

Relational Disjoint-Set Forests

Walter Guttman

September 13, 2023

Abstract

We give a simple relation-algebraic semantics of read and write operations on associative arrays. The array operations seamlessly integrate with assignments in the Hoare-logic library. Using relation algebras and Kleene algebras we verify the correctness of an array-based implementation of disjoint-set forests using the union-by-rank strategy and find operations with path compression, path splitting and path halving.

Contents

1	Overview	2
2	Relation-Algebraic Semantics of Associative Array Access	3
3	Relation-Algebraic Semantics of Disjoint-Set Forests	8
4	Verifying Operations on Disjoint-Set Forests	22
4.1	Make-Set	22
4.2	Find-Set	23
4.3	Path Compression	29
4.4	Find-Set with Path Compression	44
4.5	Union-Sets	47
5	More on Array Access and Disjoint-Set Forests	53
6	Verifying Further Operations on Disjoint-Set Forests	66
6.1	Init-Sets	68
6.2	Path Halving	69
6.3	Path Splitting	81
7	Verifying Union by Rank	96
7.1	Peano structures	97
7.2	Initialising Ranks	104
7.3	Union by Rank	106

1 Overview

Relation algebras and Kleene algebras have previously been used to reason about graphs and graph algorithms [2, 3, 4, 5, 9, 13, 16]. The operations of these algebras manipulate entire graphs, which is useful for specification but not directly intended for implementation. Low-level array access is a key ingredient for efficient algorithms [6]. We give a relation-algebraic semantics for such read/write access to associative arrays. This allows us to extend relation-algebraic verification methods to a lower level of more efficient implementations.

In this theory we focus on arrays with the same index and value sets, which can be modelled as homogeneous relations and therefore as elements of relation algebras and Kleene algebras [14, 18]. We implement and verify the correctness of disjoint-set forests with path compression strategies and union-by-rank [6, 8, 17].

In order to prepare this theory for future applications with weighted graphs, the verification uses Stone relation algebras, which have weaker axioms than relation algebras [10].

Section 2 contains the simple relation-algebraic semantics of associative array read and write and basic properties of these access operations. In Section 3 we give a Kleene-relation-algebraic semantics of disjoint-set forests. The make-set operation, find-set with path compression and the naive union-sets operation are implemented and verified in Section 4. Section 5 presents further results on disjoint-set forests and relational array access. The initialisation of disjoint-set forests, path halving and path splitting are implemented and verified in Section 6. In Section 7 we study relational Peano structures and implement and verify union-by-rank. Section 8 instantiates the Peano axioms by Boolean matrices.

This Isabelle/HOL theory formally verifies results in [11] and an extended version of that paper [12]. Theorem numbers from the extended version of the paper are mentioned in the theories for reference. See the paper for further details and related work.

Several Isabelle/HOL theories are related to disjoint sets. The theory `HOL/Library/Disjoint_Sets.thy` contains results about partitions and sets of disjoint sets and does not consider their implementation. An implementation of disjoint-set forests with path compression and a size-based heuristic in the Imperative/HOL framework is verified in Archive of Formal Proofs entry [15]. Improved automation of this proof is considered in Archive of Formal Proofs entry [19]. These approaches are based on logical specifications whereas the present theory uses relation algebras and Kleene algebras.

```

theory Disjoint-Set-Forests

imports
  HOL-Hoare.Hoare-Logic
  Stone-Kleene-Relation-Algebras.Kleene-Relation-Algebras
begin

no-notation
  minus (infixl - 65) and
  trancl ((-+) [1000] 999)

context p-algebra
begin

abbreviation minus :: 'a ⇒ 'a ⇒ 'a (infixl - 65)
  where  $x - y \equiv x \sqcap -y$ 

end

```

An arc in a Stone relation algebra corresponds to an atom in a relation algebra and represents a single edge in a graph. A point represents a set of nodes. A rectangle represents the Cartesian product of two sets of nodes [4].

```

context times-top
begin

abbreviation rectangle :: 'a ⇒ bool
  where  $rectangle\ x \equiv x * top * x = x$ 

end

context stone-relation-algebra
begin

lemma arc-rectangle:
   $arc\ x \implies rectangle\ x$ 
  using arc-top-arc by blast

```

2 Relation-Algebraic Semantics of Associative Array Access

The following two operations model updating array x at index y to value z , and reading the content of array x at index y , respectively. The read operation uses double brackets to avoid ambiguity with list syntax. The remainder of this section shows basic properties of these operations.

abbreviation *rel-update* :: 'a ⇒ 'a ⇒ 'a ⇒ 'a (([-⟶-]) [70, 65, 65] 61)
where $x[y⟶z] \equiv (y \sqcap z^T) \sqcup (-y \sqcap x)$

abbreviation *rel-access* :: 'a ⇒ 'a ⇒ 'a ((2-[-]) [70, 65] 65)
where $x[[y]] \equiv x^T * y$

Theorem 1.1

lemma *update-univalent*:

assumes *univalent x*

and *vector y*

and *injective z*

shows *univalent (x[y⟶z])*

proof –

have 1: *univalent (y ⊔ z^T)*

using *assms(3) inf-commute univalent-inf-closed by force*

have $(y \sqcap z^T)^T * (-y \sqcap x) = (y^T \sqcap z) * (-y \sqcap x)$

by (*simp add: conv-dist-inf*)

also have ... = $z * (y \sqcap -y \sqcap x)$

by (*metis assms(2) covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute*)

finally have 2: $(y \sqcap z^T)^T * (-y \sqcap x) = \text{bot}$

by *simp*

have 3: *vector (-y)*

using *assms(2) vector-complement-closed by simp*

have $(-y \sqcap x)^T * (y \sqcap z^T) = (-y^T \sqcap x^T) * (y \sqcap z^T)$

by (*simp add: conv-complement conv-dist-inf*)

also have ... = $x^T * (-y \sqcap y \sqcap z^T)$

using 3 **by** (*metis (mono-tags, opaque-lifting) conv-complement covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute*)

finally have 4: $(-y \sqcap x)^T * (y \sqcap z^T) = \text{bot}$

by *simp*

have 5: *univalent (-y ⊔ x)*

using *assms(1) inf-commute univalent-inf-closed by fastforce*

have $(x[y⟶z])^T * (x[y⟶z]) = (y \sqcap z^T)^T * (x[y⟶z]) \sqcup (-y \sqcap x)^T * (x[y⟶z])$

by (*simp add: conv-dist-sup mult-right-dist-sup*)

also have ... = $(y \sqcap z^T)^T * (y \sqcap z^T) \sqcup (y \sqcap z^T)^T * (-y \sqcap x) \sqcup (-y \sqcap x)^T * (y \sqcap z^T) \sqcup (-y \sqcap x)^T * (-y \sqcap x)$

by (*simp add: mult-left-dist-sup sup-assoc*)

finally show *?thesis*

using 1 2 4 5 **by** *simp*

qed

Theorem 1.2

lemma *update-total*:

assumes *total x*

and *vector y*

and *regular y*

and *surjective z*

shows $total (x[y \mapsto z])$
proof –
have $(x[y \mapsto z]) * top = x * top[y \mapsto top * z]$
by (*simp add: assms(2) semiring.distrib-right vector-complement-closed*
vector-inf-comp conv-dist-comp)
also have $... = top[y \mapsto top]$
using *assms(1) assms(4)* **by** *simp*
also have $... = top$
using *assms(3) regular-complement-top* **by** *auto*
finally show *?thesis*
by *simp*
qed

Theorem 1.3

lemma *update-mapping*:
assumes *mapping x*
and *vector y*
and *regular y*
and *bijjective z*
shows *mapping (x[y \mapsto z])*
using *assms update-univalent update-total* **by** *simp*

Theorem 1.4

lemma *read-injective*:
assumes *injective y*
and *univalent x*
shows *injective (x[[y]])*
using *assms injective-mult-closed univalent-conv-injective* **by** *blast*

Theorem 1.5

lemma *read-surjective*:
assumes *surjective y*
and *total x*
shows *surjective (x[[y]])*
using *assms surjective-mult-closed total-conv-surjective* **by** *blast*

Theorem 1.6

lemma *read-bijjective*:
assumes *bijjective y*
and *mapping x*
shows *bijjective (x[[y]])*
by (*simp add: assms read-injective read-surjective*)

Theorem 1.7

lemma *read-point*:
assumes *point p*
and *mapping x*
shows *point (x[[p]])*
using *assms comp-associative read-injective read-surjective* **by** *auto*

Theorem 1.8

lemma *update-postcondition*:

assumes *point x point y*

shows $x \sqcap p = x * y^T \longleftrightarrow p[[x]] = y$

apply (*rule iffI*)

subgoal by (*metis assms comp-associative conv-dist-comp conv-involutive covector-inf-comp-3 equivalence-top-closed vector-covector*)

subgoal

apply (*rule order.antisym*)

subgoal by (*metis assms conv-dist-comp conv-involutive inf.boundedI inf.cobounded1 vector-covector vector-restrict-comp-conv*)

subgoal by (*smt assms comp-associative conv-dist-comp conv-involutive covector-restrict-comp-conv dense-conv-closed equivalence-top-closed inf.boundedI shunt-mapping vector-covector preorder-idempotent*)

done

done

Back and von Wright's array independence requirements [1], later also lens laws [7]

lemma *put-get-sub*:

assumes *vector y surjective u vector z u ≤ y*

shows $(x[y \rightarrow z])[[u]] = z$

proof –

have $(x[y \rightarrow z])[[u]] = (y^T \sqcap z) * u \sqcup (-y^T \sqcap x^T) * u$

by (*simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)

also have ... = $z * u$

proof –

have $(-y^T \sqcap x^T) * u \leq (-y^T \sqcap x^T) * y$

by (*simp add: assms(4) mult-right-isotone*)

also have ... = *bot*

by (*metis assms(1) covector-inf-comp-3 inf-commute conv-complement mult-right-zero p-inf vector-complement-closed*)

finally have $(-y^T \sqcap x^T) * u = \text{bot}$

by (*simp add: bot-unique*)

thus *?thesis*

using *assms(1,4) covector-inf-comp-3 inf.absorb-iff1 inf-commute* **by** *auto*

qed

also have ... = z

by (*metis assms(2,3) mult-assoc*)

finally show *?thesis*

·

qed

Theorem 2.1

lemma *put-get*:

assumes *vector y surjective y vector z*

shows $(x[y \rightarrow z])[[y]] = z$

by (*simp add: assms put-get-sub*)

Theorem 2.3

lemma *put-put*:

$$(x[y \mapsto z])[y \mapsto w] = x[y \mapsto w]$$

by (*metis inf-absorb2 inf-commute inf-le1 inf-sup-distrib1 maddux-3-13 sup-inf-absorb*)

Theorem 2.5

lemma *get-put*:

assumes *point y*

shows $x[y \mapsto x[[y]]] = x$

proof –

have $x[y \mapsto x[[y]]] = (y \sqcap y^T * x) \sqcup (-y \sqcap x)$

by (*simp add: conv-dist-comp*)

also have $\dots = (y \sqcap x) \sqcup (-y \sqcap x)$

proof –

have $y \sqcap y^T * x = y \sqcap x$

proof (*rule order.antisym*)

have $y \sqcap y^T * x = (y \sqcap y^T) * x$

by (*simp add: assms vector-inf-comp*)

also have $(y \sqcap y^T) * x = y * y^T * x$

by (*simp add: assms vector-covector*)

also have $\dots \leq x$

using *assms comp-isotone* **by** *fastforce*

finally show $y \sqcap y^T * x \leq y \sqcap x$

by *simp*

have $y \sqcap x \leq y^T * x$

by (*simp add: assms vector-restrict-comp-conv*)

thus $y \sqcap x \leq y \sqcap y^T * x$

by *simp*

qed

thus *?thesis*

by *simp*

qed

also have $\dots = x$

proof –

have *regular y*

using *assms bijective-regular* **by** *blast*

thus *?thesis*

by (*metis inf.sup-monoid.add-commute maddux-3-11-pp*)

qed

finally show *?thesis*

.

qed

lemma *update-inf*:

$$u \leq y \implies (x[y \mapsto z]) \sqcap u = z^T \sqcap u$$

by (*smt comp-inf.mult-right-dist-sup comp-inf.semiring.mult-zero-right inf.left-commute inf.sup-monoid.add-assoc inf-absorb2 p-inf sup-bot-right inf.sup-monoid.add-commute*)

lemma *update-inf-same*:
 $(x[y \rightarrow z]) \sqcap y = z^T \sqcap y$
by (*simp add: update-inf*)

lemma *update-inf-different*:
 $u \leq -y \implies (x[y \rightarrow z]) \sqcap u = x \sqcap u$
by (*smt inf.right-idem inf.sup-monoid.add-commute*
inf.sup-relative-same-increasing inf-import-p maddux-3-13 sup.cobounded2
update-inf-same)

end

3 Relation-Algebraic Semantics of Disjoint-Set Forests

A disjoint-set forest represents a partition of a set into equivalence classes. We take the represented equivalence relation as the semantics of a forest. It is obtained by operation *fc* below. Additionally, operation *wcc* giving the weakly connected components of a graph will be used for the semantics of the union of two disjoint sets. Finally, operation *root* yields the root of a component tree, that is, the representative of a set containing a given element. This section defines these operations and derives their properties.

context *stone-kleene-relation-algebra*
begin

Theorem 5.2

lemma *omit-redundant-points*:
assumes *point p*
shows $p \sqcap x^* = (p \sqcap 1) \sqcup (p \sqcap x) * (-p \sqcap x)^*$
proof (*rule order.antisym*)
let $?p = p \sqcap 1$
have $?p * x * (-p \sqcap x)^* * ?p \leq ?p * \text{top} * ?p$
by (*metis comp-associative mult-left-isotone mult-right-isotone top.extremum*)
also have $\dots \leq ?p$
by (*simp add: assms injective-codomain vector-inf-one-comp*)
finally have $?p * x * (-p \sqcap x)^* * ?p * x \leq ?p * x$
using *mult-left-isotone* **by** *blast*
hence $?p * x * (-p \sqcap x)^* * (p \sqcap x) \leq ?p * x$
by (*simp add: assms comp-associative vector-inf-one-comp*)
also have $1: \dots \leq ?p * x * (-p \sqcap x)^*$
using *mult-right-isotone star.circ-reflexive* **by** *fastforce*
finally have $?p * x * (-p \sqcap x)^* * (p \sqcap x) \sqcup ?p * x * (-p \sqcap x)^* * (-p \sqcap x) \leq$
 $?p * x * (-p \sqcap x)^*$
by (*simp add: mult-right-isotone star.circ-plus-same star.left-plus-below-circ*
mult-assoc)
hence $?p * x * (-p \sqcap x)^* * ((p \sqcup -p) \sqcap x) \leq ?p * x * (-p \sqcap x)^*$

by (*simp add: comp-inf.mult-right-dist-sup mult-left-dist-sup*)
 hence $?p * x * (-p \sqcap x)^* * x \leq ?p * x * (-p \sqcap x)^*$
 by (*metis assms bijective-regular inf.absorb2 inf.cobounded1*
inf.sup-monoid.add-commute shunting-p)
 hence $?p * x * (-p \sqcap x)^* * x \sqcup ?p * x \leq ?p * x * (-p \sqcap x)^*$
 using 1 by *simp*
 hence $?p * (1 \sqcup x * (-p \sqcap x)^*) * x \leq ?p * x * (-p \sqcap x)^*$
 by (*simp add: comp-associative mult-left-dist-sup mult-right-dist-sup*)
 also have $\dots \leq ?p * (1 \sqcup x * (-p \sqcap x)^*)$
 by (*simp add: comp-associative mult-right-isotone*)
 finally have $?p * x^* \leq ?p * (1 \sqcup x * (-p \sqcap x)^*)$
 using *star-right-induct* by (*meson dual-order.trans le-supI*
mult-left-sub-dist-sup-left mult-sub-right-one)
 also have $\dots = ?p \sqcup ?p * x * (-p \sqcap x)^*$
 by (*simp add: comp-associative semiring.distrib-left*)
 finally show $p \sqcap x^* \leq ?p \sqcup (p \sqcap x) * (-p \sqcap x)^*$
 by (*simp add: assms vector-inf-one-comp*)
 show $?p \sqcup (p \sqcap x) * (-p \sqcap x)^* \leq p \sqcap x^*$
 by (*metis assms comp-isotone inf.boundedI inf.cobounded1 inf.coboundedI2*
inf.sup-monoid.add-commute le-supI star.circ-increasing star.circ-transitive-equal
star-isotone star-left-unfold-equal sup.cobounded1 vector-export-comp)
 qed

Weakly connected components

abbreviation $wcc\ x \equiv (x \sqcup x^T)^*$

Theorem 7.1

lemma *wcc-equivalence*:

equivalence (*wcc x*)

apply (*intro conjI*)

subgoal by (*simp add: star.circ-reflexive*)

subgoal by (*simp add: star.circ-transitive-equal*)

subgoal by (*simp add: conv-dist-sup conv-star-commute sup-commute*)

done

Theorem 7.2

lemma *wcc-increasing*:

$x \leq wcc\ x$

by (*simp add: star.circ-sub-dist-1*)

lemma *wcc-isotone*:

$x \leq y \implies wcc\ x \leq wcc\ y$

using *conv-isotone star-isotone sup-mono* **by** *blast*

lemma *wcc-idempotent*:

$wcc\ (wcc\ x) = wcc\ x$

using *star-involutive wcc-equivalence* **by** *auto*

Theorem 7.3

lemma *wcc-below-wcc*:
 $x \leq wcc\ y \implies wcc\ x \leq wcc\ y$
using *wcc-idempotent wcc-isotone* **by** *fastforce*

lemma *wcc-galois*:
 $x \leq wcc\ y \iff wcc\ x \leq wcc\ y$
using *order-trans star.circ-sub-dist-1 wcc-below-wcc* **by** *blast*

[Theorem 7.4](#)

lemma *wcc-bot*:
 $wcc\ bot = 1$
by (*simp add: star.circ-zero*)

lemma *wcc-one*:
 $wcc\ 1 = 1$
by (*simp add: star-one*)

[Theorem 7.5](#)

lemma *wcc-top*:
 $wcc\ top = top$
by (*simp add: star.circ-top*)

[Theorem 7.6](#)

lemma *wcc-with-loops*:
 $wcc\ x = wcc\ (x \sqcup 1)$
by (*metis conv-dist-sup star-decompose-1 star-sup-one sup-commute symmetric-one-closed*)

lemma *wcc-without-loops*:
 $wcc\ x = wcc\ (x - 1)$
by (*metis conv-star-commute star-sum reachable-without-loops*)

lemma *forest-components-wcc*:
injective $x \implies wcc\ x = forest-components\ x$
by (*simp add: cancel-separate-1*)

[Theorem 7.8](#)

lemma *wcc-sup-wcc*:
 $wcc\ (x \sqcup y) = wcc\ (x \sqcup wcc\ y)$
by (*smt (verit, ccfv-SIG) le-sup-iff order.antisym sup-right-divisibility wcc-below-wcc wcc-increasing*)

Components of a forest, which is represented using edges directed towards the roots

abbreviation $fc\ x \equiv x^* * x^{T^*}$

[Theorem 3.1](#)

lemma *fc-equivalence*:
univalent $x \implies equivalence\ (fc\ x)$

apply (*intro conjI*)
subgoal by (*simp add: reflexive-mult-closed star.circ-reflexive*)
subgoal by (*metis cancel-separate-1 order.eq-iff star.circ-transitive-equal*)
subgoal by (*simp add: conv-dist-comp conv-star-commute*)
done

Theorem 3.2

lemma *fc-increasing*:
 $x \leq fc\ x$
by (*metis le-supE mult-left-isotone star.circ-back-loop-fixpoint star.circ-increasing*)

Theorem 3.3

lemma *fc-isotone*:
 $x \leq y \implies fc\ x \leq fc\ y$
by (*simp add: comp-isotone conv-isotone star-isotone*)

Theorem 3.4

lemma *fc-idempotent*:
 $univalent\ x \implies fc\ (fc\ x) = fc\ x$
by (*metis fc-equivalence cancel-separate-1 star.circ-transitive-equal star-involutive*)

Theorem 3.5

lemma *fc-star*:
 $univalent\ x \implies (fc\ x)^* = fc\ x$
using *fc-equivalence fc-idempotent star.circ-transitive-equal* **by** *simp*

lemma *fc-plus*:
 $univalent\ x \implies (fc\ x)^+ = fc\ x$
by (*metis fc-star star.circ-decompose-9*)

Theorem 3.6

lemma *fc-bot*:
 $fc\ bot = 1$
by (*simp add: star.circ-zero*)

lemma *fc-one*:
 $fc\ 1 = 1$
by (*simp add: star-one*)

Theorem 3.7

lemma *fc-top*:
 $fc\ top = top$
by (*simp add: star.circ-top*)

Theorem 7.7

lemma *fc-wcc*:
 $univalent\ x \implies wcc\ x = fc\ x$

by (*simp add: fc-star star-decompose-1*)

lemma *fc-via-root*:

assumes *total* ($p^* * (p \sqcap 1)$)

shows $fc\ p = p^* * (p \sqcap 1) * p^{T^*}$

proof (*rule order.antisym*)

have $1 \leq p^* * (p \sqcap 1) * p^{T^*}$

by (*smt assms comp-associative conv-dist-comp conv-star-commute coreflexive-idempotent coreflexive-symmetric inf.cobounded2 total-var*)

hence $fc\ p \leq p^* * p^* * (p \sqcap 1) * p^{T^*} * p^{T^*}$

by (*metis comp-right-one mult-left-isotone mult-right-isotone mult-assoc*)

thus $fc\ p \leq p^* * (p \sqcap 1) * p^{T^*}$

by (*simp add: star.circ-transitive-equal mult-assoc*)

show $p^* * (p \sqcap 1) * p^{T^*} \leq fc\ p$

by (*metis comp-isotone inf.cobounded2 mult-1-right order.refl*)

qed

Theorem 5.1

lemma *update-acyclic-1*:

assumes *acyclic* ($p - 1$)

and *point* y

and *vector* w

and $w \leq p^* * y$

shows *acyclic* ($(p[w \longrightarrow y]) - 1$)

proof –

let $?p = p[w \longrightarrow y]$

have $w * y^T \leq p^*$

using *assms(2,4) shunt-bijective* **by** *blast*

hence $w * y^T \leq (p - 1)^*$

using *reachable-without-loops* **by** *auto*

hence $w * y^T - 1 \leq (p - 1)^* - 1$

by (*simp add: inf.coboundedI2 inf.sup-monoid.add-commute*)

also have $\dots \leq (p - 1)^+$

by (*simp add: star-plus-without-loops*)

finally have $1: w \sqcap y^T \sqcap -1 \leq (p - 1)^+$

using *assms(2,3) vector-covector* **by** *auto*

have $?p - 1 = (w \sqcap y^T \sqcap -1) \sqcup (-w \sqcap p \sqcap -1)$

by (*simp add: inf-sup-distrib2*)

also have $\dots \leq (p - 1)^+ \sqcup (-w \sqcap p \sqcap -1)$

using *1 sup-left-isotone* **by** *blast*

also have $\dots \leq (p - 1)^+ \sqcup (p - 1)$

using *comp-inf.mult-semi-associative sup-right-isotone* **by** *auto*

also have $\dots = (p - 1)^+$

by (*metis star.circ-back-loop-fixpoint sup.right-idem*)

finally have $(?p - 1)^+ \leq (p - 1)^+$

by (*metis comp-associative comp-isotone star.circ-transitive-equal star.left-plus-circ star-isotone*)

also have $\dots \leq -1$

using *assms(1)* **by** *blast*

finally show *?thesis*
 by *simp*
qed

lemma *update-acyclic-2:*

assumes *acyclic (p - 1)*
and *point y*
and *point x*
and $y \leq p^{T^*} * x$
and *univalent p*
and $p^T * y \leq y$
shows *acyclic ((p[p^{T*}*x→y]) - 1)*
proof –
have $p^T * p^* * y = p^T * p * p^* * y \sqcup p^T * y$
by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint*)
also have $\dots \leq p^* * y$
by (*metis assms(5,6) comp-right-one le-supI le-supI2 mult-left-isotone star.circ-loop-fixpoint star.circ-transitive-equal*)
finally have $p^{T^*} * x \leq p^* * y$
by (*simp add: assms(2-4) bijective-reverse conv-star-commute comp-associative star-left-induct*)
thus *?thesis*
by (*simp add: assms(1-3) vector-mult-closed update-acyclic-1*)
qed

lemma *update-acyclic-3:*

assumes *acyclic (p - 1)*
and *point y*
and *point w*
and $y \leq p^{T^*} * w$
shows *acyclic ((p[w→y]) - 1)*
by (*simp add: assms bijective-reverse conv-star-commute update-acyclic-1*)

lemma *rectangle-star-rectangle:*

rectangle a $\implies a * x^* * a \leq a$
by (*metis mult-left-isotone mult-right-isotone top.extremum*)

lemma *arc-star-arc:*

arc a $\implies a * x^* * a \leq a$
using *arc-top-arc rectangle-star-rectangle* **by** *blast*

lemma *star-rectangle-decompose:*

assumes *rectangle a*
shows $(a \sqcup x)^* = x^* \sqcup x^* * a * x^*$
proof (*rule order.antisym*)
have $1: 1 \leq x^* \sqcup x^* * a * x^*$
by (*simp add: star.circ-reflexive sup.coboundedI1*)
have $(a \sqcup x) * (x^* \sqcup x^* * a * x^*) = a * x^* \sqcup a * x^* * a * x^* \sqcup x^+ \sqcup x^+ * a * x^*$
qed

by (*metis comp-associative semiring.combine-common-factor semiring.distrib-left sup-commute*)
also have $\dots = a * x^* \sqcup x^+ \sqcup x^+ * a * x^*$
using *assms rectangle-star-rectangle* **by** (*simp add: mult-left-isotone sup-absorb1*)
also have $\dots = x^+ \sqcup x^* * a * x^*$
by (*metis comp-associative star.circ-loop-fixpoint sup-assoc sup-commute*)
also have $\dots \leq x^* \sqcup x^* * a * x^*$
using *star.left-plus-below-circ sup-left-isotone* **by** *auto*
finally show $(a \sqcup x)^* \leq x^* \sqcup x^* * a * x^*$
using *1* **by** (*metis comp-right-one le-supI star-left-induct*)
next
show $x^* \sqcup x^* * a * x^* \leq (a \sqcup x)^*$
by (*metis comp-isotone le-supE le-supI star.circ-increasing star.circ-transitive-equal star-isotone sup-ge2*)
qed

lemma *star-arc-decompose*:
 $\text{arc } a \implies (a \sqcup x)^* = x^* \sqcup x^* * a * x^*$
using *arc-top-arc star-rectangle-decompose* **by** *blast*

lemma *plus-rectangle-decompose*:
assumes *rectangle a*
shows $(a \sqcup x)^+ = x^+ \sqcup x^* * a * x^*$
proof –
have $(a \sqcup x)^+ = (a \sqcup x) * (x^* \sqcup x^* * a * x^*)$
by (*simp add: assms star-rectangle-decompose*)
also have $\dots = a * x^* \sqcup a * x^* * a * x^* \sqcup x^+ \sqcup x^+ * a * x^*$
by (*metis comp-associative semiring.combine-common-factor semiring.distrib-left sup-commute*)
also have $\dots = a * x^* \sqcup x^+ \sqcup x^+ * a * x^*$
using *assms rectangle-star-rectangle* **by** (*simp add: mult-left-isotone sup-absorb1*)
also have $\dots = x^+ \sqcup x^* * a * x^*$
by (*metis comp-associative star.circ-loop-fixpoint sup-assoc sup-commute*)
finally show *?thesis*
by *simp*
qed

Theorem 8.1

lemma *plus-arc-decompose*:
 $\text{arc } a \implies (a \sqcup x)^+ = x^+ \sqcup x^* * a * x^*$
using *arc-top-arc plus-rectangle-decompose* **by** *blast*

Theorem 8.2

lemma *update-acyclic-4*:
assumes *acyclic (p - 1)*
and *point y*
and *point w*

and $y \sqcap p^* * w = \text{bot}$
shows *acyclic* $((p[w \longrightarrow y]) - 1)$
proof –
let $?p = p[w \longrightarrow y]$
have $y^T * p^* * w \leq -1$
using *assms(4) comp-associative pseudo-complement schroeder-3-p* **by** *auto*
hence $1: p^* * w * y^T * p^* \leq -1$
by *(metis comp-associative comp-commute-below-diversity star.circ-transitive-equal)*
have $?p - 1 \leq (w \sqcap y^T) \sqcup (p - 1)$
by *(metis comp-inf.mult-right-dist-sup dual-order.trans inf.cobounded1 inf.coboundedI2 inf.sup-monoid.add-assoc le-supI sup.cobounded1 sup-ge2)*
also have $\dots = w * y^T \sqcup (p - 1)$
using *assms(2,3)* **by** *(simp add: vector-covector)*
finally have $(?p - 1)^+ \leq (w * y^T \sqcup (p - 1))^+$
by *(simp add: comp-isotone star-isotone)*
also have $\dots = (p - 1)^+ \sqcup (p - 1)^* * w * y^T * (p - 1)^*$
using *assms(2,3) plus-arc-decompose points-arc* **by** *(simp add: comp-associative)*
also have $\dots \leq (p - 1)^+ \sqcup p^* * w * y^T * p^*$
using *reachable-without-loops* **by** *auto*
also have $\dots \leq -1$
using 1 *assms(1)* **by** *simp*
finally show *?thesis*
by *simp*
qed

Theorem 8.3

lemma *update-acyclic-5*:
assumes *acyclic* $(p - 1)$
and *point* w
shows *acyclic* $((p[w \longrightarrow w]) - 1)$
proof –
let $?p = p[w \longrightarrow w]$
have $?p - 1 \leq (w \sqcap w^T \sqcap -1) \sqcup (p - 1)$
by *(metis comp-inf.mult-right-dist-sup inf.cobounded2 inf.sup-monoid.add-assoc sup-right-isotone)*
also have $\dots = p - 1$
using *assms(2)* **by** *(metis comp-inf.covector-complement-closed equivalence-top-closed inf-top.right-neutral maddux-3-13 pseudo-complement regular-closed-top regular-one-closed vector-covector vector-top-closed)*
finally show *?thesis*
using *assms(1) acyclic-down-closed* **by** *blast*
qed

Root of the tree containing point x in the disjoint-set forest p

abbreviation *roots* $p \equiv (p \sqcap 1) * \text{top}$
abbreviation *root* $p \ x \equiv p^{T^*} * x \sqcap \text{roots } p$

Theorem 4.1

lemma *root-var*:
 $root\ p\ x = (p \sqcap 1) * p^{T*} * x$
by (*simp add: coreflexive-comp-top-inf inf-commute mult-assoc*)

Theorem 4.2

lemma *root-successor-loop*:
 $univalent\ p \implies root\ p\ x = p[[root\ p\ x]]$
by (*metis root-var injective-codomain comp-associative conv-dist-inf coreflexive-symmetric equivalence-one-closed inf.cobounded2 univalent-conv-injective*)

lemma *root-transitive-successor-loop*:
 $univalent\ p \implies root\ p\ x = p^{T*} * (root\ p\ x)$
by (*metis mult-1-right star-one star-simulation-right-equal root-successor-loop*)

lemma *roots-successor-loop*:
 $univalent\ p \implies p[[roots\ p]] = roots\ p$
by (*metis conv-involutive inf-commute injective-codomain one-inf-conv mult-assoc*)

lemma *roots-transitive-successor-loop*:
 $univalent\ p \implies p^{T*} * (roots\ p) = roots\ p$
by (*metis comp-associative star.circ-left-top star-simulation-right-equal roots-successor-loop*)

The root of a tree of a node belongs to the same component as the node.

lemma *root-same-component*:
 $injective\ x \implies root\ p\ x * x^T \leq fc\ p$
by (*metis comp-associative coreflexive-comp-top-inf eq-refl inf.sup-left-divisibility inf.sup-monoid.add-commute mult-isotone star.circ-circ-mult star.circ-right-top star.circ-transitive-equal star-one star-outer-increasing test-preserves-equation top-greatest*)

lemma *root-vector*:
 $vector\ x \implies vector\ (root\ p\ x)$
by (*simp add: vector-mult-closed root-var*)

lemma *root-vector-inf*:
 $vector\ x \implies root\ p\ x * x^T = root\ p\ x \sqcap x^T$
by (*simp add: vector-covector root-vector*)

lemma *root-same-component-vector*:
 $injective\ x \implies vector\ x \implies root\ p\ x \sqcap x^T \leq fc\ p$
using *root-same-component root-vector-inf* **by** *fastforce*

lemma *univalent-root-successors*:
assumes *univalent p*
shows $(p \sqcap 1) * p^* = p \sqcap 1$
proof (*rule order.antisym*)

have $(p \sqcap 1) * p \leq p \sqcap 1$
by (*smt assms(1) comp-inf.mult-semi-associative conv-dist-comp conv-dist-inf conv-order equivalence-one-closed inf.absorb1 inf.sup-monoid.add-assoc injective-codomain*)
thus $(p \sqcap 1) * p^* \leq p \sqcap 1$
using *star-right-induct-mult* **by** *blast*
show $p \sqcap 1 \leq (p \sqcap 1) * p^*$
by (*metis coreflexive-idempotent inf-le1 inf-le2 mult-right-isotone order-trans star.circ-increasing*)
qed

lemma *same-component-same-root-sub*:

assumes *univalent p*
and *bijjective y*
and $x * y^T \leq fc p$
shows $root p x \leq root p y$
proof –
have $root p x * y^T \leq (p \sqcap 1) * p^{T*}$
by (*smt assms(1,3) mult-isotone mult-assoc root-var fc-plus fc-star order.eq-iff univalent-root-successors*)
thus *?thesis*
by (*simp add: assms(2) shunt-bijjective root-var*)
qed

lemma *same-component-same-root*:

assumes *univalent p*
and *bijjective x*
and *bijjective y*
and $x * y^T \leq fc p$
shows $root p x = root p y$
proof (*rule order.antisym*)
show $root p x \leq root p y$
using *assms(1,3,4) same-component-same-root-sub* **by** *blast*
have $y * x^T \leq fc p$
using *assms(1,4) fc-equivalence conv-dist-comp conv-isotone* **by** *fastforce*
thus $root p y \leq root p x$
using *assms(1,2) same-component-same-root-sub* **by** *blast*
qed

lemma *same-roots-sub*:

assumes *univalent q*
and $p \sqcap 1 \leq q \sqcap 1$
and $fc p \leq fc q$
shows $p^* * (p \sqcap 1) \leq q^* * (q \sqcap 1)$
proof –
have $p^* * (p \sqcap 1) \leq p^* * (q \sqcap 1)$
using *assms(2) mult-right-isotone* **by** *auto*
also have $\dots \leq fc p * (q \sqcap 1)$
using *mult-left-isotone mult-right-isotone star.circ-reflexive* **by** *fastforce*

also have $\dots \leq fc\ q * (q \sqcap 1)$
by (*simp add: assms(3) mult-left-isotone*)
also have $\dots = q^* * (q \sqcap 1)$
by (*metis assms(1) conv-dist-comp conv-dist-inf conv-star-commute*
inf-commute one-inf-conv symmetric-one-closed mult-assoc
univalent-root-successors)
finally show *?thesis*
qed

lemma same-roots:
assumes *univalent p*
and *univalent q*
and $p \sqcap 1 = q \sqcap 1$
and $fc\ p = fc\ q$
shows $p^* * (p \sqcap 1) = q^* * (q \sqcap 1)$
by (*smt assms conv-dist-comp conv-dist-inf conv-involutive conv-star-commute*
inf-commute one-inf-conv symmetric-one-closed root-var univalent-root-successors)

lemma same-root:
assumes *univalent p*
and *univalent q*
and $p \sqcap 1 = q \sqcap 1$
and $fc\ p = fc\ q$
shows $root\ p\ x = root\ q\ x$
by (*metis assms mult-assoc root-var univalent-root-successors*)

lemma loop-root:
assumes *injective x*
and $x = p[[x]]$
shows $x = root\ p\ x$
proof (*rule order.antisym*)
have $x \leq p * x$
by (*metis assms comp-associative comp-right-one conv-order*
equivalence-one-closed ex231c inf.orderE inf.sup-monoid.add-commute
mult-left-isotone mult-right-isotone one-inf-conv)
hence $x = (p \sqcap 1) * x$
by (*simp add: assms(1) inf-absorb2 injective-comp-right-dist-inf*)
thus $x \leq root\ p\ x$
by (*metis assms(2) coreflexive-comp-top-inf inf.boundedI inf.cobounded1*
inf.cobounded2 mult-isotone star.circ-increasing)
next
show $root\ p\ x \leq x$
using *assms(2) le-infI1 star-left-induct-mult* **by** *auto*
qed

lemma one-loop:
assumes *acyclic (p - 1)*
and *univalent p*

shows $(p \sqcap 1) * (p^T - 1)^+ * (p \sqcap 1) = \text{bot}$
proof –
have $p^{T+} \sqcap (p \sqcap 1) * \text{top} * (p \sqcap 1) = (p \sqcap 1) * p^{T+} * (p \sqcap 1)$
by (*simp add: test-comp-test-top*)
also have $\dots \leq p^{T*} * (p \sqcap 1)$
by (*simp add: inf.coboundedI2 mult-left-isotone star.circ-mult-upper-bound star.circ-reflexive star.left-plus-below-circ*)
also have $\dots = p \sqcap 1$
by (*metis assms(2) conv-dist-comp conv-dist-inf conv-star-commute inf-commute one-inf-conv symmetric-one-closed univalent-root-successors*)
also have $\dots \leq 1$
by *simp*
finally have $(p \sqcap 1) * \text{top} * (p \sqcap 1) \leq -(p^{T+} - 1)$
using *p-antitone p-antitone-iff p-shunting-swap* **by** *blast*
hence $(p \sqcap 1)^T * (p^{T+} - 1) * (p \sqcap 1)^T \leq \text{bot}$
using *triple-schroeder-p p-top* **by** *blast*
hence $(p \sqcap 1) * (p^{T+} - 1) * (p \sqcap 1) = \text{bot}$
by (*simp add: coreflexive-symmetric le-bot*)
thus *?thesis*
by (*smt assms(1) conv-complement conv-dist-comp conv-dist-inf conv-star-commute inf-absorb1 star.circ-plus-same symmetric-one-closed reachable-without-loops star-plus-without-loops*)
qed

lemma *root-root*:

$\text{root } p \ x = \text{root } p \ (\text{root } p \ x)$
by (*smt comp-associative comp-inf.mult-right-sub-dist-sup-right dual-order.eq-iff inf.cobounded1 inf.cobounded2 inf.orderE mult-right-isotone star.circ-loop-fixpoint star.circ-transitive-equal root-var*)

lemma *loop-root-2*:

assumes *acyclic* $(p - 1)$
and *univalent* p
and *injective* x
and $x \leq p^{T+} * x$
shows $x = \text{root } p \ x$
proof (*rule order.antisym*)
have $1: x = x - (-1 * x)$
by (*metis assms(3) comp-injective-below-complement inf.orderE mult-1-left regular-one-closed*)
have $x \leq (p^T - 1)^+ * x \sqcup (p \sqcap 1) * x$
by (*metis assms(4) inf-commute mult-right-dist-sup one-inf-conv plus-reachable-without-loops*)
also have $\dots \leq -1 * x \sqcup (p \sqcap 1) * x$
by (*metis assms(1) conv-complement conv-dist-inf conv-isotone conv-plus-commute mult-left-isotone semiring.add-right-mono symmetric-one-closed*)
also have $\dots \leq -1 * x \sqcup \text{root } p \ x$
using *comp-isotone inf.coboundedI2 star.circ-reflexive sup-right-isotone* **by**

auto
finally have $x \leq (-1 * x \sqcup \text{root } p \ x) - (-1 * x)$
using *1 inf.boundedI inf.order-iff* **by** *blast*
also have $\dots \leq \text{root } p \ x$
using *inf.sup-left-divisibility* **by** *auto*
finally show *2: x ≤ root p x*

have $\text{root } p \ x = (p \sqcap 1) * x \sqcup (p \sqcap 1) * (p^T - 1)^+ * x$
by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint*
sup-commute reachable-without-loops root-var)
also have $\dots \leq x \sqcup (p \sqcap 1) * (p^T - 1)^+ * \text{root } p \ x$
using *2* **by** (*metis coreflexive-comp-top-inf inf.cobounded2 mult-right-isotone*
semiring.add-mono)
also have $\dots = x$
by (*metis assms(1,2) one-loop root-var mult-assoc semiring.mult-not-zero*
sup-bot-right)
finally show $\text{root } p \ x \leq x$

qed

lemma *path-compression-invariant-simplify:*

assumes *point w*
and $p^{T+} * w \leq -w$
and $w \neq y$
shows $p[[w]] \neq w$
proof
assume $p[[w]] = w$
hence $w \leq p^{T+} * w$
by (*metis comp-isotone eq-refl star.circ-mult-increasing*)
also have $\dots \leq -w$
by (*simp add: assms(2)*)
finally have $w = \text{bot}$
using *inf.orderE* **by** *fastforce*
thus *False*
using *assms(1,3) le-bot* **by** *force*
qed

end

context *stone-relation-algebra-tarski*
begin

Theorem 5.4 *distinct-points* has been moved to theory *Relation-Algebras*
in entry *Stone-Relation-Algebras*

Back and von Wright's array independence requirements [1]

Theorem 2.2

lemma *put-get-different-vector:*

assumes *vector y w ≤ -y*

shows $(x[y \multimap z])[w] = x[w]$
proof –
have $(x[y \multimap z])[w] = (y^T \sqcap z) * w \sqcup (-y^T \sqcap x^T) * w$
by (*simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have $\dots = z * (w \sqcap y) \sqcup x^T * (w - y)$
by (*metis assms(1) conv-complement covector-inf-comp-3 inf-commute vector-complement-closed*)
also have $\dots = z * (w \sqcap y) \sqcup x^T * w$
by (*simp add: assms(2) inf.absorb1*)
also have $\dots = z * \text{bot} \sqcup x^T * w$
by (*metis assms(2) comp-inf.semiring.mult-zero-right inf.absorb1 inf.sup-monoid.add-assoc p-inf*)
also have $\dots = x^T * w$
by *simp*
finally show *?thesis*
.

qed

lemma *put-get-different*:

assumes *point y point w w \neq y*
shows $(x[y \multimap z])[w] = x[w]$
proof –
have $w \sqcap y = \text{bot}$
using *assms distinct-points by simp*
hence $w \leq -y$
using *pseudo-complement by simp*
thus *?thesis*
by (*simp add: assms(1) assms(2) put-get-different-vector*)

qed

Theorem 2.4

lemma *put-put-different-vector*:

assumes *vector y vector v v \sqcap y = bot*
shows $(x[y \multimap z])[v \multimap w] = (x[v \multimap w])[y \multimap z]$
proof –
have $(x[y \multimap z])[v \multimap w] = (v \sqcap w^T) \sqcup (-v \sqcap y \sqcap z^T) \sqcup (-v \sqcap -y \sqcap x)$
by (*simp add: comp-inf.semiring.distrib-left inf-assoc sup-assoc*)
also have $\dots = (v \sqcap w^T) \sqcup (y \sqcap z^T) \sqcup (-v \sqcap -y \sqcap x)$
by (*metis assms(3) inf-commute inf-import-p p-inf selection-closed-id*)
also have $\dots = (y \sqcap z^T) \sqcup (v \sqcap w^T) \sqcup (-y \sqcap -v \sqcap x)$
by (*simp add: inf-commute sup-commute*)
also have $\dots = (y \sqcap z^T) \sqcup (-y \sqcap v \sqcap w^T) \sqcup (-y \sqcap -v \sqcap x)$
using *assms distinct-points pseudo-complement inf.absorb2 by simp*
also have $\dots = (x[v \multimap w])[y \multimap z]$
by (*simp add: comp-inf.semiring.distrib-left inf-assoc sup-assoc*)
finally show *?thesis*

.

qed

```

lemma put-put-different:
  assumes point y point v v ≠ y
  shows (x[y→z])[v→w] = (x[v→w])[y→z]
  using assms distinct-points put-put-different-vector by blast

end

```

4 Verifying Operations on Disjoint-Set Forests

In this section we verify the make-set, find-set and union-sets operations of disjoint-set forests. We start by introducing syntax for updating arrays in programs. Updating the value at a given array index means updating the whole array.

```

syntax
  -rel-update :: idt ⇒ 'a ⇒ 'a ⇒ 'b com ((2-[-] :=/ -) [70, 65, 65] 61)

```

```

translations
  x[y] := z => (x := (y □ zT) □ (CONST uminus y □ x))

```

The finiteness requirement in the following class is used for proving that the operations terminate.

```

class finite-regular-p-algebra = p-algebra +
  assumes finite-regular: finite { x . regular x }
begin

```

```

abbreviation card-down-regular :: 'a ⇒ nat (-↓ [100] 100)
  where x↓ ≡ card { z . regular z ∧ z ≤ x }

```

end

```

class stone-kleene-relation-algebra-tarski-finite-regular =
  stone-kleene-relation-algebra-tarski + finite-regular-p-algebra
begin

```

4.1 Make-Set

We prove two correctness results about make-set. The first shows that the forest changes only to the extent of making one node the root of a tree. The second result adds that only singleton sets are created.

definition make-set-postcondition p x p0 ≡ x □ p = x * x^T ∧ -x □ p = -x □ p0

```

theorem make-set:
  VARS p
  [ point x ∧ p0 = p ]
  p[x] := x
  [ make-set-postcondition p x p0 ]
  apply vcg-tc-simp

```

by (*simp add: make-set-postcondition-def inf-sup-distrib1 inf-assoc[THEN sym]*
vector-covector[THEN sym])

theorem *make-set-2*:

VARs p
 [*point x* \wedge *p0* = *p* \wedge *p* \leq 1]
p[x] := *x*
 [*make-set-postcondition p x p0* \wedge *p* \leq 1]

proof *vcg-tc*

fix *p*

assume 1: *point x* \wedge *p0* = *p* \wedge *p* \leq 1

show *make-set-postcondition (p[x \mapsto x]) x p0* \wedge *p[x \mapsto x]* \leq 1

proof (*rule conjI*)

show *make-set-postcondition (p[x \mapsto x]) x p0*

using 1 **by** (*simp add: make-set-postcondition-def inf-sup-distrib1*
inf-assoc[THEN sym] vector-covector[THEN sym])

show *p[x \mapsto x]* \leq 1

using 1 **by** (*metis coreflexive-sup-closed dual-order.trans inf.cobounded2*
vector-covector)

qed

qed

The above total-correctness proof allows us to extract a function, which can be used in other implementations below. This is a technique of [10].

lemma *make-set-exists*:

point x \implies $\exists p'$. *make-set-postcondition p' x p*

using *tc-extract-function make-set* **by** *blast*

definition *make-set p x* \equiv (*SOME p'* . *make-set-postcondition p' x p*)

lemma *make-set-function*:

assumes *point x*

and *p' = make-set p x*

shows *make-set-postcondition p' x p*

proof –

let *?P* = $\lambda p'$. *make-set-postcondition p' x p*

have *?P (SOME z . ?P z)*

using *assms(1) make-set-exists* **by** (*meson someI*)

thus *?thesis*

using *assms(2) make-set-def* **by** *auto*

qed

end

4.2 Find-Set

Disjoint-set forests are represented by their parent mapping. It is a forest except each root of a component tree points to itself.

