text{hasRead } s \ q \ d \)
by (auto simp add: \text{EndPhase2-def} \text{hasRead-def} \text{InitializePhase-def})

with \text{p35} \text{phase-q} \text{bal hasRead dD}
have \( \{ \text{block}=\text{dblock } s \ p, \ \text{proc}=\text{p}\} \in \text{blocksRead } s' \ q \ d \)
by auto
thus \( \exists \text{br} \in \text{blocksRead } s' \ q \ d. \ \text{bal}(\text{block } \text{br}) = \text{bal}(\text{dblock } s \ p) \)
by force
qed

hence \text{p36-2}: \( \forall q. \forall d \in D. \ \text{phase } s' \ q = 1 \land \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s' \ q) \land \text{hasRead } s' \ q \ d \ p \)
\( \longrightarrow (\exists \text{br} \in \text{blocksRead } s' \ q \ d. \ \text{bal}(\text{block } \text{br}) \leq \text{bal}(\text{dblock } s \ p)) \)
by force
from \text{act}
have \( \text{bal-dblock}: \text{bal}(\text{dblock } s' \ p) = \text{bal}(\text{dblock } s \ p) \)
and \( \text{disk}: \text{disk } s' = \text{disk } s \)
by (auto simp add: \text{EndPhase2-def})
from \text{bal-dblock p33}
have \( \text{maxBalInp } s' (\text{bal}(\text{dblock } s \ p)) (\text{chosen } s') \)
by auto
moreover
from disk p34
have \( \forall d \in D. \text{bal}(\text{dblock } s \ p) \leq \text{bal}(\text{disk } s' \ d \ p) \)
  by auto
ultimately
have \( \text{maxBalInp } s' (\text{bal}(\text{dblock } s \ p)) (\text{chosen } s') \land \)
  \( (\exists D \in \text{MajoritySet}. \forall d \in D. \text{bal}(\text{dblock } s \ p) \leq \text{bal}(\text{disk } s' \ d \ p) \land \)
  \( (\forall q. \text{phase } s'^' q = \text{Suc } 0 \land \)
  \( \text{bal}(\text{dblock } s \ p) \leq \text{mbal}(\text{dblock } s' \ q) \land \text{hasRead } s' q d p \rightarrow \)
  \( (\exists br \in \text{blocksRead } s' q d. \text{bal}(\text{dblock } s \ p) \leq \text{bal}(\text{block } br))) \)
  using p36-2 Dmaj
  by auto
moreover
from phase inv2c
have \( \text{bal}(\text{dblock } s \ p) \in \text{Ballot } p \)
  by (auto simp add: Inv2c-def Inv2c-inner-def)
ultimately
show \(?thesis \)
  by (auto simp add: valueChosen-def)
next
case False
with act
have p31: chosen s' = chosen s
  by (auto simp add: HNextPart-def)
from False inv
have valueChosen s (chosen s)
  by (auto simp add: HInv6-def)
from HEndPhase2-valueChosen[OF act this] p31 False InputsOrNi
show \(?thesis \)
  by auto
qed

lemma valueChosen-equal-case:
assumes max-v: \( \text{maxBalInp } s \ b \ v \)
and Dmaj: \( D \in \text{MajoritySet} \)
and asm-v: \( \forall d \in D. \ b \leq \text{bal}(\text{disk } s \ d \ p) \)
and max-w: \( \text{maxBalInp } s \ ba \ w \)
and Damaj: \( Da \in \text{MajoritySet} \)
and asm-w: \( \forall d \in Da. \ ba \leq \text{bal}(\text{disk } s \ d \ pa) \)
and b-ba: \( b \leq ba \)
shows \( v = w \)
proof –
  have \( \forall d. \text{disk } s \ d \ pa \in \text{allBlocks } s \)
    by (auto simp add: allBlocks-def blocksOf-def)
with majorities-intersect Dmaj Damaj
have \( \exists d \in D \cap Da. \text{disk } s \ d \ pa \in \text{allBlocks } s \)
  by (auto simp add: MajoritySet-def, blast)
then obtain \( d \)
where \( \text{dinmaj} \): \( d \in D \cap Da \) and \( \text{dab} \): disk \( s \ d \ pa \in \text{allBlocks} \ s \) by auto with \( \text{asm-w} \)

have \( \text{ba} \): \( \text{ba} \leq \text{bal} (\text{disk} \ s \ d \ pa) \) by auto with \( b-\text{ba} \)

have \( b \leq \text{bal} (\text{disk} \ s \ d \ pa) \) by auto with \( \text{max-v} \ \text{dab} \)