We prove that *find-set* returns the root of the component tree of the

given node.

context *pd-kleene-allegory*
begin

abbreviation *disjoint-set-forest* $p \equiv \text{mapping } p \wedge \text{acyclic } (p - 1)$

end

context *stone-kleene-relation-algebra-tarski*
begin

If two nodes are mutually reachable from each other in a disjoint-set forest, they must be equal.

lemma *forest-mutually-reachable*:

assumes *acyclic* $(p - 1)$ *point* x *point* y $x \leq p^* * y$ $y \leq p^* * x$
shows $x = y$

proof (*rule ccontr*)

assume $1: x \neq y$

hence $2: x \leq -y$

by (*meson* *assms*(2,3) *bijjective-regular dual-order.eq-iff*
point-in-vector-or-complement point-in-vector-or-complement-2)

have $x \leq (p - 1)^* * y$

using *assms*(4) *reachable-without-loops* **by** *auto*

also have $\dots = (p - 1)^+ * y \sqcup y$

by (*simp* *add: star.circ-loop-fixpoint mult-assoc*)

finally have $3: x \leq (p - 1)^+ * y$

using 2 **by** (*metis* *half-shunting inf.orderE*)

have $4: y \leq -x$

using 1 **by** (*meson* *assms*(2,3) *bijjective-regular dual-order.eq-iff*
point-in-vector-or-complement point-in-vector-or-complement-2)

have $y \leq (p - 1)^* * x$

using *assms*(5) *reachable-without-loops* **by** *auto*

also have $\dots = (p - 1)^+ * x \sqcup x$

by (*simp* *add: star.circ-loop-fixpoint mult-assoc*)

finally have $y \leq (p - 1)^+ * x$

using 4 **by** (*metis* *half-shunting inf.orderE*)

also have $\dots \leq (p - 1)^+ * (p - 1)^+ * y$

using 3 **by** (*simp* *add: comp-associative mult-right-isotone*)

also have $\dots \leq (p - 1)^+ * y$

by (*simp* *add: mult-left-isotone plus-transitive*)

finally have $y * y^T \leq (p - 1)^+$

using *assms*(3) *shunt-bijjective* **by** *blast*

also have $\dots \leq -1$

by (*simp* *add: assms*(1))

finally have $y = \text{bot}$

using *inf.absorb-iff1 schroeder-4-p* **by** *auto*

thus *False*

using 1 *assms*(3) *bot-least top-unique* **by** *auto*

qed

lemma *forest-mutually-reachable-2*:
assumes *acyclic* $(p - 1)$ *point* x *point* y $x \leq p^{T^*} * y$ $y \leq p^{T^*} * x$
shows $x = y$
proof –
have $1: x \leq p^* * y$
by (*simp add: assms(2,3,5) bijective-reverse conv-star-commute*)
have $y \leq p^* * x$
by (*simp add: assms(2-4) bijective-reverse conv-star-commute*)
thus *?thesis*
using 1 *assms(1-3) forest-mutually-reachable* **by** *blast*
qed

end

context *stone-kleene-relation-algebra-tarski-finite-regular*
begin

definition *find-set-precondition* p $x \equiv$ *disjoint-set-forest* $p \wedge$ *point* x

definition *find-set-invariant* p x $y \equiv$ *find-set-precondition* p $x \wedge$ *point* $y \wedge y \leq p^{T^*} * x$

definition *find-set-postcondition* p x $y \equiv$ *point* $y \wedge y = \text{root } p$ x

lemma *find-set-1*:

find-set-precondition p $x \implies$ *find-set-invariant* p x

apply (*unfold find-set-invariant-def*)

using *mult-left-isotone star.circ-reflexive find-set-precondition-def* **by** *fastforce*

lemma *find-set-2*:

find-set-invariant p x $y \wedge y \neq p[[y]] \implies$ *find-set-invariant* p x $(p[[y]]) \wedge (p^{T^*} * (p[[y]])) \downarrow < (p^{T^*} * y) \downarrow$

proof –

let $?s = \{ z . \text{regular } z \wedge z \leq p^{T^*} * y \}$

let $?t = \{ z . \text{regular } z \wedge z \leq p^{T^*} * (p[[y]]) \}$

assume $1: \text{find-set-invariant } p$ x $y \wedge y \neq p[[y]]$

have $2: \text{point } (p[[y]])$

using 1 *read-point find-set-invariant-def find-set-precondition-def* **by** *simp*

show *find-set-invariant* p x $(p[[y]]) \wedge \text{card } ?t < \text{card } ?s$

proof (*unfold find-set-invariant-def, intro conjI*)

show *find-set-precondition* p x

using 1 *find-set-invariant-def* **by** *simp*

show *vector* $(p[[y]])$

using 2 **by** *simp*

show *injective* $(p[[y]])$

using 2 **by** *simp*

show *surjective* $(p[[y]])$

using 2 **by** *simp*

show $p[[y]] \leq p^{T^*} * x$

using 1 **by** (*metis (opaque-lifting) find-set-invariant-def comp-associative*)

```

comp-isotone star.circ-increasing star.circ-transitive-equal)
show card ?t < card ?s
proof -
  have p[[y]] = (pT  $\sqcap$  1) * y  $\sqcup$  (pT - 1) * y
    by (metis maddux-3-11-pp mult-right-dist-sup regular-one-closed)
  also have ...  $\leq$  ((p[[y]]  $\sqcap$  y)  $\sqcup$  (pT - 1) * y)
    by (metis comp-left-subdist-inf mult-1-left semiring.add-right-mono)
  also have ... = (pT - 1) * y
    using 1 2 find-set-invariant-def distinct-points by auto
  finally have 3: (pT - 1)* * (p[[y]])  $\leq$  (pT - 1)+ * y
    by (simp add: mult-right-isotone star-simulation-right-equal mult-assoc)
  have pT* * (p[[y]])  $\leq$  pT* * y
    by (metis mult-left-isotone star.right-plus-below-circ mult-assoc)
  hence 4: ?t  $\subseteq$  ?s
    using order-trans by auto
  have 5: y  $\in$  ?s
    using 1 find-set-invariant-def bijective-regular mult-left-isotone
star.circ-reflexive by fastforce
  have 6:  $\neg$  y  $\in$  ?t
  proof
    assume y  $\in$  ?t
    hence y  $\leq$  (pT - 1)+ * y
      using 3 by (metis reachable-without-loops mem-Collect-eq order-trans)
    hence y * yT  $\leq$  (pT - 1)+
      using 1 find-set-invariant-def shunt-bijective by simp
    also have ...  $\leq$  -1
      using 1 by (metis (mono-tags, lifting) find-set-invariant-def
find-set-precondition-def conv-dist-comp conv-dist-inf conv-isotone
conv-star-commute equivalence-one-closed star.circ-plus-same
symmetric-complement-closed)
    finally have y  $\leq$  -y
      using schroeder-4-p by auto
    thus False
      using 1 by (metis find-set-invariant-def comp-inf.coreflexive-idempotent
conv-complement covector-vector-comp inf.absorb1 inf.sup-monoid.add-commute
pseudo-complement surjective-conv-total top.extremum vector-top-closed
regular-closed-top)
  qed
show card ?t < card ?s
  apply (rule psubset-card-mono)
  subgoal using finite-regular by simp
  subgoal using 4 5 6 by auto
done
qed
qed
qed

```

lemma find-set-3:

find-set-invariant p x y \wedge y = p[[y]] \implies *find-set-postcondition* p x y

proof –

assume 1: *find-set-invariant* $p\ x\ y \wedge y = p[[y]]$

show *find-set-postcondition* $p\ x\ y$

proof (*unfold find-set-postcondition-def, rule conjI*)

show *point* y

using 1 *find-set-invariant-def* **by** *simp*

show $y = \text{root } p\ x$

proof (*rule order.antisym*)

have $y * y^T \leq p$

using 1 **by** (*metis find-set-invariant-def find-set-precondition-def shunt-bijective shunt-mapping top-right-mult-increasing*)

hence $y * y^T \leq p \sqcap 1$

using 1 *find-set-invariant-def le-infI* **by** *blast*

hence $y \leq \text{roots } p$

using 1 **by** (*metis find-set-invariant-def order-lesseq-imp shunt-bijective top-right-mult-increasing mult-assoc*)

thus $y \leq \text{root } p\ x$

using 1 *find-set-invariant-def* **by** *simp*

next

have 2: $x \leq p^* * y$

using 1 *find-set-invariant-def find-set-precondition-def bijective-reverse conv-star-commute* **by** *auto*

have $p^T * p^* * y = p^T * p * p^* * y \sqcup (p[[y]])$

by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint*)

also have $\dots \leq p^* * y \sqcup y$

using 1 **by** (*metis find-set-invariant-def find-set-precondition-def comp-isotone mult-left-sub-dist-sup semiring.add-right-mono star.circ-back-loop-fixpoint star.circ-circ-mult star.circ-top star.circ-transitive-equal star-involutive star-one*)

also have $\dots = p^* * y$

by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)

finally have 3: $p^{T*} * x \leq p^* * y$

using 2 **by** (*simp add: comp-associative star-left-induct*)

have $p * y \sqcap \text{roots } p = (p \sqcap 1) * p * y$

using *comp-associative coreflexive-comp-top-inf inf-commute* **by** *auto*

also have $\dots \leq p^T * p * y$

by (*metis inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone one-inf-conv*)

also have $\dots \leq y$

using 1 *find-set-invariant-def find-set-precondition-def mult-left-isotone* **by** *fastforce*

finally have 4: $p * y \leq y \sqcup -\text{roots } p$

using 1 **by** (*metis find-set-invariant-def shunting-p bijective-regular*)

have $p * -\text{roots } p \leq -\text{roots } p$

using 1 **by** (*metis find-set-invariant-def find-set-precondition-def conv-complement-sub-leq conv-involutive roots-successor-loop*)

hence $p * y \sqcup p * -\text{roots } p \leq y \sqcup -\text{roots } p$

using 4 *dual-order.trans le-supI sup-ge2* **by** *blast*

hence $p * (y \sqcup -\text{roots } p) \leq y \sqcup -\text{roots } p$

```

    by (simp add: mult-left-dist-sup)
  hence  $p^* * y \leq y \sqcup -\text{roots } p$ 
    by (simp add: star-left-induct)
  hence  $p^{T^*} * x \leq y \sqcup -\text{roots } p$ 
    using 3 dual-order.trans by blast
  thus  $\text{root } p \ x \leq y$ 
    using 1 by (metis find-set-invariant-def shunting-p bijective-regular)
qed
qed
qed

```

```

theorem find-set:
  VARS  $y$ 
  [ find-set-precondition  $p \ x$  ]
   $y := x$ ;
  WHILE  $y \neq p[[y]]$ 
    INV { find-set-invariant  $p \ x \ y$  }
    VAR {  $(p^{T^*} * y)\downarrow$  }
    DO  $y := p[[y]]$ 
    OD
  [ find-set-postcondition  $p \ x \ y$  ]
apply vcg-tc-simp
  apply (fact find-set-1)
  apply (fact find-set-2)
  by (fact find-set-3)

```

```

lemma find-set-exists:
  find-set-precondition  $p \ x \implies \exists y . \text{find-set-postcondition } p \ x \ y$ 
  using tc-extract-function find-set by blast

```

The root of a component tree is a point, that is, represents a singleton set of nodes. This could be proved from the definitions using Kleene-relation algebraic calculations. But they can be avoided because the property directly follows from the postcondition of the previous correctness proof. The corresponding algorithm shows how to obtain the root. We therefore have an essentially constructive proof of the following result.

Theorem 4.3

```

lemma root-point:
  disjoint-set-forest  $p \implies \text{point } x \implies \text{point } (\text{root } p \ x)$ 
  using find-set-exists find-set-precondition-def find-set-postcondition-def by simp

```

```

definition find-set  $p \ x \equiv (\text{SOME } y . \text{find-set-postcondition } p \ x \ y)$ 

```

```

lemma find-set-function:
  assumes find-set-precondition  $p \ x$ 
  and  $y = \text{find-set } p \ x$ 
  shows find-set-postcondition  $p \ x \ y$ 
  by (metis assms find-set-def find-set-exists someI)

```

4.3 Path Compression

The path-compression technique is frequently implemented in recursive implementations of find-set modifying the tree on the way out from recursive calls. Here we implement it using a second while-loop, which iterates over the same path to the root and changes edges to point to the root of the component, which is known after the while-loop in find-set completes. We prove that path compression preserves the equivalence-relational semantics of the disjoint-set forest and also preserves the roots of the component trees. Additionally we prove the exact effect of path compression.

definition *path-compression-precondition* $p\ x\ y \equiv \text{disjoint-set-forest } p \wedge \text{point } x \wedge \text{point } y \wedge y = \text{root } p\ x$

definition *path-compression-invariant* $p\ x\ y\ p0\ w \equiv$
path-compression-precondition $p\ x\ y \wedge \text{point } w \wedge$
 $p \sqcap 1 = p0 \sqcap 1 \wedge \text{fc } p = \text{fc } p0 \wedge$
 $\text{root } p\ w = y \wedge p0[p0^{T^*} * x - p0^{T^*} * w \longrightarrow y] = p \wedge$
disjoint-set-forest $p0 \wedge w \leq p0^{T^*} * x$

definition *path-compression-postcondition* $p\ x\ y\ p0 \equiv$
path-compression-precondition $p\ x\ y \wedge p \sqcap 1 = p0 \sqcap 1 \wedge \text{fc } p = \text{fc } p0 \wedge$
 $p0[p0^{T^*} * x \longrightarrow y] = p$

We first consider a variant that achieves the effect as a single update. The parents of all nodes reachable from x are simultaneously updated to the root of the component of x .

lemma *path-compression-exact*:

assumes *path-compression-precondition* $p0\ x\ y$
and $p0[p0^{T^*} * x \longrightarrow y] = p$
shows $p \sqcap 1 = p0 \sqcap 1 \wedge \text{fc } p = \text{fc } p0$

proof –

have $a1$: *disjoint-set-forest* $p0$ **and** $a2$: *point* x **and** $a3$: *point* y **and** $a4$: $y = \text{root } p0\ x$

using *path-compression-precondition-def* $assms(1)$ **by** *auto*

have 1 : *regular* $(p0^{T^*} * x)$

using $a1\ a2$ *bijective-regular mapping-regular regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*

have $p \sqcap 1 = (p0^{T^*} * x \sqcap y^T \sqcap 1) \sqcup (-(p0^{T^*} * x) \sqcap p0 \sqcap 1)$

using $assms(2)$ *inf-sup-distrib2* **by** *auto*

also have $\dots = (p0^{T^*} * x \sqcap p0 \sqcap 1) \sqcup (-(p0^{T^*} * x) \sqcap p0 \sqcap 1)$

proof –

have $p0^{T^*} * x \sqcap y^T \sqcap 1 = p0^{T^*} * x \sqcap p0 \sqcap 1$

proof (*rule order.antisym*)

have $(p0 \sqcap 1) * p0^{T^*} * x \sqcap 1 \leq p0$

by (*smt coreflexive-comp-top-inf-one inf.absorb-iff2 inf.cobounded2 inf.sup-monoid.add-assoc root-var*)

hence $p0^{T^*} * x \sqcap y^T \sqcap 1 \leq p0$

by (*metis inf-le1 a4 conv-dist-inf coreflexive-symmetric inf.absorb2 inf.cobounded2 inf.sup-monoid.add-assoc root-var symmetric-one-closed*)

thus $p0^{T^*} * x \sqcap y^T \sqcap 1 \leq p0^{T^*} * x \sqcap p0 \sqcap 1$

by (*meson inf.le-sup-iff order.refl*)
 have $p0^{T^*} * x \sqcap p0 \sqcap 1 \leq y$
 by (*metis a4 coreflexive-comp-top-inf-one inf.cobounded1 inf-assoc inf-le2*)
 thus $p0^{T^*} * x \sqcap p0 \sqcap 1 \leq p0^{T^*} * x \sqcap y^T \sqcap 1$
 by (*smt conv-dist-inf coreflexive-symmetric inf.absorb-iff2 inf.cobounded2*
inf.sup-monoid.add-assoc)
 qed
 thus ?thesis
 by *simp*
 qed
 also have $\dots = p0 \sqcap 1$
 using 1 by (*metis inf.sup-monoid.add-commute inf-sup-distrib1*
maddux-3-11-pp)
 finally show $p \sqcap 1 = p0 \sqcap 1$
 .
 show *fc p = fc p0*
 proof (*rule order.antisym*)
 have 2: *univalent* ($p0[p0^{T^*} * x \mapsto y]$)
 by (*simp add: a1 a2 a3 update-univalent mult-assoc*)
 have 3: $-(p0^{T^*} * x) \sqcap p0 \leq (p0[p0^{T^*} * x \mapsto y])^* * (p0[p0^{T^*} * x \mapsto y])^{T^*}$
 using *fc-increasing inf.order-trans sup.cobounded2* by *blast*
 have $p0^{T^*} * x \sqcap p0 \leq (p0^{T^*} \sqcap p0 * x^T) * (x \sqcap p0^* * p0)$
 by (*metis conv-involutive conv-star-commute dedekind*)
 also have $\dots \leq p0^{T^*} * x \sqcap p0 * x^T * p0^* * p0$
 by (*metis comp-associative inf.boundedI inf.cobounded2 inf-le1 mult-isotone*)
 also have $\dots \leq p0^{T^*} * x \sqcap top * x^T * p0^*$
 using *comp-associative comp-inf.mult-right-isotone mult-isotone*
star.right-plus-below-circ by *auto*
 also have $\dots = p0^{T^*} * x * x^T * p0^*$
 by (*metis a2 symmetric-top-closed vector-covector vector-inf-comp*
vector-mult-closed)
 also have $\dots \leq (p0^{T^*} * x * y^T) * (y * x^T * p0^*)$
 by (*metis a3 order.antisym comp-inf.top-right-mult-increasing*
conv-involutive dedekind-1 inf.sup-left-divisibility inf.sup-monoid.add-commute
mult-right-isotone surjective-conv-total mult-assoc)
 also have $\dots = (p0^{T^*} * x \sqcap y^T) * (y \sqcap x^T * p0^*)$
 by (*metis a2 a3 vector-covector vector-inf-comp vector-mult-closed*)
 also have $\dots = (p0^{T^*} * x \sqcap y^T) * (p0^{T^*} * x \sqcap y^T)^T$
 by (*simp add: conv-dist-comp conv-dist-inf conv-star-commute inf-commute*)
 also have $\dots \leq (p0[p0^{T^*} * x \mapsto y])^* * (p0[p0^{T^*} * x \mapsto y])^{T^*}$
 by (*meson conv-isotone dual-order.trans mult-isotone star.circ-increasing*
sup.cobounded1)
 finally have $p0^{T^*} * x \sqcap p0 \leq (p0[p0^{T^*} * x \mapsto y])^* * (p0[p0^{T^*} * x \mapsto y])^{T^*}$
 .
 hence $(p0^{T^*} * x \sqcap p0) \sqcup (-(p0^{T^*} * x) \sqcap p0) \leq (p0[p0^{T^*} * x \mapsto y])^* * (p0[p0^{T^*} * x \mapsto y])^{T^*}$
 using 3 *le-supI* by *blast*
 hence $p0 \leq (p0[p0^{T^*} * x \mapsto y])^* * (p0[p0^{T^*} * x \mapsto y])^{T^*}$
 using 1 by (*metis inf-commute maddux-3-11-pp*)

hence $fc\ p0 \leq (p0[p0^{T*} * x \longrightarrow y])^* * (p0[p0^{T*} * x \longrightarrow y])^{T*}$
using 2 *fc-idempotent fc-isotone* **by** *fastforce*
thus $fc\ p0 \leq fc\ p$
by (*simp add: assms(2)*)
have $((p0^{T*} * x \sqcap y^T) \sqcup (-(p0^{T*} * x) \sqcap p0))^* = (-(p0^{T*} * x) \sqcap p0)^* * ((p0^{T*} * x \sqcap y^T) \sqcup 1)$
proof (*rule star-sup-2*)
have 4: *transitive* ($p0^{T*} * x$)
using a2 *comp-associative mult-right-isotone rectangle-star-rectangle* **by** *auto*
have *transitive* (y^T)
by (*metis a3 conv-dist-comp inf.eq-refl mult-assoc*)
thus *transitive* ($p0^{T*} * x \sqcap y^T$)
using 4 *transitive-inf-closed* **by** *auto*
have 5: $p0^{T*} * x * (-(p0^{T*} * x) \sqcap p0) \leq p0^{T*} * x$
by (*metis a2 mult-right-isotone top-greatest mult-assoc*)
have $(-(p0^{T*} * x) \sqcap p0)^T * y \leq p0^T * y$
by (*simp add: conv-dist-inf mult-left-isotone*)
also have $\dots \leq y$
using a1 a4 *root-successor-loop* **by** *auto*
finally have $y^T * (-(p0^{T*} * x) \sqcap p0) \leq y^T$
using *conv-dist-comp conv-isotone* **by** *fastforce*
thus $(p0^{T*} * x \sqcap y^T) * (-(p0^{T*} * x) \sqcap p0) \leq p0^{T*} * x \sqcap y^T$
using 5 *comp-left-subdist-inf inf-mono order-trans* **by** *blast*
qed
hence $p^* = (-(p0^{T*} * x) \sqcap p0)^* * ((p0^{T*} * x \sqcap y^T) \sqcup 1)$
by (*simp add: assms(2)*)
also have $\dots \leq p0^* * ((p0^{T*} * x \sqcap y^T) \sqcup 1)$
by (*simp add: mult-left-isotone star-isotone*)
also have $\dots = p0^* * (p0^{T*} * x * y^T \sqcup 1)$
by (*simp add: a2 a3 vector-covector vector-mult-closed*)
also have $\dots = p0^* * (p0^{T*} * (x * x^T) * p0^* * (p0 \sqcap 1) \sqcup 1)$
by (*metis a4 coreflexive-symmetric inf.cobounded2 root-var comp-associative conv-dist-comp conv-involutive conv-star-commute*)
also have $\dots \leq p0^* * (p0^{T*} * 1 * p0^* * (p0 \sqcap 1) \sqcup 1)$
by (*metis a2 mult-left-isotone mult-right-isotone semiring.add-left-mono sup-commute*)
also have $\dots = p0^* * (p0^{T*} * (p0 \sqcap 1) \sqcup p0^* * (p0 \sqcap 1) \sqcup 1)$
by (*simp add: a1 cancel-separate-eq mult-right-dist-sup*)
also have $\dots = p0^* * ((p0 \sqcap 1) \sqcup p0^* * (p0 \sqcap 1) \sqcup 1)$
by (*smt univalent-root-successors a1 conv-dist-comp conv-dist-inf coreflexive-idempotent coreflexive-symmetric inf.cobounded2 injective-codomain loop-root root-transitive-successor-loop symmetric-one-closed*)
also have $\dots = p0^* * (p0^* * (p0 \sqcap 1) \sqcup 1)$
by (*metis inf.sup-left-divisibility inf-commute sup.left-idem sup-commute sup-relative-same-increasing*)
also have $\dots \leq p0^* * p0^*$
by (*metis inf.cobounded2 inf-commute order.refl order-lesseq-imp star.circ-mult-upper-bound star.circ-reflexive star.circ-transitive-equal*)

```

sup.boundedI sup-monoid.add-commute)
  also have ... = p0*
    by (simp add: star.circ-transitive-equal)
  finally show fc p ≤ fc p0
    by (metis conv-order conv-star-commute mult-isotone)
qed
qed

lemma update-acyclic-6:
  assumes disjoint-set-forest p
    and point x
  shows acyclic ((p[pT**x]→root p x]) - 1)
  using assms root-point root-successor-loop update-acyclic-2 by auto

theorem path-compression-assign:
  VARS p
  [ path-compression-precondition p x y ∧ p0 = p ]
  p[pT**x] := y
  [ path-compression-postcondition p x y p0 ]
  apply vcg-tc-simp
  apply (unfold path-compression-precondition-def
path-compression-postcondition-def)
  apply (intro conjI)
  subgoal using update-univalent mult-assoc by auto
  subgoal using bijective-regular mapping-regular regular-closed-star
regular-conv-closed regular-mult-closed update-mapping mult-assoc by auto
  subgoal using update-acyclic-6 by blast
  subgoal by blast
  subgoal by blast
  subgoal by blast
  subgoal by blast
  subgoal by blast
  subgoal by (smt same-root path-compression-exact
path-compression-precondition-def update-univalent vector-mult-closed)
  subgoal using path-compression-exact(1) path-compression-precondition-def
by blast
  subgoal using path-compression-exact(2) path-compression-precondition-def
by blast
  by blast
  by blast

```

We next look at implementing these updates using a loop.

```

lemma path-compression-1a:
  assumes point x
    and disjoint-set-forest p
    and x ≠ root p x
  shows pT+*x ≤ - x
  by (meson assms bijective-regular mapping-regular regular-closed-star
regular-conv-closed regular-mult-closed vector-mult-closed)

```


point-in-vector-or-complement-2 loop-root-2)

lemma *path-compression-1b*:

$$x \leq p^{T^*} * x$$

using *mult-left-isotone star.circ-reflexive* **by** *fastforce*

lemma *path-compression-1*:

path-compression-precondition $p \ x \ y \implies$ *path-compression-invariant* $p \ x \ y \ p \ x$

using *path-compression-invariant-def path-compression-precondition-def*

loop-root path-compression-1a path-compression-1b **by** *auto*

lemma *path-compression-2*:

path-compression-invariant $p \ x \ y \ p0 \ w \ \wedge \ y \neq p[[w]] \implies$

path-compression-invariant $(p[w \rightarrow y]) \ x \ y \ p0 \ (p[[w]]) \ \wedge \ ((p[w \rightarrow y])^{T^*} * (p[[w]])) \downarrow < (p^{T^*} * w) \downarrow$

proof *-*

let $?p = p[w \rightarrow y]$

let $?s = \{ z . \text{regular } z \ \wedge \ z \leq p^{T^*} * w \}$

let $?t = \{ z . \text{regular } z \ \wedge \ z \leq ?p^{T^*} * (p[[w]]) \}$

assume 1 : *path-compression-invariant* $p \ x \ y \ p0 \ w \ \wedge \ y \neq p[[w]]$

have $i1$: *disjoint-set-forest* p **and** $i2$: *point* x **and** $i3$: *point* y **and** $i4$: $y = \text{root } p \ x$

using 1 *path-compression-invariant-def path-compression-precondition-def* **by** *meson+*

have $i5$: *point* w

and $i8$: $p \sqcap 1 = p0 \sqcap 1$ **and** $i9$: $fc \ p = fc \ p0$

and $i10$: $\text{root } p \ w = y$ **and** $i12$: $p0[p0^{T^*} * x - p0^{T^*} * w \rightarrow y] = p$

using 1 *path-compression-invariant-def* **by** *blast+*

have $i13$: *disjoint-set-forest* $p0$ **and** $i15$: $w \leq p0^{T^*} * x$

using 1 *path-compression-invariant-def* **by** *auto*

have $i6$: $y \leq p^{T^*} * w$

using $i10$ **by** *force*

have $i11$: $p[[w]] = p0[[w]]$

by (*smt (verit) i12 i2 i5 dual-order.trans inf-le2 p-antitone-iff put-get-different-vector vector-complement-closed vector-inf-closed vector-mult-closed path-compression-1b*)

have $i14$: $y = \text{root } p0 \ x$

using $i1 \ i13 \ i4 \ i8 \ i9$ *same-root* **by** *blast*

have 2 : *point* $(p[[w]])$

using $i1 \ i5$ *read-point* **by** *blast*

show *path-compression-invariant* $?p \ x \ y \ p0 \ (p[[w]]) \ \wedge \ \text{card } ?t < \text{card } ?s$

proof (*unfold path-compression-invariant-def, intro conjI*)

have 3 : *mapping* $?p$

by (*simp add: i1 i3 i5 bijective-regular update-total update-univalent*)

have 4 : $w \neq y$

using $1 \ i1 \ i4$ *root-successor-loop* **by** *blast*

hence 5 : $w \sqcap y = \text{bot}$

by (*simp add: i3 i5 distinct-points*)

hence $y * w^T \leq -1$

using *pseudo-complement schroeder-4-p* by auto
 hence $y * w^T \leq p^{T^*} - 1$
 using *i5 i6 shunt-bijective* by auto
 also have $\dots \leq p^{T^+}$
 by (*simp add: star-plus-without-loops*)
 finally have 6: $y \leq p^{T^+} * w$
 using *i5 shunt-bijective* by auto
 have 7: $w * w^T \leq -p^{T^+}$
 proof (rule *ccontr*)
 assume $\neg w * w^T \leq -p^{T^+}$
 hence $w * w^T \leq --p^{T^+}$
 using *i5 point-arc arc-in-partition* by blast
 hence $w * w^T \leq p^{T^+} \sqcap 1$
 using *i1 i5 mapping-regular regular-conv-closed regular-closed-star*
regular-mult-closed by *simp*
 also have $\dots = ((p^T \sqcap 1) * p^{T^*} \sqcap 1) \sqcup ((p^T - 1) * p^{T^*} \sqcap 1)$
 by (*metis comp-inf.mult-right-dist-sup maddux-3-11-pp mult-right-dist-sup*
regular-one-closed)
 also have $\dots = ((p^T \sqcap 1) * p^{T^*} \sqcap 1) \sqcup ((p - 1)^+ \sqcap 1)^T$
 by (*metis conv-complement conv-dist-inf conv-plus-commute*
equivalence-one-closed reachable-without-loops)
 also have $\dots \leq ((p^T \sqcap 1) * p^{T^*} \sqcap 1) \sqcup (-1 \sqcap 1)^T$
 by (*metis (no-types, opaque-lifting) i1 sup-right-isotone inf.sup-left-isotone*
conv-isotone)
 also have $\dots = (p^T \sqcap 1) * p^{T^*} \sqcap 1$
 by *simp*
 also have $\dots \leq (p^T \sqcap 1) * top \sqcap 1$
 by (*metis comp-inf.comp-isotone coreflexive-comp-top-inf*
equivalence-one-closed inf.cobounded1 inf.cobounded2)
 also have $\dots \leq p^T$
 by (*simp add: coreflexive-comp-top-inf-one*)
 finally have $w * w^T \leq p^T$
 by *simp*
 hence $w \leq p[[w]]$
 using *i5 shunt-bijective* by blast
 hence $w = p[[w]]$
 using 2 by (*metis i5 epm-3 mult-semi-associative*)
 thus *False*
 using 2 4 *i10 loop-root* by auto
 qed
 have 10: *acyclic* (?*p* - 1)
 using *i1 i10 i3 i5 inf-le1 update-acyclic-3* by blast
 have ?*p*[[$p^{T^+} * w$]] $\leq p^{T^+} * w$
 proof -
 have $(w^T \sqcap y) * p^{T^+} * w = y \sqcap w^T * p^{T^+} * w$
 by (*metis i3 inf-vector-comp vector-inf-comp*)
 hence ?*p*[[$p^{T^+} * w$]] = $(y \sqcap w^T * p^{T^+} * w) \sqcup (-w^T \sqcap p^T) * p^{T^+} * w$
 by (*simp add: comp-associative conv-complement conv-dist-inf*
conv-dist-sup mult-right-dist-sup)

also have $\dots \leq y \sqcup (-w^T \sqcap p^T) * p^{T+} * w$
using *sup-left-isotone* **by** *auto*
also have $\dots \leq y \sqcup p^T * p^{T+} * w$
using *mult-left-isotone sup-right-isotone* **by** *auto*
also have $\dots \leq y \sqcup p^{T+} * w$
using *semiring.add-left-mono mult-left-isotone mult-right-isotone*
star.left-plus-below-circ **by** *auto*
also have $\dots = p^{T+} * w$
using *6* **by** (*simp add: sup-absorb2*)
finally show *?thesis*
by *simp*
qed
hence *11*: $?p^{T*} * (p[[w]]) \leq p^{T+} * w$
using *star-left-induct* **by** (*simp add: mult-left-isotone*
star.circ-mult-increasing)
have *13*: $?p[[x]] = y$
proof (*cases w = x*)
case *True*
hence $?p[[x]] = (w^T \sqcap y) * w \sqcup (-w^T \sqcap p^T) * w$
by (*simp add: conv-complement conv-dist-inf conv-dist-sup*
mult-right-dist-sup)
also have $\dots = (w^T \sqcap y) * w \sqcup p^T * (-w \sqcap w)$
by (*metis i5 conv-complement covector-inf-comp-3*
inf.sup-monoid.add-commute vector-complement-closed)
also have $\dots = (w^T \sqcap y) * w$
by *simp*
also have $\dots = y * w$
by (*simp add: i5 covector-inf-comp-3 inf.sup-monoid.add-commute*)
also have $\dots = y$
by (*metis i3 i5 comp-associative*)
finally show *?thesis*
·
next
case *False*
hence $\neg x \leq p0^{T*} * w$
using *forest-mutually-reachable-2 i13 i15 i2 i5* **by** *blast*
hence $x \leq - p0^{T*} * w$
by (*metis (mono-tags, lifting) i13 i2 i5 comp-bijective-complement*
mapping-regular point-in-vector-or-complement regular-closed-star
regular-conv-closed vector-mult-closed)
hence $x \leq p0^{T*} * x - p0^{T*} * w$
by (*simp add: i5 comp-bijective-complement path-compression-1b*)
hence $p[[x]] = y$
by (*smt (verit) i12 i2 i3 i5 comp-bijective-complement put-get-sub*
vector-inf-comp vector-mult-closed)
thus $?p[[x]] = y$
using *False i2 i5 put-get-different* **by** *blast*
qed
have *14*: $?p^{T*} * x = x \sqcup y$

```

proof (rule order.antisym)
  have  $?p^T * (x \sqcup y) = y \sqcup ?p^T * y$ 
    using 13 by (simp add: mult-left-dist-sup)
  also have  $\dots = y \sqcup (w^T \sqcap y) * y \sqcup (-w^T \sqcap p^T) * y$ 
    by (simp add: conv-complement conv-dist-inf conv-dist-sup
mult-right-dist-sup sup-assoc)
  also have  $\dots \leq y \sqcup (w^T \sqcap y) * y \sqcup p^T * y$ 
    using mult-left-isotone sup-right-isotone by auto
  also have  $\dots = y \sqcup (w^T \sqcap y) * y$ 
    using i1 i10 root-successor-loop sup-commute by auto
  also have  $\dots \leq y \sqcup y * y$ 
    using mult-left-isotone sup-right-isotone by auto
  also have  $\dots = y$ 
    by (metis i3 comp-associative sup.idem)
  also have  $\dots \leq x \sqcup y$ 
    by simp
  finally show  $?p^{T*} * x \leq x \sqcup y$ 
    by (simp add: star-left-induct)
next
  show  $x \sqcup y \leq ?p^{T*} * x$ 
    using 13 by (metis mult-left-isotone star.circ-increasing
star.circ-loop-fixpoint sup.boundedI sup-ge2)
qed
have 15:  $y = \text{root } ?p \ x$ 
proof -
  have  $(p \sqcap 1) * y = (p \sqcap 1) * (p \sqcap 1) * p^{T*} * x$ 
    by (simp add: i4 comp-associative root-var)
  also have  $\dots = (p \sqcap 1) * p^{T*} * x$ 
    using coreflexive-idempotent by auto
  finally have 16:  $(p \sqcap 1) * y = y$ 
    by (simp add: i4 root-var)
  have 17:  $(p \sqcap 1) * x \leq y$ 
    by (metis (no-types, lifting) i4 comp-right-one mult-left-isotone
mult-right-isotone star.circ-reflexive root-var)
  have  $\text{root } ?p \ x = (?p \sqcap 1) * (x \sqcup y)$ 
    using 14 by (metis mult-assoc root-var)
  also have  $\dots = (w \sqcap y^T \sqcap 1) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$ 
    by (simp add: inf-sup-distrib2 semiring.distrib-right)
  also have  $\dots = (w \sqcap 1 \sqcap y^T) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$ 
    by (simp add: inf.left-commute inf.sup-monoid.add-commute)
  also have  $\dots = (w \sqcap 1) * (y \sqcap (x \sqcup y)) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$ 
    by (simp add: i3 covector-inf-comp-3)
  also have  $\dots = (w \sqcap 1) * y \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$ 
    by (simp add: inf.absorb1)
  also have  $\dots = (w \sqcap 1 * y) \sqcup (-w \sqcap (p \sqcap 1) * (x \sqcup y))$ 
    by (simp add: i5 inf-assoc vector-complement-closed vector-inf-comp)
  also have  $\dots = (w \sqcap y) \sqcup (-w \sqcap ((p \sqcap 1) * x \sqcup y))$ 
    using 16 by (simp add: mult-left-dist-sup)
  also have  $\dots = (w \sqcap y) \sqcup (-w \sqcap y)$ 

```

```

    using 17 by (simp add: sup.absorb2)
  also have ... = y
    using 5 inf.sup-monoid.add-commute le-iff-inf pseudo-complement
sup-monoid.add-0-left by fastforce
  finally show ?thesis
    by simp
qed
show path-compression-precondition ?p x y
  using 3 10 15 i2 i3 path-compression-precondition-def by blast
show vector (p[[w]])
  using 2 by simp
show injective (p[[w]])
  using 2 by simp
show surjective (p[[w]])
  using 2 by simp
have  $w \sqcap p \sqcap 1 \leq w \sqcap w^T \sqcap p$ 
  by (metis inf.boundedE inf.boundedI inf.cobounded1 inf.cobounded2
one-inf-conv)
  also have ... =  $w * w^T \sqcap p$ 
    by (simp add: i5 vector-covector)
  also have ...  $\leq -p^{T+} \sqcap p$ 
    using 7 by (simp add: inf.coboundedI2 inf.sup-monoid.add-commute)
  finally have  $w \sqcap p \sqcap 1 = \text{bot}$ 
    by (metis (no-types, opaque-lifting) conv-dist-inf coreflexive-symmetric
inf.absorb1 inf.boundedE inf.cobounded2 pseudo-complement
star.circ-mult-increasing)
  also have  $w \sqcap y^T \sqcap 1 = \text{bot}$ 
    using 5 antisymmetric-bot-closed asymmetric-bot-closed comp-inf.schroeder-2
inf.absorb1 one-inf-conv by fastforce
  finally have  $w \sqcap p \sqcap 1 = w \sqcap y^T \sqcap 1$ 
    by simp
  thus 18:  $?p \sqcap 1 = p0 \sqcap 1$ 
    by (metis i5 i8 bijective-regular inf.sup-monoid.add-commute inf-sup-distrib2
maddux-3-11-pp)
  show 19:  $fc ?p = fc p0$ 
  proof -
    have  $p[[w]] = p^T * (w \sqcap p^* * y)$ 
      by (metis i3 i5 i6 bijective-reverse conv-star-commute inf.absorb1)
    also have ... =  $p^T * (w \sqcap p^*) * y$ 
      by (simp add: i5 vector-inf-comp mult-assoc)
    also have ... =  $p^T * ((w \sqcap 1) \sqcup (w \sqcap p) * (-w \sqcap p)^*) * y$ 
      by (simp add: i5 omit-redundant-points)
    also have ... =  $p^T * (w \sqcap 1) * y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^* * y$ 
      by (simp add: comp-associative mult-left-dist-sup mult-right-dist-sup)
    also have ...  $\leq p^T * y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^* * y$ 
      by (metis semiring.add-right-mono comp-isotone order.eq-iff
inf.cobounded1 inf.sup-monoid.add-commute mult-1-right)
    also have ... =  $y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^* * y$ 
      using i1 i4 root-successor-loop by auto

```

also have $\dots \leq y \sqcup p^T * p * (-w \sqcap p)^* * y$
using *comp-isotone sup-right-isotone* **by** *auto*
also have $\dots \leq y \sqcup (-w \sqcap p)^* * y$
by (*metis i1 comp-associative eq-refl shunt-mapping sup-right-isotone*)
also have $\dots = (-w \sqcap p)^* * y$
by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)
finally have 20: $p[[w]] \leq (-w \sqcap p)^* * y$
by *simp*
have $p^T * (-w \sqcap p)^* * y = p^T * y \sqcup p^T * (-w \sqcap p) * (-w \sqcap p)^* * y$
by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint sup-commute*)
also have $\dots = y \sqcup p^T * (-w \sqcap p) * (-w \sqcap p)^* * y$
using *i1 i4 root-successor-loop* **by** *auto*
also have $\dots \leq y \sqcup p^T * p * (-w \sqcap p)^* * y$
using *comp-isotone sup-right-isotone* **by** *auto*
also have $\dots \leq y \sqcup (-w \sqcap p)^* * y$
by (*metis i1 comp-associative eq-refl shunt-mapping sup-right-isotone*)
also have $\dots = (-w \sqcap p)^* * y$
by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)
finally have 21: $p^{T*} * p^T * w \leq (-w \sqcap p)^* * y$
using 20 **by** (*simp add: comp-associative star-left-induct*)
have $w^T \sqcap p^T = p^T * (w^T \sqcap 1)$
by (*metis i5 comp-right-one covector-inf-comp-3*)
inf.sup-monoid.add-commute one-inf-conv
also have $\dots \leq p[[w]]$
by (*metis comp-right-subdist-inf inf.boundedE inf.sup-monoid.add-commute one-inf-conv*)
also have $\dots \leq p^{T*} * p^T * w$
by (*simp add: mult-left-isotone star.circ-mult-increasing-2*)
also have $\dots \leq (-w \sqcap p)^* * y$
using 21 **by** *simp*
finally have $w \sqcap p \leq y^T * (-w \sqcap p)^{T*}$
by (*metis conv-dist-comp conv-dist-inf conv-involutive conv-isotone conv-star-commute*)
hence $w \sqcap p \leq (w \sqcap y^T) * (-w \sqcap p)^{T*}$
by (*simp add: i5 vector-inf-comp*)
also have $\dots \leq (w \sqcap y^T) * ?p^{T*}$
by (*simp add: conv-isotone mult-right-isotone star-isotone*)
also have $\dots \leq ?p * ?p^{T*}$
by (*simp add: mult-left-isotone*)
also have $\dots \leq fc ?p$
by (*simp add: mult-left-isotone star.circ-increasing*)
finally have 22: $w \sqcap p \leq fc ?p$
by *simp*
have $-w \sqcap p \leq ?p$
by *simp*
also have $\dots \leq fc ?p$
by (*simp add: fc-increasing*)
finally have $(w \sqcup -w) \sqcap p \leq fc ?p$

using 22 **by** (*simp add: comp-inf.semiring.distrib-left*
inf.sup-monoid.add-commute)
hence $p \leq fc \ ?p$
by (*metis i5 bijective-regular inf.sup-monoid.add-commute inf-sup-distrib1*
maddux-3-11-pp)
hence 23: $fc \ p \leq fc \ ?p$
using 3 *fc-idempotent fc-isotone* **by** *fastforce*
have $?p \leq (w \sqcap y^T) \sqcup p$
using *sup-right-isotone* **by** *auto*
also have $\dots = w * y^T \sqcup p$
by (*simp add: i3 i5 vector-covector*)
also have $\dots \leq p^* \sqcup p$
by (*smt i5 i6 conv-dist-comp conv-involutive conv-isotone*
conv-star-commute le-supI shunt-bijective star.circ-increasing sup-absorb1)
also have $\dots \leq fc \ p$
using *fc-increasing star.circ-back-loop-prefixpoint* **by** *auto*
finally have $fc \ ?p \leq fc \ p$
using *i1 fc-idempotent fc-isotone* **by** *fastforce*
thus *?thesis*
using 23 *i9* **by** *auto*
qed
have 24: $root \ ?p \ (p[[w]]) = root \ p0 \ (p[[w]])$
using 3 18 19 *i13 same-root* **by** *blast*
also have $\dots = root \ p0 \ (p0[[w]])$
by (*simp add: i11*)
also have 25: $\dots = root \ p0 \ w$
by (*metis i5 i13 conv-involutive forest-components-increasing*
mult-left-isotone shunt-bijective injective-mult-closed read-surjective
same-component-same-root)
finally show 26: $root \ ?p \ (p[[w]]) = y$
by (*metis i1 i10 i13 i8 i9 same-root*)
show *univalent p0 total p0 acyclic (p0 - 1)*
by (*simp-all add: i13*)
show $p[[w]] \leq p0^{T^*} * x$
by (*metis i11 i15 mult-isotone star.circ-increasing star.circ-transitive-equal*
mult-assoc)
let $?q = p0[p0^{T^*} * x - p0^{T^*} * (p[[w]]) \dashv\rightarrow y]$
show $?q = ?p$
proof –
have 27: $w \sqcup p0^{T^+} * w = p0^{T^*} * w$
using *comp-associative star.circ-loop-fixpoint sup-commute* **by** *auto*
hence 28: $p0^{T^+} * w = p0^{T^*} * w - w$
using 4 24 25 26 **by** (*metis i11 i13 i5 inf.orderE maddux-3-13*
path-compression-1a)
hence $p0^{T^*} * (p[[w]]) \leq -w$
by (*metis i11 inf-le2 star-plus mult.assoc*)
hence $w \leq -(p0^{T^*} * (p[[w]]))$
by (*simp add: p-antitone-iff*)
hence $w \leq p0^{T^*} * x - p0^{T^*} * (p[[w]])$

by (*simp add: i15*)
 hence 29: $?q \sqcap w = ?p \sqcap w$
 by (*metis update-inf update-inf-same*)
 have 30: $?q \sqcap p0^{T+} * w = ?p \sqcap p0^{T+} * w$
proof –
 have $?q \sqcap p0^{T+} * w = p0 \sqcap p0^{T+} * w$
 by (*metis i11 comp-associative inf.cobounded2 p-antitone-iff star.circ-plus-same update-inf-different*)
 also have $\dots = p \sqcap p0^{T+} * w$
 using 28 by (*metis i12 inf.cobounded2 inf.sup-monoid.add-assoc p-antitone-iff update-inf-different*)
 also have $\dots = ?p \sqcap p0^{T+} * w$
 using 28 by (*simp add: update-inf-different*)
 finally show *?thesis*
 .
qed
 have 31: $?q \sqcap p0^{T*} * w = ?p \sqcap p0^{T*} * w$
 using 27 29 30 by (*metis inf-sup-distrib1*)
 have 32: $?q \sqcap (p0^{T*} * x - p0^{T*} * w) = ?p \sqcap (p0^{T*} * x - p0^{T*} * w)$
proof –
 have $p0^{T*} * x - p0^{T*} * w \leq p0^{T*} * x - p0^{T*} * (p[[w]])$
 using 28 by (*metis i11 inf.sup-right-isotone mult.semigroup-axioms p-antitone-inf star-plus semigroup.assoc*)
 hence $?q \sqcap (p0^{T*} * x - p0^{T*} * w) = y^T \sqcap p0^{T*} * x \sqcap -(p0^{T*} * w)$
 by (*metis inf-assoc update-inf*)
 also have $\dots = p \sqcap (p0^{T*} * x - p0^{T*} * w)$
 by (*metis i12 inf-assoc update-inf-same*)
 also have $\dots = ?p \sqcap (p0^{T*} * x - p0^{T*} * w)$
 by (*simp add: inf.coboundedI2 p-antitone path-compression-1b inf-assoc update-inf-different*)
 finally show *?thesis*
 .
qed
 have $p0^{T*} * w \sqcup (p0^{T*} * x - p0^{T*} * w) = p0^{T*} * x$
proof –
 have 33: *regular* ($p0^{T*} * w$)
 using i13 i5 *bijjective-regular mapping-regular regular-closed-star regular-conv-closed regular-mult-closed* by *auto*
 have $p0^{T*} * w \leq p0^{T*} * x$
 by (*metis i15 comp-associative mult-right-isotone star.circ-transitive-equal*)
 hence $p0^{T*} * w \sqcup (p0^{T*} * x - p0^{T*} * w) = p0^{T*} * x \sqcap (p0^{T*} * w \sqcup -(p0^{T*} * w))$
 by (*simp add: comp-inf.semiring.distrib-left inf.absorb2*)
 also have $\dots = p0^{T*} * x$
 using 33 by (*metis inf-sup-distrib1 maddux-3-11-pp*)
 finally show *?thesis*
 .
qed

hence 34: $?q \sqcap p0^{T^*} * x = ?p \sqcap p0^{T^*} * x$
using 31 32 by (*metis inf-sup-distrib1*)
have 35: *regular* ($p0^{T^*} * x$)
using i13 i2 *bijjective-regular mapping-regular regular-closed-star*
regular-conv-closed regular-mult-closed by auto
have $-(p0^{T^*} * x) \leq -w$
by (*simp add: i15 p-antitone*)
hence $?q - p0^{T^*} * x = ?p - p0^{T^*} * x$
by (*metis i12 p-antitone-inf update-inf-different*)
thus *?thesis*
using 34 35 by (*metis maddux-3-11-pp*)
qed
show *card ?t < card ?s*
proof –
have $?p^T * p^{T^*} * w = (w^T \sqcap y) * p^{T^*} * w \sqcup (-w^T \sqcap p^T) * p^{T^*} * w$
by (*simp add: conv-complement conv-dist-inf conv-dist-sup*)
mult-right-dist-sup
also have $\dots \leq (w^T \sqcap y) * p^{T^*} * w \sqcup p^T * p^{T^*} * w$
using *mult-left-isotone sup-right-isotone by auto*
also have $\dots \leq (w^T \sqcap y) * p^{T^*} * w \sqcup p^{T^*} * w$
using *mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast*
also have $\dots \leq y * p^{T^*} * w \sqcup p^{T^*} * w$
using *semiring.add-right-mono mult-left-isotone by auto*
also have $\dots \leq y * top \sqcup p^{T^*} * w$
by (*simp add: comp-associative le-supI1 mult-right-isotone*)
also have $\dots = p^{T^*} * w$
by (*simp add: i3 i6 sup-absorb2*)
finally have $?p^{T^*} * p^T * w \leq p^{T^*} * w$
using 11 by (*metis dual-order.trans star.circ-loop-fixpoint sup-commute*)
sup-ge2 mult-assoc
hence 36: $?t \subseteq ?s$
using *order-lesseq-imp mult-assoc by auto*
have 37: $w \in ?s$
by (*simp add: i5 bijjective-regular path-compression-1b*)
have 38: $\neg w \in ?t$
proof
assume $w \in ?t$
hence 39: $w \leq (?p^T - 1)^* * (p[[w]])$
using *reachable-without-loops by auto*
hence $p[[w]] \leq (?p - 1)^* * w$
using 2 by (*smt i5 bijjective-reverse conv-star-commute*)
reachable-without-loops
also have $\dots \leq p^* * w$
proof –
have $p^{T^*} * y = y$
using *i1 i4 root-transitive-successor-loop by auto*
hence $y^T * p^* * w = y^T * w$
by (*metis conv-dist-comp conv-involutive conv-star-commute*)
also have $\dots = bot$

```

    using 5 by (metis i5 inf.idem inf.sup-monoid.add-commute
mult-left-zero schroeder-1 vector-inf-comp)
    finally have 40:  $y^T * p^* * w = \text{bot}$ 
    by simp
    have ( $?p - 1$ ) *  $p^* * w = (w \sqcap y^T \sqcap -1) * p^* * w \sqcup (-w \sqcap p \sqcap -1) * p^* * w$ 
    by (simp add: comp-inf.mult-right-dist-sup mult-right-dist-sup)
    also have  $\dots \leq (w \sqcap y^T \sqcap -1) * p^* * w \sqcup p * p^* * w$ 
    by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-right-isotone)
    also have  $\dots \leq (w \sqcap y^T \sqcap -1) * p^* * w \sqcup p^* * w$ 
    using mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast
    also have  $\dots \leq y^T * p^* * w \sqcup p^* * w$ 
    by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-left-isotone)
    also have  $\dots = p^* * w$ 
    using 40 by simp
    finally show ?thesis
    by (metis comp-associative le-supI star.circ-loop-fixpoint sup-ge2
star-left-induct)
qed
finally have  $w \leq p^{T^*} * p^T * w$ 
using 11 39 reachable-without-loops star-plus by auto
thus False
using 4 i1 i10 i5 loop-root-2 star.circ-plus-same by auto
qed
show card ?t < card ?s
apply (rule psubset-card-mono)
subgoal using finite-regular by simp
subgoal using 36 37 38 by auto
done
qed
qed
qed
qed
lemma path-compression-3a:
  assumes path-compression-invariant p x (p[[w]]) p0 w
  shows  $p0[p0^{T^*} * x \mapsto p[[w]]] = p$ 
proof -
  let ?y = p[[w]]
  let ?p = p0[p0^{T^*} * x \mapsto ?y]
  have i1: disjoint-set-forest p and i2: point x and i3: point ?y and i4: ?y =
  root p x
  using assms path-compression-invariant-def path-compression-precondition-def
  by meson+
  have i5: point w
  and i8:  $p \sqcap 1 = p0 \sqcap 1$  and i9:  $fc\ p = fc\ p0$ 
  and i10:  $root\ p\ w = ?y$  and i12:  $p0[p0^{T^*} * x - p0^{T^*} * w \mapsto ?y] = p$ 
  and i13: disjoint-set-forest p0 and i15:  $w \leq p0^{T^*} * x$ 
  using assms path-compression-invariant-def by blast+
  have i11:  $p[[w]] = p0[[w]]$ 