have \( \text{v-value} \): \( \text{inp} (\text{disk} \ s \ d \ pa) = v \) by (auto simp add: \text{maxBalInp-def})

from \( \text{ba} \ \text{max-w} \ \text{dab} \)

have \( \text{w-value} \): \( \text{inp} (\text{disk} \ s \ d \ pa) = w \) by (auto simp add: \text{maxBalInp-def})

with \( \text{v-value} \)

show \(?\text{thesis}\) by auto

qed

lemma \text{valueChosen-equal}: assumes \( v \): \text{valueChosen} \ s \ v \ and \( w \): \text{valueChosen} \ s \ w \ shows \( v=w \) using \text{assms}

proof (auto simp add: \text{valueChosen-def})

fix \( a \ b \ aa \ ba \ p \ D \ pa \ Da \)

assume \( \text{max-v}: \text{maxBalInp} \ s \ b \ v \)

and \( \text{Dmaj}: D \in \text{MajoritySet} \)

and \( \text{asm-v}: \forall d \in D. \ b \leq \text{bal} (\text{disk} \ s \ d \ p) \land \
(\forall q. \text{phase} \ s \ q = \text{Suc} \ 0 \land \\ b \leq \text{mbal} (\text{dblock} \ s \ q) \land \text{hasRead} \ s \ q \ d \ p \longrightarrow \ (\exists \text{br} \in \text{blocksRead} \ s \ q \ d. \ b \leq \text{bal} (\text{block} \ \text{br}))) \)

and \( \text{max-w}: \text{maxBalInp} \ s \ ba \ w \)

and \( \text{Damaj}: Da \in \text{MajoritySet} \)

and \( \text{asm-w}: \forall d \in Da. \ ba \leq \text{bal} (\text{disk} \ s \ d \ pa) \land \n(\forall q. \text{phase} \ s \ q = \text{Suc} \ 0 \land \\ ba \leq \text{mbal} (\text{dblock} \ s \ q) \land \text{hasRead} \ s \ q \ d \ pa \longrightarrow \ (\exists \text{br} \in \text{blocksRead} \ s \ q \ d. \ ba \leq \text{bal} (\text{block} \ \text{br}))) \)

from \( \text{asm-v} \)

have \( \text{asm-v}: \forall d \in D. \ b \leq \text{bal} (\text{disk} \ s \ d \ p) \) by auto

from \( \text{asm-w} \)

have \( \text{asm-w}: \forall d \in Da. \ ba \leq \text{bal} (\text{disk} \ s \ d \ pa) \) by auto

show \( v=w \)

proof(cases \( b\leq ba \))

  case True
  from \text{valueChosen-equal-case}[OF \text{max-v} \ \text{Dmaj} \ \text{asm-v} \ \text{max-w} \ \text{Damaj} \ \text{asm-w} \ True]
  show \(?\text{thesis}\).

next

  case False
  from \text{valueChosen-equal-case}[OF \text{max-w} \ \text{Damaj} \ \text{asm-w} \ \text{max-v} \ \text{Dmaj} \ \text{asm-v}]
  False
show ?thesis
by auto
qed

lemma HEndPhase2-Inv6-2:
  assumes act: HEndPhase2 s s' p
  and inv: HInv6 s
  and inv2b: Inv2b s
  and inv2c: Inv2c s
  and inv3: HInv3 s
  and inv5: HInv5-inner s p
  and asm: outpt s' r ≠ NotAnInput
  shows outpt s' r = chosen s'
proof (cases chosen s = NotAnInput)
  case True
  with inv2c
  have ∀ q. outpt s q = NotAnInput
    by (auto simp add: Inv2c-def Inv2c-inner-def)
  with True act asm
  show ?thesis
    by (auto simp add: EndPhase2-def HNextPart-def
      split: if-split-asm)
  next
  case False
  with inv
  have p31: valueChosen s (chosen s)
    by (auto simp add: HInv6-def)
  with False act
  have chosen s' ≠ NotAnInput
    by (auto simp add: HNextPart-def)
  from HEndPhase2-Inv6-1 [OF act inv inv2b inv2c inv3 inv5 this]
  have p32: valueChosen s' (chosen s')
    from False InputsOrNi
  have chosen s ∈ Inputs by auto
  from valueChosen-equal [OF HEndPhase2-valueChosen [OF act p31 this] p32]
  have p33: chosen s = chosen s'
    from act
  have maj: IsMajority {d . d ∈ disksWritten s p
    ∧ (∀ q ∈ UNIV − {p}. hasRead s p d q)} (is IsMajority ?D)
    and phase: phase s p = 2
    by (auto simp add: EndPhase2-def)
  show ?thesis
  proof (cases outpt s r = NotAnInput)
    case True
    with asm act
    have p41: r = p
      by (auto simp add: EndPhase2-def split: if-split-asm)
    from maj
have \( p42 \): \( \exists D \in \text{MajoritySet}. \forall d \in D. \forall q \in \text{UNIV} - \{p\}. \text{hasRead} s p d q \\
\text{by} (\text{auto simp add: MajoritySet-def})

have \( p43 \): \( \neg (\exists D \in \text{MajoritySet}. \exists q. (\forall d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q)) \land \neg \text{hasRead} s p d q) \)
\begin{proof}
\text{fix} D q \\
\text{assume} \text{Dmaj}: D \in \text{MajoritySet} \\
\text{show} \exists d \in D. \text{bal}(\text{dblock} s p) < \text{mbal}(\text{disk} s d q) \longrightarrow \text{hasRead} s p d q \\
\text{proof} (\text{cases} \ p=\bar{q}) \\
\text{assume} \ pq: \ p=\bar{q} \\
\text{thus} \ ?\text{thesis} \\
\text{proof} \ (\text{auto}) \\
\text{from} \ \text{maj majorities-intersect Dmaj} \\
\text{have} \ ?D\cap\bar{D}\neq\{\} \\
\text{by} (\text{auto simp add: MajoritySet-def}) \\
\text{hence} \exists d \in ?D\cap\bar{D}. \ d \in \text{disksWritten} s p \text{ by auto} \\
\text{then obtain} \ d \ \text{where} \ d: \ d \in \text{disksWritten} s p \ \text{and} \ d\in?D\cap\bar{D} \\
\text{by auto} \\
\text{hence} \ dD: \ d\in\bar{D} \text{ by auto} \\
\text{from} \ d \ \text{inv2b} \\
\text{have} \ \text{disk} s d p = \text{dblock} s p \\
\text{by} (\text{auto simp add: Inv2b-def Inv2b-inner-def}) \\
\text{with} \ \text{inv2c phase} \\
\text{have} \ \text{bal}(\text{dblock} s p) = \text{mbal}(\text{disk} s d p) \\
\text{by} (\text{auto simp add: Inv2c-def Inv2c-inner-def}) \\
\text{with} \ d\bar{D} \ pq \\
\text{show} \ \exists d \in D. \ \text{bal}(\text{dblock} s q) < \text{mbal}(\text{disk} s d q) \longrightarrow \text{hasRead} s q d q \\
\text{by} (\text{auto}) \\
\text{qed}
\end{proof}
\begin{next}
\text{next} \\
\text{case} \ False \\
\text{with} \ \text{p42} \\
\text{have} \ \exists D \in \text{MajoritySet}. \ \forall d \in D. \text{hasRead} s p d q \\
\text{by} (\text{auto}) \\
\text{with} \ \text{majorities-intersect Dmaj} \\
\text{show} \ ?\text{thesis} \\
\text{by} (\text{auto simp add: MajoritySet-def, blast}) \\
\text{qed}
\end{next}
\begin{qed}
\text{qed}
\text{with} \ \text{inv5 act} \\
\text{have} \ \text{p44}: \ \text{maxBalInp} s (\text{bal}(\text{dblock} s p)) (\text{inp}(\text{dblock} s p)) \\
\text{by} (\text{auto simp add: EndPhase2-def} \text{Hinv5-inner-def} \text{Hinv5-inner-R-def}) \\
\text{have} \ \exists bk \in \text{allBlocks} s. \ \exists b \in (\text{UN} p. \ \text{Ballot} p). (\text{maxBalInp} s b (\text{chosen} s)) \land b \leq \text{bal} \ bk \\
\text{proof} - \\
\text{have} \ \text{disk-allblks}: \ \forall d \ p. \ \text{disk} s d p \in \text{allBlocks} s \\
\text{by} (\text{auto simp add: allBlocks-def blocksOf-def})
from p31
have \( \exists b \in (\bigcup p. \text{Ballot } p). \max\text{BalInp } s \ b \ (\text{chosen } s) \land \)
\( (\exists p, \exists D \in \text{MajoritySet}. (\forall d \in D. \ b \leq \text{bal}(\text{disk } s \ d \ p))) \)
  by(auto simp add: valueChosen-def, force)
with majority-nonempty obtain b p D d
  where IsMajority D \land b \in (\bigcup p. \text{Ballot } p) \land
  \max\text{BalInp } s \ b \ (\text{chosen } s) \land d \in D \land b \leq \text{bal}(\text{disk } s \ d \ p)
  by(auto simp add: MajoritySet-def, blast)
with disk-allblks
show \(?thesis \)
  by(auto)
qed
then obtain bk b
  where p45-bk: bk \in \text{allBlocks } s \land b \leq \text{bal } bk
  and p45-b: bc(\bigcup p. \text{Ballot } p) \land (\max\text{BalInp } s \ b \ (\text{chosen } s))
  by auto
have p46: \text{inp}(\text{dblock } s \ p) = \text{chosen } s
proof(cases b \leq \text{bal}(\text{dblock } s \ p))
case True
have \text{dblock } s \ p \in \text{allBlocks } s
  by(auto simp add: allBlocks-def blocksOf-def)
with p45-b True
show \(?thesis \)
  by(auto simp add: maxBalInp-def)
next
case False
from p44 p45-bk False
have \text{inp } bk = \text{inp}(\text{dblock } s \ p)
  by(auto simp add: maxBalInp-def)
with p45-b p45-bk
show \(?thesis \)
  by(auto simp add: maxBalInp-def)
qed
with p41 p33 act
show \(?thesis \)
  by(auto simp add: EndPhase2-def)
next
case False
from inv2c
have Inv2c-inner s r
  by(auto simp add: Inv2c-def)
with False asm inv2c act
have \text{outpt } s^' r = \text{outpt } s \ r
  by(auto simp add: Inv2c-inner-def EndPhase2-def
split: if-split-asm)
with inv p33 False
show \(?thesis \)
  by(auto simp add: HInv6-def)
qed
qed