```

by (*smt (verit) i12 i2 i5 dual-order.trans inf-le2 p-antitone-iff put-get-different-vector vector-complement-closed vector-inf-closed vector-mult-closed path-compression-1b*)
have $i14$: $?y = \text{root } p0 \ x$
by (*metis i1 i13 i4 i8 i9 same-root*)
have 1 : $?p \sqcap ?y = p \sqcap ?y$
by (*metis i1 i14 i3 i4 get-put inf-le1 root-successor-loop update-inf update-inf-same*)
have 2 : $?p \sqcap w = p \sqcap w$
by (*metis i5 i11 i15 get-put update-inf update-inf-same*)
have $?y = \text{root } p0 \ w$
by (*metis i1 i10 i13 i8 i9 same-root*)
hence $p0^{T^*} * w = w \sqcup ?y$
by (*metis i11 i13 root-transitive-successor-loop star.circ-loop-fixpoint star-plus sup-monoid.add-commute mult-assoc*)
hence 3 : $?p \sqcap p0^{T^*} * w = p \sqcap p0^{T^*} * w$
using $1 \ 2$ **by** (*simp add: inf-sup-distrib1*)
have $p0^{T^*} * w \leq p0^{T^*} * x$
by (*metis i15 comp-associative mult-right-isotone star.circ-transitive-equal*)
hence 4 : $?p \sqcap (p0^{T^*} * x \sqcap p0^{T^*} * w) = p \sqcap (p0^{T^*} * x \sqcap p0^{T^*} * w)$
using 3 **by** (*simp add: inf.absorb2*)
have 5 : $?p \sqcap (p0^{T^*} * x - p0^{T^*} * w) = p \sqcap (p0^{T^*} * x - p0^{T^*} * w)$
by (*metis i12 inf-le1 update-inf update-inf-same*)
have *regular* ($p0^{T^*} * w$)
using $i13 \ i5$ *bijection-regular mapping-regular regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
hence 6 : $?p \sqcap p0^{T^*} * x = p \sqcap p0^{T^*} * x$
using $4 \ 5$ **by** (*smt inf-sup-distrib1 maddux-3-11-pp*)
have 7 : $?p - p0^{T^*} * x = p - p0^{T^*} * x$
by (*smt i12 inf.sup-monoid.add-commute inf-import-p inf-sup-absorb le-iff-inf p-dist-inf update-inf-different inf.idem p-antitone-inf*)
have *regular* ($p0^{T^*} * x$)
using $i13 \ i2$ *bijection-regular mapping-regular regular-closed-star regular-conv-closed regular-mult-closed* **by** *auto*
thus $?p = p$
using $6 \ 7$ **by** (*smt inf-sup-distrib1 maddux-3-11-pp*)
qed

lemma *path-compression-3*:

path-compression-invariant $p \ x \ (p[[w]]) \ p0 \ w \implies$ *path-compression-postcondition*
 $p \ x \ (p[[w]]) \ p0$
using *path-compression-invariant-def path-compression-postcondition-def path-compression-precondition-def path-compression-3a* **by** *blast*

theorem *path-compression*:

VARs $p \ t \ w$
 $[\textit{path-compression-precondition} \ p \ x \ y \wedge \ p0 = p]$
 $w := x;$
 $WHILE \ y \neq p[[w]]$

```

INV { path-compression-invariant p x y p0 w }
VAR { (pT* * w)↓ }
DO t := w;
  w := p[[w]];
  p[t] := y
OD
[ path-compression-postcondition p x y p0 ]
apply vcg-tc-simp
  apply (fact path-compression-1)
  apply (fact path-compression-2)
using path-compression-3 by auto

```

lemma *path-compression-exists*:
path-compression-precondition p x y $\implies \exists p' . \text{path-compression-postcondition } p' x y p$
using *tc-extract-function path-compression* **by** *blast*

definition *path-compression* p x y $\equiv (\text{SOME } p' . \text{path-compression-postcondition } p' x y p)$

lemma *path-compression-function*:
assumes *path-compression-precondition p x y*
and $p' = \text{path-compression } p x y$
shows *path-compression-postcondition p' x y p*
by (*metis assms path-compression-def path-compression-exists someI*)

4.4 Find-Set with Path Compression

We sequentially combine find-set and path compression. We consider implementations which use the previously derived functions and implementations which unfold their definitions.

theorem *find-set-path-compression*:
VARs p y
[*find-set-precondition p x* \wedge $p0 = p$]
y := find-set p x;
p := path-compression p x y
[*path-compression-postcondition p x y p0*]
apply *vcg-tc-simp*
using *find-set-function find-set-postcondition-def find-set-precondition-def path-compression-function path-compression-precondition-def* **by** *fastforce*

theorem *find-set-path-compression-1*:
VARs p t w y
[*find-set-precondition p x* \wedge $p0 = p$]
y := find-set p x;
w := x;
WHILE $y \neq p[[w]]$
 INV { *path-compression-invariant p x y p0 w* }
 VAR { $(p^{T*} * w)\downarrow$ }

```

    DO t := w;
      w := p[[w]];
      p[t] := y
    OD
  [ path-compression-postcondition p x y p0 ]
apply vcg-tc-simp
  using find-set-function find-set-postcondition-def find-set-precondition-def
  path-compression-1 path-compression-precondition-def apply fastforce
  apply (fact path-compression-2)
  by (fact path-compression-3)

```

theorem *find-set-path-compression-2*:

```

  VARS p y
  [ find-set-precondition p x  $\wedge$  p0 = p ]
  y := x;
  WHILE y  $\neq$  p[[y]]
    INV { find-set-invariant p x y  $\wedge$  p0 = p }
    VAR { (pT* * y)↓ }
    DO y := p[[y]]
    OD;
  p := path-compression p x y
  [ path-compression-postcondition p x y p0 ]
apply vcg-tc-simp
  apply (fact find-set-1)
  apply (fact find-set-2)
  by (smt find-set-3 find-set-invariant-def find-set-postcondition-def
  find-set-precondition-def path-compression-function
  path-compression-precondition-def)

```

theorem *find-set-path-compression-3*:

```

  VARS p t w y
  [ find-set-precondition p x  $\wedge$  p0 = p ]
  y := x;
  WHILE y  $\neq$  p[[y]]
    INV { find-set-invariant p x y  $\wedge$  p0 = p }
    VAR { (pT* * y)↓ }
    DO y := p[[y]]
    OD;
  w := x;
  WHILE y  $\neq$  p[[w]]
    INV { path-compression-invariant p x y p0 w }
    VAR { (pT* * w)↓ }
    DO t := w;
      w := p[[w]];
      p[t] := y
    OD
  [ path-compression-postcondition p x y p0 ]
apply vcg-tc-simp
  apply (simp add: find-set-1)

```

```

apply (fact find-set-2)
using find-set-3 find-set-invariant-def find-set-postcondition-def
find-set-precondition-def path-compression-1 path-compression-precondition-def
apply blast
apply (fact path-compression-2)
by (fact path-compression-3)

```

Find-set with path compression returns two results: the representative of the tree and the modified disjoint-set forest.

```

lemma find-set-path-compression-exists:
  find-set-precondition p x  $\implies \exists p' y . \text{path-compression-postcondition } p' x y p$ 
using tc-extract-function find-set-path-compression by blast

```

```

definition find-set-path-compression p x  $\equiv (\text{SOME } (p',y) .$ 
  path-compression-postcondition p' x y p)

```

```

lemma find-set-path-compression-function:
assumes find-set-precondition p x
and (p',y) = find-set-path-compression p x
shows path-compression-postcondition p' x y p
proof –
let ?P =  $\lambda(p',y) . \text{path-compression-postcondition } p' x y p$ 
have ?P (SOME z . ?P z)
apply (unfold some-eq-ex)
using assms(1) find-set-path-compression-exists by simp
thus ?thesis
using assms(2) find-set-path-compression-def by auto
qed

```

We prove that *find-set-path-compression* returns the same representative as *find-set*.

```

lemma find-set-path-compression-find-set:
assumes find-set-precondition p x
shows find-set p x = snd (find-set-path-compression p x)
proof –
let ?r = find-set p x
let ?p = fst (find-set-path-compression p x)
let ?y = snd (find-set-path-compression p x)
have 1: find-set-postcondition p x ?r
by (simp add: assms find-set-function)
have path-compression-postcondition ?p x ?y p
using assms find-set-path-compression-function prod.collapse by blast
thus ?r = ?y
using 1 by (smt assms same-root find-set-precondition-def
  find-set-postcondition-def path-compression-precondition-def
  path-compression-postcondition-def)
qed

```

A weaker postcondition suffices to prove that the two forests have the same semantics; that is, they describe the same disjoint sets and have the

same roots.

lemma *find-set-path-compression-path-compression-antics:*

```

assumes find-set-precondition p x
shows fc (path-compression p x (find-set p x)) = fc (fst
(find-set-path-compression p x))
  and path-compression p x (find-set p x)  $\sqcap$  1 = fst (find-set-path-compression p
x)  $\sqcap$  1
proof –
  let ?r = find-set p x
  let ?q = path-compression p x ?r
  let ?p = fst (find-set-path-compression p x)
  let ?y = snd (find-set-path-compression p x)
  have 1: path-compression-postcondition (path-compression p x ?r) x ?r p
    using assms find-set-function find-set-postcondition-def
find-set-precondition-def path-compression-function
path-compression-precondition-def by auto
  have 2: path-compression-postcondition ?p x ?y p
    using assms find-set-path-compression-function prod.collapse by blast
  show fc ?q = fc ?p
    using 1 2 by (simp add: path-compression-postcondition-def)
  show ?q  $\sqcap$  1 = ?p  $\sqcap$  1
    using 1 2 by (simp add: path-compression-postcondition-def)
qed

```

With the current, stronger postcondition of path compression describing the precise effect of how links change, we can prove that the two forests are actually equal.

lemma *find-set-path-compression-find-set-pathcompression:*

```

assumes find-set-precondition p x
shows path-compression p x (find-set p x) = fst (find-set-path-compression p x)
proof –
  let ?r = find-set p x
  let ?q = path-compression p x ?r
  let ?p = fst (find-set-path-compression p x)
  let ?y = snd (find-set-path-compression p x)
  have 1: path-compression-postcondition (path-compression p x ?r) x ?r p
    using assms find-set-function find-set-postcondition-def
find-set-precondition-def path-compression-function
path-compression-precondition-def by auto
  have 2: path-compression-postcondition ?p x ?y p
    using assms find-set-path-compression-function prod.collapse by blast
  have ?r = ?y
    by (simp add: assms find-set-path-compression-find-set)
  thus ?q = ?p
    using 1 2 by (simp add: path-compression-postcondition-def)
qed

```

4.5 Union-Sets

We only consider a naive union-sets operation (without ranks). The semantics is the equivalence closure obtained after adding the link between the two given nodes, which requires those two elements to be in the same set. The implementation uses temporary variable t to store the two results returned by find-set with path compression. The disjoint-set forest, which keeps being updated, is threaded through the sequence of operations.

definition *union-sets-precondition* $p\ x\ y \equiv \text{disjoint-set-forest } p \wedge \text{point } x \wedge \text{point } y$

definition *union-sets-postcondition* $p\ x\ y\ p0 \equiv \text{union-sets-precondition } p\ x\ y \wedge \text{fc } p = \text{ucc } (p0 \sqcup x * y^T)$

lemma *union-sets-1:*

assumes *union-sets-precondition* $p0\ x\ y$
and *path-compression-postcondition* $p1\ x\ r\ p0$
and *path-compression-postcondition* $p2\ y\ s\ p1$
shows *union-sets-postcondition* $(p2[r \mapsto s])\ x\ y\ p0$
proof (*unfold union-sets-postcondition-def union-sets-precondition-def, intro conjI*)
let $?p = p2[r \mapsto s]$
have $1: \text{disjoint-set-forest } p1 \wedge \text{point } r \wedge r = \text{root } p1\ x \wedge p1 \sqcap 1 = p0 \sqcap 1 \wedge \text{fc } p1 = \text{fc } p0$
using $\text{assms}(2)$ *path-compression-precondition-def path-compression-postcondition-def* **by** *auto*
have $2: \text{disjoint-set-forest } p2 \wedge \text{point } s \wedge s = \text{root } p2\ y \wedge p2 \sqcap 1 = p1 \sqcap 1 \wedge \text{fc } p2 = \text{fc } p1$
using $\text{assms}(3)$ *path-compression-precondition-def path-compression-postcondition-def* **by** *auto*
hence $3: \text{fc } p2 = \text{fc } p0$
using 1 **by** *simp*
show $4: \text{univalent } ?p$
using $1\ 2$ *update-univalent* **by** *blast*
show *total* $?p$
using $1\ 2$ *bijjective-regular update-total* **by** *blast*
show *acyclic* $(?p - 1)$
proof (*cases* $r = s$)
case *True*
thus $?thesis$
using 2 *update-acyclic-5* **by** *fastforce*
next
case *False*
hence $\text{bot} = r \sqcap s$
using $1\ 2$ *distinct-points* **by** *blast*
also have $\dots = r \sqcap p2^{T*} * s$
using 2 **by** (*smt root-transitive-successor-loop*)
finally have $s \sqcap p2^* * r = \text{bot}$
using *schroeder-1 conv-star-commute inf.sup-monoid.add-commute* **by** *fastforce*
thus $?thesis$


```

    using 1 2 update-acyclic-4 by blast
qed
show vector x
  using assms(1) by (simp add: union-sets-precondition-def)
show injective x
  using assms(1) by (simp add: union-sets-precondition-def)
show surjective x
  using assms(1) by (simp add: union-sets-precondition-def)
show vector y
  using assms(1) by (simp add: union-sets-precondition-def)
show injective y
  using assms(1) by (simp add: union-sets-precondition-def)
show surjective y
  using assms(1) by (simp add: union-sets-precondition-def)
show fc ?p = wcc (p0  $\sqcup$  x * yT)
proof (rule order.antisym)
  have r = p1[[r]]
    using 1 by (metis root-successor-loop)
  hence r * rT ≤ p1T
    using 1 eq-refl shunt-bijective by blast
  hence r * rT ≤ p1
    using 1 conv-order coreflexive-symmetric by fastforce
  hence r * rT ≤ p1  $\sqcap$  1
    using 1 inf.boundedI by blast
  also have ... = p2  $\sqcap$  1
    using 2 by simp
  finally have r * rT ≤ p2
    by simp
  hence r ≤ p2 * r
    using 1 shunt-bijective by blast
  hence 5: p2[[r]] ≤ r
    using 2 shunt-mapping by blast
  have r  $\sqcap$  p2 ≤ r * (top  $\sqcap$  rT * p2)
    using 1 by (metis dedekind-1)
  also have ... = r * rT * p2
    by (simp add: mult-assoc)
  also have ... ≤ r * rT
    using 5 by (metis comp-associative conv-dist-comp conv-involutive
conv-order mult-right-isotone)
  also have ... ≤ 1
    using 1 by blast
  finally have 6: r  $\sqcap$  p2 ≤ 1
    by simp
  have p0 ≤ wcc p0
    by (simp add: star.circ-sub-dist-1)
  also have ... = wcc p2
    using 3 by (simp add: star-decompose-1)
  also have 7: ... ≤ wcc ?p
proof -

```

have $wcc\ p2 = wcc\ ((-r \sqcap p2) \sqcup (r \sqcap p2))$
using 1 **by** (*metis bijective-regular inf.sup-monoid.add-commute*
maddux-3-11-pp)
also have $\dots \leq wcc\ ((-r \sqcap p2) \sqcup 1)$
using 6 *wcc-isotone sup-right-isotone* **by** *simp*
also have $\dots = wcc\ (-r \sqcap p2)$
using *wcc-with-loops* **by** *simp*
also have $\dots \leq wcc\ ?p$
using *wcc-isotone sup-ge2* **by** *blast*
finally show *?thesis*
by *simp*
qed
finally have 8: $p0 \leq wcc\ ?p$
by *force*
have $r \leq p1^{T^*} * x$
using 1 **by** (*metis inf-le1*)
hence 9: $r * x^T \leq p1^{T^*}$
using *assms(1) shunt-bijective union-sets-precondition-def* **by** *blast*
hence $x * r^T \leq p1^*$
using *conv-dist-comp conv-order conv-star-commute* **by** *force*
also have $\dots \leq wcc\ p1$
by (*simp add: star.circ-sub-dist*)
also have $\dots = wcc\ p2$
using 1 2 **by** (*simp add: fc-wcc*)
also have $\dots \leq wcc\ ?p$
using 7 **by** *simp*
finally have 10: $x * r^T \leq wcc\ ?p$
by *simp*
have 11: $r * s^T \leq wcc\ ?p$
using 1 2 *star.circ-sub-dist-1 sup-assoc vector-covector* **by** *auto*
have $s \leq p2^{T^*} * y$
using 2 **by** (*metis inf-le1*)
hence 12: $s * y^T \leq p2^{T^*}$
using *assms(1) shunt-bijective union-sets-precondition-def* **by** *blast*
also have $\dots \leq wcc\ p2$
using *star-isotone sup-ge2* **by** *blast*
also have $\dots \leq wcc\ ?p$
using 7 **by** *simp*
finally have 13: $s * y^T \leq wcc\ ?p$
by *simp*
have $x \leq x * r^T * r \wedge y \leq y * s^T * s$
using 1 2 *shunt-bijective* **by** *blast*
hence $x * y^T \leq x * r^T * r * (y * s^T * s)^T$
using *comp-isotone conv-isotone* **by** *blast*
also have $\dots = x * r^T * r * s^T * s * y^T$
by (*simp add: comp-associative conv-dist-comp*)
also have $\dots \leq wcc\ ?p * (r * s^T) * (s * y^T)$
using 10 **by** (*metis mult-left-isotone mult-assoc*)
also have $\dots \leq wcc\ ?p * wcc\ ?p * (s * y^T)$

using 11 by (*metis mult-left-isotone mult-right-isotone*)
 also have $\dots \leq wcc \ ?p * wcc \ ?p * wcc \ ?p$
 using 13 by (*metis mult-right-isotone*)
 also have $\dots = wcc \ ?p$
 by (*simp add: star.circ-transitive-equal*)
 finally have $p0 \sqcup x * y^T \leq wcc \ ?p$
 using 8 by *simp*
 hence $wcc (p0 \sqcup x * y^T) \leq wcc \ ?p$
 using *wcc-below-wcc* by *simp*
 thus $wcc (p0 \sqcup x * y^T) \leq fc \ ?p$
 using 4 *fc-wcc* by *simp*
 have $-r \sqcap p2 \leq wcc \ p2$
 by (*simp add: inf.coboundedI2 star.circ-sub-dist-1*)
 also have $\dots = wcc \ p0$
 using 3 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc (p0 \sqcup x * y^T)$
 by (*simp add: wcc-isotone*)
 finally have 14: $-r \sqcap p2 \leq wcc (p0 \sqcup x * y^T)$
 by *simp*
 have $r * x^T \leq wcc \ p1$
 using 9 *inf.order-trans star.circ-sub-dist sup-commute* by *fastforce*
 also have $\dots = wcc \ p0$
 using 1 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc (p0 \sqcup x * y^T)$
 by (*simp add: wcc-isotone*)
 finally have 15: $r * x^T \leq wcc (p0 \sqcup x * y^T)$
 by *simp*
 have 16: $x * y^T \leq wcc (p0 \sqcup x * y^T)$
 using *le-supE star.circ-sub-dist-1* by *blast*
 have $y * s^T \leq p2^*$
 using 12 *conv-dist-comp conv-order conv-star-commute* by *fastforce*
 also have $\dots \leq wcc \ p2$
 using *star.circ-sub-dist sup-commute* by *fastforce*
 also have $\dots = wcc \ p0$
 using 3 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc (p0 \sqcup x * y^T)$
 by (*simp add: wcc-isotone*)
 finally have 17: $y * s^T \leq wcc (p0 \sqcup x * y^T)$
 by *simp*
 have $r \leq r * x^T * x \wedge s \leq s * y^T * y$
 using *assms(1) shunt-bijective union-sets-precondition-def* by *blast*
 hence $r * s^T \leq r * x^T * x * (s * y^T * y)^T$
 using *comp-isotone conv-isotone* by *blast*
 also have $\dots = r * x^T * x * y^T * y * s^T$
 by (*simp add: comp-associative conv-dist-comp*)
 also have $\dots \leq wcc (p0 \sqcup x * y^T) * (x * y^T) * (y * s^T)$
 using 15 by (*metis mult-left-isotone mult-assoc*)
 also have $\dots \leq wcc (p0 \sqcup x * y^T) * wcc (p0 \sqcup x * y^T) * (y * s^T)$
 using 16 by (*metis mult-left-isotone mult-right-isotone*)

also have $\dots \leq wcc (p0 \sqcup x * y^T) * wcc (p0 \sqcup x * y^T) * wcc (p0 \sqcup x * y^T)$
using 17 **by** (*metis mult-right-isotone*)
also have $\dots = wcc (p0 \sqcup x * y^T)$
by (*simp add: star.circ-transitive-equal*)
finally have $?p \leq wcc (p0 \sqcup x * y^T)$
using 1 2 14 *vector-covector* **by** *auto*
hence $wcc ?p \leq wcc (p0 \sqcup x * y^T)$
using *wcc-below-wcc* **by** *blast*
thus *fc* $?p \leq wcc (p0 \sqcup x * y^T)$
using 4 *fc-wcc* **by** *simp*
qed
qed

theorem *union-sets*:

VARs $p \ r \ s \ t$
 $[\text{union-sets-precondition } p \ x \ y \wedge p0 = p]$
 $t := \text{find-set-path-compression } p \ x;$
 $p := \text{fst } t;$
 $r := \text{snd } t;$
 $t := \text{find-set-path-compression } p \ y;$
 $p := \text{fst } t;$
 $s := \text{snd } t;$
 $p[r] := s$
 $[\text{union-sets-postcondition } p \ x \ y \ p0]$

proof *vcg-tc-simp*

let $?t1 = \text{find-set-path-compression } p0 \ x$
let $?p1 = \text{fst } ?t1$
let $?r = \text{snd } ?t1$
let $?t2 = \text{find-set-path-compression } ?p1 \ y$
let $?p2 = \text{fst } ?t2$
let $?s = \text{snd } ?t2$
let $?p = ?p2[?r \mapsto ?s]$
assume 1: *union-sets-precondition* $p0 \ x \ y$
hence 2: *path-compression-postcondition* $?p1 \ x \ ?r \ p0$
by (*simp add: find-set-precondition-def union-sets-precondition-def*
find-set-path-compression-function)
hence *path-compression-postcondition* $?p2 \ y \ ?s \ ?p1$
using 1 **by** (*meson find-set-precondition-def union-sets-precondition-def*
find-set-path-compression-function path-compression-postcondition-def
path-compression-precondition-def prod.collapse)
thus *union-sets-postcondition* $(?p2[?r \mapsto ?s]) \ x \ y \ p0$
using 1 2 **by** (*simp add: union-sets-1*)
qed

lemma *union-sets-exists*:

union-sets-precondition $p \ x \ y \implies \exists p' . \text{union-sets-postcondition } p' \ x \ y \ p$
using *tc-extract-function union-sets* **by** *blast*

definition *union-sets* $p \ x \ y \equiv (\text{SOME } p' . \text{union-sets-postcondition } p' \ x \ y \ p)$

```

lemma union-sets-function:
  assumes union-sets-precondition  $p$   $x$   $y$ 
    and  $p' = \text{union-sets } p$   $x$   $y$ 
  shows union-sets-postcondition  $p'$   $x$   $y$   $p$ 
  by (metis assms union-sets-def union-sets-exists someI)

theorem union-sets-2:
  VARs  $p$   $r$   $s$ 
  [ union-sets-precondition  $p$   $x$   $y$   $\wedge$   $p0 = p$  ]
   $r := \text{find-set } p$   $x$ ;
   $p := \text{path-compression } p$   $x$   $r$ ;
   $s := \text{find-set } p$   $y$ ;
   $p := \text{path-compression } p$   $y$   $s$ ;
   $p[r] := s$ 
  [ union-sets-postcondition  $p$   $x$   $y$   $p0$  ]
proof vcg-tc-simp
  let  $?r = \text{find-set } p0$   $x$ 
  let  $?p1 = \text{path-compression } p0$   $x$   $?r$ 
  let  $?s = \text{find-set } ?p1$   $y$ 
  let  $?p2 = \text{path-compression } ?p1$   $y$   $?s$ 
  assume 1: union-sets-precondition  $p0$   $x$   $y$ 
  hence 2: path-compression-postcondition  $?p1$   $x$   $?r$   $p0$ 
    using find-set-function find-set-postcondition-def find-set-precondition-def
path-compression-function path-compression-precondition-def
union-sets-precondition-def by auto
  hence path-compression-postcondition  $?p2$   $y$   $?s$   $?p1$ 
    using 1 find-set-function find-set-postcondition-def find-set-precondition-def
path-compression-function path-compression-precondition-def
union-sets-precondition-def path-compression-postcondition-def by meson
  thus union-sets-postcondition ( $?p2[?r \mapsto ?s]$ )  $x$   $y$   $p0$ 
    using 1 2 by (simp add: union-sets-1)
qed

end

end

```

```

theory More-Disjoint-Set-Forests

```

```

imports Disjoint-Set-Forests

```

```

begin

```

5 More on Array Access and Disjoint-Set Forests

This section contains further results about directed acyclic graphs and relational array operations.

context *stone-relation-algebra*
begin

Theorem 6.4

lemma *update-square*:

assumes *point y*

shows $x[y \mapsto x[x[[y]]]] \leq x * x \sqcup x$

proof –

have $x[y \mapsto x[x[[y]]]] = (y \sqcap y^T * x * x) \sqcup (-y \sqcap x)$

by (*simp add: conv-dist-comp*)

also have $\dots \leq (y \sqcap y^T) * x * x \sqcup x$

by (*smt assms inf.eq-refl inf.sup-monoid.add-commute inf-le1 sup-mono vector-inf-comp*)

also have $\dots \leq x * x \sqcup x$

by (*smt (z3) assms comp-associative conv-dist-comp coreflexive-comp-top-inf inf.cobounded2 sup-left-isotone symmetric-top-closed*)

finally show *?thesis*

qed

Theorem 2.13

lemma *update-ub*:

$x[y \mapsto z] \leq x \sqcup z^T$

by (*meson dual-order.trans inf.cobounded2 le-supI sup.cobounded1 sup-ge2*)

Theorem 6.7

lemma *update-square-ub*:

$x[y \mapsto (x * x)^T] \leq x \sqcup x * x$

by (*metis conv-involutive update-ub*)

Theorem 2.14

lemma *update-same-sub*:

assumes $u \sqcap x = u \sqcap z$

and $y \leq u$

and *regular y*

shows $x[y \mapsto z^T] = x$

by (*smt (z3) assms conv-involutive inf.sup-monoid.add-commute inf.sup-relative-same-increasing maddux-3-11-pp*)

Theorem 2.15

lemma *update-point-get*:

point y $\implies x[y \mapsto z[[y]]] = x[y \mapsto z^T]$

by (*metis conv-involutive get-put inf-commute update-inf-same*)

Theorem 2.11

lemma *update-bot*:

$x[\text{bot} \rightarrow z] = x$
by *simp*

Theorem 2.12

lemma *update-top*:
 $x[\text{top} \rightarrow z] = z^T$
by *simp*

Theorem 2.6

lemma *update-same*:

assumes *regular u*

shows $(x[y \rightarrow z])[u \rightarrow z] = x[y \sqcup u \rightarrow z]$

proof –

have $(x[y \rightarrow z])[u \rightarrow z] = (u \sqcap z^T) \sqcup (-u \sqcap y \sqcap z^T) \sqcup (-u \sqcap -y \sqcap x)$

using *inf.sup-monoid.add-assoc inf-sup-distrib1 sup-assoc* **by** *force*

also have $\dots = (u \sqcap z^T) \sqcup (y \sqcap z^T) \sqcup -(u \sqcup y) \sqcap x$

by (*metis assms inf-sup-distrib2 maddux-3-21-pp p-dist-sup*)

also have $\dots = x[y \sqcup u \rightarrow z]$

using *comp-inf.mult-right-dist-sup sup-commute* **by** *auto*

finally show *?thesis*

qed

lemma *update-same-3*:

assumes *regular u*

and *regular v*

shows $((x[y \rightarrow z])[u \rightarrow z])[v \rightarrow z] = x[y \sqcup u \sqcup v \rightarrow z]$

by (*metis assms update-same*)

Theorem 2.7

lemma *update-split*:

assumes *regular w*

shows $x[y \rightarrow z] = (x[y - w \rightarrow z])[y \sqcap w \rightarrow z]$

by (*smt (z3) assms comp-inf.semiring.distrib-left inf.left-commute*

inf.sup-monoid.add-commute inf-import-p maddux-3-11-pp maddux-3-12 p-dist-inf sup-assoc)

Theorem 2.8

lemma *update-injective-swap*:

assumes *injective x*

and *point y*

and *injective z*

and *vector z*

shows *injective* $((x[y \rightarrow x[[z]]])[z \rightarrow x[[y]])$

proof –

have $1: (z \sqcap y^T * x) * (z \sqcap y^T * x)^T \leq 1$

using *assms(3) injective-inf-closed* **by** *auto*

have $(z \sqcap y^T * x) * (-z \sqcap y \sqcap z^T * x)^T \leq (z \sqcap y^T * x) * (y^T \sqcap x^T * z)$

by (*metis conv-dist-comp conv-involutive conv-order inf.boundedE*

inf.boundedI inf.cobounded1 inf.cobounded2 mult-right-isotone)

also have $\dots = (z \sqcap z^T * x) * (y^T \sqcap x^T * y)$
by (*smt* (*z3*) *assms*(2,4) *covector-inf-comp-3* *inf.left-commute*
inf.sup-monoid.add-commute *comp-associative* *conv-dist-comp* *conv-involutive*)
also have $\dots = (z \sqcap z^T) * x * x^T * (y \sqcap y^T)$
by (*smt* (*z3*) *assms*(2,4) *comp-associative* *inf.sup-monoid.add-commute*
vector-covector *vector-inf-comp*)
also have $\dots \leq x * x^T$
by (*metis* *assms*(2-4) *comp-associative* *comp-right-one*
coreflexive-comp-top-inf *inf.coboundedI2* *mult-right-isotone* *vector-covector*)
also have $\dots \leq 1$
by (*simp* *add: assms*(1))
finally have 2: $(z \sqcap y^T * x) * (-z \sqcap y \sqcap z^T * x)^T \leq 1$

have $(z \sqcap y^T * x) * (-z \sqcap -y \sqcap x)^T \leq y^T * x * (-y^T \sqcap x^T)$
by (*smt* *comp-isotone* *conv-complement* *conv-dist-inf* *inf.cobounded2*
inf.sup-monoid.add-assoc)
also have $\dots = y^T * x * x^T \sqcap -y^T$
by (*simp* *add: inf.commute* *assms*(2) *covector-comp-inf* *vector-conv-compl*)
also have $\dots \leq y^T \sqcap -y^T$
by (*metis* *assms*(1) *comp-associative* *comp-inf.mult-left-isotone* *comp-isotone*
comp-right-one *mult-sub-right-one*)
finally have 3: $(z \sqcap y^T * x) * (-z \sqcap -y \sqcap x)^T \leq 1$
using *pseudo-complement* **by** *fastforce*
have 4: $(-z \sqcap y \sqcap z^T * x) * (z \sqcap y^T * x)^T \leq 1$
using 2 *conv-dist-comp* *conv-order* **by** *force*
have 5: $(-z \sqcap y \sqcap z^T * x) * (-z \sqcap y \sqcap z^T * x)^T \leq 1$
by (*simp* *add: assms*(2) *inf-assoc* *inf-left-commute* *injective-inf-closed*)
have $(-z \sqcap y \sqcap z^T * x) * (-z \sqcap -y \sqcap x)^T \leq z^T * x * (-z^T \sqcap x^T)$
using *comp-inf.mult-left-isotone* *comp-isotone* *conv-complement* *conv-dist-inf*
inf.cobounded1 *inf.cobounded2* **by** *auto*
also have $\dots = z^T * x * x^T \sqcap -z^T$
by (*metis* *assms*(4) *covector-comp-inf* *inf.sup-monoid.add-commute*
vector-conv-compl)
also have $\dots \leq z^T \sqcap -z^T$
by (*metis* *assms*(1) *comp-associative* *comp-inf.mult-left-isotone* *comp-isotone*
comp-right-one *mult-sub-right-one*)
finally have 6: $(-z \sqcap y \sqcap z^T * x) * (-z \sqcap -y \sqcap x)^T \leq 1$
using *pseudo-complement* **by** *fastforce*
have 7: $(-z \sqcap -y \sqcap x) * (z \sqcap y^T * x)^T \leq 1$
using 3 *conv-dist-comp* *coreflexive-symmetric* **by** *fastforce*
have 8: $(-z \sqcap -y \sqcap x) * (-z \sqcap y \sqcap z^T * x)^T \leq 1$
using 6 *conv-dist-comp* *coreflexive-symmetric* **by** *fastforce*
have 9: $(-z \sqcap -y \sqcap x) * (-z \sqcap -y \sqcap x)^T \leq 1$
using *assms*(1) *inf.sup-monoid.add-commute* *injective-inf-closed* **by** *auto*
have $(x[y \mapsto x[[z]]][z \mapsto x[[y]]]) = (z \sqcap y^T * x) \sqcup (-z \sqcap y \sqcap z^T * x) \sqcup (-z \sqcap -y \sqcap x)$
by (*simp* *add: comp-inf.comp-left-dist-sup* *conv-dist-comp* *inf-assoc*
sup-monoid.add-assoc)
hence $((x[y \mapsto x[[z]]][z \mapsto x[[y]]]) * ((x[y \mapsto x[[z]]][z \mapsto x[[y]]])^T) = ((z \sqcap y^T * x$

$x) \sqcup (-z \sqcap y \sqcap z^T * x) \sqcup (-z \sqcap -y \sqcap x)) * ((z \sqcap y^T * x)^T \sqcup (-z \sqcap y \sqcap z^T * x)^T \sqcup (-z \sqcap -y \sqcap x)^T) \sqcup$
by (*simp add: conv-dist-sup*)
also have ... = $(z \sqcap y^T * x) * ((z \sqcap y^T * x)^T \sqcup (-z \sqcap y \sqcap z^T * x)^T \sqcup (-z \sqcap -y \sqcap x)^T) \sqcup$
 $(-z \sqcap y \sqcap z^T * x) * ((z \sqcap y^T * x)^T \sqcup (-z \sqcap y \sqcap z^T * x)^T \sqcup (-z \sqcap -y \sqcap x)^T) \sqcup$
 $(-z \sqcap -y \sqcap x) * ((z \sqcap y^T * x)^T \sqcup (-z \sqcap y \sqcap z^T * x)^T \sqcup (-z \sqcap -y \sqcap x)^T)$
using *mult-right-dist-sup by auto*
also have ... = $(z \sqcap y^T * x) * (z \sqcap y^T * x)^T \sqcup (z \sqcap y^T * x) * (-z \sqcap y \sqcap z^T * x)^T \sqcup$
 $(z \sqcap y^T * x) * (-z \sqcap -y \sqcap x)^T \sqcup$
 $(-z \sqcap y \sqcap z^T * x) * (z \sqcap y^T * x)^T \sqcup (-z \sqcap y \sqcap z^T * x) * (-z \sqcap y \sqcap z^T * x)^T \sqcup$
 $(-z \sqcap y \sqcap z^T * x) * (-z \sqcap -y \sqcap x)^T \sqcup$
 $(-z \sqcap -y \sqcap x) * (z \sqcap y^T * x)^T \sqcup (-z \sqcap -y \sqcap x) * (-z \sqcap y \sqcap z^T * x)^T \sqcup$
 $(-z \sqcap -y \sqcap x) * (-z \sqcap -y \sqcap x)^T$
using *mult-left-dist-sup sup.left-commute sup-commute by auto*
also have ... ≤ 1
using 1 2 3 4 5 6 7 8 9 **by** *simp-all*
finally show *?thesis*

qed

lemma *update-injective-swap-2:*

assumes *injective x*

shows *injective ((x[y \mapsto x[[bot]]]) [bot \mapsto x[[y]]])*

by (*simp add: assms inf.sup-monoid.add-commute injective-inf-closed*)

[Theorem 2.9](#)

lemma *update-univalent-swap:*

assumes *univalent x*

and *injective y*

and *vector y*

and *injective z*

and *vector z*

shows *univalent ((x[y \mapsto x[[z]]]) [z \mapsto x[[y]]])*

by (*simp add: assms read-injective update-univalent*)

[Theorem 2.10](#)

lemma *update-mapping-swap:*

assumes *mapping x*

and *point y*

and *point z*

shows *mapping ((x[y \mapsto x[[z]]]) [z \mapsto x[[y]]])*

by (*simp add: assms bijective-regular read-injective read-surjective update-total update-univalent*)

[Theorem 2.16 mapping-inf-point-arc](#) has been moved to theory *Relation-Algebras* in entry *Stone-Relation-Algebras*

end

context *stone-kleene-relation-algebra*
begin

lemma *omit-redundant-points-2*:

assumes *point p*
shows $p \sqcap x^* = (p \sqcap 1) \sqcup (p \sqcap x \sqcap -p^T) * (x \sqcap -p^T)^*$
proof –
 let $?p = p \sqcap 1$
 let $?np = -p \sqcap 1$
 have $1: p \sqcap x^* \sqcap 1 = p \sqcap 1$
 by (*metis inf.le-iff-sup inf.left-commute inf.sup-monoid.add-commute star.circ-reflexive*)
 have $2: p \sqcap 1 \sqcap -p^T = \text{bot}$
 by (*smt (z3) inf-bot-right inf-commute inf-left-commute one-inf-conv p-inf*)
 have $p \sqcap x^* \sqcap -1 = p \sqcap x^* \sqcap -p^T$
 by (*metis assms antisymmetric-inf-diversity inf.cobounded1 inf.sup-relative-same-increasing vector-covector*)
 also have $\dots = (p \sqcap 1 \sqcap -p^T) \sqcup ((p \sqcap x) * (-p \sqcap x)^* \sqcap -p^T)$
 by (*simp add: assms omit-redundant-points comp-inf.semiring.distrib-right*)
 also have $\dots = (p \sqcap x) * (-p \sqcap x)^* \sqcap -p^T$
 using 2 **by** *simp*
 also have $\dots = ?p * x * (-p \sqcap x)^* \sqcap -p^T$
 by (*metis assms vector-export-comp-unit*)
 also have $\dots = ?p * x * (?np * x)^* \sqcap -p^T$
 by (*metis assms vector-complement-closed vector-export-comp-unit*)
 also have $\dots = ?p * x * (?np * x)^* * ?np$
 by (*metis assms conv-complement covector-comp-inf inf.sup-monoid.add-commute mult-1-right one-inf-conv vector-conv-compl*)
 also have $\dots = ?p * x * ?np * (x * ?np)^*$
 using *star-slide mult-assoc* **by** *auto*
 also have $\dots = (?p * x \sqcap -p^T) * (x * ?np)^*$
 by (*metis assms conv-complement covector-comp-inf inf.sup-monoid.add-commute mult-1-right one-inf-conv vector-conv-compl*)
 also have $\dots = (?p * x \sqcap -p^T) * (x \sqcap -p^T)^*$
 by (*metis assms conv-complement covector-comp-inf inf.sup-monoid.add-commute mult-1-right one-inf-conv vector-conv-compl*)
 also have $\dots = (p \sqcap x \sqcap -p^T) * (x \sqcap -p^T)^*$
 by (*metis assms vector-export-comp-unit*)
 finally show *?thesis*
 using 1 **by** (*metis maddux-3-11-pp regular-one-closed*)
qed

Theorem 5.3

lemma *omit-redundant-points-3*:

assumes *point p*
shows $p \sqcap x^* = (p \sqcap 1) \sqcup (p \sqcap (x \sqcap -p^T)^+)$
by (*simp add: assms inf-assoc vector-inf-comp omit-redundant-points-2*)

Theorem 6.1

lemma *even-odd-root*:

assumes *acyclic* $(x - 1)$

and *regular* x

and *univalent* x

shows $(x * x)^{T^*} \sqcap x^T * (x * x)^{T^*} = (1 \sqcap x) * ((x * x)^{T^*} \sqcap x^T * (x * x)^{T^*})$

proof –

have 1 : *univalent* $(x * x)$

by (*simp add: assms(3) univalent-mult-closed*)

have $x \sqcap 1 \leq top * (x \sqcap 1)$

by (*simp add: top-left-mult-increasing*)

hence $x \sqcap -(top * (x \sqcap 1)) \leq x - 1$

using *assms(2) p-shunting-swap pp-dist-comp* **by** *auto*

hence $x^* * (x \sqcap -(top * (x \sqcap 1))) \leq (x - 1)^* * (x - 1)$

using *mult-right-isotone reachable-without-loops* **by** *auto*

also have $\dots \leq -1$

by (*simp add: assms(1) star-plus*)

finally have $(x \sqcap -(top * (x \sqcap 1)))^T \leq -x^*$

using *schroeder-4-p* **by** *force*

hence $x^T \sqcap x^* \leq (top * (x \sqcap 1))^T$

by (*smt (z3) assms(2) conv-complement conv-dist-inf p-shunting-swap regular-closed-inf regular-closed-top regular-mult-closed regular-one-closed*)

also have $\dots = (1 \sqcap x) * top$

by (*metis conv-dist-comp conv-dist-inf inf-commute one-inf-conv symmetric-one-closed symmetric-top-closed*)

finally have 2 : $(x^T \sqcap x^*) * top \leq (1 \sqcap x) * top$

by (*metis inf.orderE inf.orderI inf-commute inf-vector-comp*)

have $1 \sqcap x^{T+} \leq (x^T \sqcap 1 * x^*) * x^{T^*}$

by (*metis conv-involutive conv-star-commute dedekind-2 inf-commute*)

also have $\dots \leq (x^T \sqcap x^*) * top$

by (*simp add: mult-right-isotone*)

also have $\dots \leq (1 \sqcap x) * top$

using 2 **by** *simp*

finally have 3 : $1 \sqcap x^{T+} \leq (1 \sqcap x) * top$

have $x^T \sqcap (x^T * x^T)^+ = 1 * x^T \sqcap (x^T * x^T)^* * x^T * x^T$

using *star-plus mult-assoc* **by** *auto*

also have $\dots = (1 \sqcap (x^T * x^T)^* * x^T) * x^T$

using *assms(3) injective-comp-right-dist-inf* **by** *force*

also have $\dots \leq (1 \sqcap x^{T^*} * x^T) * x^T$

by (*meson comp-inf.mult-right-isotone comp-isotone inf.eq-refl star.circ-square*)

also have $\dots \leq (1 \sqcap x) * top * x^T$

using 3 **by** (*simp add: mult-left-isotone star-plus*)

also have $\dots \leq (1 \sqcap x) * top$

by (*simp add: comp-associative mult-right-isotone*)

finally have 4 : $x^T \sqcap (x^T * x^T)^+ \leq (1 \sqcap x) * top$

have $x^T \sqcap (x^T * x^T)^* = (x^T \sqcap 1) \sqcup (x^T \sqcap (x^T * x^T)^+)$

by (*metis inf-sup-distrib1 star-left-unfold-equal*)
 also have ... $\leq (1 \sqcap x) * \text{top}$
 using 4 by (*metis inf.sup-monoid.add-commute le-supI one-inf-conv*
top-right-mult-increasing)
 finally have 4: $x^T \sqcap (x^T * x^T)^* \leq (1 \sqcap x) * \text{top}$