theorem HEndPhase2-Inv6:
assumes act: HEndPhase2 s s' p
and inv: HHw6 s
and inv2b: Inv2b s
and inv2c: Inv2c s
and inv3: HHinv3 s
and inv5: HHinv5-inner s p
shows HHw6 s'
proof (auto simp add: HHw6-def)
assume chosen s' ≠ NotAnInput
from HEndPhase2-Inv6-1[OF act inv inv2b inv2c inv3 inv5 this]
show valueChosen s' (chosen s') .
next
fix p
assume outpt s' p≠ NotAnInput
from HEndPhase2-Inv6-2[OF act inv inv2b inv2c inv3 inv5 this]
show outpt s' p = chosen s' .
qed

lemma outpt-chosen:
assumes outpt: outpt s = outpt s'
and inv2c: Inv2c s
and nextp: HNextPart s s'
shows chosen s' = chosen s
proof –
from inv2c
have chosen s = NotAnInput → (∀ p. outpt s p = NotAnInput)
  by (auto simp add: Inv2c-inner-def Inv2c-def)
with outpt nextp
show ?thesis
by (auto simp add: HNextPart-def)
qed

lemma outpt-Inv6:
[ outpt s = outpt s'; ∀ p. outpt s p ∈ {chosen s, NotAnInput};
  Inv2c s; HNextPart s s' ] → ∀ p. outpt s' p ∈ {chosen s', NotAnInput}
using outpt-chosen.
by auto

theorem HStartBallot-Inv6:
assumes act: HStartBallot s s' p
and inv: HHw6 s
and inv2c: Inv2c s
shows HHw6 s'
proof –
from outpt-chosen act inv2c inv
have chosen s' ≠ NotAnInput → valueChosen s (chosen s')
by (auto simp add: StartBallot-def Hinv6-def)
from HStartBallot-valueChosen [OF act] this InputsOrNi
have t1: chosen s' ≠ NotAnInput ⟹ valueChosen s' (chosen s')
  by auto
from act
have outpt: outpt s = outpt s'
  by (auto simp add: StartBallot-def)
from outpt-Inv6 [OF outpt] act inv2c inv
have ∀p. outpt s' p = chosen s' ∨ outpt s' p = NotAnInput
  by (auto simp add: Hinv6-def)
with t1
show ?thesis
  by (simp add: Hinv6-def)
qed

theorem HPhase1or2Write-Inv6:
  assumes act: HPhase1or2Write s s' p d
  and inv: Hinv6 s
  and inv4: Hinv4a s p
  and inv2c: Inv2c s
  shows Hinv6 s'
proof –
  from outpt-chosen act inv2c inv
  have chosen s' ≠ NotAnInput ⟹ valueChosen s (chosen s')
    by (auto simp add: Phase1or2Write-def Hinv6-def)
  from HPhase1or2Write-valueChosen [OF act] inv4 this InputsOrNi
  have t1: chosen s' ≠ NotAnInput ⟹ valueChosen s' (chosen s')
    by auto
  from act
  have outpt: outpt s = outpt s'
    by (auto simp add: Phase1or2Write-def)
  from outpt-Inv6 [OF outpt] act inv2c inv
  have ∀p. outpt s' p = chosen s' ∨ outpt s' p = NotAnInput
    by (auto simp add: Hinv6-def)
  with t1
  show ?thesis
    by (simp add: Hinv6-def)
qed