•
 have $x^T \sqcap (x * x)^* \sqcap -1 \leq x^T \sqcap x^* \sqcap -1$
 by (*simp add: inf.coboundedI2 inf.sup-monoid.add-commute star.circ-square*)
 also have ... $= (x - 1)^* \sqcap (x - 1)^T$
 using *conv-complement conv-dist-inf inf-assoc inf-left-commute*
reachable-without-loops symmetric-one-closed by *auto*
 also have ... $= \text{bot}$
 using *assms(1) acyclic-star-below-complement-1* by *auto*
 finally have 5: $x^T \sqcap (x * x)^* \sqcap -1 = \text{bot}$
 by (*simp add: le-bot*)
 have $x^T \sqcap (x * x)^* = (x^T \sqcap (x * x)^* \sqcap 1) \sqcup (x^T \sqcap (x * x)^* \sqcap -1)$
 by (*metis maddux-3-11-pp regular-one-closed*)
 also have ... $= x^T \sqcap (x * x)^* \sqcap 1$
 using 5 by *simp*
 also have ... $= x^T \sqcap 1$
 by (*metis calculation comp-inf.semiring.distrib-left*
inf.sup-monoid.add-commute star.circ-transitive-equal star-involutive
star-left-unfold-equal sup-inf-absorb)
 finally have $(x^T \sqcap (x * x)^*) \sqcup (x^T \sqcap (x^T * x^T)^*) \leq (1 \sqcap x) * \text{top}$
 using 4 *inf.sup-monoid.add-commute one-inf-conv top-right-mult-increasing*
 by *auto*
 hence $x^T \sqcap ((x * x)^* \sqcup (x * x)^{T*}) \leq (1 \sqcap x) * \text{top}$
 by (*simp add: comp-inf.semiring.distrib-left conv-dist-comp*)
 hence 6: $x^T \sqcap (x * x)^{T*} * (x * x)^* \leq (1 \sqcap x) * \text{top}$
 using 1 by (*simp add: cancel-separate-eq sup-commute*)
 have $(x * x)^{T*} \sqcap x^T * (x * x)^{T*} \leq (x^T \sqcap (x * x)^{T*} * (x * x)^*) * (x * x)^{T*}$
 by (*metis conv-involutive conv-star-commute dedekind-2 inf-commute*)
 also have ... $\leq (1 \sqcap x) * \text{top} * (x * x)^{T*}$
 using 6 by (*simp add: mult-left-isotone*)
 also have ... $= (1 \sqcap x) * \text{top}$
 by (*simp add: comp-associative star.circ-left-top*)
 finally have $(x * x)^{T*} \sqcap x^T * (x * x)^{T*} = (x * x)^{T*} \sqcap x^T * (x * x)^{T*} \sqcap (1 \sqcap x) * \text{top}$
 using *inf.order-iff* by *auto*
 also have ... $= (1 \sqcap x) * ((x * x)^{T*} \sqcap x^T * (x * x)^{T*})$
 by (*metis coreflexive-comp-top-inf inf.cobounded1*
inf.sup-monoid.add-commute)
 finally show ?thesis

qed

lemma *update-square-plus*:

point $y \implies x[y \longrightarrow x[[x[[y]]]]] \leq x^+$

by (*meson update-square comp-isotone dual-order.trans le-supI order-refl*)

star.circ-increasing star.circ-mult-increasing)

lemma *update-square-ub-plus*:

$x[y \mapsto (x * x)^T] \leq x^+$

by (*simp add: comp-isotone inf.coboundedI2 star.circ-increasing star.circ-mult-increasing*)

Theorem 6.2

lemma *acyclic-square*:

assumes *acyclic* ($x - 1$)

shows $x * x \sqcap 1 = x \sqcap 1$

proof (*rule order.antisym*)

have $1 \sqcap x * x = 1 \sqcap ((x - 1) * x \sqcup (x \sqcap 1) * x)$

by (*metis maddux-3-11-pp regular-one-closed semiring.distrib-right*)

also have $\dots \leq 1 \sqcap ((x - 1) * x \sqcup x)$

by (*metis inf.cobounded2 mult-1-left mult-left-isotone inf.sup-right-isotone semiring.add-left-mono*)

also have $\dots = 1 \sqcap ((x - 1) * (x - 1) \sqcup (x - 1) * (x \sqcap 1) \sqcup x)$

by (*metis maddux-3-11-pp mult-left-dist-sup regular-one-closed*)

also have $\dots \leq (1 \sqcap (x - 1) * (x - 1)) \sqcup (x - 1) * (x \sqcap 1) \sqcup x$

by (*metis inf-le2 inf-sup-distrib1 semiring.add-left-mono sup-monoid.add-assoc*)

also have $\dots \leq (1 \sqcap (x - 1)^+) \sqcup (x - 1) * (x \sqcap 1) \sqcup x$

by (*metis comp-isotone inf.eq-refl inf.sup-right-isotone star.circ-increasing sup-monoid.add-commute sup-right-isotone*)

also have $\dots = (x - 1) * (x \sqcap 1) \sqcup x$

by (*metis assms inf.le-iff-sup inf.sup-monoid.add-commute inf-import-p inf-p regular-one-closed sup-inf-absorb sup-monoid.add-commute*)

also have $\dots = x$

by (*metis comp-isotone inf.cobounded1 inf-le2 mult-1-right sup.absorb2*)

finally show $x * x \sqcap 1 \leq x \sqcap 1$

by (*simp add: inf.sup-monoid.add-commute*)

show $x \sqcap 1 \leq x * x \sqcap 1$

by (*metis coreflexive-idempotent inf-le1 inf-le2 le-infI mult-isotone*)

qed

lemma *diagonal-update-square-aux*:

assumes *acyclic* ($x - 1$)

and point y

shows $1 \sqcap y \sqcap y^T * x * x = 1 \sqcap y \sqcap x$

proof –

have $1: 1 \sqcap y \sqcap x \leq y^T * x * x$

by (*metis comp-isotone coreflexive-idempotent inf.boundedE inf.cobounded1 inf.cobounded2 one-inf-conv*)

have $1 \sqcap y \sqcap y^T * x * x = 1 \sqcap (y \sqcap y^T) * x * x$

by (*simp add: assms(2) inf.sup-monoid.add-assoc vector-inf-comp*)

also have $\dots = 1 \sqcap (y \sqcap 1) * x * x$

by (*metis assms(2) inf.cobounded1 inf.sup-monoid.add-commute inf.sup-same-context one-inf-conv vector-covector*)

also have $\dots \leq 1 \sqcap x * x$
by (*metis comp-left-subdist-inf inf.sup-right-isotone le-infE mult-left-isotone mult-left-one*)
also have $\dots \leq x$
using *assms(1) acyclic-square inf.sup-monoid.add-commute* **by** *auto*
finally show *?thesis*
using *1* **by** (*metis inf.absorb2 inf.left-commute inf.sup-monoid.add-commute*)
qed

Theorem 6.5

lemma *diagonal-update-square*:

assumes *acyclic (x - 1)*
and *point y*
shows $(x[y \longrightarrow x[[x[[y]]]]) \sqcap 1 = x \sqcap 1$
proof –
let $?xy = x[[y]]$
let $?xxy = x[[?xy]]$
let $?xyxxy = x[y \longrightarrow ?xxy]$
have $?xyxxy \sqcap 1 = ((y \sqcap y^T * x * x) \sqcup (-y \sqcap x)) \sqcap 1$
by (*simp add: conv-dist-comp*)
also have $\dots = (y \sqcap y^T * x * x \sqcap 1) \sqcup (-y \sqcap x \sqcap 1)$
by (*simp add: inf-sup-distrib2*)
also have $\dots = (y \sqcap x \sqcap 1) \sqcup (-y \sqcap x \sqcap 1)$
using *assms* **by** (*smt (verit, ccfv-threshold) diagonal-update-square-aux find-set-precondition-def inf-assoc inf-commute*)
also have $\dots = x \sqcap 1$
by (*metis assms(2) bijective-regular comp-inf.mult-right-dist-sup inf.sup-monoid.add-commute maddux-3-11-pp*)
finally show *?thesis*

qed

Theorem 6.6

lemma *fc-update-square*:

assumes *mapping x*
and *point y*
shows $fc (x[y \longrightarrow x[[x[[y]]]]) = fc x$
proof (*rule order.antisym*)
let $?xy = x[[y]]$
let $?xxy = x[[?xy]]$
let $?xyxxy = x[y \longrightarrow ?xxy]$
have $1: y \sqcap y^T * x * x \leq x * x$
by (*smt (z3) assms(2) inf.cobounded2 inf.sup-monoid.add-commute inf.sup-same-context mult-1-left one-inf-conv vector-covector vector-inf-comp*)
have $2: ?xyxxy = (y \sqcap y^T * x * x) \sqcup (-y \sqcap x)$
by (*simp add: conv-dist-comp*)
also have $\dots \leq x * x \sqcup x$
using *1* *inf-le2 sup-mono* **by** *blast*
also have $\dots \leq x^*$

by (*simp add: star.circ-increasing star.circ-mult-upper-bound*)
finally show $fc \ ?xyxy \leq fc \ x$
 by (*metis comp-isotone conv-order conv-star-commute star-involutive star-isotone*)
have $3: y \sqcap x \sqcap 1 \leq fc \ ?xyxy$
using *inf.coboundedI1 inf.sup-monoid.add-commute reflexive-mult-closed star.circ-reflexive* **by auto**
have $4: y - 1 \leq -y^T$
using *assms(2) p-shunting-swap regular-one-closed vector-covector* **by auto**
have $y \sqcap x \leq y^T * x$
by (*simp add: assms(2) vector-restrict-comp-conv*)
also have $\dots \leq y^T * x * x * x^T$
by (*metis assms(1) comp-associative mult-1-right mult-right-isotone total-var*)
finally have $y \sqcap x \sqcap -1 \leq y \sqcap -y^T \sqcap y^T * x * x * x^T$
using 4 **by** (*smt (z3) inf.cobounded1 inf.coboundedI2 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute inf-greatest*)
also have $\dots = (y \sqcap y^T * x * x) * x^T \sqcap -y^T$
by (*metis assms(2) inf.sup-monoid.add-assoc inf.sup-monoid.add-commute vector-inf-comp*)
also have $\dots = (y \sqcap y^T * x * x) * (x^T \sqcap -y^T)$
using *assms(2) covector-comp-inf vector-conv-compl* **by auto**
also have $\dots = (y \sqcap y^T * x * x) * (-y \sqcap x)^T$
by (*simp add: conv-complement conv-dist-inf inf-commute*)
also have $\dots \leq ?xyxy * (-y \sqcap x)^T$
using 2 **by** (*simp add: comp-left-increasing-sup*)
also have $\dots \leq ?xyxy * ?xyxy^T$
by (*simp add: conv-isotone mult-right-isotone*)
also have $\dots \leq fc \ ?xyxy$
using *comp-isotone star.circ-increasing* **by blast**
finally have $5: y \sqcap x \leq fc \ ?xyxy$
using 3 **by** (*smt (z3) comp-inf.semiring.distrib-left inf.le-iff-sup maddux-3-11-pp regular-one-closed*)
have $x = (y \sqcap x) \sqcup (-y \sqcap x)$
by (*metis assms(2) bijective-regular inf.sup-monoid.add-commute maddux-3-11-pp*)
also have $\dots \leq fc \ ?xyxy$
using 5 *dual-order.trans fc-increasing sup.cobounded2 sup-least* **by blast**
finally show $fc \ x \leq fc \ ?xyxy$
by (*smt (z3) assms fc-equivalence fc-isotone fc-wcc read-injective star.circ-decompose-9 star-decompose-1 update-univalent*)
qed

Theorem 6.2

lemma *acyclic-plus-loop*:
assumes *acyclic* $(x - 1)$
shows $x^+ \sqcap 1 = x \sqcap 1$
proof –
let $?r = x \sqcap 1$
let $?i = x - 1$

have $x^+ \sqcap 1 = (?i \sqcup ?r)^+ \sqcap 1$
by (*metis maddux-3-11-pp regular-one-closed*)
also have $\dots = ((?i^* * ?r)^* * ?i^+ \sqcup (?i^* * ?r)^+) \sqcap 1$
using *plus-sup* **by** *auto*
also have $\dots \leq (?i^+ \sqcup (?i^* * ?r)^+) \sqcap 1$
by (*metis comp-associative dual-order.eq-iff maddux-3-11-pp*
reachable-without-loops regular-one-closed star.circ-plus-same star.circ-sup-9)
also have $\dots = (?i^* * ?r)^+ \sqcap 1$
by (*smt (z3) assms comp-inf.mult-right-dist-sup inf.absorb2*
inf.sup-monoid.add-commute inf-le2 maddux-3-11-pp pseudo-complement
regular-one-closed)
also have $\dots \leq ?i^* * ?r \sqcap 1$
by (*metis comp-associative dual-order.eq-iff maddux-3-11-pp*
reachable-without-loops regular-one-closed star.circ-sup-9 star-slide)
also have $\dots = (?r \sqcup ?i^+ * ?r) \sqcap 1$
using *comp-associative star.circ-loop-fixpoint sup-commute* **by** *force*
also have $\dots \leq x \sqcup (?i^+ * ?r \sqcap 1)$
by (*metis comp-inf.mult-right-dist-sup inf.absorb1 inf.cobounded1*
inf.cobounded2)
also have $\dots \leq x \sqcup (-1 * ?r \sqcap 1)$
by (*meson assms comp-inf.comp-isotone mult-left-isotone order.refl*
semiring.add-left-mono)
also have $\dots = x$
by (*metis comp-inf.semiring.mult-not-zero comp-right-one inf.cobounded2*
inf-sup-absorb mult-right-isotone pseudo-complement sup.idem sup-inf-distrib1)
finally show *?thesis*
by (*meson inf.sup-same-context inf-le1 order-trans star.circ-mult-increasing*)
qed

lemma *star-irreflexive-part-eq*:

$$x^* - 1 = (x - 1)^+ - 1$$

by (*metis reachable-without-loops star-plus-without-loops*)

Theorem 6.3

lemma *star-irreflexive-part*:

$$x^* - 1 \leq (x - 1)^+$$

using *star-irreflexive-part-eq* **by** *auto*

lemma *square-irreflexive-part*:

$$x * x - 1 \leq (x - 1)^+$$

proof –

have $x * x = (x \sqcap 1) * x \sqcup (x - 1) * x$

by (*metis maddux-3-11-pp mult-right-dist-sup regular-one-closed*)

also have $\dots \leq 1 * x \sqcup (x - 1) * x$

using *comp-isotone inf.cobounded2 semiring.add-right-mono* **by** *blast*

also have $\dots \leq 1 \sqcup (x - 1) \sqcup (x - 1) * x$

by (*metis inf.cobounded2 maddux-3-11-pp mult-1-left regular-one-closed*
sup-left-isotone)

also have $\dots = (x - 1) * (x \sqcup 1) \sqcup 1$

by (*simp add: mult-left-dist-sup sup-assoc sup-commute*)
finally have $x * x - 1 \leq (x - 1) * (x \sqcup 1)$
 using *shunting-var-p* **by** *auto*
also have $\dots = (x - 1) * (x - 1) \sqcup (x - 1)$
 by (*metis comp-right-one inf.sup-monoid.add-commute maddux-3-21-pp*
mult-left-dist-sup regular-one-closed sup-commute)
also have $\dots \leq (x - 1)^+$
 by (*metis mult-left-isotone star.circ-increasing star.circ-mult-increasing*
star.circ-plus-same sup.bounded-iff)
finally show *?thesis*

.
qed

Theorem 6.3

lemma *square-irreflexive-part-2:*

$x * x - 1 \leq x^* - 1$
 using *comp-inf.mult-left-isotone star.circ-increasing star.circ-mult-upper-bound*
by *blast*

Theorem 6.8

lemma *acyclic-update-square:*

assumes *acyclic* $(x - 1)$
shows *acyclic* $((x[y \mapsto (x * x)^T]) - 1)$
proof –
have $((x[y \mapsto (x * x)^T]) - 1)^+ \leq ((x \sqcup x * x) - 1)^+$
 by (*metis comp-inf.mult-right-isotone comp-isotone*
inf.sup-monoid.add-commute star-isotone update-square-ub)
also have $\dots = ((x - 1) \sqcup (x * x - 1))^+$
 using *comp-inf.semiring.distrib-right* **by** *auto*
also have $\dots \leq ((x - 1)^+)^+$
 by (*smt (verit, del-insts) comp-isotone reachable-without-loops*
star.circ-mult-increasing star.circ-plus-same star.circ-right-slide
star.circ-separate-5 star.circ-square star.circ-transitive-equal star.left-plus-circ
sup.bounded-iff sup-ge1 square-irreflexive-part)
also have $\dots \leq -1$
 using *assms* **by** (*simp add: acyclic-plus*)
finally show *?thesis*

.
qed

Theorem 6.9

lemma *disjoint-set-forest-update-square:*

assumes *disjoint-set-forest* x
and *vector* y
and *regular* y
shows *disjoint-set-forest* $(x[y \mapsto (x * x)^T])$
proof (*intro conjI*)
show *univalent* $(x[y \mapsto (x * x)^T])$
 using *assms update-univalent mapping-mult-closed univalent-conv-injective* **by**
blast

```

show total (x[y→(x * x)T])
  using assms update-total total-conv-surjective total-mult-closed by blast
show acyclic ((x[y→(x * x)T] - 1)
  using acyclic-update-square assms(1) by blast
qed

```

```

lemma disjoint-set-forest-update-square-point:
  assumes disjoint-set-forest x
    and point y
  shows disjoint-set-forest (x[y→(x * x)T])
  using assms disjoint-set-forest-update-square bijective-regular by blast

end

```

6 Verifying Further Operations on Disjoint-Set Forests

In this section we verify the `init-sets`, `path-halving` and `path-splitting` operations of disjoint-set forests.

```

class choose-point =
  fixes choose-point :: 'a ⇒ 'a

```

Using the `choose-point` operation we define a simple for-each-loop abstraction as syntactic sugar translated to a while-loop. Regular vector h describes the set of all elements that are yet to be processed. It is made explicit so that the invariant can refer to it.

```

syntax
  -Foreach :: idt ⇒ idt ⇒ 'assn ⇒ 'com ⇒ 'com ((1FOREACH -/ USING -/
  INV {-} //DO - /OD) [0,0,0,0] 61)
translations FOREACH x USING h INV { i } DO c OD =>
  h := CONST top;
  WHILE h ≠ CONST bot
  INV { CONST regular h ∧ CONST vector h ∧ i }
  VAR { h↓ }
  DO x := CONST choose-point h;
  c;
  h[x] := CONST bot
  OD

```

```

class stone-kleene-relation-algebra-choose-point-finite-regular =
  stone-kleene-relation-algebra + finite-regular-p-algebra + choose-point +
  assumes choose-point-point: vector x ⇒ x ≠ bot ⇒ point (choose-point x)
  assumes choose-point-decreasing: choose-point x ≤ --x
begin

```

```

subclass stone-kleene-relation-algebra-tarski-finite-regular
proof unfold-locales

```

```

fix x
let ?p = choose-point (x * top)
let ?q = choose-point ((?p  $\sqcap$  x)T * top)
let ?y = ?p  $\sqcap$  ?qT
assume 1: regular x x  $\neq$  bot
hence 2: x * top  $\neq$  bot
  using le-bot top-right-mult-increasing by auto
hence 3: point ?p
  by (simp add: choose-point-point comp-associative)
hence 4: ?p  $\neq$  bot
  using 2 mult-right-zero by force
have ?p  $\sqcap$  x  $\neq$  bot
proof
  assume ?p  $\sqcap$  x = bot
  hence 5: x  $\leq$  -?p
    using p-antitone-iff pseudo-complement by auto
  have ?p  $\leq$  --(x * top)
    by (simp add: choose-point-decreasing)
  also have ...  $\leq$  --(-?p * top)
    using 5 by (simp add: comp-isotone pp-isotone)
  also have ... = -?p * top
    using regular-mult-closed by auto
  also have ... = -?p
    using 3 vector-complement-closed by auto
  finally have ?p = bot
    using inf-absorb2 by fastforce
  thus False
    using 4 by auto
qed
hence (?p  $\sqcap$  x)T * top  $\neq$  bot
  by (metis comp-inf.semiring.mult-zero-left comp-right-one
  inf.sup-monoid.add-commute inf-top.left-neutral schroeder-1)
hence point ?q
  using choose-point-point vector-top-closed mult-assoc by auto
hence 6: arc ?y
  using 3 by (smt bijective-conv-mapping inf.sup-monoid.add-commute
  mapping-inf-point-arc)
have ?q  $\leq$  --((?p  $\sqcap$  x)T * top)
  by (simp add: choose-point-decreasing)
hence ?y  $\leq$  ?p  $\sqcap$  --((?p  $\sqcap$  x)T * top)T
  by (metis conv-complement conv-isotone inf.sup-right-isotone)
also have ... = ?p  $\sqcap$  --(top * (?p  $\sqcap$  x))
  by (simp add: conv-dist-comp)
also have ... = ?p  $\sqcap$  top * (?p  $\sqcap$  x)
  using 1 3 bijective-regular pp-dist-comp by auto
also have ... = ?p  $\sqcap$  ?pT * x
  using 3 by (metis comp-inf-vector conv-dist-comp
  inf.sup-monoid.add-commute inf-top-right symmetric-top-closed)
also have ... = (?p  $\sqcap$  ?pT) * x

```

```

    using 3 by (simp add: vector-inf-comp)
  also have ... ≤ 1 * x
    using 3 point-antisymmetric mult-left-isotone by blast
  finally have ?y ≤ x
    by simp
  thus top * x * top = top
    using 6 by (smt (verit, ccfv-SIG) mult-assoc le-iff-sup mult-left-isotone
semiring.distrib-left sup.orderE top.extremum)
qed

```

6.1 Init-Sets

A disjoint-set forest is initialised by applying *make-set* to each node. We prove that the resulting disjoint-set forest is the identity relation.

theorem *init-sets*:

```

  VARS h p x
  [ True ]
  FOREACH x
  USING h
  INV { p - h = 1 - h }
  DO p := make-set p x
  OD
  [ p = 1 ∧ disjoint-set-forest p ∧ h = bot ]

```

proof *vcg-tc-simp*

fix *h p*

let *?x = choose-point h*

let *?m = make-set p ?x*

assume 1: *regular h ∧ vector h ∧ p - h = 1 - h ∧ h ≠ bot*

show *vector (-?x □ h) ∧*

?m □ (--?x ⊔ -h) = 1 □ (--?x ⊔ -h) ∧

card { x . regular x ∧ x ≤ -?x ∧ x ≤ h } < h↓

proof (*intro conjI*)

show *vector (-?x □ h)*

using 1 *choose-point-point vector-complement-closed vector-inf-closed* **by**

blast

have 2: *point ?x ∧ regular ?x*

using 1 *bijective-regular choose-point-point* **by** *blast*

have 3: *-h ≤ -?x*

using *choose-point-decreasing p-antitone-iff* **by** *auto*

have 4: *?x □ ?m = ?x * ?x^T ∧ -?x □ ?m = -?x □ p*

using 1 *choose-point-point make-set-function make-set-postcondition-def* **by**

auto

have *?m □ (--?x ⊔ -h) = (?m □ ?x) ⊔ (?m - h)*

using 2 *comp-inf.comp-left-dist-sup* **by** *auto*

also have ... = *?x * ?x^T ⊔ (?m □ -?x □ -h)*

using 3 4 **by** (*smt (z3) inf-absorb2 inf-assoc inf-commute*)

also have ... = *?x * ?x^T ⊔ (1 - h)*

using 1 3 4 *inf.absorb2 inf.sup-monoid.add-assoc inf-commute* **by** *auto*

also have ... = *(1 □ ?x) ⊔ (1 - h)*

```

    using 2 by (metis inf.cobounded2 inf.sup-same-context one-inf-conv
vector-covector)
    also have ... = 1  $\sqcap$  ( $--?x \sqcup -h$ )
    using 2 comp-inf.semiring.distrib-left by auto
    finally show ?m  $\sqcap$  ( $--?x \sqcup -h$ ) = 1  $\sqcap$  ( $--?x \sqcup -h$ )
    .
    have 5:  $\neg ?x \leq -?x$ 
    using 1 2 by (metis comp-commute-below-diversity conv-order
inf.cobounded2 inf.absorb2 pseudo-complement strict-order-var top.extremum)
    have 6:  $?x \leq h$ 
    using 1 by (metis choose-point-decreasing)
    show card {  $x . \text{regular } x \wedge x \leq -?x \wedge x \leq h$  } <  $h \downarrow$ 
    apply (rule psubset-card-mono)
    using finite-regular apply simp
    using 2 5 6 by auto
qed
qed
end

```

6.2 Path Halving

Path halving is a variant of the path compression technique. Similarly to path compression, we implement path halving independently of find-set, using a second while-loop which iterates over the same path to the root. We prove that path halving preserves the equivalence-relational semantics of the disjoint-set forest and also preserves the roots of the component trees. Additionally we prove the exact effect of path halving, which is to replace every other parent pointer with a pointer to the respective grandparent.

```

context stone-kleene-relation-algebra-tarski-finite-regular
begin

```

```

definition path-halving-invariant p x y p0  $\equiv$ 
  find-set-precondition p x  $\wedge$  point y  $\wedge$   $y \leq p^{T^*} * x \wedge y \leq (p0 * p0)^{T^*} * x \wedge$ 
  p0[(p0 * p0)^{T^*} * x - p0^{T^*} * y  $\longrightarrow$  (p0 * p0)^T] = p  $\wedge$ 
  disjoint-set-forest p0

```

```

definition path-halving-postcondition p x y p0  $\equiv$ 
  path-compression-precondition p x y  $\wedge$  p  $\sqcap$  1 = p0  $\sqcap$  1  $\wedge$  fc p = fc p0  $\wedge$ 
  p0[(p0 * p0)^{T^*} * x  $\longrightarrow$  (p0 * p0)^T] = p

```

```

lemma path-halving-invariant-aux-1:

```

```

  assumes point x
    and point y
    and disjoint-set-forest p0
  shows p0  $\leq$  wcc (p0[(p0 * p0)^{T^*} * x - p0^{T^*} * y  $\longrightarrow$  (p0 * p0)^T])
proof -
  let ?p2 = p0 * p0
  let ?p2t = ?p2^T

```

let $?p2ts = ?p2t^*$
let $?px = ?p2ts * x$
let $?py = -(p0^{T^*} * y)$
let $?pxy = ?px \sqcap ?py$
let $?p = p0[?pxy \longrightarrow ?p2t]$
have 1: *regular* $?pxy$
using *assms(1,3) bijective-regular find-set-precondition-def mapping-regular pp-dist-comp regular-closed-star regular-conv-closed path-halving-invariant-def* **by** *auto*
have 2: *vector* $x \wedge$ *vector* $?px \wedge$ *vector* $?py$
using *assms(1,2) find-set-precondition-def vector-complement-closed vector-mult-closed path-halving-invariant-def* **by** *auto*
have 3: $?pxy \sqcap p0 \sqcap -?p2 \leq -?px^T$
proof –
have 4: *injective* $x \wedge$ *univalent* $?p2 \wedge$ *regular* $p0$
using *assms(1,3) find-set-precondition-def mapping-regular univalent-mult-closed path-halving-invariant-def* **by** *auto*
have $?p2^* * p0 \sqcap 1 \leq p0^+ \sqcap 1$
using *comp-inf.mult-left-isotone comp-isotone comp-right-one mult-sub-right-one star.circ-square star-slide* **by** *auto*
also have $\dots \leq p0$
using *acyclic-plus-loop assms(3) path-halving-invariant-def* **by** *auto*
finally have 5: $?p2^* * p0 \sqcap 1 \leq p0$
hence 6: $?p2ts * (1 - p0) \leq -p0$
by (*smt (verit, ccfv-SIG) conv-star-commute dual-order.trans inf.sup-monoid.add-assoc order.refl p-antitone-iff pseudo-complement schroeder-4-p schroeder-6-p*)
have $?p2t^+ * p0 \sqcap 1 = ?p2ts * p0^T * (p0^T * p0) \sqcap 1$
by (*metis conv-dist-comp star-plus mult-assoc*)
also have $\dots \leq ?p2ts * p0^T \sqcap 1$
by (*metis assms(3) comp-inf.mult-left-isotone comp-isotone comp-right-one mult-sub-right-one*)
also have $\dots \leq p0$
using 5 **by** (*metis conv-dist-comp conv-star-commute inf-commute one-inf-conv star-slide*)
finally have $?p2t^+ * p0 \leq -1 \sqcup p0$
by (*metis regular-one-closed shunting-var-p sup-commute*)
hence 7: $?p2^+ * (1 - p0) \leq -p0$
by (*smt (z3) conv-dist-comp conv-star-commute half-shunting inf.sup-monoid.add-assoc inf.sup-monoid.add-commute pseudo-complement schroeder-4-p schroeder-6-p star.circ-plus-same*)
have $(1 \sqcap ?px) * top * (1 \sqcap ?px \sqcap -p0) = ?px \sqcap top * (1 \sqcap ?px \sqcap -p0)$
using 2 **by** (*metis inf-commute vector-inf-one-comp mult-assoc*)
also have $\dots = ?px \sqcap ?px^T * (1 - p0)$
using 2 **by** (*smt (verit, ccfv-threshold) covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute inf-top.left-neutral*)
also have $\dots = ?px \sqcap x^T * ?p2^* * (1 - p0)$
by (*simp add: conv-dist-comp conv-star-commute*)

also have $\dots = (?px \sqcap x^T) * ?p2^* * (1 - p0)$
using 2 *vector-inf-comp* **by** *auto*
also have $\dots = ?p2ts * (x * x^T) * ?p2^* * (1 - p0)$
using 2 *vector-covector mult-assoc* **by** *auto*
also have $\dots \leq ?p2ts * ?p2^* * (1 - p0)$
using 4 **by** (*metis inf.order-lesseq-imp mult-left-isotone*
star.circ-mult-upper-bound star.circ-reflexive)
also have $\dots = (?p2ts \sqcup ?p2^*) * (1 - p0)$
using 4 **by** (*simp add: cancel-separate-eq*)
also have $\dots = (?p2ts \sqcup ?p2^+) * (1 - p0)$
by (*metis star.circ-plus-one star-plus-loops sup-assoc sup-commute*)
also have $\dots \leq -p0$
using 6 7 **by** (*simp add: mult-right-dist-sup*)
finally have $(1 \sqcap ?px)^T * p0 * (1 \sqcap ?px \sqcap -p0)^T \leq bot$
by (*smt (z3) inf.boundedI inf-p top.extremum triple-schroeder-p*)
hence 8: $(1 \sqcap ?px) * p0 * (1 \sqcap ?px \sqcap -p0) = bot$
by (*simp add: coreflexive-inf-closed coreflexive-symmetric le-bot*)
have $?px \sqcap p0 \sqcap ?px^T = (1 \sqcap ?px) * p0 \sqcap ?px^T$
using 2 *inf-commute vector-inf-one-comp* **by** *fastforce*
also have $\dots = (1 \sqcap ?px) * p0 * (1 \sqcap ?px)$
using 2 **by** (*metis comp-inf-vector mult-1-right vector-conv-covector*)
also have $\dots = (1 \sqcap ?px) * p0 * (1 \sqcap ?px \sqcap p0) \sqcup (1 \sqcap ?px) * p0 * (1 \sqcap$
 $?px \sqcap -p0)$
using 4 **by** (*metis maddux-3-11-pp mult-left-dist-sup*)
also have $\dots = (1 \sqcap ?px) * p0 * (1 \sqcap ?px \sqcap p0)$
using 8 **by** *simp*
also have $\dots \leq ?p2$
by (*metis comp-isotone coreflexive-comp-top-inf inf.cobounded1*
inf.cobounded2)
finally have $?px \sqcap p0 \sqcap -?p2 \leq -?px^T$
using 4 *p-shunting-swap regular-mult-closed* **by** *fastforce*
thus *?thesis*
by (*meson comp-inf.mult-left-isotone dual-order.trans inf.cobounded1*)
qed
have $p0 \leq ?p2 * p0^T$
by (*metis assms(3) comp-associative comp-isotone comp-right-one eq-refl*
total-var)
hence $?pxy \sqcap p0 \sqcap -?p2 \leq ?p2 * p0^T$
by (*metis inf.coboundedI1 inf.sup-monoid.add-commute*)
hence $?pxy \sqcap p0 \sqcap -?p2 \leq ?pxy \sqcap ?p2 * p0^T \sqcap -?px^T$
using 3 **by** (*meson dual-order.trans inf.boundedI inf.cobounded1*)
also have $\dots = (?pxy \sqcap ?p2) * p0^T \sqcap -?px^T$
using 2 *vector-inf-comp* **by** *auto*
also have $\dots = (?pxy \sqcap ?p2) * (-?px \sqcap p0)^T$
using 2 **by** (*simp add: covector-comp-inf inf.sup-monoid.add-commute*
vector-conv-compl conv-complement conv-dist-inf)
also have $\dots \leq ?p * (-?px \sqcap p0)^T$
using *comp-left-increasing-sup* **by** *auto*
also have $\dots \leq ?p * ?p^T$

by (*metis comp-inf.mult-right-isotone comp-isotone conv-isotone inf.eq-refl*
inf.sup-monoid.add-commute le-supI1 p-antitone-inf sup-commute)
 also have $\dots \leq wcc \ ?p$
 using *star.circ-sub-dist-2* by *auto*
 finally have $9: \ ?pxy \sqcap p0 \sqcap -?p2 \leq wcc \ ?p$
 .
 have $p0 = (\ ?pxy \sqcap p0) \sqcup (-?pxy \sqcap p0)$
 using *1* by (*metis inf.sup-monoid.add-commute maddux-3-11-pp*)
 also have $\dots \leq (\ ?pxy \sqcap p0) \sqcup \ ?p$
 using *sup-right-isotone* by *auto*
 also have $\dots = (\ ?pxy \sqcap p0 \sqcap -?p2) \sqcup (\ ?pxy \sqcap p0 \sqcap \ ?p2) \sqcup \ ?p$
 by (*smt (z3) assms(3) maddux-3-11-pp mapping-regular pp-dist-comp*
path-halving-invariant-def)
 also have $\dots \leq (\ ?pxy \sqcap p0 \sqcap -?p2) \sqcup (\ ?pxy \sqcap \ ?p2) \sqcup \ ?p$
 by (*meson comp-inf.comp-left-subdist-inf inf.boundedE semiring.add-left-mono*
semiring.add-right-mono)
 also have $\dots = (\ ?pxy \sqcap p0 \sqcap -?p2) \sqcup \ ?p$
 using *sup-assoc* by *auto*
 also have $\dots \leq wcc \ ?p \sqcup \ ?p$
 using *9 sup-left-isotone* by *blast*
 also have $\dots \leq wcc \ ?p$
 by (*simp add: star.circ-sub-dist-1*)
 finally show *?thesis*
 .
 qed

lemma *path-halving-invariant-aux*:

assumes *path-halving-invariant p x y p0*
 shows $p[[y]] = p0[[y]]$
 and $p[[p[[y]]]] = p0[[p0[[y]]]]$
 and $p[[p[[p[[y]]]]]] = p0[[p0[[p0[[y]]]]]]$
 and $p \sqcap 1 = p0 \sqcap 1$
 and $fc \ p = fc \ p0$
 proof –
 let $\ ?p2 = p0 * p0$
 let $\ ?p2t = \ ?p2^T$
 let $\ ?p2ts = \ ?p2t^*$
 let $\ ?px = \ ?p2ts * x$
 let $\ ?py = -(p0^{T*} * y)$
 let $\ ?pxy = \ ?px \sqcap \ ?py$
 let $\ ?p = p0[\ ?pxy \mapsto \ ?p2t]$
 have $\ ?p[[y]] = p0[[y]]$
 apply (*rule put-get-different-vector*)
 using *assms find-set-precondition-def vector-complement-closed*
vector-inf-closed vector-mult-closed path-halving-invariant-def apply *force*
 by (*meson inf.cobounded2 order-lesseq-imp p-antitone-iff path-compression-1b*)
 thus $1: p[[y]] = p0[[y]]$
 using *assms path-halving-invariant-def* by *auto*
 have $\ ?p[[p0[[y]]]] = p0[[p0[[y]]]]$

apply (rule put-get-different-vector)
using *assms find-set-precondition-def vector-complement-closed*
vector-inf-closed vector-mult-closed path-halving-invariant-def **apply** force
by (*metis comp-isotone inf.boundedE inf.coboundedI2 inf.eq-refl p-antitone-iff*
selection-closed-id star.circ-increasing)
thus 2: $p[[p[[y]]]] = p0[[p0[[y]]]]$
using 1 *assms path-halving-invariant-def* **by** auto
have ? $p[[p0[[p0[[y]]]]]] = p0[[p0[[p0[[y]]]]]]$
apply (rule put-get-different-vector)
using *assms find-set-precondition-def vector-complement-closed*
vector-inf-closed vector-mult-closed path-halving-invariant-def **apply** force
by (*metis comp-associative comp-isotone conv-dist-comp conv-involutive*
conv-order inf.coboundedI2 inf.le-iff-sup mult-left-isotone p-antitone-iff
p-antitone-inf star.circ-increasing star.circ-transitive-equal)
thus $p[[p[[p[[y]]]]]] = p0[[p0[[p0[[y]]]]]]$
using 2 *assms path-halving-invariant-def* **by** auto
have 3: regular ? pxy
using *assms bijective-regular find-set-precondition-def mapping-regular*
pp-dist-comp regular-closed-star regular-conv-closed path-halving-invariant-def **by**
auto
have $p \sqcap 1 = ?p \sqcap 1$
using *assms path-halving-invariant-def* **by** auto
also have ... = ($?pxy \sqcap ?p2 \sqcap 1$) \sqcup ($-?pxy \sqcap p0 \sqcap 1$)
using *comp-inf.semiring.distrib-right conv-involutive* **by** auto
also have ... = ($?pxy \sqcap p0 \sqcap 1$) \sqcup ($-?pxy \sqcap p0 \sqcap 1$)
using *assms acyclic-square path-halving-invariant-def*
inf.sup-monoid.add-assoc **by** auto
also have ... = ($?pxy \sqcup -?pxy$) $\sqcap p0 \sqcap 1$
using *inf-sup-distrib2* **by** auto
also have ... = $p0 \sqcap 1$
using 3 **by** (*metis inf.sup-monoid.add-commute inf-sup-distrib1*
maddux-3-11-pp)
finally show $p \sqcap 1 = p0 \sqcap 1$
.
have $p \leq p0^+$
by (*metis assms path-halving-invariant-def update-square-ub-plus*)
hence 4: $fc p \leq fc p0$
using *conv-plus-commute fc-isotone star.left-plus-circ* **by** fastforce
have $wcc p0 \leq wcc ?p$
by (*meson assms wcc-below-wcc path-halving-invariant-aux-1*
path-halving-invariant-def find-set-precondition-def)
hence $fc p0 \leq fc ?p$
using *assms find-set-precondition-def path-halving-invariant-def fc-wcc* **by** auto
thus $fc p = fc p0$
using 4 *assms path-halving-invariant-def* **by** auto
qed

lemma *path-halving-1*:

find-set-precondition $p0 x \implies$ *path-halving-invariant* $p0 x x p0$

proof –
assume 1: *find-set-precondition* $p0$ x
show *path-halving-invariant* $p0$ x $p0$
proof (*unfold path-halving-invariant-def, intro conjI*)
show *find-set-precondition* $p0$ x
using 1 **by** *simp*
show *vector* x *injective* x *surjective* x
using 1 *find-set-precondition-def* **by** *auto*
show $x \leq p0^{T^*} * x$
by (*simp add: path-compression-1b*)
show $x \leq (p0 * p0)^{T^*} * x$
by (*simp add: path-compression-1b*)
have $(p0 * p0)^{T^*} * x \leq p0^{T^*} * x$
by (*simp add: conv-dist-comp mult-left-isotone star.circ-square*)
thus $p0[(p0 * p0)^{T^*} * x - p0^{T^*} * x \longrightarrow (p0 * p0)^T] = p0$
by (*smt (z3) inf.le-iff-sup inf-commute maddux-3-11-pp p-antitone-inf pseudo-complement*)
show *univalent* $p0$ *total* $p0$ *acyclic* ($p0 - 1$)
using 1 *find-set-precondition-def* **by** *auto*
qed
qed

lemma *path-halving-2*:

path-halving-invariant p x y $p0 \wedge y \neq p[[y]] \implies$ *path-halving-invariant*
 $(p[y \longrightarrow p[[p[[y]]]])$ x $((p[y \longrightarrow p[[p[[y]]]])[[y]])$ $p0 \wedge ((p[y \longrightarrow p[[p[[y]]]])^{T^*} * ((p[y \longrightarrow p[[p[[y]]]])[[y]])) \downarrow < (p^{T^*} * y) \downarrow$

proof –
let $?py = p[[y]]$
let $?ppy = p[[?py]]$
let $?pyppy = p[y \longrightarrow ?ppy]$
let $?p2 = p0 * p0$
let $?p2t = ?p2^T$
let $?p2ts = ?p2t^*$
let $?px = ?p2ts * x$
let $?py2 = -(p0^{T^*} * y)$
let $?pxy = ?px \sqcap ?py2$
let $?p = p0[?pxy \longrightarrow ?p2t]$
let $?pty = p0^T * y$
let $?pt2y = p0^T * p0^T * y$
let $?pt2sy = p0^{T^*} * p0^T * p0^T * y$
assume 1: *path-halving-invariant* p x y $p0 \wedge y \neq ?py$
have 2: *point* $?pty \wedge$ *point* $?pt2y$
using 1 **by** (*smt (verit) comp-associative read-injective read-surjective path-halving-invariant-def*)
show *path-halving-invariant* $?pyppy$ x $(?pyppy[[y]])$ $p0 \wedge (?pyppy^{T^*} * (?pyppy[[y]])) \downarrow < (p^{T^*} * y) \downarrow$
proof
show *path-halving-invariant* $?pyppy$ x $(?pyppy[[y]])$ $p0$
proof (*unfold path-halving-invariant-def, intro conjI*)

```

show 3: find-set-precondition ?pyppy x
proof (unfold find-set-precondition-def, intro conjI)
  show univalent ?pyppy
    using 1 find-set-precondition-def read-injective update-univalent
path-halving-invariant-def by auto
  show total ?pyppy
    using 1 bijjective-regular find-set-precondition-def read-surjective
update-total path-halving-invariant-def by force
  show acyclic (?pyppy - 1)
    apply (rule update-acyclic-3)
    using 1 find-set-precondition-def path-halving-invariant-def apply blast
    using 1 2 comp-associative path-halving-invariant-aux(2) apply force
    using 1 path-halving-invariant-def apply blast
    by (metis inf.order-lesseq-imp mult-isotone star.circ-increasing
star.circ-square mult-assoc)
  show vector x injective x surjective x
    using 1 find-set-precondition-def path-halving-invariant-def by auto
qed
show vector (?pyppy[[y]])
  using 1 comp-associative path-halving-invariant-def by auto
show injective (?pyppy[[y]])
  using 1 3 read-injective path-halving-invariant-def find-set-precondition-def
by auto
  show surjective (?pyppy[[y]])
    using 1 3 read-surjective path-halving-invariant-def
find-set-precondition-def by auto
show ?pyppy[[y]] ≤ ?pyppyT* * x
proof -
  have y = (y ∩ pT*) * x
    using 1 le-iff-inf vector-inf-comp path-halving-invariant-def by auto
  also have ... = ((y ∩ 1) ∪ (y ∩ (pT ∩ -yT)+)) * x
    using 1 omit-redundant-points-3 path-halving-invariant-def by auto
  also have ... ≤ (1 ∪ (y ∩ (pT ∩ -yT)+)) * x
    using 1 sup-inf-distrib2 vector-inf-comp path-halving-invariant-def by
auto
  also have ... ≤ (1 ∪ (pT ∩ -yT)+) * x
    by (simp add: inf.coboundedI2 mult-left-isotone)
  also have ... = (p ∩ -y)T* * x
    by (simp add: conv-complement conv-dist-inf star-left-unfold-equal)
  also have ... ≤ ?pyppyT* * x
    by (simp add: conv-isotone inf.sup-monoid.add-commute mult-left-isotone
star-isotone)
  finally show ?thesis
    by (metis mult-isotone star.circ-increasing star.circ-transitive-equal
mult-assoc)
qed
show ?pyppy[[y]] ≤ ?px
proof -
  have ?pyppy[[y]] = p[[?py]]

```

```

    using 1 put-get vector-mult-closed path-halving-invariant-def by force
  also have ... = p0[[p0[[y]]]]
    using 1 path-halving-invariant-aux(2) by blast
  also have ... = ?p2t * y
    by (simp add: conv-dist-comp mult-assoc)
  also have ... ≤ ?p2t * ?px
    using 1 path-halving-invariant-def comp-associative mult-right-isotone by
force
  also have ... ≤ ?px
    by (metis comp-associative mult-left-isotone star.left-plus-below-circ)
  finally show ?thesis
.
qed
show p0[?px - p0T * (?pyppy[[y]])] → ?p2t] = ?pyppy
proof -
  have ?px ⊓ ?pty = ?px ⊓ p0T * ?px ⊓ ?pty
    using 1 inf.absorb2 inf.sup-monoid.add-assoc mult-right-isotone
path-halving-invariant-def by force
  also have ... = (?p2ts ⊓ p0T * ?p2ts) * x ⊓ ?pty
    using 3 comp-associative find-set-precondition-def
injective-comp-right-dist-inf by auto
  also have ... = (1 ⊓ p0) * (?p2ts ⊓ p0T * ?p2ts) * x ⊓ ?pty
    using 1 even-odd-root mapping-regular path-halving-invariant-def by auto
  also have ... ≤ (1 ⊓ p0) * top ⊓ ?pty
    by (metis comp-associative comp-inf.mult-left-isotone
comp-inf.star.circ-sub-dist-2 comp-left-subdist-inf dual-order.trans
mult-right-isotone)
  also have 4: ... = (1 ⊓ p0T) * ?pty
    using coreflexive-comp-top-inf one-inf-conv by auto
  also have ... ≤ ?pt2y
    by (simp add: mult-assoc mult-left-isotone)
  finally have 5: ?px ⊓ ?pty ≤ ?pt2y
.
have 6: p[?px ⊓ -?pt2sy ⊓ ?pty] → ?p2t] = p
proof (cases ?pty ≤ ?px ⊓ -?pt2sy)
  case True
  hence ?pty ≤ ?pt2y
    using 5 conv-dist-comp inf.absorb2 by auto
  hence 7: ?pty = ?pt2y
    using 2 epm-3 by fastforce
  have p[?px ⊓ -?pt2sy ⊓ ?pty] → ?p2t] = p[?pty] → ?p2t]
    using True inf.absorb2 by auto
  also have ... = p[?pty] → ?p2[[?pty]]
    using 2 update-point-get by auto
  also have ... = p[?pty] → p0T * p0T * p0T * y]
    using comp-associative conv-dist-comp by auto
  also have ... = p[?pty] → ?pt2y]
    using 7 mult-assoc by simp
  also have ... = p[?pty] → p[[?pty]]

```

```

    using 1 path-halving-invariant-aux(1,2) mult-assoc by force
  also have ... = p
    using 2 get-put by auto
  finally show ?thesis
  .
next
case False
have mapping ?p2
  using 1 mapping-mult-closed path-halving-invariant-def by blast
hence 8: regular (?px  $\sqcap$  -?pt2sy)
  using 1 bijective-regular find-set-precondition-def mapping-regular
pp-dist-comp regular-closed-star regular-conv-closed path-halving-invariant-def by
auto
  have vector (?px  $\sqcap$  -?pt2sy)
    using 1 find-set-precondition-def vector-complement-closed
vector-inf-closed vector-mult-closed path-halving-invariant-def by force
  hence ?pty  $\leq$  -(?px  $\sqcap$  -?pt2sy)
    using 2 8 point-in-vector-or-complement False by blast
  hence ?px  $\sqcap$  -?pt2sy  $\sqcap$  ?pty = bot
    by (simp add: p-antitone-iff pseudo-complement)
  thus ?thesis
    by simp
qed
have 9: p[?px  $\sqcap$  -?pt2sy  $\sqcap$  y $\mapsto$ ?p2t] = ?pyppy
proof (cases y  $\leq$  -?pt2sy)
case True
hence p[?px  $\sqcap$  -?pt2sy  $\sqcap$  y $\mapsto$ ?p2t] = p[y $\mapsto$ ?p2t]
  using 1 inf.absorb2 path-halving-invariant-def by auto
also have ... = ?pyppy
  using 1 by (metis comp-associative conv-dist-comp
path-halving-invariant-aux(2) path-halving-invariant-def update-point-get)
  finally show ?thesis
  .
next
case False
have vector (-?pt2sy)
  using 1 vector-complement-closed vector-mult-closed
path-halving-invariant-def by blast
hence 10: y  $\leq$  ?pt2sy
  using 1 by (smt (verit, del-Insts) False bijective-regular
point-in-vector-or-complement regular-closed-star regular-mult-closed
total-conv-surjective univalent-conv-injective path-halving-invariant-def)
hence ?px  $\sqcap$  -?pt2sy  $\sqcap$  y = bot
  by (simp add: inf.coboundedI2 p-antitone pseudo-complement)
hence 11: p[?px  $\sqcap$  -?pt2sy  $\sqcap$  y $\mapsto$ ?p2t] = p
  by simp
have y  $\leq$  p0T+ * y
  using 10 by (metis mult-left-isotone order-lesseq-imp
star.circ-plus-same star.left-plus-below-circ)

```

hence 12: $y = \text{root } p0 \ y$
using 1 *loop-root-2 path-halving-invariant-def* **by** *blast*
have $?pyppy = p[y \mapsto p0[[p0[[y]]]]$
using 1 *path-halving-invariant-aux(2)* **by** *force*
also have $\dots = p[y \mapsto p0[[y]]]$
using 1 12 **by** (*metis root-successor-loop path-halving-invariant-def*)
also have $\dots = p[y \mapsto ?py]$
using 1 *path-halving-invariant-aux(1)* **by** *force*
also have $\dots = p$
using 1 *get-put path-halving-invariant-def* **by** *blast*
finally show *?thesis*
using 11 **by** *simp*
qed
have 13: $-\text{?pt2sy} = -(p0^{T^*} * y) \sqcup (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup (-\text{?pt2sy} \sqcap y)$
proof (*rule order.antisym*)
have 14: *regular* $(p0^{T^*} * y) \wedge \text{regular } ?pt2sy$
using 1 **by** (*metis order.antisym conv-complement conv-dist-comp*
conv-involutive conv-star-commute forest-components-increasing mapping-regular
pp-dist-star regular-mult-closed top.extremum path-halving-invariant-def)
have $p0^{T^*} = p0^{T^*} * p0^T * p0^T \sqcup p0^T \sqcup 1$
using *star.circ-back-loop-fixpoint star.circ-plus-same*
star-left-unfold-equal sup-commute **by** *auto*
hence $p0^{T^*} * y \leq ?pt2sy \sqcup ?pty \sqcup y$
by (*metis inf.eq-refl mult-1-left mult-right-dist-sup*)
also have $\dots = ?pt2sy \sqcup (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup y$
using 14 **by** (*metis maddux-3-21-pp*)
also have $\dots = ?pt2sy \sqcup (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup (-\text{?pt2sy} \sqcap y)$
using 14 **by** (*smt (z3) maddux-3-21-pp sup.left-commute sup-assoc*)
hence $p0^{T^*} * y \sqcap -\text{?pt2sy} \leq (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup (-\text{?pt2sy} \sqcap y)$
using *calculation half-shunting sup-assoc sup-commute* **by** *auto*
thus $-\text{?pt2sy} \leq -(p0^{T^*} * y) \sqcup (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup (-\text{?pt2sy} \sqcap y)$
using 14 **by** (*smt (z3) inf.sup-monoid.add-commute shunting-var-p*
sup.left-commute sup-commute)
have $-(p0^{T^*} * y) \leq -\text{?pt2sy}$
by (*meson mult-left-isotone order.trans p-antitone*)
star.right-plus-below-circ
thus $-(p0^{T^*} * y) \sqcup (-\text{?pt2sy} \sqcap \text{?pty}) \sqcup (-\text{?pt2sy} \sqcap y) \leq -\text{?pt2sy}$
by *simp*
qed
have *regular* $?px$ *regular* $?pty$ *regular* y
using 1 *bijective-regular find-set-precondition-def mapping-regular*
pp-dist-comp regular-closed-star regular-conv-closed path-halving-invariant-def **by** *auto*
hence 15: *regular* $(?px \sqcap -\text{?pt2sy} \sqcap \text{?pty})$ *regular* $(?px \sqcap -\text{?pt2sy} \sqcap y)$
by *auto*
have $p0[?px - p0^{T^*} * (?pyppy[[y]]) \mapsto ?p2t] = p0[?px - p0^{T^*} * (p[[?py]]) \mapsto ?p2t]$
using 1 *put-get vector-mult-closed path-halving-invariant-def* **by** *auto*
also have $\dots = p0[?px - \text{?pt2sy} \mapsto ?p2t]$

using 1 *comp-associative path-halving-invariant-aux(2)* **by force**
also have ... = $p0[?pxy \sqcup (?px \sqcap -?pt2sy \sqcap ?pty) \sqcup (?px \sqcap -?pt2sy \sqcap y) \multimap ?p2t]$
using 13 **by** (*metis comp-inf.semiring.distrib-left*
inf.sup-monoid.add-assoc)
also have ... = $(?p[?px \sqcap -?pt2sy \sqcap ?pty \multimap ?p2t])[?px \sqcap -?pt2sy \sqcap y \multimap ?p2t]$
using 15 **by** (*smt (z3) update-same-3 comp-inf.semiring.mult-not-zero*
inf.sup-monoid.add-assoc inf.sup-monoid.add-commute)
also have ... = $(p[?px \sqcap -?pt2sy \sqcap ?pty \multimap ?p2t])[?px \sqcap -?pt2sy \sqcap y \multimap ?p2t]$
using 1 *path-halving-invariant-def* **by auto**
also have ... = $p[?px \sqcap -?pt2sy \sqcap y \multimap ?p2t]$
using 6 **by simp**
also have ... = $?pyppy$
using 9 **by auto**
finally show *?thesis*
.

qed
show *univalent p0 total p0 acyclic (p0 - 1)*
using 1 *path-halving-invariant-def* **by auto**
qed

let $?s = \{ z . \text{regular } z \wedge z \leq p^{T^*} * y \}$
let $?t = \{ z . \text{regular } z \wedge z \leq ?pyppy^{T^*} * (?pyppy[[y]]) \}$
have $?pyppy^{T^*} * (?pyppy[[y]]) = ?pyppy^{T^*} * (p[[?py]])$
using 1 *put-get vector-mult-closed path-halving-invariant-def* **by force**
also have ... $\leq p^{+T^*} * (p[[?py]])$
using 1 *path-halving-invariant-def update-square-plus conv-order*
mult-left-isotone star-isotone **by force**
also have ... = $p^{T^*} * p^T * p^T * y$
by (*simp add: conv-plus-commute star.left-plus-circ mult-assoc*)
also have ... $\leq p^{T^+} * y$
by (*metis mult-left-isotone star.left-plus-below-circ star-plus*)
finally have 16: $?pyppy^{T^*} * (?pyppy[[y]]) \leq p^{T^+} * y$
.

hence $?pyppy^{T^*} * (?pyppy[[y]]) \leq p^{T^*} * y$
using *mult-left-isotone order-lesseq-imp star.left-plus-below-circ* **by blast**
hence 17: $?t \subseteq ?s$
using *order-trans* **by auto**
have 18: $y \in ?s$
using 1 *bijective-regular path-compression-1b path-halving-invariant-def* **by force**
have 19: $\neg y \in ?t$
proof
assume $y \in ?t$
hence $y \leq ?pyppy^{T^*} * (?pyppy[[y]])$
by simp
hence $y \leq p^{T^+} * y$
using 16 *dual-order.trans* **by blast**

```

    hence  $y = \text{root } p \ y$ 
      using 1 find-set-precondition-def loop-root-2 path-halving-invariant-def by
blast
    hence  $y = ?py$ 
      using 1 by (metis find-set-precondition-def root-successor-loop
path-halving-invariant-def)
    thus False
      using 1 by simp
    qed
    show  $\text{card } ?t < \text{card } ?s$ 
      apply (rule psubset-card-mono)
      subgoal using finite-regular by simp
      subgoal using 17 18 19 by auto
      done
    qed
  qed

```

lemma path-halving-3:

$\text{path-halving-invariant } p \ x \ y \ p0 \wedge y = p[[y]] \implies \text{path-halving-postcondition } p \ x \ y \ p0$

proof –

```

    assume 1: path-halving-invariant  $p \ x \ y \ p0 \wedge y = p[[y]]$ 
    show path-halving-postcondition  $p \ x \ y \ p0$ 
    proof (unfold path-halving-postcondition-def path-compression-precondition-def,
intro conjI)
      show univalent  $p \ \text{total } p \ \text{acyclic } (p - 1)$ 
        using 1 find-set-precondition-def path-halving-invariant-def by blast+
      show vector  $x \ \text{injective } x \ \text{surjective } x$ 
        using 1 find-set-precondition-def path-halving-invariant-def by blast+
      show 2: vector  $y \ \text{injective } y \ \text{surjective } y$ 
        using 1 path-halving-invariant-def by blast+
      have find-set-invariant  $p \ x \ y$ 
        using 1 find-set-invariant-def path-halving-invariant-def by blast
      thus  $y = \text{root } p \ x$ 
        using 1 find-set-3 find-set-postcondition-def by blast
      show  $p \sqcap 1 = p0 \sqcap 1$ 
        using 1 path-halving-invariant-aux(4) by blast
      show  $\text{fc } p = \text{fc } p0$ 
        using 1 path-halving-invariant-aux(5) by blast
      have 3:  $y = p0[[y]]$ 
        using 1 path-halving-invariant-aux(1) by auto
      hence  $p0^{T*} * y = y$ 
        using order.antisym path-compression-1b star-left-induct-mult-equal by auto
      hence 4:  $p0[(p0 * p0)^{T*} * x - y \longrightarrow (p0 * p0)^T] = p$ 
        using 1 path-halving-invariant-def by auto
      have  $(p0 * p0)^T * y = y$ 
        using 3 mult-assoc conv-dist-comp by auto
      hence  $y \sqcap p0 * p0 = y \sqcap p0$ 
        using 2 3 by (metis update-postcondition)
    qed
  qed

```



```

hence 5:  $y \sqcap p = y \sqcap p0 * p0$ 
  using 1 2 3 by (smt update-postcondition)
  have  $p0[(p0 * p0)^{T*} * x \mapsto (p0 * p0)^T] = (p0[(p0 * p0)^{T*} * x - y \mapsto (p0 * p0)^T])[(p0 * p0)^{T*} * x \sqcap y \mapsto (p0 * p0)^T]$ 
    using 1 bijjective-regular path-halving-invariant-def update-split by blast
    also have  $\dots = p[(p0 * p0)^{T*} * x \sqcap y \mapsto (p0 * p0)^T]$ 
      using 4 by simp
    also have  $\dots = p$ 
      apply (rule update-same-sub)
      using 5 apply simp
      apply simp
      using 1 bijjective-regular inf.absorb2 path-halving-invariant-def by auto
    finally show  $p0[(p0 * p0)^{T*} * x \mapsto (p0 * p0)^T] = p$ 
  .
qed
qed

```

theorem *find-path-halving*:

```

  VARs p y
  [ find-set-precondition p x  $\wedge$  p0 = p ]
  y := x;
  WHILE y  $\neq$  p[[y]]
    INV { path-halving-invariant p x y p0 }
    VAR { (pT* * y)↓ }
    DO p[y] := p[[p[[y]]]];
      y := p[[y]]
    OD
  [ path-halving-postcondition p x y p0 ]
apply vcg-tc-simp
  apply (fact path-halving-1)
  apply (fact path-halving-2)
by (fact path-halving-3)

```

6.3 Path Splitting

Path splitting is another variant of the path compression technique. We implement it again independently of *find-set*, using a second while-loop which iterates over the same path to the root. We prove that path splitting preserves the equivalence-relational semantics of the disjoint-set forest and also preserves the roots of the component trees. Additionally we prove the exact effect of path splitting, which is to replace every parent pointer with a pointer to the respective grandparent.

definition *path-splitting-invariant* p x y p0 \equiv
find-set-precondition p x \wedge *point* y \wedge $y \leq p0^{T*} * x \wedge$
 $p0[p0^{T*} * x - p0^{T*} * y \mapsto (p0 * p0)^T] = p \wedge$
disjoint-set-forest p0

definition *path-splitting-postcondition* p x y p0 \equiv
path-compression-precondition p x y \wedge $p \sqcap 1 = p0 \sqcap 1 \wedge$ *fc* p = *fc* p0 \wedge

$$p0[p0^{T*} * x \longrightarrow (p0 * p0)^T] = p$$

lemma *path-splitting-invariant-aux-1*:

assumes *point x*
and *point y*
and *disjoint-set-forest p0*
shows $(p0[p0^{T*} * x - p0^{T*} * y \longrightarrow (p0 * p0)^T]) \sqcap 1 = p0 \sqcap 1$
and $fc(p0[p0^{T*} * x - p0^{T*} * y \longrightarrow (p0 * p0)^T]) = fc\ p0$
and $p0^{T*} * x \leq p0^* * \text{root } p0\ x$
proof –
let $?p2 = p0 * p0$
let $?p2t = ?p2^T$
let $?px = p0^{T*} * x$
let $?py = -(p0^{T*} * y)$
let $?pxy = ?px \sqcap ?py$
let $?q1 = ?pxy \sqcap p0$
let $?q2 = -?pxy \sqcap p0$
let $?q3 = ?pxy \sqcap ?p2$
let $?q4 = -?pxy \sqcap ?p2$
let $?p = p0[?pxy \longrightarrow ?p2t]$
let $?r0 = \text{root } p0\ x$
let $?rp = \text{root } ?p\ x$
have 1: *regular ?px* \wedge *regular* $(p0^{T*} * y) \wedge$ *regular* $?pxy$
using *assms bijective-regular find-set-precondition-def mapping-regular pp-dist-comp regular-closed-star regular-conv-closed path-halving-invariant-def regular-closed-inf* **by** *auto*
have 2: *vector x* \wedge *vector* $?px \wedge$ *vector* $?py \wedge$ *vector* $?pxy$
using *assms(1,2) find-set-precondition-def vector-complement-closed vector-mult-closed path-halving-invariant-def vector-inf-closed* **by** *auto*
have 3: $?r0 \leq p0 * ?r0$
by (*metis assms(3) dedekind-1 inf.le-iff-sup root-successor-loop top-greatest*)
hence $?pxy \sqcap p0 * ?r0 \leq ?pxy \sqcap ?p2 * ?r0$
by (*metis comp-associative inf.eq-refl inf.sup-right-isotone mult-isotone*)
hence 4: $?q1 * ?r0 \leq ?q3 * ?r0$
using 2 **by** (*simp add: vector-inf-comp*)
have 5: $?q1 * ?q2 \leq ?q3$
using 2 **by** (*smt (z3) comp-isotone inf.cobounded1 inf.cobounded2 inf-greatest vector-export-comp*)
have $?q1 * ?q2^* * ?r0 = ?q1 * ?r0 \sqcup ?q1 * ?q2 * ?q2^* * ?r0$
by (*metis comp-associative semiring.distrib-left star.circ-loop-fixpoint sup-commute*)
also have $\dots \leq ?q1 * ?r0 \sqcup ?q3 * ?q2^* * ?r0$
using 5 **by** (*meson mult-left-isotone sup-right-isotone*)
also have $\dots \leq ?q3 * ?r0 \sqcup ?q3 * ?q2^* * ?r0$
using 4 *sup-left-isotone* **by** *blast*
also have $\dots = ?q3 * ?q2^* * ?r0$
by (*smt (verit, del-insts) comp-associative semiring.distrib-left star.circ-loop-fixpoint star.circ-transitive-equal star-involutive sup-commute*)
finally have 6: $?q1 * ?q2^* * ?r0 \leq ?q3 * ?q2^* * ?r0$

have $?q1 * (-?pxy \sqcap p0^+) * ?pxy \leq (?px \sqcap p0) * (-?pxy \sqcap p0^+) * ?pxy$
by (*meson comp-inf.comp-left-subdist-inf inf.boundedE mult-left-isotone*)
also have $\dots \leq (?px \sqcap p0) * (-?pxy \sqcap p0^+) * ?py$
by (*simp add: mult-right-isotone*)
also have $\dots \leq ?px^T * (-?pxy \sqcap p0^+) * ?py$
proof –
have $?px \sqcap p0 \leq ?px^T * p0$
using 2 **by** (*simp add: vector-restrict-comp-conv*)
also have $\dots \leq ?px^T$
by (*metis comp-associative conv-dist-comp conv-involutive conv-star-commute mult-right-isotone star.circ-increasing star.circ-transitive-equal*)
finally show *?thesis*
using *mult-left-isotone* **by** *auto*
qed
also have $\dots = top * (?px \sqcap -?pxy \sqcap p0^+) * ?py$
using 2 **by** (*smt (z3) comp-inf.star-plus conv-dist-inf covector-inf-comp-3 inf-top.right-neutral vector-complement-closed vector-inf-closed*)
also have $\dots \leq top * (-?py \sqcap p0^+) * ?py$
by (*metis comp-inf.comp-isotone comp-isotone inf.cobounded2 inf.eq-refl inf-import-p*)
also have $\dots = top * (-?py \sqcap p0^+ \sqcap ?py^T) * top$
using 2 **by** (*simp add: comp-associative covector-inf-comp-3*)
also have $\dots = bot$
proof –
have $p0^{T^*} * y - y^T * p0^* = p0^{T^*} * y * y^T * -p0^*$
using 2 **by** (*metis assms(2) bijective-conv-mapping comp-mapping-complement vector-covector vector-export-comp vector-mult-closed*)
also have $\dots \leq p0^{T^*} * -p0^*$
by (*meson assms(2) mult-left-isotone order-refl shunt-bijective*)
also have $\dots \leq -p0^*$
by (*simp add: conv-complement conv-star-commute pp-increasing schroeder-6-p star.circ-transitive-equal*)
also have $\dots \leq -p0^+$
by (*simp add: p-antitone star.left-plus-below-circ*)
finally have $-?py \sqcap p0^+ \sqcap ?py^T = bot$
by (*metis comp-inf.p-pp-comp conv-complement conv-dist-comp conv-involutive conv-star-commute p-shunting-swap pp-isotone pseudo-complement-pp regular-closed-p*)
thus *?thesis*
by *simp*
qed
finally have $?q1 * (-?pxy \sqcap p0^+) * ?pxy = bot$
using *le-bot* **by** *blast*
have $?q2^+ \leq -?pxy$
using 2 **by** (*smt (z3) comp-isotone complement-conv-sub inf.order-trans inf.sup-right-divisibility inf-commute symmetric-top-closed top-greatest*)
hence $?q2^+ \leq -?pxy \sqcap p0^+$
by (*simp add: comp-isotone star-isotone*)

hence 8: $?q1 * ?q2^+ * ?pxy = bot$
using 7 *mult-left-isotone mult-right-isotone le-bot* **by** *auto*
have $?q1 * ?q2^+ * ?q3^* = ?q1 * ?q2^+ \sqcup ?q1 * ?q2^+ * ?q3^+$
by (*smt (z3) comp-associative star.circ-back-loop-fixpoint star.circ-plus-same sup-commute*)
also have $... \leq ?q1 * ?q2^+ \sqcup ?q1 * ?q2^+ * ?pxy$
using 2 **by** (*smt (z3) inf.cobounded1 mult-right-isotone sup-right-isotone vector-inf-comp*)
finally have 9: $?q1 * ?q2^+ * ?q3^* \leq ?q1 * ?q2^+$
using 8 **by** *simp*
have 10: $?q1 * ?q4 * ?pxy = bot$
proof –
have $?p2 \leq p0^+$
by (*simp add: mult-right-isotone star.circ-increasing*)
thus *?thesis*
using 7 **by** (*metis mult-left-isotone mult-right-isotone le-bot comp-inf.comp-isotone eq-refl*)
qed
have 11: $?q1 * ?q2 * ?pxy = bot$
proof –
have $p0 \leq p0^+$
by (*simp add: star.circ-mult-increasing*)
thus *?thesis*
using 7 **by** (*metis mult-left-isotone mult-right-isotone le-bot comp-inf.comp-isotone eq-refl*)
qed
have 12: $?q2 \leq p0 * ?q3^* * ?q2^*$
by (*smt (verit, del-insts) conv-dist-comp conv-order conv-star-commute inf.coboundedI1 inf.orderE inf.sup-monoid.add-commute path-compression-1b*)
have $?q3 * p0 * ?q3^* * ?q2^* = ?q1 * p0 * p0 * ?q3^* * ?q2^*$
using 2 *vector-inf-comp* **by** *auto*
also have $... = ?q1 * (?q3 \sqcup ?q4) * ?q3^* * ?q2^*$
using 1 **by** (*smt (z3) comp-associative comp-inf.mult-right-dist-sup comp-inf.star-slide inf-top.right-neutral regular-complement-top*)
also have $... = ?q1 * ?q3 * ?q3^* * ?q2^* \sqcup ?q1 * ?q4 * ?q3^* * ?q2^*$
using *mult-left-dist-sup mult-right-dist-sup* **by** *auto*
also have $... \leq ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q4 * ?q3^* * ?q2^*$
by (*smt (z3) mult-left-isotone mult-left-sub-dist-sup-right sup-left-isotone sup-right-divisibility mult-assoc star.left-plus-below-circ*)
also have $... = ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q4 * ?q2^* \sqcup ?q1 * ?q4 * ?q3^+ * ?q2^*$
by (*smt (z3) semiring.combine-common-factor star.circ-back-loop-fixpoint star-plus sup-monoid.add-commute mult-assoc*)
also have $... \leq ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q4 * ?q2^* \sqcup ?q1 * ?q4 * ?pxy * ?q3^* * ?q2^*$
by (*smt (verit, ccfv-threshold) comp-isotone inf.sup-right-divisibility inf-commute order.refl semiring.add-left-mono mult-assoc*)
also have $... = ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q4 * ?q2^*$
using 10 **by** *simp*

also have ... = $?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2 * p0 * ?q2^*$
using 2 **by** (*smt vector-complement-closed vector-inf-comp mult-assoc*)
also have ... = $?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2 * (?q2 \sqcup ?q1) * ?q2^*$
using 1 **by** (*smt (z3) comp-associative comp-inf.mult-right-dist-sup comp-inf.star-slide inf-top.right-neutral regular-complement-top*)
also have ... = $?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2 * ?q2 * ?q2^* \sqcup ?q1 * ?q2 * ?q1 * ?q2^*$
using *mult-left-dist-sup mult-right-dist-sup sup-commute sup-left-commute* **by** *auto*
also have ... $\leq ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2 * ?q2 * ?q2^* \sqcup ?q1 * ?q2 * ?pxy * ?q2^*$
by (*smt (verit, ccfv-threshold) comp-isotone inf.sup-right-divisibility inf-commute order.refl semiring.add-left-mono mult-assoc*)
also have ... = $?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2 * ?q2 * ?q2^*$
using 11 **by** *simp*
also have ... $\leq ?q1 * ?q3^* * ?q2^* \sqcup ?q1 * ?q2^*$
by (*smt comp-associative comp-isotone mult-right-isotone star.circ-increasing star.circ-transitive-equal star.left-plus-below-circ sup-right-isotone*)
also have ... = $?q1 * ?q3^* * ?q2^*$
by (*smt (verit, best) comp-associative semiring.distrib-left star.circ-loop-fixpoint star.circ-transitive-equal star-involutive*)
finally have 13: $?q3 * p0 * ?q3^* * ?q2^* \leq p0 * ?q3^* * ?q2^*$
by (*meson inf.cobounded2 mult-left-isotone order-lesseq-imp*)
hence $?q3 * p0 * ?q3^* * ?q2^* \sqcup ?q2 \leq p0 * ?q3^* * ?q2^*$
using 12 **by** *simp*
hence $?q3^* * ?q2 \leq p0 * ?q3^* * ?q2^*$
by (*simp add: star-left-induct mult-assoc*)
hence $?q1 * ?q3^* * ?q2 \leq ?q1 * p0 * ?q3^* * ?q2^*$
by (*simp add: comp-associative mult-right-isotone*)
hence $?q1 * ?q3^* * ?q2 \leq ?q3^+ * ?q2^*$
using 2 **by** (*simp add: vector-inf-comp*)
hence 14: $?q1 * ?q3^* * ?q2 \leq ?q3^* * ?q2^*$
using *mult-left-isotone order-lesseq-imp star.left-plus-below-circ* **by** *blast*
have $p0 * ?r0 \leq p0 * ?q3^* * ?q2^* * ?r0$
by (*metis comp-associative mult-1-right mult-left-isotone mult-right-isotone reflexive-mult-closed star.circ-reflexive*)
hence 15: $?r0 \leq p0 * ?q3^* * ?q2^* * ?r0$
using 3 *dual-order.trans* **by** *blast*
have $?q3 * p0 * ?q3^* * ?q2^* * ?r0 \leq p0 * ?q3^* * ?q2^* * ?r0$
using 13 *mult-left-isotone* **by** *blast*
hence $?q3 * p0 * ?q3^* * ?q2^* * ?r0 \sqcup ?r0 \leq p0 * ?q3^* * ?q2^* * ?r0$
using 15 **by** *simp*
hence $?q3^* * ?r0 \leq p0 * ?q3^* * ?q2^* * ?r0$
by (*simp add: star-left-induct mult-assoc*)
hence $?q1 * ?q3^* * ?r0 \leq ?q1 * p0 * ?q3^* * ?q2^* * ?r0$
by (*simp add: comp-associative mult-right-isotone*)
hence $?q1 * ?q3^* * ?r0 \leq ?q3^+ * ?q2^* * ?r0$
using 2 **by** (*simp add: vector-inf-comp*)
hence 16: $?q1 * ?q3^* * ?r0 \leq ?q3^* * ?q2^* * ?r0$

using *mult-left-isotone order-lesseq-imp star.left-plus-below-circ* **by** *blast*
have $?q1 * ?q3^* * ?q2^* * ?r0 = ?q1 * ?q3^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^+ * ?r0$
by (*smt (z3) comp-associative mult-right-dist-sup star.circ-back-loop-fixpoint*
star.circ-plus-same sup-commute)
also have $\dots \leq ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^+ * ?r0$
using *16 sup-left-isotone* **by** *blast*
also have $\dots \leq ?q3^* * ?q2^* * ?r0 \sqcup ?q3^* * ?q2^* * ?q2^* * ?r0$
using *14* **by** (*smt (z3) inf.eq-refl semiring.distrib-right*
star.circ-transitive-equal sup.absorb2 sup-monoid.add-commute mult-assoc)
also have $\dots = ?q3^* * ?q2^* * ?r0$
by (*simp add: comp-associative star.circ-transitive-equal*)
finally have *17*: $?q1 * ?q3^* * ?q2^* * ?r0 \leq ?q3^* * ?q2^* * ?r0$
.

have $?r0 \leq ?q2^* * ?r0$
using *star.circ-loop-fixpoint sup-right-divisibility* **by** *auto*
also have $\dots \leq ?q3^* * ?q2^* * ?r0$
using *comp-associative star.circ-loop-fixpoint sup-right-divisibility* **by** *force*
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0$
using *comp-associative star.circ-loop-fixpoint sup-right-divisibility* **by** *force*
finally have *18*: $?r0 \leq ?q2^* * ?q3^* * ?q2^* * ?r0$
.

have $p0 * ?q2^* * ?q3^* * ?q2^* * ?r0 = (?q2 \sqcup ?q1) * ?q2^* * ?q3^* * ?q2^* * ?r0$
using *1* **by** (*smt (z3) comp-inf.mult-right-dist-sup comp-inf.star-plus*
inf-top.right-neutral regular-complement-top)
also have $\dots = ?q2 * ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q2^* * ?q3^* * ?q2^* * ?r0$
using *mult-right-dist-sup* **by** *auto*
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q2^* * ?q3^* * ?q2^* * ?r0$
by (*smt (z3) comp-left-increasing-sup star.circ-loop-fixpoint sup-left-isotone*
mult-assoc)
also have $\dots = ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q2^+ * ?q3^* * ?q2^* * ?r0$
by (*smt (z3) mult-left-dist-sup semiring.combine-common-factor*
star.circ-loop-fixpoint sup-monoid.add-commute mult-assoc)
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q2^+ * ?q2^* * ?r0$
using *9 mult-left-isotone sup-right-isotone* **by** *auto*
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q2^* * ?r0$
by (*smt (z3) comp-associative comp-isotone inf.eq-refl*
semiring.add-right-mono star.circ-transitive-equal star.left-plus-below-circ
sup-commute)
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q1 * ?q3^* * ?q2^* * ?r0 \sqcup ?q3^* * ?q2^* * ?r0$
using *6 sup-right-isotone* **by** *blast*
also have $\dots = ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q3^* * ?q2^* * ?r0$
using *17* **by** (*smt (z3) le-iff-sup semiring.combine-common-factor*
semiring.distrib-right star.circ-loop-fixpoint sup-monoid.add-commute)
also have $\dots \leq ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?q3^* * ?q2^* * ?r0$

by (*meson mult-left-isotone star.circ-increasing sup-right-isotone*)
 also have ... = $?q2^* * ?q3^* * ?q2^* * ?r0$
 by (*smt (z3) comp-associative star.circ-loop-fixpoint star.circ-transitive-equal star-involutive*)
 finally have $p0 * ?q2^* * ?q3^* * ?q2^* * ?r0 \sqcup ?r0 \leq ?q2^* * ?q3^* * ?q2^* * ?r0$
 using 18 *sup.boundedI* by *blast*
 hence $p0^* * ?r0 \leq ?q2^* * ?q3^* * ?q2^* * ?r0$
 by (*simp add: comp-associative star-left-induct*)
 also have ... $\leq ?p^* * ?q3^* * ?q2^* * ?r0$
 by (*metis mult-left-isotone star.circ-sub-dist sup-commute*)
 also have ... $\leq ?p^* * ?p^* * ?q2^* * ?r0$
 by (*simp add: mult-left-isotone mult-right-isotone star-isotone*)
 also have ... $\leq ?p^* * ?p^* * ?p^* * ?r0$
 by (*metis mult-isotone order.refl star.circ-sub-dist sup-commute*)
 finally have 19: $p0^* * ?r0 \leq ?p^* * ?r0$
 by (*simp add: star.circ-transitive-equal*)
 have 20: $?p^* \leq p0^*$
 by (*metis star.left-plus-circ star-isotone update-square-ub-plus*)
 hence 21: $p0^* * ?r0 = ?p^* * ?r0$
 using 19 *order.antisym mult-left-isotone* by *auto*
 have $?p \sqcap 1 = (?q3 \sqcap 1) \sqcup (?q2 \sqcap 1)$
 using *comp-inf.semiring.distrib-right conv-involutive* by *auto*
 also have ... = $(?q1 \sqcap 1) \sqcup (?q2 \sqcap 1)$
 using *assms(3) acyclic-square path-splitting-invariant-def*
inf.sup-monoid.add-assoc by *auto*
 also have ... = $(?pxy \sqcup -?pxy) \sqcap p0 \sqcap 1$
 using *inf-sup-distrib2* by *auto*
 also have ... = $p0 \sqcap 1$
 using 1 2 *assms(3) disjoint-set-forest-update-square* by *blast*
maddux-3-11-pp
 finally show 22: $?p \sqcap 1 = p0 \sqcap 1$
 .
 have $?p^{T^*} * x \leq p0^{T^*} * x$
 using 20 by (*metis conv-isotone conv-star-commute mult-left-isotone*)
 hence 23: $?rp \leq ?r0$
 using 22 *comp-inf.mult-left-isotone* by *auto*
 have 24: *disjoint-set-forest ?p*
 using 1 2 *assms(3) disjoint-set-forest-update-square* by *blast*
 hence 25: *point ?rp*
 using *root-point assms(1)* by *auto*
 have $?r0 * ?rp^T = ?r0 * x^T * ?p^* * (?p \sqcap 1)$
 by (*smt (z3) comp-associative conv-dist-comp conv-dist-inf conv-involutive conv-star-commute inf.sup-monoid.add-commute one-inf-conv root-var star-one star-sup-one wcc-one*)
 also have ... $\leq (p0 \sqcap 1) * p0^{T^*} * 1 * ?p^* * (?p \sqcap 1)$
 by (*smt (z3) assms(1) comp-associative mult-left-isotone mult-right-isotone root-var*)
 also have ... $\leq (p0 \sqcap 1) * p0^{T^*} * p0^* * (p0 \sqcap 1)$
 using 20 22 *comp-isotone* by *force*

also have ... = $(p0 \sqcap 1) * p0^* * (p0 \sqcap 1) \sqcup (p0 \sqcap 1) * p0^{T^*} * (p0 \sqcap 1)$
by (*simp add: assms(3) cancel-separate-eq sup-monoid.add-commute mult-assoc mult-left-dist-sup semiring.distrib-right*)
also have ... = $(p0 \sqcap 1) * (p0 \sqcap 1) \sqcup (p0 \sqcap 1) * p0^{T^*} * (p0 \sqcap 1)$
using *univalent-root-successors assms(3)* **by** *simp*
also have ... = $(p0 \sqcap 1) * (p0 \sqcap 1) \sqcup (p0 \sqcap 1) * ((p0 \sqcap 1) * p0^*)^T$
by (*smt (z3) comp-associative conv-dist-comp conv-dist-inf conv-star-commute inf.sup-monoid.add-commute one-inf-conv star-one star-sup-one wcc-one*)
also have ... = $(p0 \sqcap 1) * (p0 \sqcap 1)$
by (*metis univalent-root-successors assms(3) conv-dist-inf inf.sup-monoid.add-commute one-inf-conv sup-idem symmetric-one-closed*)
also have ... ≤ 1
by (*simp add: coreflexive-mult-closed*)
finally have $?r0 * ?rp^T \leq 1$
.

hence $?r0 \leq 1 * ?rp$
using *25 shunt-bijective* **by** *blast*
hence *26: ?r0 = ?rp*
using *23 order.antisym* **by** *simp*
have $?px * ?r0^T = ?px * x^T * p0^* * (p0 \sqcap 1)$
by (*smt (z3) comp-associative conv-dist-comp conv-dist-inf conv-involutive conv-star-commute inf.sup-monoid.add-commute one-inf-conv root-var star-one star-sup-one wcc-one*)
also have ... $\leq p0^{T^*} * 1 * p0^* * (p0 \sqcap 1)$
by (*smt (z3) assms(1) comp-associative mult-left-isotone mult-right-isotone root-var*)
also have ... = $p0^* * (p0 \sqcap 1) \sqcup p0^{T^*} * (p0 \sqcap 1)$
by (*simp add: assms(3) cancel-separate-eq sup-monoid.add-commute mult-right-dist-sup*)
also have ... = $p0^* * (p0 \sqcap 1) \sqcup ((p0 \sqcap 1) * p0^*)^T$
by (*smt (z3) conv-dist-comp conv-dist-inf conv-star-commute inf.sup-monoid.add-commute one-inf-conv star-one star-sup-one wcc-one*)
also have ... = $p0^* * (p0 \sqcap 1) \sqcup (p0 \sqcap 1)$
by (*metis univalent-root-successors assms(3) conv-dist-inf inf.sup-monoid.add-commute one-inf-conv symmetric-one-closed*)
also have ... = $p0^* * (p0 \sqcap 1)$
by (*metis conv-involutive path-compression-1b sup.absorb2 sup-commute*)
also have ... $\leq p0^*$
by (*simp add: inf.coboundedI1 star.circ-increasing star.circ-mult-upper-bound*)
finally have *27: ?px * ?r0^T $\leq p0^*$*
.

thus *28: ?px $\leq p0^* * ?r0$*
by (*simp add: assms(1,3) root-point shunt-bijective*)
have *29: point ?r0*
using *root-point assms(1,3)* **by** *auto*
hence *30: mapping (?r0^T)*
using *bijective-conv-mapping* **by** *blast*
have $?r0 * (?px \sqcap p0) = ?r0 * top * (?px \sqcap p0)$
using *29* **by** *force*

also have $\dots = ?r0 * ?px^T * p0$
using 29 **by** (*metis* *assms(1)* *covector-inf-comp-3* *vector-covector* *vector-mult-closed*)
also have $\dots = ?r0 * x^T * p0^* * p0$
using *comp-associative conv-dist-comp conv-star-commute* **by** *auto*
also have $\dots \leq ?r0 * x^T * p0^*$
by (*simp* *add: comp-associative mult-right-isotone star.circ-plus-same* *star.left-plus-below-circ*)
also have $\dots = ?r0 * ?px^T$
by (*simp* *add: comp-associative conv-dist-comp conv-star-commute*)
also have $\dots = (?px * ?r0^T)^T$
by (*simp* *add: conv-dist-comp*)
also have $\dots \leq p0^{T^*}$
using 27 *conv-isotone conv-star-commute* **by** *fastforce*
finally have $?r0 * (?px \sqcap p0) \leq p0^{T^*}$

hence $?px \sqcap p0 \leq ?r0^T * p0^{T^*}$
using 30 *shunt-mapping* **by** *auto*
hence $?px \sqcap p0 \leq p0^* * ?r0 \sqcap ?r0^T * p0^{T^*}$
using 28 *inf.coboundedI2 inf.sup-monoid.add-commute* **by** *fastforce*
also have $\dots = p0^* * ?r0 * ?r0^T * p0^{T^*}$
using 29 **by** (*smt* (*z3*) *vector-covector vector-inf-comp vector-mult-closed*)
also have $\dots = ?p^* * ?r0 * ?r0^T * ?p^{T^*}$
using 21 **by** (*smt* *comp-associative conv-dist-comp conv-star-commute*)
also have $\dots = ?p^* * ?rp * ?rp^T * ?p^{T^*}$
using 26 **by** *auto*
also have $\dots \leq ?p^* * 1 * ?p^{T^*}$
using 25 **by** (*smt* (*z3*) *comp-associative mult-left-isotone mult-right-isotone*)
finally have 31: $?px \sqcap p0 \leq fc ?p$
by *auto*
have $-?px \sqcap p0 \leq ?p$
by (*simp* *add: inf.sup-monoid.add-commute le-supI1 sup-commute*)
also have $\dots \leq fc ?p$
using *fc-increasing* **by** *auto*
finally have $p0 \leq fc ?p$
using 1 31 **by** (*smt* (*z3*) *inf.sup-monoid.add-commute maddux-3-11-pp* *semiring.add-left-mono sup.orderE sup-commute*)
also have $\dots \leq wcc ?p$
using *star.circ-sub-dist-3* **by** *auto*
finally have 32: $wcc p0 \leq wcc ?p$
using *wcc-below-wcc* **by** *blast*
have $?p \leq wcc p0$
by (*simp* *add: inf.coboundedI1 inf.sup-monoid.add-commute* *star.circ-mult-upper-bound star.circ-sub-dist-1*)
hence $wcc ?p \leq wcc p0$
using *wcc-below-wcc* **by** *blast*
hence $wcc ?p = wcc p0$
using 32 *order.antisym* **by** *blast*
thus $fc ?p = fc p0$

using 24 assms(3) fc-wcc by auto
qed

lemma path-splitting-invariant-aux:

assumes path-splitting-invariant p x y p0
shows $p[[y]] = p0[[y]]$
and $p[[p[[y]]]] = p0[[p0[[y]]]]$
and $p[[p[[p[[y]]]]]] = p0[[p0[[p0[[y]]]]]]$
and $p \sqcap 1 = p0 \sqcap 1$
and $fc\ p = fc\ p0$
proof –
let ?p2 = p0 * p0
let ?p2t = ?p2^T
let ?px = p0^{T*} * x
let ?py = -(p0^{T*} * y)
let ?pxy = ?px \sqcap ?py
let ?p = p0[?pxy \longrightarrow ?p2t]
have ?p[[y]] = p0[[y]]
apply (rule put-get-different-vector)
using assms find-set-precondition-def vector-complement-closed
vector-inf-closed vector-mult-closed path-splitting-invariant-def apply force
by (meson inf.cobounded2 order-lesseq-imp p-antitone-iff path-compression-1b)
thus 1: $p[[y]] = p0[[y]]$
using assms path-splitting-invariant-def by auto
have ?p[[p0[[y]]]] = p0[[p0[[y]]]]
apply (rule put-get-different-vector)
using assms find-set-precondition-def vector-complement-closed
vector-inf-closed vector-mult-closed path-splitting-invariant-def apply force
by (metis comp-isotone inf.boundedE inf.coboundedI2 inf.eq-refl p-antitone-iff
selection-closed-id star.circ-increasing)
thus 2: $p[[p[[y]]]] = p0[[p0[[y]]]]$
using 1 assms path-splitting-invariant-def by auto
have ?p[[p0[[p0[[y]]]]]] = p0[[p0[[p0[[y]]]]]]
apply (rule put-get-different-vector)
using assms find-set-precondition-def vector-complement-closed
vector-inf-closed vector-mult-closed path-splitting-invariant-def apply force
by (metis comp-associative comp-isotone conv-dist-comp conv-involutive
conv-order inf.coboundedI2 inf.le-iff-sup mult-left-isotone p-antitone-iff
p-antitone-inf star.circ-increasing star.circ-transitive-equal)
thus $p[[p[[p[[y]]]]]] = p0[[p0[[p0[[y]]]]]]$
using 2 assms path-splitting-invariant-def by auto
show $p \sqcap 1 = p0 \sqcap 1$
using assms path-splitting-invariant-aux-1(1) path-splitting-invariant-def
find-set-precondition-def by auto
show $fc\ p = fc\ p0$
using assms path-splitting-invariant-aux-1(2) path-splitting-invariant-def
find-set-precondition-def by auto
qed

lemma *path-splitting-1*:
find-set-precondition $p0\ x \implies$ *path-splitting-invariant* $p0\ x\ x\ p0$
proof –
assume 1: *find-set-precondition* $p0\ x$
show *path-splitting-invariant* $p0\ x\ x\ p0$
proof (*unfold path-splitting-invariant-def, intro conjI*)
show *find-set-precondition* $p0\ x$
using 1 **by** *simp*
show *vector* x *injective* x *surjective* x
using 1 *find-set-precondition-def* **by** *auto*
show $x \leq p0^{T^*} * x$
by (*simp add: path-compression-1b*)
have $(p0 * p0)^{T^*} * x \leq p0^{T^*} * x$
by (*simp add: conv-dist-comp mult-left-isotone star.circ-square*)
thus $p0[p0^{T^*} * x - p0^{T^*} * x \longrightarrow (p0 * p0)^T] = p0$
by (*smt (z3) inf.le-iff-sup inf-commute maddux-3-11-pp p-antitone-inf pseudo-complement*)
show *univalent* $p0$ *total* $p0$ *acyclic* $(p0 - 1)$
using 1 *find-set-precondition-def* **by** *auto*
qed
qed

lemma *path-splitting-2*:
path-splitting-invariant $p\ x\ y\ p0 \wedge y \neq p[[y]] \implies$ *path-splitting-invariant*
 $(p[y \longrightarrow p[[p[[y]]]])\ x\ (p[[y]])\ p0 \wedge ((p[y \longrightarrow p[[p[[y]]]])^{T^*} * (p[[y]])) \downarrow < (p^{T^*} * y) \downarrow$
proof –
let $?py = p[[y]]$
let $?ppy = p[[?py]]$
let $?pyppy = p[y \longrightarrow ?ppy]$
let $?p2 = p0 * p0$
let $?p2t = ?p2^T$
let $?p2ts = ?p2t^*$
let $?px = p0^{T^*} * x$
let $?py2 = -(p0^{T^*} * y)$
let $?pxy = ?px \sqcap ?py2$
let $?p = p0[?pxy \longrightarrow ?p2t]$
let $?pty = p0^T * y$
let $?pt2y = p0^T * p0^T * y$
let $?pt2sy = p0^{T^*} * p0^T * p0^T * y$
let $?ptpy = p0^{T^+} * y$
assume 1: *path-splitting-invariant* $p\ x\ y\ p0 \wedge y \neq ?py$
have 2: *point* $?pty \wedge$ *point* $?pt2y$
using 1 **by** (*smt (verit) comp-associative read-injective read-surjective path-splitting-invariant-def*)
show *path-splitting-invariant* $?pyppy\ x\ (p[[y]])\ p0 \wedge (?pyppy^{T^*} * (p[[y]])) \downarrow < (p^{T^*} * y) \downarrow$
proof
show *path-splitting-invariant* $?pyppy\ x\ (p[[y]])\ p0$
proof (*unfold path-splitting-invariant-def, intro conjI*)

```

show 3: find-set-precondition ?pyppy x
proof (unfold find-set-precondition-def, intro conjI)
  show univalent ?pyppy
    using 1 find-set-precondition-def read-injective update-univalent
path-splitting-invariant-def by auto
  show total ?pyppy
    using 1 bijjective-regular find-set-precondition-def read-surjective
update-total path-splitting-invariant-def by force
  show acyclic (?pyppy - 1)
    apply (rule update-acyclic-3)
    using 1 find-set-precondition-def path-splitting-invariant-def apply blast
    using 1 2 comp-associative path-splitting-invariant-aux(2) apply force
    using 1 path-splitting-invariant-def apply blast
    by (metis inf.order-lesseq-imp mult-isotone star.circ-increasing
star.circ-square mult-assoc)
  show vector x injective x surjective x
    using 1 find-set-precondition-def path-splitting-invariant-def by auto
qed
show vector (p[[y]])
  using 1 comp-associative path-splitting-invariant-def by auto
show injective (p[[y]])
  using 1 3 read-injective path-splitting-invariant-def find-set-precondition-def
by auto
  show surjective (p[[y]])
    using 1 3 read-surjective path-splitting-invariant-def
find-set-precondition-def by auto
show p[[y]] ≤ ?px
proof -
  have p[[y]] = p0[[y]]
    using 1 path-splitting-invariant-aux(1) by blast
  also have ... ≤ p0T * ?px
    using 1 path-splitting-invariant-def mult-right-isotone by force
  also have ... ≤ ?px
    by (metis comp-associative mult-left-isotone star.left-plus-below-circ)
  finally show ?thesis
  .
qed
show p0[?px - p0T * (p[[y]])] → ?p2t = ?pyppy
proof -
  have 4: p[?px ⊓ - ?ptpy ⊓ y → ?p2t] = ?pyppy
proof (cases y ≤ - ?ptpy)
  case True
    hence p[?px ⊓ - ?ptpy ⊓ y → ?p2t] = p[y → ?p2t]
      using 1 inf.absorb2 path-splitting-invariant-def by auto
    also have ... = ?pyppy
      using 1 by (metis comp-associative conv-dist-comp
path-splitting-invariant-aux(2) path-splitting-invariant-def update-point-get)
    finally show ?thesis
  .