theorem HPhase1or2ReadThen-Inv6:
  assumes act: HPhase1or2ReadThen s s' p d q
  and inv: Hinv6 s
  and inv2c: Inv2c s
  shows Hinv6 s'
proof –
  from outpt-chosen act inv2c inv
  have chosen s' ≠ NotAnInput ⟹ valueChosen s (chosen s')
    by (auto simp add: Phase1or2ReadThen-def Hinv6-def)
  from HPhase1or2ReadThen-valueChosen [OF act] this InputsOrNi
have \( t_1: \text{chosen } s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s' (\text{chosen } s') \)
  by auto
from act
have \( \text{outpt}: \text{outpt } s = \text{outpt } s' \)
  by(auto simp add: Phase1or2ReadThen-def)
from outpt-Inv6[OF outpt] act inv2c inv
have \( \forall p. \text{outpt } s' p = \text{chosen } s' \lor \text{outpt } s' p = \text{NotAnInput} \)
  by(auto simp add: HInv6-def)
with \( t_1 \)
show \( ?\text{thesis} \)
  by(simp add: HInv6-def)
qed

theorem HPhase1or2ReadElse-Inv6:
  assumes \( \text{act}: \text{HPhase1or2ReadElse } s \ s' \ p \ d \ q \)
  and \( \text{inv}: \text{HInv6 } s \)
  and \( \text{inv2c}: \text{Inv2c } s \)
  shows \( \text{HInv6 } s' \)
  using assms and HStartBallot-Inv6
by(auto simp add: Phase1or2ReadElse-def)

theorem HEndPhase1-Inv6:
  assumes \( \text{act}: \text{HEndPhase1 } s \ s' \ p \)
  and \( \text{inv}: \text{HInv6 } s \)
  and \( \text{inv1}: \text{Inv1 } s \)
  and \( \text{inv2a}: \text{Inv2a } s \)
  and \( \text{inv2b}: \text{Inv2b } s \)
  and \( \text{inv2c}: \text{Inv2c } s \)
  shows \( \text{HInv6 } s' \)
proof –
  from outpt-chosen act inv2c inv
  have \( \text{chosen } s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s (\text{chosen } s') \)
    by(auto simp add: EndPhase1-def HInv6-def)
from HEndPhase1-1-valueChosen[OF act] inv1 inv2a inv2b this InputsOrNi
  have \( t_1: \text{chosen } s' \neq \text{NotAnInput} \rightarrow \text{valueChosen } s' (\text{chosen } s') \)
    by auto
from act
  have \( \text{outpt}: \text{outpt } s = \text{outpt } s' \)
    by(auto simp add: EndPhase1-def)
from outpt-Inv6[OF outpt] act inv2c inv
  have \( \forall p. \text{outpt } s' p = \text{chosen } s' \lor \text{outpt } s' p = \text{NotAnInput} \)
    by(auto simp add: HInv6-def)
  with \( t_1 \)
  show \( ?\text{thesis} \)
    by(simp add: HInv6-def)
qed

lemma outpt-chosen-2:
  assumes \( \text{outpt}: \text{outpt } s' = (\text{outpt } s) \ (p:= \text{NotAnInput}) \)
and $inv2c$: Inv2c $s$
and $nextp$: HNextPart $s$ $s'$
shows chosen $s = chosen s'$
proof –
from $inv2c$
have chosen $s = NotAnInput \rightarrow (\forall p. outpt s p = NotAnInput)$
  by (auto simp add: Inv2c-inner-def Inv2c-def)
with $outpt$ $nextp$
show ?thesis
  by (auto simp add: HNextPart-def)
qed

lemma $outpt$-$HInv6$-2:
assumes $outpt$: outpt $s = (outpt s) (p:=NotAnInput)$
and $inv$: $\forall p. outpt s p \in \{chosen s, NotAnInput\}$
and $inv2c$: Inv2c $s$
and $nextp$: HNextPart $s$ $s'$
shows $\forall p. outpt s' p \in \{chosen s', NotAnInput\}$
proof –
from outpt-chosen-2[OF outpt $inv2c$ $nextp$]
have chosen $s = chosen s'$.
with $inv$ $outpt$
show ?thesis
  by auto
qed

theorem $HFail$-$Inv6$:
assumes $act$: HFail $s$ $s'$ $p$
and $inv$: $Hinv6$ $s$
and $inv2c$: Inv2c $s$
shows $HInv6$ $s'$
proof –
from outpt-chosen-2 $act$ $inv2c$ $inv$
have chosen $s' \neq NotAnInput \rightarrow valueChosen s (chosen s')$
  by (auto simp add: Fail-def HInv6-def)
from $HFail$-$valueChosen$[OF $act$] this InputsOrNi
have $t1$: chosen $s' \neq NotAnInput \rightarrow valueChosen s' (chosen s')$
  by auto
from $act$
have outpt: outpt $s' = (outpt s) (p:=NotAnInput)$
  by (auto simp add: Fail-def)
from outpt-$HInv6$-2[OF outpt] $act$ $inv2c$ $inv$
have $\forall p. outpt s' p = chosen s' \lor outpt s' p = NotAnInput$
  by (auto simp add: HInv6-def)
with $t1$
show ?thesis
  by (simp add: HInv6-def)
qed
theorem HPhase0Read-Inv6:
assumes act: HPhase0Read s s’ p d and inv: HInv6 s and inv2c: Inv2c s shows HInv6 s’
proof
  from outpt-chosen act inv2c inv have chosen s’ ≠ NotAnInput → valueChosen s (chosen s’)
    by (auto simp add: Phase0Read-def HInv6-def)
  from HPhase0Read-valueChosen[OF act] this InputsOrNi have t1: chosen s’ ≠ NotAnInput → valueChosen s’ (chosen s’)
    by auto
  from act have outpt: outpt s = outpt s’
    by (auto simp add: Phase0Read-def)
  from outpt-Inv6[OF outpt] act inv2c inv have ∀ p. outpt s’ p = chosen s’ ∨ outpt s’ p = NotAnInput
    by (auto simp add: HInv6-def)
  with t1 show ?thesis
    by (simp add: HInv6-def)
qed

HInv1 ∧ HInv2 ∧ HInv2’ ∧ HInv3 ∧ HInv4 ∧ HInv5 ∧ HInv6 is an invariant of HNext.

147
lemma I2f:
  assumes nxt: HNext s s'
  and inv: HInv1 s \land HInv2 s \land HInv2 s' \land HInv3 s \land HInv4 s \land HInv5 s \land HInv6 s
  shows HInv6 s' using assms
  by(auto simp add: HNext-def Next-def,
    auto simp add: HInv2-def intro: HStartBallot-Inv6,
    auto intro: HPhase0Read-Inv6,
    auto simp add: HInv4-def intro: HPhase1or2Write-Inv6,
    auto simp add: HPhase1or2Read-def
      intro: HPhase1or2ReadThen-Inv6
            HPhase1or2ReadElse-Inv6,
    auto simp add: EndPhase1or2-Inv6 HInv1-def HInv5-def
      intro: HEndPhase1-Inv6
            HEndPhase2-Inv6,
    auto intro: HFail-Inv6,
    auto intro: HEndPhase0-Inv6)
end

theory DiskPaxos-Invariant imports DiskPaxos-Inv6 begin

C.8 The Complete Invariant

definition HInv :: state \Rightarrow bool
where
  HInv s = (HInv1 s
    \land HInv2 s
    \land HInv3 s
    \land HInv4 s
    \land HInv5 s
    \land HInv6 s)

theorem I1:
  HInit s \Rightarrow HInv s
  using HInit-HInv1 HInit-HInv2 HInit-HInv3
    HInit-HInv4 HInit-HInv5 HInit-HInv6
  by(auto simp add: HInv-def)

theorem I2:
  assumes inv: HInv s
  and nxt: HNext s s'
  shows HInv s'
  using inv I2a[OF nxt] I2b[OF nxt] I2c[OF nxt]
    I2d[OF nxt] I2e[OF nxt] I2f[OF nxt]
  by(simp add: HInv-def)

148
theory DiskPaxos imports DiskPaxos-Invariant begin

C.9 Inner Module

record
Istate =
iinput :: Proc ⇒ InputsOrNi
ioutput :: Proc ⇒ InputsOrNi
ichosen :: InputsOrNi
iallInput :: InputsOrNi set

definition IInit :: Istate ⇒ bool
where
IInit s = (range (iinput s) ⊆ Inputs
∧ ioutput s = (λp. NotAnInput)
∧ ichosen s = NotAnInput
∧ iallInput s = range (iinput s))