```

```

next
  case False
  have vector ( $\neg ?ptpy$ )
    using 1 vector-complement-closed vector-mult-closed
path-splitting-invariant-def by blast
  hence 5:  $y \leq ?ptpy$ 
    using 1 by (smt (verit, del-Insts) False bijective-regular
point-in-vector-or-complement regular-closed-star regular-mult-closed
total-conv-surjective univalent-conv-injective path-splitting-invariant-def)
  hence  $?px \sqcap \neg ?ptpy \sqcap y = \text{bot}$ 
    by (simp add: inf.coboundedI2 p-antitone pseudo-complement)
  hence 6:  $p[?px \sqcap \neg ?ptpy \sqcap y \longrightarrow ?p2t] = p$ 
    by simp
  have 7:  $y = \text{root } p0 \ y$ 
    using 1 5 loop-root-2 path-splitting-invariant-def by blast
  have  $?pyppy = p[y \longrightarrow p0[[p0[[y]]]]]$ 
    using 1 path-splitting-invariant-aux(2) by force
  also have  $\dots = p[y \longrightarrow p0[[y]]]$ 
    using 1 7 by (metis root-successor-loop path-splitting-invariant-def)
  also have  $\dots = p[y \longrightarrow ?py]$ 
    using 1 path-splitting-invariant-aux(1) by force
  also have  $\dots = p$ 
    using 1 get-put path-splitting-invariant-def by blast
  finally show ?thesis
    using 6 by simp
qed
have 8:  $\neg ?ptpy = ?py2 \sqcup (\neg ?ptpy \sqcap y)$ 
proof (rule order.antisym)
  have 9:  $\text{regular } (p0^{T^*} * y) \wedge \text{regular } ?ptpy$ 
    using 1 bijective-regular mapping-conv-bijective pp-dist-star
regular-mult-closed path-splitting-invariant-def by auto
  have  $p0^{T^*} * y \leq ?ptpy \sqcup y$ 
    by (simp add: star.circ-loop-fixpoint mult-assoc)
  also have  $\dots = ?ptpy \sqcup (\neg ?ptpy \sqcap y)$ 
    using 9 by (metis maddux-3-21-pp)
  hence  $p0^{T^*} * y \sqcap \neg ?ptpy \leq \neg ?ptpy \sqcap y$ 
    using calculation half-shunting sup-commute by auto
  thus  $\neg ?ptpy \leq ?py2 \sqcup (\neg ?ptpy \sqcap y)$ 
    using 9 by (smt (z3) inf.sup-monoid.add-commute shunting-var-p
sup.left-commute sup-commute)
  have  $\neg (p0^{T^*} * y) \leq \neg ?ptpy$ 
    by (simp add: comp-isotone p-antitone star.left-plus-below-circ)
  thus  $\neg (p0^{T^*} * y) \sqcup (\neg ?ptpy \sqcap y) \leq \neg ?ptpy$ 
    by simp
qed
have regular ?px regular y
  using 1 bijective-regular find-set-precondition-def mapping-regular
pp-dist-comp regular-closed-star regular-conv-closed path-splitting-invariant-def by
auto

```

```

    hence 10: regular (?px  $\sqcap$  -?ptpy  $\sqcap$  y)
    by auto
    have p0[?px  $\sqcap$  -(p0T* * (p[[y]])) $\mapsto$ ?p2t] = p0[?px  $\sqcap$  -?ptpy $\mapsto$ ?p2t]
    using 1 by (smt comp-associative path-splitting-invariant-aux(1)
star-plus)
    also have ... = p0[?pxy  $\sqcup$  (?px  $\sqcap$  -?ptpy  $\sqcap$  y) $\mapsto$ ?p2t]
    using 8 by (metis comp-inf.semiring.distrib-left
inf.sup-monoid.add-assoc)
    also have ... = ?p[?px  $\sqcap$  -?ptpy  $\sqcap$  y $\mapsto$ ?p2t]
    using 10 by (smt (z3) update-same comp-inf.semiring.mult-not-zero
inf.sup-monoid.add-assoc inf.sup-monoid.add-commute)
    also have ... = p[?px  $\sqcap$  -?ptpy  $\sqcap$  y $\mapsto$ ?p2t]
    using 1 path-splitting-invariant-def by auto
    also have ... = ?pyppy
    using 4 by auto
    finally show ?thesis
    .
qed
show univalent p0 total p0 acyclic (p0 - 1)
using 1 path-splitting-invariant-def by auto
qed
let ?s = { z . regular z  $\wedge$  z  $\leq$  pT* * y }
let ?t = { z . regular z  $\wedge$  z  $\leq$  ?pyppyT* * (p[[y]]) }
have ?pyppyT* * (p[[y]])  $\leq$  p+T* * (p[[y]])
using 1 path-splitting-invariant-def update-square-plus conv-order
mult-left-isotone star-isotone by force
also have ... = pT* * pT * y
by (simp add: conv-plus-commute star.left-plus-circ mult-assoc)
also have ... = pT+ * y
by (simp add: star-plus)
finally have 11: ?pyppyT* * (p[[y]])  $\leq$  pT+ * y
.
hence ?pyppyT* * (p[[y]])  $\leq$  pT* * y
using mult-left-isotone order-lesseq-imp star.left-plus-below-circ by blast
hence 12: ?t  $\subseteq$  ?s
using order-trans by auto
have 13: y  $\in$  ?s
using 1 bijective-regular path-compression-1b path-splitting-invariant-def by
force
have 14:  $\neg$  y  $\in$  ?t
proof
assume y  $\in$  ?t
hence y  $\leq$  ?pyppyT* * (p[[y]])
by simp
hence y  $\leq$  pT+ * y
using 11 dual-order.trans by blast
hence y = root p y
using 1 find-set-precondition-def loop-root-2 path-splitting-invariant-def by
blast

```

```

    hence  $y = ?py$ 
    using 1 by (metis find-set-precondition-def root-successor-loop
path-splitting-invariant-def)
    thus False
    using 1 by simp
qed
show  $\text{card } ?t < \text{card } ?s$ 
  apply (rule psubset-card-mono)
  subgoal using finite-regular by simp
  subgoal using 12 13 14 by auto
  done
qed
qed

```

lemma *path-splitting-3*:

path-splitting-invariant $p\ x\ y\ p0 \wedge y = p[[y]] \implies \text{path-splitting-postcondition } p\ x\ y\ p0$

proof –

```

  assume 1: path-splitting-invariant  $p\ x\ y\ p0 \wedge y = p[[y]]$ 
  show path-splitting-postcondition  $p\ x\ y\ p0$ 
  proof (unfold path-splitting-postcondition-def
path-compression-precondition-def, intro conjI)
    show univalent  $p$  total  $p$  acyclic  $(p - 1)$ 
      using 1 find-set-precondition-def path-splitting-invariant-def by blast+
    show vector  $x$  injective  $x$  surjective  $x$ 
      using 1 find-set-precondition-def path-splitting-invariant-def by blast+
    show 2: vector  $y$  injective  $y$  surjective  $y$ 
      using 1 path-splitting-invariant-def by blast+
    show 3:  $p \sqcap 1 = p0 \sqcap 1$ 
      using 1 path-splitting-invariant-aux(4) by blast
    show 4:  $\text{fc } p = \text{fc } p0$ 
      using 1 path-splitting-invariant-aux(5) by blast
    have  $y \leq p0^{T^*} * x$ 
      using 1 path-splitting-invariant-def by simp
    hence 5:  $y * x^T \leq \text{fc } p0$ 
      using 1 by (metis dual-order.trans fc-wcc find-set-precondition-def
shunt-bijective star.circ-decompose-11 star-decompose-1 star-outer-increasing
path-splitting-invariant-def)
    have 6:  $y = p0[[y]]$ 
      using 1 path-splitting-invariant-aux(1) by auto
    hence  $y = \text{root } p0\ y$ 
      using 2 loop-root by auto
    also have  $\dots = \text{root } p0\ x$ 
      using 1 2 5 find-set-precondition-def path-splitting-invariant-def
same-component-same-root by auto
    also have  $\dots = \text{root } p\ x$ 
      using 1 3 4 by (metis find-set-precondition-def path-splitting-invariant-def
same-root)
    finally show  $y = \text{root } p\ x$ 

```

```

    •
    have  $p0^{T^*} * y = y$ 
      using 6 order.antisym path-compression-1b star-left-induct-mult-equal by
    auto
    hence  $\gamma: p0[p0^{T^*} * x - y \longrightarrow (p0 * p0)^T] = p$ 
      using 1 path-splitting-invariant-def by auto
    have  $(p0 * p0)^T * y = y$ 
      using 6 mult-assoc conv-dist-comp by auto
    hence  $y \sqcap p0 * p0 = y \sqcap p0$ 
      using 2 6 by (metis update-postcondition)
    hence  $\delta: y \sqcap p = y \sqcap p0 * p0$ 
      using 1 2 6 by (smt update-postcondition)
    have  $p0[p0^{T^*} * x \longrightarrow (p0 * p0)^T] = (p0[p0^{T^*} * x - y \longrightarrow (p0 * p0)^T])[p0^{T^*} * x \sqcap y \longrightarrow (p0 * p0)^T]$ 
      using 1 bijective-regular path-splitting-invariant-def update-split by blast
    also have  $\dots = p[p0^{T^*} * x \sqcap y \longrightarrow (p0 * p0)^T]$ 
      using 7 by simp
    also have  $\dots = p$ 
      apply (rule update-same-sub)
      using 8 apply simp
      apply simp
      using 1 bijective-regular inf.absorb2 path-splitting-invariant-def by auto
    finally show  $p0[p0^{T^*} * x \longrightarrow (p0 * p0)^T] = p$ 
  •
  qed
  qed

```

theorem *find-path-splitting*:

```

  VARS  $p \ t \ y$ 
  [ find-set-precondition  $p \ x \wedge p0 = p$  ]
   $y := x;$ 
  WHILE  $y \neq p[[y]]$ 
    INV { path-splitting-invariant  $p \ x \ y \ p0$  }
    VAR {  $(p^{T^*} * y) \downarrow$  }
    DO  $t := p[[y]];$ 
       $p[y] := p[[p[[y]]]];$ 
       $y := t$ 
    OD
  [ path-splitting-postcondition  $p \ x \ y \ p0$  ]
  apply vcg-tc-simp
  apply (fact path-splitting-1)
  apply (fact path-splitting-2)
  by (fact path-splitting-3)

```

end

7 Verifying Union by Rank

In this section we verify the union-by-rank operation of disjoint-set forests. The rank of a node is an upper bound of the height of the subtree rooted at that node. The rank array of a disjoint-set forest maps each node to its rank. This can be represented as a homogeneous relation since the possible rank values are $0, \dots, n-1$ where n is the number of nodes of the disjoint-set forest.

7.1 Peano structures

Since ranks are natural numbers we start by introducing basic Peano arithmetic. Numbers are represented as (relational) points. Constant Z represents the number 0. Constant S represents the successor function. The successor of a number x is obtained by the relational composition $S^T * x$. The composition $S * x$ results in the predecessor of x .

```
class peano-signature =
  fixes  $Z :: 'a$ 
  fixes  $S :: 'a$ 
```

The numbers will be used in arrays, which are represented by homogeneous finite relations. Such relations can only represent finitely many numbers. This means that we weaken the Peano axioms, which are usually used to obtain (infinitely many) natural numbers. Axiom *Z-point* specifies that 0 is a number. Axiom *S-univalent* specifies that every number has at most one ‘successor’. Together with axiom *S-total*, which is added later, this means that every number has exactly one ‘successor’. Axiom *S-injective* specifies that numbers with the same successor are equal. Axiom *S-star-Z-top* specifies that every number can be obtained from 0 by finitely many applications of the successor. We omit the Peano axiom $S * Z = bot$ which would specify that 0 is not the successor of any number. Since only finitely many numbers will be represented, the remaining axioms will model successor modulo m for some m depending on the carrier of the algebra. That is, the algebra will be able to represent numbers $0, \dots, m-1$ where the successor of $m-1$ is 0.

```
class skra-peano-1 = stone-kleene-relation-algebra-tarski-consistent +
peano-signature +
  assumes Z-point: point  $Z$ 
  assumes S-univalent: univalent  $S$ 
  assumes S-injective: injective  $S$ 
  assumes S-star-Z-top:  $S^{T*} * Z = top$ 
begin
```

```
lemma conv-Z-Z:
   $Z^T * Z = top$ 
by (simp add: Z-point point-conv-comp)
```

Theorem 9.2

lemma *Z-below-S-star*:

$$Z \leq S^*$$

proof –

$$\text{have } \text{top} * Z^T \leq S^{T^*}$$

using *S-star-Z-top Z-point shunt-bijective* **by** *blast*

thus *?thesis*

using *Z-point conv-order conv-star-commute vector-conv-covector* **by** *force*

qed

Theorem 9.3

lemma *S-connected*:

$$S^{T^*} * S^* = \text{top}$$

by (*metis Z-below-S-star S-star-Z-top mult-left-dist-sup sup.orderE sup-commute top.extremum*)

Theorem 9.4

lemma *S-star-connex*:

$$S^* \sqcup S^{T^*} = \text{top}$$

using *S-connected S-univalent cancel-separate-eq sup-commute* **by** *auto*

Theorem 9.5

lemma *Z-sup-conv-S-top*:

$$Z \sqcup S^T * \text{top} = \text{top}$$

using *S-star-Z-top star.circ-loop-fixpoint sup-commute* **by** *auto*

lemma *top-S-sup-conv-Z*:

$$\text{top} * S \sqcup Z^T = \text{top}$$

by (*metis S-star-Z-top conv-dist-comp conv-involutive conv-star-commute star.circ-back-loop-fixpoint symmetric-top-closed*)

Theorem 9.1

lemma *S-inf-1-below-Z*:

$$S \sqcap 1 \leq Z$$

proof –

$$\text{have } (S \sqcap 1) * S^T \leq S \sqcap 1$$

by (*metis S-injective conv-dist-comp coreflexive-symmetric inf.boundedI inf.cobounded1 inf.cobounded2 injective-codomain*)

$$\text{hence } (S \sqcap 1) * S^{T^*} \leq S \sqcap 1$$

using *star-right-induct-mult* **by** *blast*

$$\text{hence } (S \sqcap 1) * S^{T^*} * Z \leq (S \sqcap 1) * Z$$

by (*simp add: mult-left-isotone*)

also have $\dots \leq Z$

by (*metis comp-left-subdist-inf inf.boundedE mult-1-left*)

finally show *?thesis*

using *S-star-Z-top inf.order-trans top-right-mult-increasing mult-assoc* **by**

auto

qed

lemma *S-inf-1-below-conv-Z*:

$$S \sqcap 1 \leq Z^T$$

using *S-inf-1-below-Z conv-order coreflexive-symmetric* **by** *fastforce*

The successor operation provides a convenient way to compare two natural numbers. Namely, $k < m$ if m can be reached from k by finitely many applications of the successor, formally $m \leq S^{T*} * k$ or $k \leq S^* * m$. This does not work for numbers modulo m since comparison depends on the chosen representative. We therefore work with a modified successor relation S' , which is a partial function that computes the successor for all numbers except $m - 1$. If S is surjective, the point M representing the greatest number $m - 1$ is the predecessor of 0 under S . If S is not surjective (like for the set of all natural numbers), $M = \text{bot}$.

abbreviation $S' \equiv S - Z^T$

abbreviation $M \equiv S * Z$

Theorem 11.1

lemma *M-point-iff-S-surjective:*

point M \longleftrightarrow *surjective S*

proof

assume *1: point M*

hence $1 \leq Z^T * S^T * S * Z$

using *comp-associative conv-dist-comp surjective-var* **by** *auto*

hence $Z \leq S^T * S * Z$

using *1 Z-point bijective-reverse mult-assoc* **by** *auto*

also have $\dots \leq S^T * \text{top}$

by (*simp add: comp-isotone mult-assoc*)

finally have $S^T * S^T * \text{top} \sqcup Z \leq S^T * \text{top}$

using *mult-isotone mult-assoc* **by** *force*

hence $S^{T*} * Z \leq S^T * \text{top}$

by (*simp add: star-left-induct mult-assoc*)

thus *surjective S*

by (*simp add: S-star-Z-top order.antisym surjective-conv-total*)

next

assume *surjective S*

thus *point M*

by (*metis S-injective Z-point comp-associative injective-mult-closed*)

qed

Theorem 10.1

lemma *S'-univalent:*

univalent S'

by (*simp add: S-univalent univalent-inf-closed*)

Theorem 10.2

lemma *S'-injective:*

injective S'

by (*simp add: S-injective injective-inf-closed*)

Theorem 10.9

lemma *S'-Z*:
*S' * Z = bot*
by (*simp add: Z-point covector-vector-comp injective-comp-right-dist-inf*)

Theorem 10.4

lemma *S'-irreflexive*:
irreflexive S'
using *S-inf-1-below-conv-Z order-lesseq-imp p-shunting-swap pp-increasing* **by**
blast

end

class *skra-peano-2 = skra-peano-1 +*
assumes *S-total: total S*
begin

lemma *S-mapping*:
mapping S
by (*simp add: S-total S-univalent*)

Theorem 11.2

lemma *M-bot-iff-S-not-surjective*:
M ≠ bot ↔ surjective S
proof
assume *M ≠ bot*
hence *top * S * Z = top*
by (*metis S-mapping Z-point bijective-regular comp-associative*
mapping-regular regular-mult-closed tarski)
hence *Z^T ≤ top * S*
using *M-point-iff-S-surjective S-injective Z-point comp-associative*
injective-mult-closed **by** *auto*
thus *surjective S*
using *sup.orderE top-S-sup-conv-Z* **by** *fastforce*
next
assume *surjective S*
thus *M ≠ bot*
using *M-point-iff-S-surjective consistent covector-bot-closed* **by** *force*
qed

Theorem 11.3

lemma *M-point-or-bot*:
point M ∨ M = bot
using *M-bot-iff-S-not-surjective M-point-iff-S-surjective* **by** *blast*

Alternative way to express *S'*

Theorem 12.1

lemma *S'-var*:

$S' = S - M$
proof –
have $S' = S * (1 - Z^T)$
by (*simp add: Z-point covector-comp-inf vector-conv-compl*)
also have $\dots = S * (1 - Z)$
by (*metis conv-complement one-inf-conv*)
also have $\dots = S * 1 \sqcap S * -Z$
by (*simp add: S-mapping univalent-comp-left-dist-inf*)
also have $\dots = S - M$
by (*simp add: comp-mapping-complement S-mapping*)
finally show *?thesis*
qed

Special case of just 1 number

Theorem 12.2

lemma *M-is-Z-iff-1-is-top:*

$M = Z \iff 1 = \text{top}$

proof

assume $M = Z$

hence $Z = S^T * Z$

by (*metis S-mapping Z-point order.antisym bijective-reverse inf.eq-refl shunt-mapping*)

thus $1 = \text{top}$

by (*metis S-star-Z-top Z-point inf.eq-refl star-left-induct sup.absorb2 symmetric-top-closed top-le*)

next

assume $1 = \text{top}$

thus $M = Z$

using *S-mapping comp-right-one mult-1-left* **by** *auto*

qed

Theorem 12.3

lemma *S-irreflexive:*

assumes $M \neq Z$

shows *irreflexive S*

proof –

have $(S \sqcap 1) * S^T \leq S \sqcap 1$

by (*smt (z3) S-injective S-mapping coreflexive-comp-top-inf dual-order.eq-iff inf.cobounded1 inf.sup-monoid.add-commute inf.sup-same-context mult-left-isotone one-inf-conv top-right-mult-increasing total-var*)

hence $(S \sqcap 1) * S^{T*} \leq S \sqcap 1$

using *star-right-induct-mult* **by** *blast*

hence $(S \sqcap 1) * S^{T*} * Z \leq (S \sqcap 1) * Z$

by (*simp add: mult-left-isotone*)

also have $\dots = M \sqcap Z$

by (*simp add: Z-point injective-comp-right-dist-inf*)

also have $\dots = \text{bot}$

by (*smt (verit, ccfv-threshold) M-point-or-bot assms Z-point*
bijjective-one-closed bijjective-regular comp-associative conv-complement
coreflexive-comp-top-inf epm-3 inf.sup-monoid.add-commute one-inf-conv
regular-mult-closed star.circ-increasing star.circ-zero tarski vector-conv-covector
vector-export-comp-unit)
finally have $S \sqcap 1 \leq \text{bot}$
using *S-star-Z-top comp-associative le-bot top-right-mult-increasing* **by**
fastforce
thus *?thesis*
using *le-bot pseudo-complement* **by** *blast*
qed

We show that S' satisfies most properties of S .

lemma *M-regular:*
regular M
using *S-mapping Z-point bijjective-regular mapping-regular regular-mult-closed*
by *blast*

lemma *S'-regular:*
regular S'
using *S-mapping mapping-regular* **by** *auto*

Theorem 10.3

lemma *S'-star-Z-top:*
 $S^{T^*} * Z = \text{top}$
proof –
have $S^{T^*} * Z = (S' \sqcup (S \sqcap M))^{T^*} * Z$
by (*metis M-regular maddux-3-11-pp S'-var*)
also have $\dots \leq S^{T^*} * Z$
proof (*cases M = bot*)
case *True*
thus *?thesis*
by *simp*
next
case *False*
hence *point M*
using *M-point-or-bot* **by** *auto*
hence *arc (S \sqcap M)*
using *S-mapping mapping-inf-point-arc* **by** *blast*
hence *1: arc ((S \sqcap M)^T)*
using *conv-involutive* **by** *auto*
have *2: S \sqcap M \leq Z^T*
by (*metis S'-var Z-point bijjective-regular conv-complement inf.cobounded2*
p-shunting-swap)
have $(S' \sqcup (S \sqcap M))^{T^*} * Z = (S'^T \sqcup (S \sqcap M)^T)^* * Z$
by (*simp add: S'-var conv-dist-sup*)
also have $\dots = (S'^{T^*} * (S \sqcap M)^T * S'^{T^*} \sqcup S'^{T^*}) * Z$
using *1 star-arc-decompose sup-commute* **by** *auto*
also have $\dots = S'^{T^*} * (S \sqcap M)^T * S'^{T^*} * Z \sqcup S'^{T^*} * Z$

using *mult-right-dist-sup* by *auto*
 also have $\dots \leq S^{T*} * Z^{TT} * S^{T*} * Z \sqcup S^{T*} * Z$
 using 2 by (*meson comp-isotone conv-isotone inf.eq-refl semiring.add-mono*)
 also have $\dots \leq S^{T*} * Z$
 by (*metis Z-point comp-associative conv-involutive le-supI mult-right-isotone top.extremum*)
 finally show *?thesis*
 .
 qed
 finally show *?thesis*
 using *S-star-Z-top top-le* by *auto*
 qed

Theorem 10.5

lemma *Z-below-S'-star*:

$$Z \leq S'^*$$

by (*metis S'-star-Z-top Z-point comp-associative comp-right-one conv-order conv-star-commute mult-right-isotone vector-conv-covector*)

Theorem 10.6

lemma *S'-connected*:

$$S^{T*} * S'^* = \text{top}$$

by (*metis Z-below-S'-star S'-star-Z-top mult-left-dist-sup sup.orderE sup-commute top.extremum*)

Theorem 10.7

lemma *S'-star-connex*:

$$S'^* \sqcup S^{T*} = \text{top}$$

using *S'-connected S'-univalent cancel-separate-eq sup-commute* by *auto*

Theorem 10.8

lemma *Z-sup-conv-S'-top*:

$$Z \sqcup S^{T*} * \text{top} = \text{top}$$

using *S'-star-Z-top star.circ-loop-fixpoint sup-commute* by *auto*

lemma *top-S'-sup-conv-Z*:

$$\text{top} * S' \sqcup Z^T = \text{top}$$

by (*metis S'-star-Z-top conv-dist-comp conv-involutive conv-star-commute star.circ-back-loop-fixpoint symmetric-top-closed*)

lemma *S-power-point-or-bot*:

assumes *regular S'*

shows *point (S^{T*} ^ n * Z) \vee S^{T*} ^ n * Z = bot*

proof –

have 1: *regular (S^{T*} ^ n * Z)*

using *assms Z-point bijective-regular regular-conv-closed regular-mult-closed regular-power-closed* by *auto*

have *injective (S^{T*} ^ n)*

by (*simp add: injective-power-closed S'-univalent*)

```

hence injective ( $S^{T^+} \hat{\ } n * Z$ )
  using Z-point injective-mult-closed by blast
thus ?thesis
  using 1 Z-point comp-associative tarski by force
qed

end

```

7.2 Initialising Ranks

We show that the rank array satisfies three properties which are established/preserved by the union-find operations. First, every node has a rank, that is, the rank array is a mapping. Second, the rank of a node is strictly smaller than the rank of its parent, except if the node is a root. This implies that the rank of a node is an upper bound on the height of its subtree. Third, the number of roots in the disjoint-set forest (the number of disjoint sets) is not larger than $m - k$ where m is the total number of nodes and k is the maximum rank of any node. The third property is useful to show that ranks never overflow (exceed $m - 1$). To compare the number of roots and $m - k$ we use the existence of an injective univalent relation between the set of roots and the set of $m - k$ largest numbers, both represented as vectors. The three properties are captured in *rank-property*.

```

class skra-peano-3 = stone-kleene-relation-algebra-tarski-finite-regular +
skra-peano-2
begin

```

```

definition card-less-eq  $v\ w \equiv \exists i . \text{injective } i \wedge \text{univalent } i \wedge \text{regular } i \wedge v \leq i * w$ 
definition rank-property  $p\ \text{rank} \equiv \text{mapping } \text{rank} \wedge (p - 1) * \text{rank} \leq \text{rank} * S^{T^+}$ 
 $\wedge \text{card-less-eq } (\text{roots } p) (- (S^{T^+} * \text{rank}^T * \text{top}))$ 

```

```

end

```

```

class skra-peano-4 = stone-kleene-relation-algebra-choose-point-finite-regular +
skra-peano-2
begin

```

```

subclass skra-peano-3 ..

```

The initialisation loop is augmented by setting the rank of each node to 0. The resulting rank array satisfies the desired properties explained above.

```

theorem init-ranks:

```

```

  VARs  $h\ p\ x\ \text{rank}$ 
  [ True ]
  FOREACH  $x$ 
  USING  $h$ 
  INV {  $p - h = 1 - h \wedge \text{rank} - h = Z^T - h$  }
  DO  $p := \text{make-set } p\ x;$ 

```



```

      rank[x] := Z
    OD
  [ p = 1 ∧ disjoint-set-forest p ∧ rank = ZT ∧ rank-property p rank ∧ h = bot ]
proof vcg-tc-simp
  fix h p rank
  let ?x = choose-point h
  let ?m = make-set p ?x
  let ?rank = rank[?x→Z]
  assume 1: regular h ∧ vector h ∧ p - h = 1 - h ∧ rank - h = ZT - h ∧ h ≠
  bot
  show vector (-?x ⊔ h) ∧
    ?m ⊔ (--?x ⊔ -h) = 1 ⊔ (--?x ⊔ -h) ∧
    ?rank ⊔ (--?x ⊔ -h) = ZT ⊔ (--?x ⊔ -h) ∧
    card { x . regular x ∧ x ≤ -?x ∧ x ≤ h } < h↓
  proof (intro conjI)
    show vector (-?x ⊔ h)
      using 1 choose-point-point vector-complement-closed vector-inf-closed by
  blast
    have 2: point ?x ∧ regular ?x
      using 1 bijective-regular choose-point-point by blast
    have 3: -h ≤ -?x
      using choose-point-decreasing p-antitone-iff by auto
    have 4: ?x ⊔ ?m = ?x * ?xT ∧ -?x ⊔ ?m = -?x ⊔ p
      using 1 choose-point-point make-set-function make-set-postcondition-def by
  auto
    have ?m ⊔ (--?x ⊔ -h) = (?m ⊔ ?x) ⊔ (?m - h)
      using 2 comp-inf.comp-left-dist-sup by auto
    also have ... = ?x * ?xT ⊔ (?m ⊔ -?x ⊔ -h)
      using 3 4 by (smt (z3) inf-absorb2 inf-assoc inf-commute)
    also have ... = ?x * ?xT ⊔ (1 - h)
      using 1 3 4 inf.absorb2 inf.sup-monoid.add-assoc inf-commute by auto
    also have ... = (1 ⊔ ?x) ⊔ (1 - h)
      using 2 by (metis inf.cobounded2 inf.sup-same-context one-inf-conv
  vector-covector)
    also have ... = 1 ⊔ (--?x ⊔ -h)
      using 2 comp-inf.semiring.distrib-left by auto
    finally show ?m ⊔ (--?x ⊔ -h) = 1 ⊔ (--?x ⊔ -h)
  .
    have 5: ?x ⊔ ?rank = ?x ⊔ ZT ∧ -?x ⊔ ?rank = -?x ⊔ rank
      by (smt (z3) inf-commute order-refl update-inf-different update-inf-same)
    have ?rank ⊔ (--?x ⊔ -h) = (?rank ⊔ ?x) ⊔ (?rank - h)
      using 2 comp-inf.comp-left-dist-sup by auto
    also have ... = (?x ⊔ ZT) ⊔ (?rank ⊔ -?x ⊔ -h)
      using 3 5 by (smt (z3) inf-absorb2 inf-assoc inf-commute)
    also have ... = (ZT ⊔ ?x) ⊔ (ZT - h)
      using 1 3 5 inf.absorb2 inf.sup-monoid.add-assoc inf-commute by auto
    also have ... = ZT ⊔ (--?x ⊔ -h)
      using 2 comp-inf.semiring.distrib-left by auto
    finally show ?rank ⊔ (--?x ⊔ -h) = ZT ⊔ (--?x ⊔ -h)

```

```

have 5:  $\neg ?x \leq -?x$ 
  using 1 2 by (metis comp-commute-below-diversity conv-order
inf.cobounded2 inf-absorb2 pseudo-complement strict-order-var top.extremum)
  have 6:  $?x \leq h$ 
    using 1 by (metis choose-point-decreasing)
  show  $\text{card } \{ x . \text{regular } x \wedge x \leq -?x \wedge x \leq h \} < h \downarrow$ 
    apply (rule psubset-card-mono)
    using finite-regular apply simp
    using 2 5 6 by auto
qed
next
show rank-property 1 ( $Z^T$ )
proof (unfold rank-property-def, intro conjI)
  show univalent ( $Z^T$ ) total ( $Z^T$ )
    using Z-point surjective-conv-total by auto
  show  $(1 - 1) * (Z^T) \leq (Z^T) * S'^+$ 
    by simp
  have  $\text{top} \leq 1 * -(S'^+ * Z * \text{top})$ 
    by (simp add: S'-Z comp-associative star-simulation-right-equal)
  thus card-less-eq (roots 1) ( $-(S'^+ * Z^{TT} * \text{top})$ )
    by (metis conv-involutive inf.idem mapping-one-closed regular-one-closed
card-less-eq-def bijective-one-closed)
qed
qed
end

```

7.3 Union by Rank

We show that path compression and union-by-rank preserve the rank property.

context *stone-kleene-relation-algebra-tarski-finite-regular*
begin

lemma *union-sets-1-swap*:

```

assumes union-sets-precondition  $p0\ x\ y$ 
  and path-compression-postcondition  $p1\ x\ r\ p0$ 
  and path-compression-postcondition  $p2\ y\ s\ p1$ 
shows union-sets-postcondition ( $p2[s \rightarrow r]$ )  $x\ y\ p0$ 
proof (unfold union-sets-postcondition-def union-sets-precondition-def, intro
conjI)
  let  $?p = p2[s \rightarrow r]$ 
  have 1: disjoint-set-forest  $p1 \wedge \text{point } r \wedge r = \text{root } p1\ x \wedge p1 \sqcap 1 = p0 \sqcap 1 \wedge$ 
fc  $p1 = \text{fc } p0$ 
    using assms(2) path-compression-precondition-def
path-compression-postcondition-def by auto
  have 2: disjoint-set-forest  $p2 \wedge \text{point } s \wedge s = \text{root } p2\ y \wedge p2 \sqcap 1 = p1 \sqcap 1 \wedge$ 
fc  $p2 = \text{fc } p1$ 

```

```

    using assms(3) path-compression-precondition-def
path-compression-postcondition-def by auto
  hence 3:  $fc\ p2 = fc\ p0$ 
    using 1 by simp
  show 4: univalent ?p
    using 1 2 update-univalent by blast
  show total ?p
    using 1 2 bijective-regular update-total by blast
  show acyclic (?p - 1)
  proof (cases  $r = s$ )
    case True
      thus ?thesis
        using 2 update-acyclic-5 by fastforce
    next
      case False
        hence  $bot = s \sqcap r$ 
          using 1 2 distinct-points inf-commute by blast
        also have  $\dots = s \sqcap p1^{T^*} * r$ 
          using 1 by (smt root-transitive-successor-loop)
        also have  $\dots = s \sqcap p2^{T^*} * r$ 
          using 1 2 by (smt (z3) inf-assoc inf-commute same-root)
        finally have  $r \sqcap p2^{T^*} * s = bot$ 
          using schroeder-1 conv-star-commute inf.sup-monoid.add-commute by
fastforce
        thus ?thesis
          using 1 2 update-acyclic-4 by blast
    qed
  show vector x
    using assms(1) by (simp add: union-sets-precondition-def)
  show injective x
    using assms(1) by (simp add: union-sets-precondition-def)
  show surjective x
    using assms(1) by (simp add: union-sets-precondition-def)
  show vector y
    using assms(1) by (simp add: union-sets-precondition-def)
  show injective y
    using assms(1) by (simp add: union-sets-precondition-def)
  show surjective y
    using assms(1) by (simp add: union-sets-precondition-def)
  show  $fc\ ?p = wcc\ (p0 \sqcup x * y^T)$ 
  proof (rule order.antisym)
    have  $s = p2[[s]]$ 
      using 2 by (metis root-successor-loop)
    hence  $s * s^T \leq p2^T$ 
      using 2 eq-refl shunt-bijective by blast
    hence  $s * s^T \leq p2$ 
      using 2 conv-order coreflexive-symmetric by fastforce
    hence  $s \leq p2 * s$ 
      using 2 shunt-bijective by blast
  end

```

hence 5: $p2[[s]] \leq s$
 using 2 *shunt-mapping* by *blast*
 have $s \sqcap p2 \leq s * (top \sqcap s^T * p2)$
 using 2 by (*metis dedekind-1*)
 also have $\dots = s * s^T * p2$
 by (*simp add: mult-assoc*)
 also have $\dots \leq s * s^T$
 using 5 by (*metis comp-associative conv-dist-comp conv-involutive conv-order mult-right-isotone*)
 also have $\dots \leq 1$
 using 2 by *blast*
 finally have 6: $s \sqcap p2 \leq 1$
 by *simp*
 have $p0 \leq wcc\ p0$
 by (*simp add: star.circ-sub-dist-1*)
 also have $\dots = wcc\ p2$
 using 3 by (*simp add: star-decompose-1*)
 also have 7: $\dots \leq wcc\ ?p$
proof –
 have $wcc\ p2 = wcc\ ((-s \sqcap p2) \sqcup (s \sqcap p2))$
 using 2 by (*metis bijective-regular inf.sup-monoid.add-commute maddux-3-11-pp*)
 also have $\dots \leq wcc\ ((-s \sqcap p2) \sqcup 1)$
 using 6 *wcc-isotone sup-right-isotone* by *simp*
 also have $\dots = wcc\ (-s \sqcap p2)$
 using *wcc-with-loops* by *simp*
 also have $\dots \leq wcc\ ?p$
 using *wcc-isotone sup-ge2* by *blast*
finally show *?thesis*
 by *simp*
qed
 finally have 8: $p0 \leq wcc\ ?p$
 by *force*
 have $s \leq p2^{T*} * y$
 using 2 by (*metis inf-le1*)
 hence 9: $s * y^T \leq p2^{T*}$
 using *assms(1) shunt-bijective union-sets-precondition-def* by *blast*
 hence $y * s^T \leq p2^*$
 using *conv-dist-comp conv-order conv-star-commute* by *force*
 also have $\dots \leq wcc\ p2$
 by (*simp add: star.circ-sub-dist*)
 also have $\dots \leq wcc\ ?p$
 using 7 by *simp*
 finally have 10: $y * s^T \leq wcc\ ?p$
 by *simp*
 have 11: $s * r^T \leq wcc\ ?p$
 using 1 2 *star.circ-sub-dist-1 sup-assoc vector-covector* by *auto*
 have $r \leq p1^{T*} * x$
 using 1 by (*metis inf-le1*)

hence 12: $r * x^T \leq p1^{T*}$
 using *assms(1) shunt-bijective union-sets-precondition-def* by *blast*
 also have $\dots \leq wcc\ p1$
 using *star-isotone sup-ge2* by *blast*
 also have $\dots = wcc\ p2$
 using 2 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc\ ?p$
 using 7 by *simp*
 finally have 13: $r * x^T \leq wcc\ ?p$
 by *simp*
 have $x \leq x * r^T * r \wedge y \leq y * s^T * s$
 using 1 2 *shunt-bijective* by *blast*
 hence $y * x^T \leq y * s^T * s * (x * r^T * r)^T$
 using *comp-isotone conv-isotone* by *blast*
 also have $\dots = y * s^T * s * r^T * r * x^T$
 by (*simp add: comp-associative conv-dist-comp*)
 also have $\dots \leq wcc\ ?p * (s * r^T) * (r * x^T)$
 using 10 by (*metis mult-left-isotone mult-assoc*)
 also have $\dots \leq wcc\ ?p * wcc\ ?p * (r * x^T)$
 using 11 by (*metis mult-left-isotone mult-right-isotone*)
 also have $\dots \leq wcc\ ?p * wcc\ ?p * wcc\ ?p$
 using 13 by (*metis mult-right-isotone*)
 also have $\dots = wcc\ ?p$
 by (*simp add: star.circ-transitive-equal*)
 finally have $x * y^T \leq wcc\ ?p$
 by (*metis conv-dist-comp conv-involutive conv-order wcc-equivalence*)
 hence $p0 \sqcup x * y^T \leq wcc\ ?p$
 using 8 by *simp*
 hence $wcc\ (p0 \sqcup x * y^T) \leq wcc\ ?p$
 using *wcc-below-wcc* by *simp*
 thus $wcc\ (p0 \sqcup x * y^T) \leq fc\ ?p$
 using 4 *fc-wcc* by *simp*
 have $-s \sqcap p2 \leq wcc\ p2$
 by (*simp add: inf.coboundedI2 star.circ-sub-dist-1*)
 also have $\dots = wcc\ p0$
 using 3 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc\ (p0 \sqcup y * x^T)$
 by (*simp add: wcc-isotone*)
 finally have 14: $-s \sqcap p2 \leq wcc\ (p0 \sqcup y * x^T)$
 by *simp*
 have $s * y^T \leq wcc\ p2$
 using 9 *inf.order-trans star.circ-sub-dist sup-commute* by *fastforce*
 also have $\dots = wcc\ p0$
 using 1 2 by (*simp add: star-decompose-1*)
 also have $\dots \leq wcc\ (p0 \sqcup y * x^T)$
 by (*simp add: wcc-isotone*)
 finally have 15: $s * y^T \leq wcc\ (p0 \sqcup y * x^T)$
 by *simp*
 have 16: $y * x^T \leq wcc\ (p0 \sqcup y * x^T)$

```

    using le-supE star.circ-sub-dist-1 by blast
  have  $x * r^T \leq p1^*$ 
    using 12 conv-dist-comp conv-order conv-star-commute by fastforce
  also have  $\dots \leq wcc\ p1$ 
    using star.circ-sub-dist sup-commute by fastforce
  also have  $\dots = wcc\ p0$ 
    using 1 by (simp add: star-decompose-1)
  also have  $\dots \leq wcc\ (p0 \sqcup y * x^T)$ 
    by (simp add: wcc-isotone)
  finally have 17:  $x * r^T \leq wcc\ (p0 \sqcup y * x^T)$ 
    by simp
  have  $r \leq r * x^T * x \wedge s \leq s * y^T * y$ 
    using assms(1) shunt-bijective union-sets-precondition-def by blast
  hence  $s * r^T \leq s * y^T * y * (r * x^T * x)^T$ 
    using comp-isotone conv-isotone by blast
  also have  $\dots = s * y^T * y * x^T * x * r^T$ 
    by (simp add: comp-associative conv-dist-comp)
  also have  $\dots \leq wcc\ (p0 \sqcup y * x^T) * (y * x^T) * (x * r^T)$ 
    using 15 by (metis mult-left-isotone mult-assoc)
  also have  $\dots \leq wcc\ (p0 \sqcup y * x^T) * wcc\ (p0 \sqcup y * x^T) * (x * r^T)$ 
    using 16 by (metis mult-left-isotone mult-right-isotone)
  also have  $\dots \leq wcc\ (p0 \sqcup y * x^T) * wcc\ (p0 \sqcup y * x^T) * wcc\ (p0 \sqcup y * x^T)$ 
    using 17 by (metis mult-right-isotone)
  also have  $\dots = wcc\ (p0 \sqcup y * x^T)$ 
    by (simp add: star.circ-transitive-equal)
  finally have ?p  $\leq wcc\ (p0 \sqcup y * x^T)$ 
    using 1 2 14 vector-covector by auto
  hence  $wcc\ ?p \leq wcc\ (p0 \sqcup y * x^T)$ 
    using wcc-below-wcc by blast
  also have  $\dots = wcc\ (p0 \sqcup x * y^T)$ 
    using conv-dist-comp conv-dist-sup sup-assoc sup-commute by auto
  finally show fc ?p  $\leq wcc\ (p0 \sqcup x * y^T)$ 
    using 4 fc-wcc by simp

```

qed

qed

lemma *union-sets-1-skip*:

```

  assumes union-sets-precondition p0 x y
    and path-compression-postcondition p1 x r p0
    and path-compression-postcondition p2 y r p1
  shows union-sets-postcondition p2 x y p0
proof (unfold union-sets-postcondition-def union-sets-precondition-def, intro
  conjI)
  have 1:  $point\ r \wedge r = root\ p1\ x \wedge fc\ p1 = fc\ p0 \wedge disjoint-set-forest\ p2 \wedge r =$ 
 $root\ p2\ y \wedge fc\ p2 = fc\ p1$ 
    using assms(2,3) path-compression-precondition-def
  path-compression-postcondition-def by auto
  thus univalent p2 total p2 acyclic (p2 - 1)
    by auto

```

```

show vector x injective x surjective x vector y injective y surjective y
  using assms(1) union-sets-precondition-def by auto
have  $r \leq p1^{T^*} * x$ 
  using 1 by (metis inf-le1)
hence  $r * x^T \leq p1^{T^*}$ 
  using assms(1) shunt-bijective union-sets-precondition-def by blast
hence 2:  $x * r^T \leq p1^*$ 
  using conv-dist-comp conv-order conv-star-commute by force
have  $r \leq p2^{T^*} * y$ 
  using 1 by (metis inf-le1)
hence 3:  $r * y^T \leq p2^{T^*}$ 
  using assms(1) shunt-bijective union-sets-precondition-def by blast
have  $x * y^T \leq x * r^T * r * y^T$ 
  using 1 mult-left-isotone shunt-bijective by blast
also have  $\dots \leq p1^* * p2^{T^*}$ 
  using 2 3 by (metis comp-associative comp-isotone)
also have  $\dots \leq wcc\ p0$ 
  using 1 by (metis star.circ-mult-upper-bound star-decompose-1 star-isotone
sup-ge2 star.circ-sub-dist)
finally show fc  $p2 = wcc\ (p0 \sqcup x * y^T)$ 
  using 1 by (smt (z3) fc-star star-decompose-1 sup-absorb1 wcc-sup-wcc
star.circ-sub-dist-3 sup-commute wcc-equivalence)
qed

```

end

syntax

```

-Cond1 :: 'bexp  $\Rightarrow$  'com  $\Rightarrow$  'com ((1IF -/ THEN -/ FI) [0,0] 61)
translations IF b THEN c FI == IF b THEN c ELSE SKIP FI

```

context *skra-peano-3*

begin

lemma *path-compression-preserves-rank-property*:

```

  assumes path-compression-postcondition p x y p0
    and disjoint-set-forest p0
    and rank-property p0 rank
  shows rank-property p rank
proof (unfold rank-property-def, intro conjI)
  let  $?px = p0^{T^*} * x$ 
  have 1: point y
    using assms(1,2) path-compression-postcondition-def
path-compression-precondition-def root-point by auto
  have 2: vector  $?px$ 
    using assms(1) comp-associative path-compression-postcondition-def
path-compression-precondition-def by auto
  have root p0 x = root p x
    by (smt (verit) assms(1,2) path-compression-postcondition-def
path-compression-precondition-def same-root)

```

hence $root\ p0\ x = y$
using $assms(1)\ path-compression-postcondition-def$
path-compression-precondition-def **by** *auto*
hence $?px \leq p0^* * y$
by ($meson\ assms(1,2)\ path-splitting-invariant-aux-1(3)$)
path-compression-precondition-def path-compression-postcondition-def
hence $?px * y^T \leq p0^*$
using $1\ shunt-bijective$ **by** *blast*
hence $?px \sqcap y^T \leq p0^*$
using $1\ 2$ **by** ($simp\ add: vector-covector$)
also have $\dots = (p0 - 1)^+ \sqcup 1$
using $reachable-without-loops\ star-left-unfold-equal\ sup-commute$ **by** *fastforce*
finally have $\exists: ?px \sqcap y^T \sqcap -1 \leq (p0 - 1)^+$
using $half-shunting$ **by** *blast*
have $p0[?px \rightarrow y] = p$
using $assms(1)\ path-compression-postcondition-def$ **by** *simp*
hence $(p - 1) * rank = (?px \sqcap y^T \sqcap -1) * rank \sqcup (-?px \sqcap p0 \sqcap -1) * rank$
using $inf-sup-distrib2\ mult-right-dist-sup$ **by** *force*
also have $\dots \leq (?px \sqcap y^T \sqcap -1) * rank \sqcup (p0 - 1) * rank$
by ($meson\ comp-inf.mult-semi-associative\ le-infE\ mult-left-isotone$)
sup-right-isotone
also have $\dots \leq (?px \sqcap y^T \sqcap -1) * rank \sqcup rank * S'^+$
using $assms(3)\ rank-property-def\ sup-right-isotone$ **by** *auto*
also have $\dots \leq (p0 - 1)^+ * rank \sqcup rank * S'^+$
using $3\ mult-left-isotone\ sup-left-isotone$ **by** *blast*
also have $\dots \leq rank * S'^+$
proof –
have $(p0 - 1)^* * rank \leq rank * S'^*$
using $assms(3)\ rank-property-def\ star-simulation-left\ star.left-plus-circ$ **by**
fastforce
hence $(p0 - 1)^+ * rank \leq (p0 - 1) * rank * S'^*$
by ($simp\ add: comp-associative\ mult-right-isotone$)
also have $\dots \leq rank * S'^+$
by ($smt\ (z3)\ assms(3)\ rank-property-def\ comp-associative$)
comp-left-subdist-inf\ inf.boundedE\ inf.sup-right-divisibility
star.circ-transitive-equal
finally show $?thesis$
by *simp*
qed
finally show $(p - 1) * rank \leq rank * S'^+$
 \cdot
show $univalent\ rank\ total\ rank$
using $rank-property-def\ assms(3)$ **by** *auto*
show $card-less-eq\ (roots\ p)\ (- (S'^+ * rank^T * top))$
using $assms(1,3)\ path-compression-postcondition-def\ rank-property-def$ **by**
auto
qed

theorem *union-sets-by-rank*:


```

VARS p r s rank
[ union-sets-precondition p x y  $\wedge$  rank-property p rank  $\wedge$  p0 = p ]
r := find-set p x;
p := path-compression p x r;
s := find-set p y;
p := path-compression p y s;
IF r  $\neq$  s THEN
  IF rank[[r]]  $\leq$  S+ * (rank[[s]]) THEN
    p[r] := s
  ELSE
    p[s] := r;
    IF rank[[r]] = rank[[s]] THEN
      rank[r] := ST * (rank[[r]])
    FI
  FI
FI
[ union-sets-postcondition p x y p0  $\wedge$  rank-property p rank ]
proof vcg-tc-simp
fix rank
let ?r = find-set p0 x
let ?p1 = path-compression p0 x ?r
let ?s = find-set ?p1 y
let ?p2 = path-compression ?p1 y ?s
let ?p5 = path-compression ?p1 y ?r
let ?rr = rank[[?r]]
let ?rs = rank[[?s]]
let ?rank = rank[?r  $\mapsto$  ST * ?rs]
let ?p3 = ?p2[?r  $\mapsto$  ?s]
let ?p4 = ?p2[?s  $\mapsto$  ?r]
assume 1: union-sets-precondition p0 x y  $\wedge$  rank-property p0 rank
hence 2: path-compression-postcondition ?p1 x ?r p0
  using find-set-function find-set-postcondition-def find-set-precondition-def
path-compression-function path-compression-precondition-def
union-sets-precondition-def by auto
  hence 3: path-compression-postcondition ?p2 y ?s ?p1
    using 1 find-set-function find-set-postcondition-def find-set-precondition-def
path-compression-function path-compression-precondition-def
union-sets-precondition-def path-compression-postcondition-def by meson
    have rank-property ?p1 rank
      using 1 2 path-compression-preserves-rank-property
union-sets-precondition-def by blast
    hence 4: rank-property ?p2 rank
      using 1 2 3 by (meson path-compression-preserves-rank-property
path-compression-postcondition-def path-compression-precondition-def)
    have 5: point ?r point ?s
      using 2 3 path-compression-postcondition-def
path-compression-precondition-def by auto
    hence 6: point ?rr point ?rs
      using 1 comp-associative read-injective read-surjective rank-property-def by

```

auto
have $top \leq S'^* \sqcup S'^{+T}$
by (*metis S'-star-connex conv-dist-comp conv-star-commute eq-refl star.circ-reflexive star-left-unfold-equal star-simulation-right-equal sup.orderE sup-monoid.add-assoc*)
hence $\gamma: -S'^{+T} \leq S'^*$
by (*metis comp-inf.case-split-left comp-inf.star.circ-plus-one comp-inf.star.circ-sup-2 half-shunting*)
show $(?r \neq ?s \longrightarrow (?rr \leq S'^+ * ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0 \wedge rank\text{-property } ?p3 \ rank) \wedge$
 $(\neg ?rr \leq S'^+ * ?rs \longrightarrow ((?rr = ?rs \longrightarrow$
 $union\text{-sets}\text{-postcondition } ?p4 \ x \ y \ p0 \wedge rank\text{-property } ?p4 \ ?rank) \wedge$
 $(?rr \neq ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p4 \ x$
 $y \ p0 \wedge rank\text{-property } ?p4 \ rank)))) \wedge$
 $(?r = ?s \longrightarrow union\text{-sets}\text{-postcondition } ?p5 \ x \ y \ p0 \wedge rank\text{-property } ?p5 \ rank)$
proof
show $?r \neq ?s \longrightarrow (?rr \leq S'^+ * ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0$
 $\wedge rank\text{-property } ?p3 \ rank) \wedge$
 $(\neg ?rr \leq S'^+ * ?rs \longrightarrow ((?rr = ?rs \longrightarrow$
 $union\text{-sets}\text{-postcondition } ?p4 \ x \ y \ p0 \wedge rank\text{-property } ?p4 \ ?rank) \wedge$
 $(?rr \neq ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p4 \ x$
 $y \ p0 \wedge rank\text{-property } ?p4 \ rank))))$
proof
assume $\delta: ?r \neq ?s$
show $(?rr \leq S'^+ * ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0 \wedge$
 $rank\text{-property } ?p3 \ rank) \wedge$
 $(\neg ?rr \leq S'^+ * ?rs \longrightarrow ((?rr = ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p4 \ x$
 $y \ p0 \wedge rank\text{-property } ?p4 \ ?rank) \wedge$
 $(?rr \neq ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p4 \ x \ y \ p0 \wedge$
 $rank\text{-property } ?p4 \ rank))))$
proof
show $?rr \leq S'^+ * ?rs \longrightarrow union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0 \wedge$
 $rank\text{-property } ?p3 \ rank$
proof
assume $\delta: ?rr \leq S'^+ * ?rs$
show $union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0 \wedge rank\text{-property } ?p3 \ rank$
proof
show $union\text{-sets}\text{-postcondition } ?p3 \ x \ y \ p0$
using $1 \ 2 \ 3$ **by** (*simp add: union-sets-1*)
show $rank\text{-property } ?p3 \ rank$
proof (*unfold rank-property-def, intro conjI*)
show $univalent \ rank \ total \ rank$
using 1 $rank\text{-property}\text{-def}$ **by** *auto*
have $?s \leq -?r$
using $5 \ 8$ **by** (*meson order.antisym bijective-regular point-in-vector-or-complement point-in-vector-or-complement-2*)
hence $?r \sqcap ?s^T \sqcap 1 = bot$
by (*metis (full-types) bot-least inf.left-commute inf.sup-monoid.add-commute one-inf-conv pseudo-complement*)

hence $?p3 \sqcap 1 \leq ?p2$
by (*smt half-shunting inf.cobounded2 pseudo-complement regular-one-closed semiring.add-mono sup-commute*)
hence $\text{roots } ?p3 \leq \text{roots } ?p2$
by (*simp add: mult-left-isotone*)
thus $\text{card-less-eq } (\text{roots } ?p3) \text{ } (-S^{'+} * \text{rank}^T * \text{top})$
using 4 **by** (*meson rank-property-def card-less-eq-def order-trans*)
have $(?p3 - 1) * \text{rank} = (?r \sqcap ?s^T \sqcap -1) * \text{rank} \sqcup (-?r \sqcap ?p2 \sqcap -1) * \text{rank}$
using *comp-inf.semiring.distrib-right mult-right-dist-sup* **by** *auto*
also have $\dots \leq (?r \sqcap ?s^T \sqcap -1) * \text{rank} \sqcup (?p2 - 1) * \text{rank}$
using *comp-inf.mult-semi-associative mult-left-isotone sup-right-isotone* **by** *auto*
also have $\dots \leq (?r \sqcap ?s^T \sqcap -1) * \text{rank} \sqcup \text{rank} * S^{'+}$
using 4 *sup-right-isotone rank-property-def* **by** *blast*
also have $\dots \leq (?r \sqcap ?s^T) * \text{rank} \sqcup \text{rank} * S^{'+}$
using *inf-le1 mult-left-isotone sup-left-isotone* **by** *blast*
also have $\dots = ?r * ?s^T * \text{rank} \sqcup \text{rank} * S^{'+}$
using 5 **by** (*simp add: vector-covector*)
also have $\dots = \text{rank} * S^{'+}$
proof –
have $\text{rank}^T * ?r \leq S^{'+} * \text{rank}^T * ?s$
using 9 *comp-associative* **by** *auto*
hence $?r \leq \text{rank} * S^{'+} * \text{rank}^T * ?s$
using 4 *shunt-mapping comp-associative rank-property-def* **by** *auto*
hence $?r * ?s^T \leq \text{rank} * S^{'+} * \text{rank}^T$
using 5 *shunt-bijective* **by** *blast*
hence $?r * ?s^T * \text{rank} \leq \text{rank} * S^{'+}$
using 4 *shunt-bijective rank-property-def mapping-conv-bijective* **by** *auto*
thus *?thesis*
using *sup-absorb2* **by** *blast*
qed
finally show $(?p3 - 1) * \text{rank} \leq \text{rank} * S^{'+}$
qed
qed
qed
show $\neg ?rr \leq S^{'+} * ?rs \longrightarrow ((?rr = ?rs \longrightarrow \text{union-sets-postcondition } ?p4 \ x \ y \ p0 \wedge \text{rank-property } ?p4 \ ?rank) \wedge$
 $(?rr \neq ?rs \longrightarrow \text{union-sets-postcondition } ?p4 \ x \ y \ p0 \wedge \text{rank-property } ?p4 \ \text{rank}))$
proof
assume $\neg ?rr \leq S^{'+} * ?rs$
hence $?rr \leq -(S^{'+} * ?rs)$
using 6 **by** (*meson point-in-vector-or-complement S'-regular bijective-regular regular-closed-star regular-mult-closed vector-mult-closed*)
also have $\dots = -S^{'+} * ?rs$
using 6 *comp-bijective-complement* **by** *simp*

finally have $?rs \leq -S'^{+T} * ?rr$
using 6 **by** (*metis bijective-reverse conv-complement*)
also have $\dots \leq S'^{*} * ?rr$
using 7 **by** (*simp add: mult-left-isotone*)
also have $\dots = S'^{+} * ?rr \sqcup ?rr$
using *star.circ-loop-fixpoint mult-assoc* **by** *auto*
finally have 10: $?rs - ?rr \leq S'^{+} * ?rr$
using *half-shunting* **by** *blast*
show ($(?rr = ?rs \longrightarrow \text{union-sets-postcondition } ?p4 \ x \ y \ p0 \ \wedge$
rank-property $?p4 \ ?rank) \ \wedge$
 $(?rr \neq ?rs \longrightarrow \text{union-sets-postcondition } ?p4 \ x \ y \ p0 \ \wedge \text{rank-property}$
 $?p4 \ ?rank)$)
proof
show $?rr = ?rs \longrightarrow \text{union-sets-postcondition } ?p4 \ x \ y \ p0 \ \wedge$
rank-property $?p4 \ ?rank$
proof
assume 11: $?rr = ?rs$
show $\text{union-sets-postcondition } ?p4 \ x \ y \ p0 \ \wedge \text{rank-property } ?p4 \ ?rank$
proof
show $\text{union-sets-postcondition } ?p4 \ x \ y \ p0$
using 1 2 3 **by** (*simp add: union-sets-1-swap*)
show $\text{rank-property } ?p4 \ ?rank$
proof (*unfold rank-property-def, intro conjI*)
show *univalent* $?rank$
using 4 5 6 **by** (*meson S'-univalent read-injective*
update-univalent rank-property-def)
have *card-less-eq* (*roots* $?p2$) $(-(S'^{+} * \text{rank}^T * \text{top}))$
using 4 *rank-property-def* **by** *blast*
from *this* **obtain** *i* **where** 12: $\text{injective } i \ \wedge \ \text{univalent } i \ \wedge \ \text{regular } i$
 $\wedge \ \text{roots } ?p2 \leq i * -(S'^{+} * \text{rank}^T * \text{top})$
using *card-less-eq-def* **by** *blast*
let $?i = (i[?s \longrightarrow i[[i * ?rr]]])[i * ?rr \longrightarrow i[[?s]]]$
have 13: $?i = (i * ?rr \sqcap ?s^T * i) \sqcup (-(i * ?rr) \sqcap ?s \sqcap ?rr^T * i^T$
 $* i) \sqcup (-(i * ?rr) \sqcap -?s \sqcap i)$
by (*smt (z3) conv-dist-comp conv-involutive*
inf.sup-monoid.add-assoc inf-sup-distrib1 sup-assoc)
have 14: *injective* $?i$
apply (*rule update-injective-swap*)
subgoal **using** 12 **by** *simp*
subgoal **using** 5 **by** *simp*
subgoal **using** 6 12 *injective-mult-closed* **by** *simp*
subgoal **using** 5 *comp-associative* **by** *simp*
done
have 15: *univalent* $?i$
apply (*rule update-univalent-swap*)
subgoal **using** 12 **by** *simp*
subgoal **using** 5 **by** *simp*
subgoal **using** 5 **by** *simp*
subgoal **using** 6 12 *injective-mult-closed* **by** *simp*

```

    subgoal using 5 comp-associative by simp
    done
  have 16: regular ?i
    using 5 6 12 by (smt (z3) bijective-regular p-dist-inf p-dist-sup
pp-dist-comp regular-closed-inf regular-conv-closed)
  have 17: regular (i * ?rr)
    using 6 12 bijective-regular regular-mult-closed by blast
  have 18: find-set-precondition ?p1 y
    using 2 3 find-set-precondition-def
path-compression-postcondition-def path-compression-precondition-def by blast
  hence ?s = root ?p1 y
    by (meson find-set-function find-set-postcondition-def)
  also have ... = root ?p2 y
    using 3 18 by (smt (z3) find-set-precondition-def
path-compression-postcondition-def path-compression-precondition-def same-root)
  also have ... ≤ roots ?p2
    by simp
  also have ... ≤ i * -(S+ * rankT * top)
    using 12 by simp
  finally have 19: ?s ≤ i * -(S+ * rankT * top)
    .
  have roots ?p4 ≤ ?i * -(S+ * ?rankT * top)
  proof -
    have ?r ≤ -?s
      using 5 8 by (meson order.antisym bijective-regular
point-in-vector-or-complement point-in-vector-or-complement-2)
    hence ?s ⊓ ?rT ⊓ 1 = bot
      by (metis (full-types) bot-least inf.left-commute
inf.sup-monoid.add-commute one-inf-conv pseudo-complement)
    hence ?p4 ⊓ 1 ≤ -?s ⊓ ?p2
      by (smt (z3) bot-least comp-inf.semiring.distrib-left
inf.cobounded2 inf.sup-monoid.add-commute le-supI)
    hence roots ?p4 ≤ roots (-?s ⊓ ?p2)
      by (simp add: mult-left-isotone)
    also have ... = -?s ⊓ roots ?p2
      using 5 inf-assoc vector-complement-closed vector-inf-comp by
auto
  also have ... = (i * ?rr ⊓ -?s ⊓ roots ?p2) ⊔ (-(i * ?rr) ⊓ -?s
⊓ roots ?p2)
    using 17 by (smt (z3) comp-inf.star-plus inf-sup-distrib2
inf-top.right-neutral regular-complement-top)
  also have ... ≤ ?i * -(S+ * ?rankT * top)
  proof (rule sup-least)
    have ?rankT * top = (?r ⊓ (ST * ?rs)T)T * top ⊔ (-?r ⊓
rank)T * top
      using conv-dist-sup mult-right-dist-sup by auto
    also have ... = (?rT ⊓ ST * ?rs) * top ⊔ (-?rT ⊓ rankT) * top
      using conv-complement conv-dist-inf conv-involutive by auto
    also have ... = ST * ?rs * (?r ⊓ top) ⊔ (-?rT ⊓ rankT) * top

```

using 5 by (*smt (z3) covector-inf-comp-3 inf-commute*)
also have $\dots = S^{T^*} * ?rs * (?r \sqcap top) \sqcup rank^T * (-?r \sqcap top)$
using 5 by (*smt (z3) conv-complement*)
vector-complement-closed covector-inf-comp-3 inf-commute
also have $\dots = S^{T^*} * ?rs * ?r \sqcup rank^T * -?r$
by simp
also have $\dots \leq S^{T^*} * ?rs * ?r \sqcup rank^T * top$
using *mult-right-isotone sup-right-isotone* **by force**
also have $\dots \leq S^{T^*} * ?rs \sqcup rank^T * top$
using 5 6 by (*metis inf.eq-refl mult-assoc*)
finally have $S^{'+} * ?rank^T * top \leq S^{'+} * S^{T^*} * ?rs \sqcup S^{'+} * rank^T * top$
*rank^T * top*
by (*smt comp-associative mult-left-dist-sup mult-right-isotone*)
also have $\dots = S^{'+} * (S' * S^{T^*}) * ?rs \sqcup S^{'+} * rank^T * top$
by (*smt star-plus mult-assoc*)
also have $\dots \leq S^{'+} * ?rs \sqcup S^{'+} * rank^T * top$
by (*metis S'-injective comp-right-one mult-left-isotone*)
mult-right-isotone sup-left-isotone
also have $\dots = ?rs \sqcup S^{'+} * ?rs \sqcup S^{'+} * rank^T * top$
using *comp-associative star.circ-loop-fixpoint sup-commute* **by**
fastforce
also have $\dots = ?rs \sqcup S^{'+} * rank^T * top$
by (*smt (verit, del-insts) comp-associative mult-left-dist-sup*)
sup.orderE sup-assoc sup-commute top.extremum
finally have 20: $S^{'+} * ?rank^T * top \leq ?rs \sqcup S^{'+} * rank^T * top$
 \cdot
have $?s * ?s^T = (?s \sqcap i * -(S^{'+} * rank^T * top)) * ?s^T$
using 19 inf.orderE **by fastforce**
also have $\dots = (?s \sqcap i * -(S^{'+} * rank^T * top)) * top \sqcap ?s^T$
using 5 by (*smt (z3) covector-comp-inf vector-conv-covector*)
vector-covector vector-top-closed
also have $\dots = ?s \sqcap i * -(S^{'+} * rank^T * top) * top \sqcap ?s^T$
using 5 vector-inf-comp **by auto**
also have $\dots \leq 1 \sqcap i * -(S^{'+} * rank^T * top) * top$
using 5 by (*smt (verit, ccfv-SIG) inf.cobounded1*)
inf.cobounded2 inf-greatest order-trans vector-covector
also have $\dots = 1 \sqcap i * -(S^{'+} * rank^T * top)$
using *comp-associative vector-complement-closed*
vector-top-closed **by auto**
also have $\dots \leq 1 \sqcap i * -(S^{'+} * rank^T)$
by (*meson comp-inf.mult-right-isotone mult-right-isotone*)
p-antitone top-right-mult-increasing
also have $\dots \leq 1 \sqcap i * S^{'+} * rank^T$
proof –
have $S^{'+} * rank^T \sqcup S^{'+} * rank^T = (S^{T^*} \sqcup S^{'+}) * rank^T$
by (*simp add: conv-star-commute mult-right-dist-sup*)
also have $\dots = (S^{T^*} \sqcup S^{'+}) * rank^T$
by (*smt (z3) comp-associative semiring.distrib-right*)
star.circ-loop-fixpoint sup.left-commute sup-commute sup-idem

also have $\dots = top * rank^T$
by (*simp add: S'-star-connex sup-commute*)
also have $\dots = top$
using 4 *rank-property-def total-conv-surjective* **by** *blast*
finally have $-(S'^+ * rank^T) \leq S'^+ * rank^T$
by (*metis half-shunting inf.idem top-greatest*)
thus *?thesis*
using *comp-associative inf.sup-right-isotone*
mult-right-isotone **by** *auto*
qed
also have $\dots = 1 \sqcap rank * S'^+ * i^T$
by (*metis comp-associative conv-dist-comp conv-involutive*)
one-inf-conv)
also have $\dots \leq rank * S'^+ * i^T$
by *simp*
finally have $?s \leq rank * S'^+ * i^T * ?s$
using 5 *shunt-bijective* **by** *auto*
hence $?rs \leq S'^+ * i^T * ?s$
using 4 *shunt-mapping comp-associative rank-property-def* **by**
auto
hence $?s * (i * ?rr \sqcap -?s \sqcap roots ?p2) \leq ?s * (i * S'^+ * i^T * ?s \sqcap -?s \sqcap roots ?p2)$
using 11 *comp-associative comp-inf.mult-left-isotone*
comp-isotone inf.eq-refl **by** *auto*
also have $\dots = ?s * ((i * S'^+ * i^T * ?s \sqcup i * i^T * ?s) \sqcap -?s \sqcap roots ?p2)$
by (*metis comp-associative mult-left-dist-sup*)
star.circ-loop-fixpoint)
also have $\dots \leq ?s * ((i * S'^+ * i^T * ?s \sqcup 1 * ?s) \sqcap -?s \sqcap roots ?p2)$
using 12 **by** (*metis mult-left-isotone sup-right-isotone*
comp-inf.mult-left-isotone mult-right-isotone)
also have $\dots = ?s * (i * S'^+ * i^T * ?s \sqcap -?s \sqcap roots ?p2)$
using *comp-inf.comp-right-dist-sup* **by** *simp*
also have $\dots \leq ?s * (i * S'^+ * i^T * ?s \sqcap roots ?p2)$
using *comp-inf.mult-left-isotone inf.cobounded1*
mult-right-isotone **by** *blast*
also have $\dots \leq ?s * (i * S'^+ * i^T * ?s \sqcap i * -(S'^+ * rank^T * top))$
using 12 *comp-inf.mult-right-isotone mult-right-isotone* **by** *auto*
also have $\dots = ?s * (i * S'^+ * i^T * ?s \sqcap i) * -(S'^+ * rank^T * top)$
using 5 **by** (*simp add: comp-associative vector-inf-comp*)
also have $\dots = (?s \sqcap (i * S'^+ * i^T * ?s)^T) * i * -(S'^+ * rank^T * top)$
using 5 *covector-inf-comp-3 mult-assoc* **by** *auto*
also have $\dots = (?s \sqcap ?s^T * i * S'^+ * i^T) * i * -(S'^+ * rank^T * top)$
using *conv-dist-comp conv-involutive mult-assoc* **by** *auto*

$* \text{ top}$
also have $\dots = (?s \sqcap ?s^T) * i * S'^{+T} * i^T * i * -(S'^{+} * \text{rank}^T$
using *5 vector-inf-comp* **by** *auto*
also have $\dots \leq i * S'^{+T} * i^T * i * -(S'^{+} * \text{rank}^T * \text{top})$
using *5* **by** (*metis point-antisymmetric mult-left-isotone*)
 mult-left-one
also have $\dots \leq i * S'^{+T} * -(S'^{+} * \text{rank}^T * \text{top})$
using *12* **by** (*smt mult-left-isotone mult-right-isotone*)
 $\text{mult-assoc comp-right-one}$
also have $\dots \leq i * -(S'^{*} * \text{rank}^T * \text{top})$
proof $-$
have $S'^{+} * S'^{*} * \text{rank}^T * \text{top} \leq S'^{+} * \text{rank}^T * \text{top}$
by (*simp add: comp-associative star.circ-transitive-equal*)
hence $S'^{+T} * -(S'^{+} * \text{rank}^T * \text{top}) \leq -(S'^{*} * \text{rank}^T * \text{top})$
by (*smt (verit, ccfv-SIG) comp-associative*
 $\text{conv-complement-sub-leq mult-right-isotone order.trans p-antitone}$)
thus *?thesis*
by (*simp add: comp-associative mult-right-isotone*)
qed
also have $\dots \leq i * -(S'^{+} * ?\text{rank}^T * \text{top})$
proof $-$
have $S'^{+} * ?\text{rank}^T * \text{top} \leq ?rs \sqcup S'^{+} * \text{rank}^T * \text{top}$
using *20* **by** *simp*
also have $\dots \leq \text{rank}^T * \text{top} \sqcup S'^{+} * \text{rank}^T * \text{top}$
using *mult-right-isotone sup-left-isotone top.extremum* **by**
 blast
also have $\dots = S'^{*} * \text{rank}^T * \text{top}$
using *comp-associative star.circ-loop-fixpoint sup-commute*
by *auto*
finally show *?thesis*
using *mult-right-isotone p-antitone* **by** *blast*
qed
finally have $?s * (i * ?rr \sqcap -?s \sqcap \text{roots } ?p2) \leq i * -(S'^{+} *$
 $?\text{rank}^T * \text{top})$
 \cdot
hence $i * ?rr \sqcap -?s \sqcap \text{roots } ?p2 \leq ?s^T * i * -(S'^{+} * ?\text{rank}^T *$
 $\text{top})$
using *5 shunt-mapping bijective-conv-mapping mult-assoc* **by**
 auto
hence $i * ?rr \sqcap -?s \sqcap \text{roots } ?p2 \leq i * ?rr \sqcap ?s^T * i * -(S'^{+} *$
 $?\text{rank}^T * \text{top})$
by (*simp add: inf.sup-monoid.add-assoc*)
also have $\dots = (i * ?rr \sqcap ?s^T * i) * -(S'^{+} * ?\text{rank}^T * \text{top})$
using *6 vector-inf-comp vector-mult-closed* **by** *simp*
also have $\dots \leq ?i * -(S'^{+} * ?\text{rank}^T * \text{top})$
using *13 comp-left-increasing-sup sup-assoc* **by** *auto*
finally show $i * ?rr \sqcap -?s \sqcap \text{roots } ?p2 \leq ?i * -(S'^{+} * ?\text{rank}^T$
 $* \text{top})$
 \cdot

have $-(i * ?rr) \sqcap \text{roots } ?p2 \leq -(i * ?rr) \sqcap i * -(S'^+ * \text{rank}^T$
** top)*

using *12 inf.sup-right-isotone by auto*
also have $\dots \leq -(i * ?rr) \sqcap i * -(?rs \sqcup S'^+ * \text{rank}^T * \text{top})$
proof –

have *21: regular (?rs \sqcup $S'^+ * \text{rank}^T * \text{top}$)*
using *4 6 rank-property-def mapping-regular S'-regular*
pp-dist-star regular-conv-closed regular-mult-closed bijective-regular
regular-closed-sup by auto

have $?rs \sqcup S'^+ * \text{rank}^T * \text{top} \leq S'^+ * \text{rank}^T * \text{top} \sqcup ?rr$
using *11 by simp*
hence $(?rs \sqcup S'^+ * \text{rank}^T * \text{top}) - S'^+ * \text{rank}^T * \text{top} \leq ?rr$
using *half-shunting sup-commute by auto*
hence $-(S'^+ * \text{rank}^T * \text{top}) \leq -(?rs \sqcup S'^+ * \text{rank}^T * \text{top}) \sqcup$
?rr

using *21 by (metis inf.sup-monoid.add-commute*
shunting-var-p sup-commute)
hence $i * -(S'^+ * \text{rank}^T * \text{top}) \leq i * -(?rs \sqcup S'^+ * \text{rank}^T * \text{top}) \sqcup i * ?rr$

by *(metis mult-left-dist-sup mult-right-isotone)*
hence $-(i * ?rr) \sqcap i * -(S'^+ * \text{rank}^T * \text{top}) \leq i * -(?rs \sqcup$
 $S'^+ * \text{rank}^T * \text{top})$

using *half-shunting inf.sup-monoid.add-commute by fastforce*
thus *?thesis*
using *inf.le-sup-iff by blast*
qed

also have $\dots \leq -(i * ?rr) \sqcap i * -(S'^+ * ?\text{rank}^T * \text{top})$
using *20 by (meson comp-inf.mult-right-isotone*
mult-right-isotone p-antitone)

finally have $-(i * ?rr) \sqcap -?s \sqcap \text{roots } ?p2 \leq -(i * ?rr) \sqcap -?s$
 $\sqcap i * -(S'^+ * ?\text{rank}^T * \text{top})$

by *(smt (z3) inf.boundedI inf.cobounded1 inf.coboundedI2*
inf.sup-monoid.add-assoc inf.sup-monoid.add-commute)
also have $\dots \leq ?i * -(S'^+ * ?\text{rank}^T * \text{top})$
using *5 6 13 by (smt (z3) sup-commute*
vector-complement-closed vector-inf-comp vector-mult-closed
comp-left-increasing-sup)

finally show $-(i * ?rr) \sqcap -?s \sqcap \text{roots } ?p2 \leq ?i * -(S'^+ *$
 $?\text{rank}^T * \text{top})$

qed

finally show *?thesis*

qed

thus *card-less-eq (roots ?p4) $-(S'^+ * ?\text{rank}^T * \text{top})$*
using *14 15 16 card-less-eq-def by auto*
have $?s \leq i * -(S'^+ * \text{rank}^T * \text{top})$
using *19 by simp*
also have $\dots \leq i * -(S'^+ * ?rr)$

using *mult-right-isotone p-antitone top.extremum mult-assoc* by
auto
 also have $\dots = i * -S'^+ * ?rr$
 using *6 comp-bijective-complement mult-assoc* by *fastforce*
 finally have $?rr \leq -S'^{T+} * i^T * ?s$
 using *5 6* by (*metis conv-complement conv-dist-comp*
conv-plus-commute bijective-reverse)
 also have $\dots \leq S'^* * i^T * ?s$
 using *7 conv-plus-commute mult-left-isotone* by *auto*
 finally have *22*: $?rr \leq S'^* * i^T * ?s$
 .
 have $?r = \text{root } ?p1 \ x$
 using *2 path-compression-postcondition-def*
path-compression-precondition-def by *blast*
 also have $\dots = \text{root } ?p2 \ x$
 using *3 18* by (*smt (z3) find-set-precondition-def*
path-compression-postcondition-def path-compression-precondition-def same-root)
 also have $\dots \leq \text{roots } ?p2$
 by *simp*
 also have $\dots \leq i * -(S'^+ * \text{rank}^T * \text{top})$
 using *12* by *simp*
 also have $\dots \leq i * -(S'^+ * ?rr)$
 using *mult-right-isotone p-antitone top.extremum mult-assoc* by
auto
 also have $\dots = i * -S'^+ * ?rr$
 using *6 comp-bijective-complement mult-assoc* by *fastforce*
 finally have $?rr \leq -S'^{T+} * i^T * ?r$
 using *5 6* by (*metis conv-complement conv-dist-comp*
conv-plus-commute bijective-reverse)
 also have $\dots \leq S'^* * i^T * ?r$
 using *7 conv-plus-commute mult-left-isotone* by *auto*
 finally have $?rr \leq S'^* * i^T * ?r$
 .
 hence $?rr \leq S'^* * i^T * ?r \sqcap S'^* * i^T * ?s$
 using *22 inf.boundedI* by *blast*
 also have $\dots = (S'^+ * i^T * ?r \sqcup i^T * ?r) \sqcap S'^* * i^T * ?s$
 by (*simp add: star.circ-loop-fixpoint mult-assoc*)
 also have $\dots \leq S'^+ * i^T * ?r \sqcup (i^T * ?r \sqcap S'^* * i^T * ?s)$
 by (*metis comp-inf.mult-right-dist-sup eq-refl inf.cobounded1*
semiring.add-mono)
 also have $\dots \leq S' * \text{top} \sqcup (i^T * ?r \sqcap S'^* * i^T * ?s)$
 using *comp-associative mult-right-isotone sup-left-isotone*
top.extremum by *auto*
 also have $\dots = S' * \text{top} \sqcup (i^T * ?r \sqcap (S'^+ * i^T * ?s \sqcup i^T * ?s))$
 by (*simp add: star.circ-loop-fixpoint mult-assoc*)
 also have $\dots \leq S' * \text{top} \sqcup S'^+ * i^T * ?s \sqcup (i^T * ?r \sqcap i^T * ?s)$
 by (*smt (z3) comp-inf.semiring.distrib-left inf.sup-right-divisibility*
star.circ-loop-fixpoint sup-assoc sup-commute sup-inf-distrib1)
 also have $\dots \leq S' * \text{top} \sqcup (i^T * ?r \sqcap i^T * ?s)$

by (*metis comp-associative mult-right-isotone order.refl*
sup.orderE top.extremum)
 also have $\dots = S' * top \sqcup i^T * (?r \sqcap ?s)$
 using 12 *conv-involutive univalent-comp-left-dist-inf* by auto
 also have $\dots = S' * top$
 using 5 8 *distinct-points* by auto
 finally have $top \leq ?rr^T * S' * top$
 using 6 by (*smt conv-involutive shunt-mapping*
bijjective-conv-mapping mult-assoc)
 hence *surjective* ($S'^T * ?rs$)
 using 6 11 by (*smt conv-dist-comp conv-involutive*
point-conv-comp top-le)
 thus *total* ?rank
 using 4 5 *bijjective-regular update-total rank-property-def* by blast
 show $(?p4 - 1) * ?rank \leq ?rank * S'^+$
 proof –
 have 23: *univalent* ?p2
 using 3 *path-compression-postcondition-def*
path-compression-precondition-def by blast
 have 24: $?r \sqcap (?p4 - 1) * ?rank \leq ?s^T * rank * S' * S'^+$
 proof –
 have $?r \sqcap (?p4 - 1) * ?rank = (?r \sqcap ?p4 \sqcap -1) * ?rank$
 using 5 *vector-complement-closed vector-inf-comp inf-assoc* by
 auto
 also have $\dots = (?r \sqcap -?s \sqcap ?p4 \sqcap -1) * ?rank$
 using 5 8 by (*smt (z3) order.antisym bijjective-regular*
point-in-vector-or-complement point-in-vector-or-complement-2 inf-absorb1)
 also have $\dots = (?r \sqcap -?s \sqcap ?p2 \sqcap -1) * ?rank$
 by (*simp add: inf.left-commute inf.sup-monoid.add-commute*
inf-sup-distrib1 inf-assoc)
 also have $\dots \leq (?r \sqcap ?p2 \sqcap -1) * ?rank$
 using *inf.sup-left-isotone inf-le1 mult-left-isotone* by blast
 also have $\dots = bot$
 proof –
 have $?r = root ?p1 x$
 using 2 *path-compression-postcondition-def*
path-compression-precondition-def by blast
 also have $\dots = root ?p2 x$
 using 3 18 by (*smt (z3) find-set-precondition-def*
path-compression-postcondition-def path-compression-precondition-def same-root)
 also have $\dots \leq roots ?p2$
 by *simp*
 finally have $?r \sqcap ?p2 \leq roots ?p2 \sqcap ?p2$
 using *inf.sup-left-isotone* by blast
 also have $\dots \leq (?p2 \sqcap 1) * (?p2 \sqcap 1)^T * ?p2$
 by (*smt (z3) comp-associative comp-inf.star-plus dedekind-1*
inf-top-right order-lesseq-imp)
 also have $\dots = (?p2 \sqcap 1) * (?p2 \sqcap 1) * ?p2$
 using *coreflexive-symmetric* by force

also have $\dots \leq (?p2 \sqcap 1) * ?p2$
by (*metis coreflexive-comp-top-inf inf.cobounded2*)

mult-left-isotone)

also have $\dots \leq ?p2 \sqcap 1$
by (*smt 23 comp-inf.mult-semi-associative conv-dist-comp conv-dist-inf conv-order equivalence-one-closed inf.absorb1 inf.sup-monoid.add-assoc injective-codomain*)

also have $\dots \leq 1$
by *simp*

finally have $?r \sqcap ?p2 \leq 1$

\cdot

thus *?thesis*
by (*metis pseudo-complement regular-one-closed semiring.mult-not-zero*)

qed

finally show *?thesis*
using *bot-least le-bot by blast*

qed

have $?5: -?r \sqcap (?p4 - 1) * ?rank \leq rank * S'^+$
proof –

have $-?r \sqcap (?p4 - 1) * ?rank = (-?r \sqcap ?p4 \sqcap -1) * ?rank$
using *5 vector-complement-closed vector-inf-comp inf-assoc by*

auto

also have $\dots = (-?r \sqcap (?s \sqcup -?s) \sqcap ?p4 \sqcap -1) * ?rank$
using *5 bijective-regular inf-top-right regular-complement-top*

by *auto*

also have $\dots = (-?r \sqcap ?s \sqcap ?p4 \sqcap -1) * ?rank \sqcup (-?r \sqcap -?s \sqcap ?p4 \sqcap -1) * ?rank$
by (*smt (z3) inf-sup-distrib1 inf-sup-distrib2*)

mult-right-dist-sup)

also have $\dots = (-?r \sqcap ?s \sqcap ?r^T \sqcap -1) * ?rank \sqcup (-?r \sqcap -?s \sqcap ?p2 \sqcap -1) * ?rank$
using *5 by (smt (z3) bijective-regular comp-inf.comp-left-dist-sup inf-assoc inf-commute inf-top-right mult-right-dist-sup regular-complement-top)*

also have $\dots \leq (?s \sqcap ?r^T \sqcap -1) * ?rank \sqcup (-?s \sqcap ?p2 \sqcap -1) * ?rank$

by (*smt (z3) comp-inf.semiring.distrib-left inf.cobounded2 inf.sup-monoid.add-assoc mult-left-isotone mult-right-dist-sup*)

also have $\dots \leq (?s \sqcap ?r^T) * ?rank \sqcup (?p2 - 1) * ?rank$
by (*smt (z3) inf.cobounded1 inf.cobounded2 inf.sup-monoid.add-assoc mult-left-isotone semiring.add-mono*)

also have $\dots = ?s * (?r \sqcap ?rank) \sqcup (?p2 - 1) * ?rank$
using *5 by (simp add: covector-inf-comp-3)*

also have $\dots = ?s * (?r \sqcap (S'^T * ?rs)^T) \sqcup (?p2 - 1) * ?rank$
using *inf-commute update-inf-same mult-assoc by force*

also have $\dots = ?s * (?r \sqcap ?s^T * rank * S') \sqcup (?p2 - 1) * ?rank$
using *comp-associative conv-dist-comp conv-involutive by auto*

also have $\dots \leq ?s * ?s^T * rank * S' \sqcup (?p2 - 1) * ?rank$

using *comp-associative inf.cobounded2 mult-right-isotone semiring.add-right-mono* **by auto**
also have $\dots \leq 1 * \text{rank} * S' \sqcup (?p2 - 1) * ?\text{rank}$
using 5 **by** (*meson mult-left-isotone order.refl semiring.add-mono*)
also have $\dots = \text{rank} * S' \sqcup (?p2 - 1) * (?r \sqcap (S'^T * ?rr)^T) \sqcup$
 $(?p2 - 1) * (-?r \sqcap \text{rank})$
using 11 *comp-associative mult-1-left mult-left-dist-sup sup-assoc* **by auto**
also have $\dots \leq \text{rank} * S' \sqcup (?p2 - 1) * (?r \sqcap ?r^T * \text{rank} * S')$
 $\sqcup (?p2 - 1) * \text{rank}$
using *comp-associative conv-dist-comp conv-involutive inf.cobounded1 inf.sup-monoid.add-commute mult-right-isotone semiring.add-left-mono* **by auto**
also have $\dots = \text{rank} * S' \sqcup (?p2 - 1) * (?r \sqcap ?r^T) * \text{rank} * S'$
 $\sqcup (?p2 - 1) * \text{rank}$
using 5 *comp-associative vector-inf-comp* **by auto**
also have $\dots \leq \text{rank} * S' \sqcup (?p2 - 1) * \text{rank} * S' \sqcup (?p2 - 1)$
 $* \text{rank}$
using 5 **by** (*metis point-antisymmetric mult-left-isotone mult-right-isotone sup-left-isotone sup-right-isotone comp-right-one*)
also have $\dots \leq \text{rank} * S' \sqcup \text{rank} * S'^+ * S' \sqcup (?p2 - 1) * \text{rank}$
using 4 **by** (*metis rank-property-def mult-left-isotone sup-left-isotone sup-right-isotone*)
also have $\dots \leq \text{rank} * S' \sqcup \text{rank} * S'^+ * S' \sqcup \text{rank} * S'^+$
using 4 **by** (*metis rank-property-def sup-right-isotone*)
also have $\dots \leq \text{rank} * S'^+$
using *comp-associative eq-refl le-sup-iff mult-right-isotone star.circ-mult-increasing star.circ-plus-same star.left-plus-below-circ* **by auto**
finally show *?thesis*

qed
have $(?p4 - 1) * ?\text{rank} = (?r \sqcap (?p4 - 1) * ?\text{rank}) \sqcup (-?r \sqcap$
 $(?p4 - 1) * ?\text{rank})$
using 5 **by** (*smt (verit, ccfv-threshold) bijective-regular inf-commute inf-sup-distrib2 inf-top-right regular-complement-top*)
also have $\dots \leq (?r \sqcap ?s^T * \text{rank} * S' * S'^+) \sqcup (-?r \sqcap \text{rank} *$
 $S'^+)$
using 24 25 **by** (*meson inf.boundedI inf.cobounded1 semiring.add-mono*)
also have $\dots = (?r \sqcap ?s^T * \text{rank} * S') * S'^+ \sqcup (-?r \sqcap \text{rank}) *$
 S'^+
using 5 *vector-complement-closed vector-inf-comp* **by auto**
also have $\dots = ?\text{rank} * S'^+$
using *conv-dist-comp mult-right-dist-sup* **by auto**
finally show *?thesis*

qed
qed

```

      qed
    qed
  show ?rr ≠ ?rs → union-sets-postcondition ?p4 x y p0 ∧
rank-property ?p4 rank
  proof
    assume ?rr ≠ ?rs
    hence ?rs ⊓ ?rr = bot
    using 6 by (meson bijective-regular dual-order.eq-iff
point-in-vector-or-complement point-in-vector-or-complement-2
pseudo-complement)
    hence 26: ?rs ≤ S+ * ?rr
    using 10 le-iff-inf pseudo-complement by auto
    show union-sets-postcondition ?p4 x y p0 ∧ rank-property ?p4 rank
    proof
      show union-sets-postcondition ?p4 x y p0
      using 1 2 3 by (simp add: union-sets-1-swap)
      show rank-property ?p4 rank
      proof (unfold rank-property-def, intro conjI)
        show univalent rank total rank
        using 1 rank-property-def by auto
        have ?r ≤ -?s
        using 5 8 by (meson order.antisym bijective-regular
point-in-vector-or-complement point-in-vector-or-complement-2)
        hence ?s ⊓ ?rT ⊓ 1 = bot
        by (metis (full-types) bot-least inf.left-commute
inf.sup-monoid.add-commute one-inf-conv pseudo-complement)
        hence ?p4 ⊓ 1 ≤ ?p2
        by (smt half-shunting inf.cobounded2 pseudo-complement
regular-one-closed semiring.add-mono sup-commute)
        hence roots ?p4 ≤ roots ?p2
        by (simp add: mult-left-isotone)
        thus card-less-eq (roots ?p4) (-(S+ * rankT * top))
        using 4 by (meson rank-property-def card-less-eq-def order-trans)
        have (?p4 - 1) * rank = (?s ⊓ ?rT ⊓ -1) * rank ⊔ (-?s ⊓ ?p2
⊓ -1) * rank
        using comp-inf.semiring.distrib-right mult-right-dist-sup by auto
        also have ... ≤ (?s ⊓ ?rT ⊓ -1) * rank ⊔ (?p2 - 1) * rank
        using comp-inf.mult-semi-associative mult-left-isotone
sup-right-isotone by auto
        also have ... ≤ (?s ⊓ ?rT ⊓ -1) * rank ⊔ rank * S+
        using 4 sup-right-isotone rank-property-def by blast
        also have ... ≤ (?s ⊓ ?rT) * rank ⊔ rank * S+
        using inf-le1 mult-left-isotone sup-left-isotone by blast
        also have ... = ?s * ?rT * rank ⊔ rank * S+
        using 5 by (simp add: vector-covector)
        also have ... = rank * S+
      proof -
        have rankT * ?s ≤ S+ * rankT * ?r
        using 26 comp-associative by auto

```

```

      hence ?s ≤ rank * S'+ * rankT * ?r
      using 4 shunt-mapping comp-associative rank-property-def by
auto
      hence ?s * ?rT ≤ rank * S'+ * rankT
      using 5 shunt-bijective by blast
      hence ?s * ?rT * rank ≤ rank * S'+
      using 4 shunt-bijective rank-property-def mapping-conv-bijective
by auto
      thus ?thesis
      using sup-absorb2 by blast
qed
finally show (?p4 - 1) * rank ≤ rank * S'+
.
qed
qed
qed
qed
qed
qed
qed
show ?r = ?s → union-sets-postcondition ?p5 x y p0 ∧ rank-property ?p5
rank
proof
  assume 27: ?r = ?s
  show union-sets-postcondition ?p5 x y p0 ∧ rank-property ?p5 rank
  proof
    show union-sets-postcondition ?p5 x y p0
    using 1 2 3 27 by (simp add: union-sets-1-skip)
    show rank-property ?p5 rank
    using 4 27 by simp
  qed
qed
qed
qed
end
end

```

8 Matrix Peano Algebras

We define a Boolean matrix representation of natural numbers up to n , where n the size of an enumeration type $'a::enum$. Numbers (obtained by Z -matrix for 0 and N -matrix n for n) are represented as relational vectors. The total successor function (S -matrix, modulo n) and the partial successor function (S' -matrix, for numbers up to $n-1$) are relations that are (partial) functions.

We give an order-embedding of nat into this representation. We show

that this representation satisfies a relational version of the Peano axioms. We also implement a function *CP-matrix* that chooses a number in a non-empty set.

theory *Matrix-Peano-Algebras*

imports *Aggregation-Algebras.M-Choose-Component*
Relational-Disjoint-Set-Forests.More-Disjoint-Set-Forests

begin

no-notation

uminus ($-$ - [81] 80) **and**
minus-class.minus (**infixl** $-$ 65)

definition *Z-matrix* :: ('a::enum,'b::{bot,top}) square (*mZero*) **where** *mZero* =
 $(\lambda(i,j) . \text{if } i = \text{hd } \text{enum-class.enum} \text{ then top else bot})$

definition *S-matrix* :: ('a::enum,'b::{bot,top}) square (*msuccmod*) **where**
msuccmod = $(\lambda(i,j) . \text{let } e = (\text{enum-class.enum} :: 'a \text{ list}) \text{ in if } (\exists k . \text{Suc } k < \text{length } e \wedge i = e ! k \wedge j = e ! \text{Suc } k) \vee (i = e ! \text{minus-class.minus } (\text{length } e) 1 \wedge j = \text{hd } e) \text{ then top else bot})$

definition *S'-matrix* :: ('a::enum,'b::{bot,top}) square (*msucc*) **where** *msucc* =
 $(\lambda(i,j) . \text{let } e = (\text{enum-class.enum} :: 'a \text{ list}) \text{ in if } \exists k . \text{Suc } k < \text{length } e \wedge i = e ! k \wedge j = e ! \text{Suc } k \text{ then top else bot})$

definition *N-matrix* :: nat \Rightarrow ('a::enum,'b::{bot,top}) square (*mnat*) **where**
mnat *n* = $(\lambda(i,j) . \text{if } i = \text{enum-class.enum} ! n \text{ then top else bot})$

definition *CP-matrix* :: ('a::enum,'b::{bot,uminus}) square \Rightarrow ('a,'b) square
(*mcp*) **where** *mcp* *f* = $(\lambda(i,j) . \text{if } \text{Some } i = \text{find } (\lambda x . f(x,x)) \neq \text{bot} \text{ then } \text{enum-class.enum} \text{ then } \text{uminus-class.uminus } (\text{uminus-class.uminus } (f(i,j))) \text{ else bot})$

lemma *S'-matrix-S-matrix*:

$(\text{msucc} :: ('a::\text{enum},'b::\text{stone-relation-algebra}) \text{ square}) = \text{msuccmod} \ominus \text{mZero}^t$

proof (*rule ext*, *rule prod-cases*)

let *?e* = *enum-class.enum* :: 'a list

let *?h* = *hd ?e*

let *?s* = *msuccmod* :: ('a,'b) square

let *?s'* = *msucc* :: ('a,'b) square

let *?z* = *mZero* :: ('a,'b) square

fix *i j*

have *?s' (i,j) = ?s (i,j) - ?z (j,i)*

proof (*cases j = ?h*)

case *True*

have *?s' (i,j) = bot*

proof (*unfold S'-matrix-def*, *clarsimp*)

fix *k*

assume *1: Suc k < length ?e j = ?e ! Suc k*

have (*UNIV* :: 'a set) $\neq \{\}$

by *simp*

hence *?e ! Suc k = ?e ! 0*


```

    using 1 by (simp add: hd-conv-nth UNIV-enum True)
  hence Suc k = 0
    apply (subst nth-eq-iff-index-eq[THEN sym, of ?e])
    using 1 enum-distinct by auto
  thus top = bot
    by simp
qed
thus ?thesis
  by (simp add: Z-matrix-def True)
next
case False
thus ?thesis
  by (simp add: Z-matrix-def S-matrix-def S'-matrix-def)
qed
thus ?s' (i,j) = (?s ⊖ ?zt) (i,j)
  by (simp add: minus-matrix-def conv-matrix-def Z-matrix-def)
qed

```

lemma *N-matrix-power-S:*

$n < \text{length } (\text{enum-class.enum} :: 'a \text{ list}) \longrightarrow \text{mnat } n = \text{matrix-monoid.power } (\text{msuccmod}^t) \ n \odot (\text{mZero} :: ('a::\text{enum}, 'b::\text{stone-relation-algebra}) \text{ square})$

proof (*induct* n)

let $?z = \text{mZero} :: ('a, 'b) \text{ square}$

let $?s = \text{msuccmod} :: ('a, 'b) \text{ square}$

let $?e = \text{enum-class.enum} :: 'a \text{ list}$

let $?h = \text{hd } ?e$

let $?l = \text{length } ?e$

let $?g = ?e ! \text{minus-class.minus } ?l \ 1$

let $?p = \text{matrix-monoid.power } (?s^t)$

case 0

have $(\text{UNIV} :: 'a \text{ set}) \neq \{\}$

by *simp*

hence $?h = ?e ! 0$

by (*simp* add: *hd-conv-nth UNIV-enum*)

thus *?case*

by (*simp* add: *N-matrix-def Z-matrix-def*)

case (*Suc* n)

assume 1: $n < ?l \longrightarrow \text{mnat } n = ?p \ n \odot ?z$

show $\text{Suc } n < ?l \longrightarrow \text{mnat } (\text{Suc } n) = ?p (\text{Suc } n) \odot ?z$

proof

assume 2: $\text{Suc } n < ?l$

hence $n < ?l$

by *simp*

hence $\forall l < ?l . (?e ! l = ?e ! n \longrightarrow l = n)$

using *nth-eq-iff-index-eq enum-distinct* by *auto*

hence 3: $\bigwedge i . (\exists l < ?l . ?e ! n = ?e ! l \wedge i = ?e ! \text{Suc } l) \longrightarrow (i = ?e ! \text{Suc } n)$

by *auto*

have 4: $\bigwedge i . (\exists l . \text{Suc } l < ?l \wedge ?e ! n = ?e ! l \wedge i = ?e ! \text{Suc } l) \longleftrightarrow (i = ?e ! \text{Suc } n)$

```

apply (rule iffI)
using 3 apply (metis Suc-lessD)
using 2 by auto
show mnat (Suc n) = ?p (Suc n)  $\odot$  ?z
proof (rule ext, rule prod-cases)
  fix i j :: 'a
  have (?p (Suc n)  $\odot$  ?z) (i,j) = (?st  $\odot$  mnat n) (i,j)
    using 1 2 by (simp add: matrix-monoid.mult-assoc)
  also have ... = ( $\bigsqcup_k$  ((?s (k,i))T * mnat n (k,j)))
    by (simp add: times-matrix-def conv-matrix-def)
  also have ... = ( $\bigsqcup_k$  ((if ( $\exists l$  . Suc l < length ?e  $\wedge$  k = ?e ! l  $\wedge$  i = ?e ! Suc
l)  $\vee$  (k = ?g  $\wedge$  i = ?h) then top else bot)T * (if k = ?e ! n then top else bot)))
    by (simp add: S-matrix-def N-matrix-def)
  also have ... = ( $\bigsqcup_k$  ((if ( $\exists l$  . Suc l < length ?e  $\wedge$  k = ?e ! l  $\wedge$  i = ?e ! Suc
l)  $\vee$  (k = ?g  $\wedge$  i = ?h) then top else bot) * (if k = ?e ! n then top else bot)))
    by (smt (verit, best) sup-monoid.sum.cong symmetric-bot-closed
symmetric-top-closed)
  also have ... = ( $\bigsqcup_k$  (if ( $\exists l$  . Suc l < length ?e  $\wedge$  k = ?e ! l  $\wedge$  i = ?e ! Suc l
 $\wedge$  k = ?e ! n)  $\vee$  (k = ?g  $\wedge$  i = ?h  $\wedge$  k = ?e ! n) then top else bot))
    by (smt (verit, best) covector-bot-closed idempotent-bot-closed
sup-monoid.sum.cong surjective-top-closed vector-bot-closed)
  also have ... = (if  $\exists l$  . Suc l < length ?e  $\wedge$  ?e ! n = ?e ! l  $\wedge$  i = ?e ! Suc l
then top else bot)
  proof -
    have  $\bigwedge k$  .  $\neg$ (k = ?g  $\wedge$  i = ?h  $\wedge$  k = ?e ! n)
      using 2 distinct-conv-nth[of ?e] enum-distinct by auto
    thus ?thesis
      by (smt (verit, del-insts) comp-inf.ub-sum sup.order-iff
sup-monoid.sum.neutral sup-top-right)
  qed
  also have ... = (if i = ?e ! Suc n then top else bot)
    using 4 by simp
  also have ... = mnat (Suc n) (i,j)
    by (simp add: N-matrix-def)
  finally show mnat (Suc n) (i,j) = (?p (Suc n)  $\odot$  ?z) (i,j)
    by simp
qed
qed
qed

```

lemma N-matrix-power-S':

$n < \text{length } (\text{enum-class.enum} :: 'a \text{ list}) \longrightarrow \text{mnat } n = \text{matrix-monoid.power } (\text{msucc}^t) n \odot (\text{mZero} :: ('a::\text{enum}, 'b::\text{stone-relation-algebra}) \text{ square})$

proof (induct n)

```

let ?z = mZero :: ('a,'b) square
let ?s = msucc :: ('a,'b) square
let ?e = enum-class.enum :: 'a list
let ?h = hd ?e
let ?l = length ?e