definition IChoose :: Istate ⇒ Istate ⇒ Proc ⇒ bool
where
IChoose s s′ p = (ioutput s p = NotAnInput
∧ (if (ichosen s = NotAnInput)
then (∃ip ∈ iallInput s. ichosen s′ = ip
∧ ioutput s′ = (ioutput s) (p := ip))
else ( ioutput s′ = (ioutput s) (p:= ichosen s)
∧ ichosen s′ = ichosen s))
∧ iinput s′ = iinput s ∧ iallInput s′ = iallInput s)

definition IFail :: Istate ⇒ Istate ⇒ Proc ⇒ bool
where
IFail s s′ p = (ioutput s′ = (ioutput s) (p:= NotAnInput)
∧ (∃ip ∈ Inputs. iinput s′ = (iinput s)(p:= ip)
∧ iallInput s′ = iallInput s ∪ {ip})
∧ ichosen s′ = ichosen s)

definition INext :: Istate ⇒ Istate ⇒ bool
where
INext s s′ = (∃p. IChoose s s′ p ∨ IFail s s′ p)

definition s2is :: state ⇒ Istate
where
s2is s = {iinput = inpt s,
ioutput = outpt s,
ichosen=chosen s,
iallInput = allInput s}

theorem R1:
[HInit s; is = s2is s] \Rightarrow \text{IInit is}

by (auto simp add: HInit-def IInit-def s2is-def Init-def)

theorem R2b:
  assumes inv: HInv s
  and inv': HInv s'
  and nxt: HNext s s'
  and srel: is = s2is s \land is' = s2is s'
  shows (\exists p. IFail is is' p \lor IChoose is is' p) \lor is = is'

proof (auto)
  assume chg-vars: is \neq is'
  with srel
  have s-change: inpt s \neq inpt s' \lor outpt s \neq outpt s'
  \lor chosen s \neq chosen s' \lor allInput s \neq allInput s'
    by (auto simp add: s2is-def)
  from inv
  have inv2c5: \forall p. inpt s \in allInput s
    \land (chosen s = NotAnInput \longrightarrow outpt s = NotAnInput)
    by (auto simp add: HInv-def HInv2-def Inv2c-def Inv2c-inner-def)
  from nxt s-change inv2c5
  have inpt s' \neq inpt s \lor outpt s' \neq outpt s
    by (auto simp add: HNext-def Next-def HNextPart-def)
  with nxt
  have \exists p. Fail s s' p \lor EndPhase2 s s' p
    by (auto simp add: HNext-def Next-def
      StartBallot-def Phase0Read-def Phase1or2Write-def
      Phase1or2Read-def Phase1or2ReadThen-def Phase1or2ReadElse-def
      EndPhase1or2-def EndPhase1or2-def EndPhase1or2-def)
  then obtain p where fail-or-endphase2: Fail s s' p \lor EndPhase2 s s' p
    by auto
  from inv
  have inv2c: Inv2c-inner s p
    by (auto simp add: HInv-def HInv2-def Inv2c-def)
  from fail-or-endphase2 have IFail is is' p \lor IChoose is is' p
    proof
      assume fail: Fail s s' p
      hence phase': phase s' p = 0
        and outpt: outpt s' = (outpt s) (p:= NotAnInput)
        by (auto simp add: Fail-def)
      have IFail is is' p
        proof
          from fail srel
          have ioutput s' = (ioutput is) (p:= NotAnInput)
            by (auto simp add: Fail-def s2is-def)
          moreover
          from nxt
          have all-nxt: allInput s' = allInput s \cup (range (inpt s'))
            by (auto simp add: HNext-def HNextPart-def)
          from fail srel

150
\[
\begin{align*}
\text{have } & \exists ip \in \text{Inputs}. \ iinput is' = (iinput is)(p:= ip) \\
& \text{by (auto simp add: Fail-def s2is-def)} \\
\text{then obtain } & ip \text{ where } ip-\text{Input: } ip \in \text{Inputs} \text{ and } iinput is' = (iinput is)(p:= ip) \\
& \text{by auto} \\
& \text{with } inv2c5 \text{ srel all-nxt} \\
& \text{have } iinput is' = (iinput is)(p:= ip) \\
& \land i\text{allInput is’} = \text{allInput is} \cup \{ip\} \\
& \text{by (auto simp add: s2is-def)} \\
\text{moreover} \\
& \text{from } outpt \text{ srel nxt inv2c} \\
& i\text{chosen is’} = i\text{chosen is} \\
& \text{by (auto simp add: HNext-def HNextPart-def s2is-def Inv2c-inner-def)} \\
\text{ultimately} \\
& \text{show } \text{thesis} \\
& \text{using } ip-\text{Input} \\
& \text{by (auto simp add: IFail-def)} \\
& \text{qed} \\
& \text{thus } \text{thesis} \\
& \text{by auto} \\
\text{next} \\
& \text{assume } endphase2: \text{EndPhase2 } s \ s' \ p \\
& \text{from } endphase2 \\
& \text{have phase } s \ p =2 \\
& \text{by (auto simp add: EndPhase2-def)} \\
& \text{with } inv2c: \text{Ballot-nzero} \\
& \text{have bal-dblk-nzero: } bal(\text{dblock } s \ p) \neq 0 \\
& \text{by (auto simp add: Inv2c-inner-def)} \\
\text{moreover} \\
& \text{from } inv \\
& \text{have inv2a-dblock: Inv2a-innermost } s \ p \ (\text{dblock } s \ p) \\
& \text{by (auto simp add: HInv-def HInv1-def Inv2a-def Inv2a-inner-def blocksOf-def)} \\
\text{ultimately} \\
& \text{have p22: } \text{inp} \ (\text{dblock } s \ p) \in \text{allInput } s \\
& \text{by (auto simp add: Inv2a-innermost-def)} \\
& \text{from } inv \\
& \text{have allInput } s \subseteq \text{Inputs} \\
& \text{by (auto simp add: HInv-def HInv1-def)} \\
& \text{with } p22 \text{ NotAnInput endphase2} \\
& \text{have outpt-uni: outpt } s' \ p \neq \text{NotAnInput} \\
& \text{by (auto simp add: EndPhase2-def)} \\
& \text{show } \text{thesis} \\
& \text{proof (cases chosen } s = \text{NotAnInput)} \\
& \text{case True} \\
& \text{with } inv2c5 \\
& \text{have p31: } \forall q. \text{ outpt } s \ q = \text{NotAnInput} \\
& \text{by auto} \\
& \text{with } endphase2 \\
& \text{have p32: } \forall q \in \text{UNIV }\setminus\{p\}. \text{ outpt } s' \ q = \text{NotAnInput} \\
\end{align*}
\]
by (auto simp add: EndPhase2-def)

hence some-eq: (∀x. outpt s' x ≠ NotAnInput → x = p)
by auto

from p32 True nxt some-equality[of λp. outpt s' p ≠ NotAnInput, OF outpt-nni some-eq]

have p33: chosen s' = outpt s' p
by (auto simp add: HNext-def HNextPart-def)

with endphase2

have chosen s' = inp(dblock s p) ∧ outpt s' = (outpt s)(p:=inp(dblock s p))
by (auto simp add: EndPhase2-def)

with True p32

have if (chosen s = NotAnInput)
then (∃ip ∈ allInput s. chosen s' = ip
∧ outpt s' = (outpt s)(p := ip))
else ( outpt s' = (outpt s)(p:= chosen s)
∧ chosen s' = chosen s)

by auto

moreover
from endphase2 inv2c5 nxt

have inp s' = inp s ∧ allInput s' = allInput s
by (auto simp add: EndPhase2-def HNext-def HNextPart-def)

ultimately

show ?thesis
using srel p31
by (auto simp add: IChoose-def s2is-def)

next

case False

with nxt

have p31: chosen s' = chosen s
by (auto simp add: HNext-def HNextPart-def)

from inv'

have inv6: HInv6 s'
by (auto simp add: HInv-def)

have p32: outpt s' p = chosen s

proof

from endphase2

have outpt s' p = inp(dblock s p)
by (auto simp add: EndPhase2-def)

moreover
from inv6 p31

have outpt s' p ∈ {chosen s, NotAnInput}
by (auto simp add: HInv6-def)

ultimately

show ?thesis
using outpt-nni
by auto

qed

from srel False

have IChoose is is' p
proof (clarsimp simp add: IChoose-def s2is-def)
  from endphase2 inv2c
  have outpt s p = NotAnInput
    by (auto simp add: EndPhase2-def Inv2c-inner-def)
  moreover
  from endphase2 p31 p32 False
  have outpt s' = (outpt s) (p:= chosen s) ∧ chosen s' = chosen s
    by (auto simp add: EndPhase2-def)
  moreover
  from endphase2 nxt inv2c5
  have inpt s' = inpt s ∧ allInput s' = allInput s
    by (auto simp add: EndPhase2-def HNext-def HNextPart-def)
  ultimately
  show outpt s p = NotAnInput
    ∧ outpt s' = (outpt s)(p:= chosen s) ∧ chosen s' = chosen s
    ∧ inpt s' = inpt s ∧ allInput s' = allInput s
    by auto
  qed
  thus ?thesis
    by auto
  qed
  qed
  thus ∃ p. IFail is is' p ∨ IChoose is is' p
    by auto
  qed
end