```

```

let ?p = matrix-monoid.power (?st)
case 0
have (UNIV :: 'a set) ≠ {}
  by simp
hence ?h = ?e ! 0
  by (simp add: hd-conv-nth UNIV-enum)
thus ?case
  by (simp add: N-matrix-def Z-matrix-def)
case (Suc n)
assume 1: n < ?l → mnat n = ?p n ⊙ ?z
show Suc n < ?l → mnat (Suc n) = ?p (Suc n) ⊙ ?z
proof
  assume 2: Suc n < ?l
  hence n < ?l
    by simp
  hence ∀ l < ?l . (?e ! l = ?e ! n → l = n)
    using nth-eq-iff-index-eq enum-distinct by auto
  hence 3: ∧ i . (∃ l < ?l . ?e ! n = ?e ! l ∧ i = ?e ! Suc l) → (i = ?e ! Suc n)
    by auto
  have 4: ∧ i . (∃ l . Suc l < ?l ∧ ?e ! n = ?e ! l ∧ i = ?e ! Suc l) ↔ (i = ?e !
Suc n)
    apply (rule iffI)
    using 3 apply (metis Suc-lessD)
    using 2 by auto
  show mnat (Suc n) = ?p (Suc n) ⊙ ?z
proof (rule ext, rule prod-cases)
  fix i j :: 'a
  have (?p (Suc n) ⊙ ?z) (i,j) = (?st ⊙ mnat n) (i,j)
    using 1 2 by (simp add: matrix-monoid.mult-assoc)
  also have ... = (⊔k ((?s (k,i))T * mnat n (k,j)))
    by (simp add: times-matrix-def conv-matrix-def)
  also have ... = (⊔k ((if ∃ l . Suc l < length ?e ∧ k = ?e ! l ∧ i = ?e ! Suc l
then top else bot)T * (if k = ?e ! n then top else bot)))
    by (simp add: S'-matrix-def N-matrix-def)
  also have ... = (⊔k ((if ∃ l . Suc l < length ?e ∧ k = ?e ! l ∧ i = ?e ! Suc l
then top else bot) * (if k = ?e ! n then top else bot)))
    by (smt (verit, best) sup-monoid.sum.cong symmetric-bot-closed
symmetric-top-closed)
  also have ... = (⊔k (if ∃ l . Suc l < length ?e ∧ k = ?e ! l ∧ i = ?e ! Suc l ∧
k = ?e ! n then top else bot))
    by (smt (verit, best) covector-bot-closed idempotent-bot-closed
sup-monoid.sum.cong surjective-top-closed vector-bot-closed)
  also have ... = (if ∃ l . Suc l < length ?e ∧ ?e ! n = ?e ! l ∧ i = ?e ! Suc l
then top else bot)
    by (smt (verit, del-insts) comp-inf.ub-sum sup.order-iff
sup-monoid.sum.neutral sup-top-right)
  also have ... = (if i = ?e ! Suc n then top else bot)
    using 4 by simp
  also have ... = mnat (Suc n) (i,j)

```

```

    by (simp add: N-matrix-def)
  finally show  $mnat (Suc n) (i,j) = (?p (Suc n) \odot ?z) (i,j)$ 
    by simp
  qed
qed
qed

```

```

lemma N-matrix-power-S'-hom-zero:
   $mnat 0 = (mZero :: ('a::enum,'b::stone-relation-algebra) square)$ 
proof -
  let ?e = enum-class.enum :: 'a list
  have (UNIV :: 'a set) = set ?e
    using UNIV-enum by simp
  hence  $0 < length ?e$ 
    by auto
  thus ?thesis
    using N-matrix-power-S' by force
qed

```

```

lemma N-matrix-power-S'-hom-succ:
  assumes  $Suc n < length (enum-class.enum :: 'a list)$ 
  shows  $mnat (Suc n) = msucc^t \odot (mnat n :: ('a::enum,'b::stone-relation-algebra) square)$ 
proof -
  let ?e = enum-class.enum :: 'a list
  let ?z = mZero :: ('a,'b) square
  have  $1: n < length ?e$ 
    using assms by simp
  have  $mnat (Suc n) = matrix-monoid.power (msucc^t) (Suc n) \odot ?z$ 
    using assms N-matrix-power-S' by blast
  also have  $\dots = msucc^t \odot matrix-monoid.power (msucc^t) n \odot ?z$ 
    by simp
  also have  $\dots = msucc^t \odot (matrix-monoid.power (msucc^t) n \odot ?z)$ 
    by (simp add: matrix-monoid.mult-assoc)
  also have  $\dots = msucc^t \odot mnat n$ 
    using 1 by (metis N-matrix-power-S')
  finally show ?thesis
    .
qed

```

```

lemma N-matrix-power-S'-hom-inj:
  assumes  $m < length (enum-class.enum :: 'a list)$ 
  and  $n < length (enum-class.enum :: 'a list)$ 
  and  $m \neq n$ 
  shows  $mnat m \neq (mnat n :: ('a::enum,'b::stone-relation-algebra-consistent) square)$ 
proof -
  let ?e = enum-class.enum :: 'a list
  let ?m = ?e ! m

```

```

have 1: mnat m (?m,?m) = top
  by (simp add: N-matrix-def)
have mnat n (?m,?m) = bot
  apply (unfold N-matrix-def)
  using assms enum-distinct nth-eq-iff-index-eq by auto
thus ?thesis
  using 1 by (metis consistent)
qed

```

syntax

```

-sum-sup-monoid :: idt ⇒ nat ⇒ 'a::bounded-semilattice-sup-bot ⇒ 'a (( $\sqcup$ )-<- .
-) [0,51,10] 10)

```

translations

```

 $\sqcup$  x < y . t => XCONST sup-monoid.sum ( $\lambda$ x . t) { x . x < y }

```

context *bounded-semilattice-sup-bot*

begin

lemma *ub-sum-nat*:

fixes *f* :: *nat* ⇒ 'a

assumes *i* < *l*

shows *f* *i* ≤ (\sqcup *k* < *l* . *f* *k*)

by (*metis* (*no-types*, *lifting*) *assms* *finite-Collect-less-nat* *sup-ge1* *sup-monoid.sum.remove* *mem-Collect-eq*)

lemma *lub-sum-nat*:

fixes *f* :: *nat* ⇒ 'a

assumes \forall *k* < *l* . *f* *k* ≤ *x*

shows (\sqcup *k* < *l* . *f* *k*) ≤ *x*

apply (*rule* *finite-subset-induct*[**where** *A*={*k* . *k* < *l*}])

by (*simp-all* *add*: *assms*)

end

lemma *ext-sum-nat*:

fixes *l* :: *nat*

shows (\sqcup *k* < *l* . *f* *k* *x*) = (\sqcup *k* < *l* . *f* *k*) *x*

apply (*rule* *finite-subset-induct*[**where** *A*={*k* . *k* < *l*}])

apply *simp*

apply *simp*

apply (*metis* (*no-types*, *lifting*) *bot-apply* *sup-monoid.sum.empty*)

by (*metis* (*mono-tags*, *lifting*) *sup-apply* *sup-monoid.sum.insert*)

interpretation *matrix-skra-peano-1*: *skra-peano-1* **where** *sup* = *sup-matrix* **and** *inf* = *inf-matrix* **and** *less-eq* = *less-eq-matrix* **and** *less* = *less-matrix* **and** *bot* = *bot-matrix* ::

('a::*enum*, 'b::*linorder-stone-kleene-relation-algebra-tarski-consistent-expansion*)

square **and** *top* = *top-matrix* **and** *uminus* = *uminus-matrix* **and** *one* =

one-matrix **and** *times* = *times-matrix* **and** *conv* = *conv-matrix* **and** *star* =

star-matrix and $Z = Z\text{-matrix}$ and $S = S\text{-matrix}$

proof

let $?z = mZero :: ('a, 'b)$ square
let $?s = msuccmod :: ('a, 'b)$ square
let $?e = enum\text{-class.enum} :: 'a$ list
let $?h = hd ?e$
let $?l = length ?e$
let $?g = ?e ! minus\text{-class.minus} ?l 1$
let $?p = matrix\text{-monoid.power} (?s^t)$
have 1: $?z \odot mtop = ?z$
proof (*rule ext, rule prod-cases*)
fix $i j :: 'a$
have $(?z \odot mtop) (i, j) = (\bigsqcup_k (?z (i, k) * top))$
by (*simp add: times-matrix-def top-matrix-def*)
also have $\dots = (\bigsqcup_{k::'a} (if\ i = ?h\ then\ top\ else\ bot) * top)$
by (*simp add: Z-matrix-def*)
also have $\dots = (if\ i = ?h\ then\ top\ else\ bot) * (top :: 'b)$
using *sum-const* **by** *blast*
also have $\dots = ?z (i, j)$
by (*simp add: Z-matrix-def*)
finally show $(?z \odot mtop) (i, j) = ?z (i, j)$

qed

have 2: $?z \odot ?z^t \preceq mone$

proof (*unfold less-eq-matrix-def, rule allI, rule prod-cases*)

fix $i j :: 'a$
have $(?z \odot ?z^t) (i, j) = (\bigsqcup_k (?z (i, k) * (?z (j, k))^T))$
by (*simp add: times-matrix-def conv-matrix-def*)
also have $\dots = (\bigsqcup_{k::'a} (if\ i = ?h\ then\ top\ else\ bot) * (if\ j = ?h\ then\ top\ else\ bot))$
by (*simp add: Z-matrix-def*)
also have $\dots = (if\ i = ?h\ then\ top\ else\ bot) * (if\ j = ?h\ then\ top\ else\ bot)$
using *sum-const* **by** *blast*
also have $\dots \leq mone (i, j)$
by (*simp add: one-matrix-def*)
finally show $(?z \odot ?z^t) (i, j) \leq mone (i, j)$

qed

have 3: $mtop \odot ?z = mtop$

proof (*rule ext, rule prod-cases*)

fix $i j :: 'a$
have $mtop (i, j) = (top :: 'b) * (if\ ?h = ?h\ then\ top\ else\ bot)$
by (*simp add: top-matrix-def*)
also have $\dots \leq (\bigsqcup_{k::'a} (top * (if\ k = ?h\ then\ top\ else\ bot)))$
by (*rule ub-sum*)
also have $\dots = (\bigsqcup_k (top * ?z (k, j)))$
by (*simp add: Z-matrix-def*)
also have $\dots = (mtop \odot ?z) (i, j)$
by (*simp add: times-matrix-def top-matrix-def*)

finally show $(m_{top} \odot ?z)(i,j) = m_{top}(i,j)$
by (*simp add: inf.le-bot top-matrix-def*)
qed
show *matrix-stone-relation-algebra.point ?z*
using 1 2 3 **by** *simp*
have $\forall i (j::'a) (k::'a) . (\exists l < ?l . \exists m < ?l . k = ?e ! l \wedge i = ?e ! \text{Suc } l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m) \longrightarrow i = j$
using *distinct-conv-nth enum-distinct by auto*
hence 4: $\forall i (j::'a) (k::'a) . (\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m) \longrightarrow i = j$
by (*metis Suc-lessD*)
show $?s^t \odot ?s \preceq mone$
proof (*unfold less-eq-matrix-def, rule allI, rule prod-cases*)
fix $i j :: 'a$
have $(?s^t \odot ?s)(i,j) = (\bigsqcup_k (?s(k,i) * ?s(k,j)))$
by (*simp add: times-matrix-def conv-matrix-def*)
also have $\dots = (\bigsqcup_{k::'a} ((\text{if } (\exists l . \text{Suc } l < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l) \vee (k = ?g \wedge i = ?h) \text{ then top else bot}) * (\text{if } (\exists m . \text{Suc } m < ?l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m) \vee (k = ?g \wedge j = ?h) \text{ then top else bot})))$
by (*simp add: S-matrix-def*)
also have $\dots = (\bigsqcup_{k::'a} (\text{if } (\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m) \vee (k = ?g \wedge i = ?h \wedge j = ?h) \text{ then top else bot}))$
proof –
have 5: $\bigwedge k . \neg((\exists l . \text{Suc } l < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l) \wedge (k = ?g \wedge j = ?h))$
using *distinct-conv-nth[of ?e] enum-distinct by auto*
have $\bigwedge k . \neg((k = ?g \wedge i = ?h) \wedge (\exists m . \text{Suc } m < ?l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m))$
using *distinct-conv-nth[of ?e] enum-distinct by auto*
thus *?thesis*
using 5 **by** (*smt (verit) covector-bot-closed idempotent-bot-closed sup-monoid.sum.cong surjective-top-closed vector-bot-closed*)
qed
also have $\dots \leq (\bigsqcup_{k::'a} (\text{if } i = j \text{ then top else bot}))$
using 4 **by** (*smt (verit, best) comp-inf.ub-sum order-top-class.top-greatest sup-monoid.sum.not-neutral-contains-not-neutral top.extremum-uniqueI*)
also have $\dots \leq (\text{if } i = j \text{ then top else bot})$
by (*simp add: sum-const*)
also have $\dots = mone(i,j)$
by (*simp add: one-matrix-def*)
finally show $(?s^t \odot ?s)(i,j) \leq mone(i,j)$
qed
have 6: $\forall i (j::'a) (k::'a) . (\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l \wedge j = ?e ! m \wedge k = ?e ! \text{Suc } m) \longrightarrow i = j$
using *distinct-conv-nth enum-distinct by auto*
show $?s \odot ?s^t \preceq mone$
proof (*unfold less-eq-matrix-def, rule allI, rule prod-cases*)

```

fix i j :: 'a
have (?s ⊙ ?st) (i,j) = (⊔k (?s (i,k) * ?s (j,k)))
  by (simp add: times-matrix-def conv-matrix-def)
also have ... = (⊔k::'a ((if (∃ l . Suc l < ?l ∧ i = ?e ! l ∧ k = ?e ! Suc l) ∨ (i
= ?g ∧ k = ?h) then top else bot) * (if (∃ m . Suc m < ?l ∧ j = ?e ! m ∧ k = ?e !
Suc m) ∨ (j = ?g ∧ k = ?h) then top else bot)))
  by (simp add: S-matrix-def)
also have ... = (⊔k::'a (if (∃ l m . Suc l < ?l ∧ Suc m < ?l ∧ i = ?e ! l ∧ k =
?e ! Suc l ∧ j = ?e ! m ∧ k = ?e ! Suc m) ∨ (i = ?g ∧ k = ?h ∧ j = ?g) then
top else bot))
proof -
  have 7: ∧ l . Suc l < ?l → 0 < ?l
    by auto
  have 8: ?h = ?e ! 0
proof (rule hd-conv-nth, rule)
  assume ?e = []
  hence (UNIV::'a set) = {}
    by (auto simp add: UNIV-enum)
  thus False
    by simp
qed
have 9: ∧ k . ¬((∃ l . Suc l < ?l ∧ i = ?e ! l ∧ k = ?e ! Suc l) ∧ (j = ?g ∧ k
= ?h))
  using 7 8 distinct-conv-nth[of ?e] enum-distinct by auto
  have ∧ k . ¬((i = ?g ∧ k = ?h) ∧ (∃ m . Suc m < ?l ∧ j = ?e ! m ∧ k = ?e !
Suc m))
  using 7 8 distinct-conv-nth[of ?e] enum-distinct by auto
  thus ?thesis
  using 9 by (smt (verit) covector-bot-closed idempotent-bot-closed
sup-monoid.sum.cong surjective-top-closed vector-bot-closed)
qed
also have ... ≤ (⊔k::'a (if i = j then top else bot))
  using 6 by (smt (verit, best) comp-inf.ub-sum order-top-class.top-greatest
sup-monoid.sum.not-neutral-contains-not-neutral top.extremum-uniqueI)
also have ... ≤ (if i = j then top else bot)
  by simp
also have ... = mone (i,j)
  by (simp add: one-matrix-def)
finally show (?s ⊙ ?st) (i,j) ≤ mone (i,j)
  .
qed
have (mtop :: ('a,'b) square) = (⊔k < ?l . mnat k)
proof (rule ext, rule prod-cases)
  fix i j :: 'a
  have mtop (i,j) = (top :: 'b)
    by (simp add: top-matrix-def)
  also have ... = (⊔k < ?l . (if i = ?e ! k then top else bot))
proof -
  have i ∈ set ?e

```



```

    using UNIV-enum by auto
    from this obtain k where 6: k < ?l ∧ i = ?e ! k
    by (metis in-set-conv-nth)
    hence (λk . if i = ?e ! k then top else bot) k ≤ (⊔ k < ?l . (if i = ?e ! k then
top else bot :: 'b))
    by (metis ub-sum-nat)
    hence top ≤ (⊔ k < ?l . (if i = ?e ! k then top else bot :: 'b))
    using 6 by simp
    thus ?thesis
    using top.extremum-uniqueI by force
qed
also have ... = (⊔ k < ?l . mnat k (i,j))
    by (simp add: N-matrix-def)
also have ... = (⊔ k < ?l . mnat k) (i,j)
    by (simp add: ext-sum-nat)
finally show (mtop (i,j) :: 'b) = (⊔ k < ?l . mnat k) (i,j)
    .
qed
also have ... = (⊔ k < ?l . ?p k ⊙ ?z)
proof -
    have ∧k . k < ?l → mnat k = ?p k ⊙ ?z
    using N-matrix-power-S by auto
    thus ?thesis
    by (metis (no-types, lifting) mem-Collect-eq sup-monoid.sum.cong)
qed
also have ... ≤ ?st ⊙ ?z
proof (unfold less-eq-matrix-def, rule allI, rule prod-cases)
    fix i j :: 'a
    have (⊔ k < ?l . ?p k ⊙ ?z) (i,j) = (⊔ k < ?l . (?p k ⊙ ?z) (i,j))
    by (metis ext-sum-nat)
    also have ... ≤ (?st ⊙ ?z) (i,j)
    apply (rule lub-sum-nat)
    by (metis less-eq-matrix-def matrix-idempotent-semiring.mult-left-isotone
matrix-kleene-algebra.star.power-below-circ)
    finally show (⊔ k < ?l . ?p k ⊙ ?z) (i,j) ≤ (?st ⊙ ?z) (i,j)
    .
qed
finally show ?st ⊙ ?z = mtop
    by (simp add: matrix-order.antisym-conv)
qed

```

interpretation *matrix-skra-peano-2: skra-peano-2* where **sup** = *sup-matrix* and **inf** = *inf-matrix* and **less-eq** = *less-eq-matrix* and **less** = *less-matrix* and **bot** = *bot-matrix* ::
('a::enum,'b::linorder-stone-kleene-relation-algebra-tarski-consistent-expansion)
square and **top** = *top-matrix* and **uminus** = *uminus-matrix* and **one** =
one-matrix and **times** = *times-matrix* and **conv** = *conv-matrix* and **star** =
star-matrix and **Z** = *Z-matrix* and **S** = *S-matrix*
proof

```

let ?s = msuccmod :: ('a,'b) square
let ?e = enum-class.enum :: 'a list
let ?h = hd ?e
let ?l = length ?e
let ?g = ?e ! minus-class.minus ?l 1
show matrix-bounded-idempotent-semiring.total ?s
proof (rule ext, rule prod-cases)
  fix i j :: 'a
  have (?s  $\odot$  mtop) (i,j) = ( $\bigsqcup_k$  (?s (i,k) * top))
    by (simp add: times-matrix-def top-matrix-def)
  also have ... = ( $\bigsqcup_{k::'a}$  if ( $\exists l . \text{Suc } l < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l$ )  $\vee$  (i
= ?g  $\wedge$  k = ?h) then top else bot)
    by (simp add: S-matrix-def)
  also have ... = top
  proof -
    have  $\bigwedge i . (\exists l . \text{Suc } l < ?l \wedge i = ?e ! l) \vee i = ?g$ 
      by (metis in-set-conv-nth in-enum Suc-lessI diff-Suc-1)
    hence  $\bigwedge i . \exists k . (\exists l . \text{Suc } l < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l) \vee (i = ?g \wedge k$ 
= ?h)
      by blast
    thus ?thesis
      by (smt (verit, ccfv-threshold) comp-inf.ub-sum top.extremum-uniqueI)
  qed
  finally show (?s  $\odot$  mtop) (i,j) = mtop (i,j)
    by (simp add: top-matrix-def)
  qed
qed

```

interpretation *matrix-skra-peano-3*: *skra-peano-3* where *sup* = *sup-matrix* and *inf* = *inf-matrix* and *less-eq* = *less-eq-matrix* and *less* = *less-matrix* and *bot* = *bot-matrix* ::

```

('a::enum,'b::linorder-stone-kleene-relation-algebra-tarski-consistent-expansion)
square and top = top-matrix and uminus = uminus-matrix and one =
one-matrix and times = times-matrix and conv = conv-matrix and star =
star-matrix and Z = Z-matrix and S = S-matrix

```

proof (*unfold-locales*, *rule finite-surj*)

```

show finite (UNIV :: 'a rel set)

```

```

  by simp

```

```

let ?f =  $\lambda R p . \text{if } p \in R \text{ then top else bot}$ 

```

```

show { f :: ('a,'b) square . matrix-p-algebra.regular f }  $\subseteq$  range ?f

```

proof

```

  fix f :: ('a,'b) square

```

```

  let ?R = { (x,y) . f (x,y) = top }

```

```

  assume f  $\in$  { f . matrix-p-algebra.regular f }

```

```

  hence matrix-p-algebra.regular f

```

```

    by simp

```

```

  hence  $\bigwedge p . f p \neq \text{top} \longrightarrow f p = \text{bot}$ 

```

```

    by (metis linorder-stone-algebra-expansion-class.uminus-def

```

```

uminus-matrix-def)

```

hence $f = ?f ?R$
by *fastforce*
thus $f \in \text{range } ?f$
by *blast*
qed
qed

interpretation *matrix-skra-peano-4: skra-peano-4* **where** $\text{sup} = \text{sup-matrix}$ **and**
 $\text{inf} = \text{inf-matrix}$ **and** $\text{less-eq} = \text{less-eq-matrix}$ **and** $\text{less} = \text{less-matrix}$ **and** $\text{bot} =$
 $\text{bot-matrix} ::$
 $(?a::\text{enum}, ?b::\text{linorder-stone-kleene-relation-algebra-tarski-consistent-plus-expansion})$
square **and** $\text{top} = \text{top-matrix}$ **and** $\text{uminus} = \text{uminus-matrix}$ **and** $\text{one} =$
 one-matrix **and** $\text{times} = \text{times-matrix}$ **and** $\text{conv} = \text{conv-matrix}$
and $\text{star} = \text{star-matrix}$ **and** $Z = Z\text{-matrix}$ **and** $S = S\text{-matrix}$ **and** $\text{choose-point} =$
 $\text{agg-square-m-kleene-algebra-2.m-choose-component-algebra-tarski.choose-component-point}$
apply *unfold-locales*
apply *(simp add:*
 $\text{agg-square-m-kleene-algebra-2.m-choose-component-algebra-tarski.choose-component-point-point})$
by *(simp add:*
 $\text{agg-square-m-kleene-algebra-2.m-choose-component-algebra-tarski.choose-component-point-decreasing})$

interpretation *matrix'-skra-peano-1: skra-peano-1* **where** $\text{sup} = \text{sup-matrix}$
and $\text{inf} = \text{inf-matrix}$ **and** $\text{less-eq} = \text{less-eq-matrix}$ **and** $\text{less} = \text{less-matrix}$ **and**
 $\text{bot} = \text{bot-matrix} ::$
 $(?a::\text{enum}, ?b::\text{linorder-stone-kleene-relation-algebra-tarski-consistent-expansion})$
square **and** $\text{top} = \text{top-matrix}$ **and** $\text{uminus} = \text{uminus-matrix}$ **and** $\text{one} =$
 one-matrix **and** $\text{times} = \text{times-matrix}$ **and** $\text{conv} = \text{conv-matrix}$ **and** $\text{star} =$
 star-matrix **and** $Z = Z\text{-matrix}$ **and** $S = S'\text{-matrix}$
proof
let $?z = \text{mZero} :: (?a, ?b)$ *square*
let $?s = \text{msucc} :: (?a, ?b)$ *square*
let $?e = \text{enum-class.enum} :: ?a$ *list*
let $?l = \text{length } ?e$
let $?p = \text{matrix-monoid.power } (?s^t)$
show *matrix-stone-relation-algebra.point ?z*
using *matrix-skra-peano-1.Z-point* **by** *auto*
have $\forall i (j::?a) (k::?a) . (\exists l < ?l . \exists m < ?l . k = ?e ! l \wedge i = ?e ! \text{Suc } l \wedge k = ?e$
 $! m \wedge j = ?e ! \text{Suc } m) \longrightarrow i = j$
using *distinct-conv-nth enum-distinct* **by** *auto*
hence $\not\vdash \forall i (j::?a) (k::?a) . (\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge k = ?e ! l \wedge i =$
 $?e ! \text{Suc } l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m) \longrightarrow i = j$
by *(metis Suc-lessD)*
show $?s^t \odot ?s \preceq \text{mone}$
proof *(unfold less-eq-matrix-def, rule allI, rule prod-cases)*
fix $i j :: ?a$
have $(?s^t \odot ?s) (i, j) = (\bigsqcup_k (?s (k, i) * ?s (k, j)))$
by *(simp add: times-matrix-def conv-matrix-def)*
also have $\dots = (\bigsqcup_k :: ?a ((\text{if } \exists l . \text{Suc } l < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l \text{ then}$
 $\text{top else bot}) * (\text{if } \exists m . \text{Suc } m < ?l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m \text{ then top else$

```

bot)))
  by (simp add: S'-matrix-def)
  also have ... = ( $\bigsqcup_{k::'a}$  (if  $(\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge k = ?e ! l \wedge i = ?e ! \text{Suc } l \wedge k = ?e ! m \wedge j = ?e ! \text{Suc } m)$  then top else bot))
  by (smt (verit) covector-bot-closed idempotent-bot-closed
sup-monoid.sum.cong surjective-top-closed vector-bot-closed)
  also have ...  $\leq$  ( $\bigsqcup_{k::'a}$  (if  $i = j$  then top else bot))
  using 4 by (smt (verit, best) comp-inf.ub-sum order-top-class.top-greatest
sup-monoid.sum.not-neutral-contains-not-neutral top.extremum-uniqueI)
  also have ...  $\leq$  (if  $i = j$  then top else bot)
  by (simp add: sum-const)
  also have ... = mone (i,j)
  by (simp add: one-matrix-def)
  finally show  $(?s^t \odot ?s)$  (i,j)  $\leq$  mone (i,j)
  .
qed
have 5:  $\forall i (j::'a) (k::'a) . (\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l \wedge j = ?e ! m \wedge k = ?e ! \text{Suc } m) \longrightarrow i = j$ 
  using distinct-conv-nth enum-distinct by auto
  show  $?s \odot ?s^t \preceq$  mone
  proof (unfold less-eq-matrix-def, rule allI, rule prod-cases)
    fix i j :: 'a
    have  $(?s \odot ?s^t)$  (i,j) = ( $\bigsqcup_k$  ( $?s$  (i,k) *  $?s$  (j,k)))
    by (simp add: times-matrix-def conv-matrix-def)
    also have ... = ( $\bigsqcup_{k::'a}$  ((if  $\exists l . \text{Suc } l < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l$  then top else bot) * (if  $\exists m . \text{Suc } m < ?l \wedge j = ?e ! m \wedge k = ?e ! \text{Suc } m$  then top else bot)))
    by (simp add: S'-matrix-def)
    also have ... = ( $\bigsqcup_{k::'a}$  (if  $(\exists l m . \text{Suc } l < ?l \wedge \text{Suc } m < ?l \wedge i = ?e ! l \wedge k = ?e ! \text{Suc } l \wedge j = ?e ! m \wedge k = ?e ! \text{Suc } m)$  then top else bot))
    by (smt (verit) covector-bot-closed idempotent-bot-closed
sup-monoid.sum.cong surjective-top-closed vector-bot-closed)
    also have ...  $\leq$  ( $\bigsqcup_{k::'a}$  (if  $i = j$  then top else bot))
    using 5 by (smt (verit, best) comp-inf.ub-sum order-top-class.top-greatest
sup-monoid.sum.not-neutral-contains-not-neutral top.extremum-uniqueI)
    also have ...  $\leq$  (if  $i = j$  then top else bot)
    by simp
    also have ... = mone (i,j)
    by (simp add: one-matrix-def)
    finally show  $(?s \odot ?s^t)$  (i,j)  $\leq$  mone (i,j)
    .
  qed
have (mtop :: ('a,'b) square) = ( $\bigsqcup_{k < ?l} . \text{mnat } k$ )
  proof (rule ext, rule prod-cases)
    fix i j :: 'a
    have mtop (i,j) = (top :: 'b)
    by (simp add: top-matrix-def)
    also have ... = ( $\bigsqcup_{k < ?l} .$  (if  $i = ?e ! k$  then top else bot))
    proof -

```

```

have  $i \in \text{set } ?e$ 
  using UNIV-enum by auto
from this obtain  $k$  where  $6: k < ?l \wedge i = ?e ! k$ 
  by (metis in-set-conv-nth)
hence  $(\lambda k . \text{if } i = ?e ! k \text{ then top else bot}) k \leq (\bigsqcup_{k < ?l} . (\text{if } i = ?e ! k \text{ then top else bot} :: 'b))$ 
top else bot :: 'b))
  by (metis ub-sum-nat)
hence  $\text{top} \leq (\bigsqcup_{k < ?l} . (\text{if } i = ?e ! k \text{ then top else bot} :: 'b))$ 
  using  $6$  by simp
thus ?thesis
  using top.extremum-uniqueI by force
qed
also have  $\dots = (\bigsqcup_{k < ?l} . \text{mnat } k (i,j))$ 
  by (simp add: N-matrix-def)
also have  $\dots = (\bigsqcup_{k < ?l} . \text{mnat } k) (i,j)$ 
  by (simp add: ext-sum-nat)
finally show  $(\text{mtop } (i,j) :: 'b) = (\bigsqcup_{k < ?l} . \text{mnat } k) (i,j)$ 
  .
qed
also have  $\dots = (\bigsqcup_{k < ?l} . ?p k \odot ?z)$ 
proof –
  have  $\bigwedge k . k < ?l \longrightarrow \text{mnat } k = ?p k \odot ?z$ 
    using N-matrix-power-S' by auto
  thus ?thesis
    by (metis (no-types, lifting) mem-Collect-eq sup-monoid.sum.cong)
qed
also have  $\dots \preceq ?s^{t\odot} \odot ?z$ 
proof (unfold less-eq-matrix-def, rule allI, rule prod-cases)
  fix  $i j :: 'a$ 
  have  $(\bigsqcup_{k < ?l} . ?p k \odot ?z) (i,j) = (\bigsqcup_{k < ?l} . (?p k \odot ?z) (i,j))$ 
    by (metis ext-sum-nat)
  also have  $\dots \leq (?s^{t\odot} \odot ?z) (i,j)$ 
    apply (rule lub-sum-nat)
    by (metis less-eq-matrix-def matrix-idempotent-semiring.mult-left-isotone matrix-kleene-algebra.star.power-below-circ)
  finally show  $(\bigsqcup_{k < ?l} . ?p k \odot ?z) (i,j) \leq (?s^{t\odot} \odot ?z) (i,j)$ 
  .
qed
finally show  $?s^{t\odot} \odot ?z = \text{mtop}$ 
  by (simp add: matrix-order.antisym-conv)
qed

lemma nat-less-lesseq-pred:
   $(m :: \text{nat}) < n \implies m \leq \text{minus-class.minus } n \ 1$ 
  by simp

lemma S'-matrix-acyclic:
  matrix-stone-kleene-relation-algebra.acyclic (msucc ::
  ('a::enum, 'b::linorder-stone-kleene-relation-algebra-tarski-consistent-expansion)

```

square)
proof (*rule ccontr*)
let $?e = \text{enum-class.enum} :: 'a \text{ list}$
let $?l = \text{length } ?e$
let $?l1 = \text{minus-class.minus } ?l \ 1$
let $?s = \text{msucc} :: ('a, 'b) \text{ square}$
have ($\text{UNIV} :: 'a \text{ set}$) $\neq \{\}$
by *simp*
hence 1: $?e \neq []$
by (*simp add: UNIV-enum*)
hence 2: $?l \neq 0$
by *simp*
assume $\neg \text{matrix-stone-kleene-relation-algebra.acyclic } ?s$
hence $?s \odot ?s^\odot \otimes \text{mone} \neq \text{mbot}$
by (*simp add: matrix-p-algebra.pseudo-complement*)
from this obtain $i1 \ i2$ **where** $(?s \odot ?s^\odot \otimes \text{mone}) (i1, i2) \neq \text{bot}$
by (*metis bot-matrix-def ext surj-pair*)
hence 3: $(?s \odot ?s^\odot) (i1, i2) \sqcap \text{mone} (i1, i2) \neq \text{bot}$
by (*simp add: inf-matrix-def*)
hence $\text{mone} (i1, i2) \neq (\text{bot} :: 'b)$
by *force*
hence $i1 = i2$
by (*metis (mono-tags, lifting) prod.simps(2) one-matrix-def*)
hence $(?s \odot ?s^\odot) (i1, i1) \neq \text{bot}$
using 3 **by** *force*
hence $(\bigsqcup_k ?s (i1, k) * (?s^\odot) (k, i1)) \neq \text{bot}$
by (*smt (verit, best) times-matrix-def case-prod-conv sup-monoid.sum.cong*)
from this obtain $i3$ **where** 4: $?s (i1, i3) * (?s^\odot) (i3, i1) \neq \text{bot}$
by *force*
hence $?s (i1, i3) \neq \text{bot}$
by *force*
hence (*if* $\exists j1 . \text{Suc } j1 < ?l \wedge i1 = ?e ! j1 \wedge i3 = ?e ! \text{Suc } j1$ *then top else bot*) $::$
 $'b) \neq \text{bot}$
by (*simp add: S'-matrix-def*)
from this obtain $j1$ **where** 5: $\text{Suc } j1 < ?l \wedge i1 = ?e ! j1 \wedge i3 = ?e ! \text{Suc } j1$
by *meson*
have $j1 \neq ?l1$
using 5 *enum-distinct* **by** *auto*
hence $i1 \neq \text{last } ?e$
apply (*subst last-conv-nth*)
using 1 **apply** *simp*
apply (*subst 5*)
apply (*subst nth-eq-iff-index-eq[of ?e]*)
using 1 5 *enum-distinct* **by** *auto*
hence 6: $\text{mone} (\text{last } ?e, i1) = (\text{bot} :: 'b)$
by (*simp add: one-matrix-def*)
have 7: $(?s^\odot) (i3, i1) \neq \text{bot}$
using 4 **by** *force*
have $\bigwedge j2 . \text{Suc } j1 + j2 < ?l \longrightarrow (?s^\odot) (?e ! (\text{Suc } j1 + j2), i1) \neq \text{bot}$

```

proof –
  fix  $j2$ 
  show  $Suc\ j1 + j2 < ?l \longrightarrow (?s^\odot) (?e ! (Suc\ j1 + j2), i1) \neq bot$ 
  proof (induct  $j2$ )
    case 0
    show  $?case$ 
    using 5 7 by simp
  next
  case ( $Suc\ j3$ )
  show  $?case$ 
  proof
    assume 8:  $Suc\ j1 + Suc\ j3 < ?l$ 
    hence  $(?s^\odot) (?e ! (Suc\ j1 + j3), i1) \neq bot$ 
    using Suc by simp
    hence  $(mone \oplus ?s \odot ?s^\odot) (?e ! (Suc\ j1 + j3), i1) \neq bot$ 
    by (metis matrix-kleene-algebra.star-left-unfold-equal)
    hence 9:  $mone (?e ! (Suc\ j1 + j3), i1) \sqcup (?s \odot ?s^\odot) (?e ! (Suc\ j1 +$ 
 $j3), i1) \neq bot$ 
    by (simp add: sup-matrix-def)
    have  $?e ! (Suc\ j1 + j3) \neq i1$ 
    using 5 8 distinct-conv-nth[of  $?e$ ] enum-distinct by auto
    hence  $mone (?e ! (Suc\ j1 + j3), i1) = (bot :: 'b)$ 
    by (simp add: one-matrix-def)
    hence  $(?s \odot ?s^\odot) (?e ! (Suc\ j1 + j3), i1) \neq bot$ 
    using 9 by simp
    hence  $(\bigsqcup_k ?s (?e ! (Suc\ j1 + j3), k) * (?s^\odot) (k, i1)) \neq bot$ 
    by (smt (verit, best) times-matrix-def case-prod-conv
sup-monoid.sum.cong)
    from this obtain  $i4$  where 10:  $?s (?e ! (Suc\ j1 + j3), i4) * (?s^\odot) (i4, i1)$ 
 $\neq bot$ 
    by force
    hence  $?s (?e ! (Suc\ j1 + j3), i4) \neq bot$ 
    by force
    hence (if  $\exists j4 . Suc\ j4 < ?l \wedge ?e ! (Suc\ j1 + j3) = ?e ! j4 \wedge i4 = ?e ! Suc$ 
 $j4$  then top else  $bot :: 'b$ )  $\neq bot$ 
    by (simp add: S'-matrix-def)
    from this obtain  $j4$  where 11:  $Suc\ j4 < ?l \wedge ?e ! (Suc\ j1 + j3) = ?e ! j4$ 
 $\wedge i4 = ?e ! Suc\ j4$ 
    by meson
    hence  $Suc\ j1 + j3 = j4$ 
    apply (subst nth-eq-iff-index-eq[of  $?e$ , THEN sym])
    using 8 enum-distinct by auto
    hence  $i4 = ?e ! (Suc\ j1 + Suc\ j3)$ 
    using 11 by simp
    thus  $(?s^\odot) (?e ! (Suc\ j1 + Suc\ j3), i1) \neq bot$ 
    using 10 by force
  qed
qed
qed

```

hence $\bigwedge j5 . \text{Suc } j1 \leq j5 \wedge j5 < ?l \longrightarrow (?s^\odot) (?e ! j5, i1) \neq \text{bot}$
using *le-Suc-ex* **by** *blast*
hence $(?s^\odot) (\text{last } ?e, i1) \neq \text{bot}$
apply (*subst last-conv-nth*)
using 1 2 5 *nat-less-lesseq-pred* **by** *auto*
hence $(\text{mone} \oplus ?s \odot ?s^\odot) (\text{last } ?e, i1) \neq \text{bot}$
by (*metis matrix-kleene-algebra.star-left-unfold-equal*)
hence $\text{mone} (\text{last } ?e, i1) \sqcup (?s \odot ?s^\odot) (\text{last } ?e, i1) \neq \text{bot}$
by (*simp add: sup-matrix-def*)
hence $(?s \odot ?s^\odot) (\text{last } ?e, i1) \neq \text{bot}$
using 6 **by** *simp*
hence $(\bigsqcup_k ?s (\text{last } ?e, k) * (?s^\odot) (k, i1)) \neq \text{bot}$
by (*smt (verit, best) times-matrix-def case-prod-conv sup-monoid.sum.cong*)
from this obtain *i5* **where** $?s (\text{last } ?e, i5) * (?s^\odot) (i5, i1) \neq \text{bot}$
by *force*
hence $?s (\text{last } ?e, i5) \neq \text{bot}$
by *force*
hence (*if* $\exists j6 . \text{Suc } j6 < ?l \wedge \text{last } ?e = ?e ! j6 \wedge i5 = ?e ! \text{Suc } j6$ *then top else*
bot $:: 'b) \neq \text{bot}$
by (*simp add: S'-matrix-def*)
from this obtain *j6* **where** 12: $\text{Suc } j6 < ?l \wedge \text{last } ?e = ?e ! j6 \wedge i5 = ?e ! \text{Suc } j6$
by *force*
hence $?e ! ?l1 = ?e ! j6$
using 1 5 **by** (*metis last-conv-nth*)
hence $?l1 = j6$
apply (*subst nth-eq-iff-index-eq[of ?e, THEN sym]*)
using 2 12 *enum-distinct* **by** *auto*
thus *False*
using 12 **by** *auto*
qed

lemma *N-matrix-point*:

assumes $n < \text{length} (\text{enum-class.enum} :: 'a \text{ list})$
shows *matrix-stone-relation-algebra.point* (*mnat* $n :: ('a :: \text{enum}, 'b :: \text{linorder-stone-kleene-relation-algebra-tarski-consistent-expansion})$
square)
proof –
let $?e = \text{enum-class.enum} :: 'a \text{ list}$
let $?n = \text{mnat } n :: ('a, 'b) \text{ square}$
let $?s = \text{msucc} :: ('a, 'b) \text{ square}$
let $?z = \text{mZero} :: ('a, 'b) \text{ square}$
have 1: $?n = \text{matrix-monoid.power} (?s^t) n \odot ?z$
using *assms N-matrix-power-S'* **by** *blast*
have $?s = \text{matrix-skra-peano-1.S}'$
by (*simp add: S'-matrix-S-matrix inf-matrix-def minus-matrix-def*
uminus-matrix-def)
hence 2: *matrix-p-algebra.regular* $?s$
by (*metis matrix-skra-peano-2.S'-regular*)


```

have ?n ≠ mbot
proof
  assume ?n = mbot
  hence ?n (?e ! n, ?e ! n) = mbot (?e ! n, ?e ! n)
  by simp
  hence top = (bot :: 'b)
  by (simp add: N-matrix-def bot-matrix-def)
  thus False
  by (metis bot-not-top)
qed
thus matrix-stone-relation-algebra.point ?n
  using 1 2 by (metis (no-types, lifting) matrix'-skra-peano-1.S-univalent
matrix'-skra-peano-1.Z-point matrix-stone-relation-algebra.injective-power-closed
ma-
trix-stone-relation-algebra-tarski-consistent.regular-injective-vector-point-xor-bot
matrix-stone-relation-algebra.regular-power-closed
matrix-stone-relation-algebra.bijjective-regular
matrix-stone-relation-algebra.comp-associative
matrix-stone-relation-algebra.injective-mult-closed
matrix-stone-relation-algebra.regular-conv-closed
matrix-stone-relation-algebra.regular-mult-closed
matrix-stone-relation-algebra.univalent-conv-injective)
qed

```

lemma *N-matrix-power-S'-hom-lesseq*:

```

assumes m < length (enum-class.enum :: 'a list)
  and n < length (enum-class.enum :: 'a list)
  shows m < n ⟷ mnat m ≲ msucc ∘ msucc⊙ ∘ (mnat n ::
('a::enum, 'b::linorder-stone-kleene-relation-algebra-tarski-consistent-expansion)
square)

```

proof –

```

let ?m = mnat m :: ('a, 'b) square
let ?n = mnat n :: ('a, 'b) square
let ?s = msucc :: ('a, 'b) square
let ?z = mZero :: ('a, 'b) square
have 1: ?m = matrix-monoid.power (?st) m ∘ ?z
  using assms(1) N-matrix-power-S' by blast
have 2: ?n = matrix-monoid.power (?st) n ∘ ?z
  using assms(2) N-matrix-power-S' by blast
have 3: matrix-stone-relation-algebra.point ?m
  by (simp add: assms(1) N-matrix-point)
have 4: matrix-stone-relation-algebra.point ?n
  by (simp add: assms(2) N-matrix-point)
show m < n ⟷ ?m ≲ ?s ∘ ?s⊙ ∘ ?n
proof
  assume m < n
  from this obtain k where n = Suc k + m
  using less-iff-Suc-add by auto
  hence ?n = matrix-monoid.power (?st) (Suc k) ∘ matrix-monoid.power (?st)

```

$m \odot ?z$
using 2 **by** (*metis matrix-monoid.power-add*)
also have ... = *matrix-monoid.power* ($?s^t$) (*Suc k*) \odot $?m$
using 1 **by** (*simp add: matrix-monoid.mult-assoc*)
also have ... = (*matrix-monoid.power* $?s$ (*Suc k*))^t \odot $?m$
by (*metis matrix-stone-relation-algebra.power-conv-commute*)
finally have $?m \preceq$ *matrix-monoid.power* $?s$ (*Suc k*) \odot $?n$
using 3 4 **by** (*simp add: matrix-stone-relation-algebra.bijective-reverse*)
also have ... = $?s \odot$ *matrix-monoid.power* $?s$ $k \odot$ $?n$
by *simp*
also have ... \preceq $?s \odot ?s^\odot \odot ?n$
using *matrix-idempotent-semiring.mult-left-isotone*
matrix-idempotent-semiring.mult-right-isotone
matrix-kleene-algebra.star.power-below-circ **by** *blast*
finally show $?m \preceq ?s \odot ?s^\odot \odot ?n$
.

next
assume 5: $?m \preceq ?s \odot ?s^\odot \odot ?n$
show $m < n$
proof (*rule ccontr*)
assume $\neg m < n$
from *this* **obtain** k **where** $m = k + n$
by (*metis add.commute add-diff-inverse-nat*)
hence $?m =$ *matrix-monoid.power* ($?s^t$) $k \odot$ *matrix-monoid.power* ($?s^t$) n
 \odot $?z$
using 1 **by** (*metis matrix-monoid.power-add*)
also have ... = *matrix-monoid.power* ($?s^t$) $k \odot$ $?n$
using 2 **by** (*simp add: matrix-monoid.mult-assoc*)
also have ... = (*matrix-monoid.power* $?s$ k)^t \odot $?n$
by (*metis matrix-stone-relation-algebra.power-conv-commute*)
finally have $?n \preceq$ *matrix-monoid.power* $?s$ $k \odot$ $?m$
using 3 4 **by** (*simp add: matrix-stone-relation-algebra.bijective-reverse*)
also have ... \preceq $?s^\odot \odot ?m$
using *matrix-kleene-algebra.star.power-below-circ*
matrix-stone-relation-algebra.comp-left-isotone **by** *blast*
finally have $?n \preceq ?s^\odot \odot ?m$
.

hence $?m \preceq ?s \odot ?s^\odot \odot ?s^\odot \odot ?m$
using 5 **by** (*metis (no-types, opaque-lifting) matrix-monoid.mult-assoc*
matrix-order.dual-order.trans matrix-stone-relation-algebra.comp-right-isotone)
hence $?m \preceq ?s \odot ?s^\odot \odot ?m$
by (*metis matrix-kleene-algebra.star.circ-transitive-equal*
matrix-monoid.mult-assoc)
thus *False*
using 3 *S'-matrix-acyclic*
matrix-stone-kleene-relation-algebra-consistent.acyclic-reachable-different **by** *blast*
qed
qed
qed

end

References

- [1] R.-J. Back and J. von Wright. *Refinement Calculus*. Springer, New York, 1998.
- [2] R. C. Backhouse and B. A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15(2):161–186, 1975.
- [3] R. Berghammer. Combining relational calculus and the Dijkstra–Gries method for deriving relational programs. *Information Sciences*, 119(3–4):155–171, 1999.
- [4] R. Berghammer and G. Struth. On automated program construction and verification. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction (MPC 2010)*, volume 6120 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2010.
- [5] R. Berghammer, B. von Karger, and A. Wolf. Relation-algebraic derivation of spanning tree algorithms. In J. Jeuring, editor, *Mathematics of Program Construction (MPC 1998)*, volume 1422 of *Lecture Notes in Computer Science*, pages 23–43. Springer, 1998.
- [6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [7] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Trans. Prog. Lang. Syst.*, 29(3:17):1–65, 2007.
- [8] B. A. Galler and M. J. Fisher. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, 1964.
- [9] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings*. Springer, 2008.
- [10] W. Guttman. Verifying minimum spanning tree algorithms with Stone relation algebras. *Journal of Logical and Algebraic Methods in Programming*, 101:132–150, 2018.
- [11] W. Guttman. Verifying the correctness of disjoint-set forests with Kleene relation algebras. In U. Fahrenberg, P. Jipsen, and M. Winter,

- editors, *Relational and Algebraic Methods in Computer Science (RAM-iCS 2020)*, volume 12062 of *Lecture Notes in Computer Science*, pages 134–151. Springer, 2020.
- [12] W. Guttman. Relation-algebraic verification of disjoint-set forests. *arXiv*, 2301.10311, 2023. <https://arxiv.org/abs/2301.10311>.
 - [13] P. Höfner and B. Möller. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing*, 24(4):459–476, 2012.
 - [14] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
 - [15] P. Lammich and R. Meis. A separation logic framework for Imperative HOL. *Archive of Formal Proofs*, 2012.
 - [16] B. Möller. Derivation of graph and pointer algorithms. In B. Möller, H. A. Partsch, and S. A. Schuman, editors, *Formal Program Development*, volume 755 of *Lecture Notes in Computer Science*, pages 123–160. Springer, 1993.
 - [17] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.
 - [18] A. Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.
 - [19] B. Zhan. Verifying imperative programs using Auto2. *Archive of Formal Proofs*, 2018.