Disjoint-Set Forests

Walter Guttmann

October 5, 2019

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 Isabelle/HOL Hoare-logic library. We verify the correctness of an array-based implementation of disjoint-set forests with a naive union operation and a find operation with path compression.

Contents

1 Relation-Algebraic Semantics of Associative Array Access 2

2 Relation-Algebraic Semantics of Disjoint-Set Forests 4

3 Verifying Operations on Disjoint-Set Forests 13
   3.1 Make-set ........................................... 14
   3.2 Find-set ........................................... 14
   3.3 Path Compression ................................. 18
   3.4 Find-set with Path Compression ................. 29
   3.5 Union-sets ........................................ 31

This theory has been developed with Isabelle2019.

theory Disjoint-Set-Forests


begin

no-notation trancl ((+) [1000] 999)

context stone-relation-algebra
begin
An arc in a Stone relation algebra corresponds to an atom in a relation algebra.

**lemma** points-arc:
point \( x \Rightarrow point \ y \Rightarrow arc \ (x \ast y^T) \)
by (metis comp-associative cone-dist-comp conv-involutive equivalence-top-closed)

**lemma** point-arc:
point \( x \Rightarrow arc \ (x \ast x^T) \)
by (simp add: points-arc)

**lemma** injective-codomain:
assumes injective \( x \)
shows \( x \ast (x \land 1) = x \land 1 \)
**proof (rule antisym)**
show \( x \land 1 \leq x \land 1 \)
by (metis assms comp-right-one dual-order.trans inf.boundedI inf.cobounded1 inf.sup-monoid.add-commute mult-right-isotone one-inf-conv)
next
show \( x \land 1 \leq x \ast (x \land 1) \)
by (metis coreflexive-idempotent inf.cobounded1 inf.cobounded2 mult-left-isotone)
qed

1 Relation-Algebraic Semantics of Associative Array Access

**abbreviation** rel-update :: \( 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \) \([70, 65, 65] 61\)
where \( x[y \mapsto z] \equiv (y \land z^T) \sqcup (\neg y \land x) \)

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

**Theorem 1.1**

**lemma** update-univalent:
assumes univalent \( x \)
and vector \( y \)
and injective \( z \)
shows univalent \( (x[y \mapsto z]) \)
**proof**
−
have 1: univalent \( (y \land z^T) \)
using assms(3) inf-commute univalent-inf-closed by force
have \( (y \land z^T)^T \ast (\neg y \land x) = (y^T \land z) \ast (\neg y \land x) \)
by (simp add: conv-dist-inf)
also have \( ... = z \ast (y \land \neg y \land x) \)
by (metis assms(2) covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute)
finally have 2: \( (y \land z^T)^T \ast (\neg y \land x) = bot \)

2
by simp
have 3: \(-y\) using assms(2) vector-complement-closed by simp
have \((-y \cap x)^T * (y \cap z^T) = (-y^T \cap x^T) * (y \cap z^T)\)
  by (simp add: conv-complement conv-dist-inf)
also have \(... = x^T * (-y \cap y \cap z^T)\)
  using 3 by (metis (mono-tags, hide-lams) conv-complement covector-inf-comp-3 inf.sup-monoid.add-conv inf.sup-monoid.add-commute)
finally have 4: \((-y \cap x)^T * (y \cap z^T) = bot\)
  by simp
have 5: univalent \((-y \cap x)\)
  using assms(1) inf-commute univalent-inf-closed by fastforce
have \((x[y\mapsto z])^T * (x[y\mapsto z]) = (y \cap z^T)^T * (x[y\mapsto z]) \sqcup (-y \cap x)^T * \(x[y\mapsto z]\)\)
  by (simp add: conv-dist-sup mult-right-dist-sup)
also have \(... = (y \cap z^T)^T * (y \cap z^T) \sqcup (y \cap z^T)^T * (-y \cap x)^T * \(y \cap z^T\) \sqcup (-y \cap x)^T * (y \cap z^T)\)
  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 \((x[y\mapsto z]) \times top = x \times top[y\mapsto top]z\)
    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 bijective z
  shows mapping \((x[y\mapsto z])\)
using assms update-univalent update-total by simp

3
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-bijective:
assumes bijective y
and mapping x
shows bijective (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

end

2 Relation-Algebraic Semantics of Disjoint-Set Forests

context stone-kleene-relation-algebra
begin

lemma equivalence-star-closed:
equivalence x \implies equivalence (x^*)
by (simp add: conv-star-commute star.circ-reflexive star.circ-transitive-equal)

lemma equivalence-plus-closed:
equivalence x \implies equivalence (x^+)
by (simp add: conv-star-commute star.circ-reflexive star.circ-sup-one-left-unfold star.circ-transitive-equal)

lemma reachable-without-loops:
x^* = (x \cap -1)^*
proof (rule antisym)
have $x \ast (x \sqcap -1)^* = (x \sqcap 1) \ast (x \sqcap -1)^* \sqcup (x \sqcap -1) \ast (x \sqcap -1)^*$
  by (metis maddux-3-11-pp mult-right-dist-sup regular-one-closed)
also have $\ldots \leq (x \sqcap -1)^*$
  by (metis inf.cobounded2 le-supI mult-left-isotone star.circ-circ-mult star.left-plus-below-circ star-involutive star-one)
finally show $x^* \leq (x \sqcap -1)^*$
  by (metis inf.cobounded2 maddux-3-11-pp regular-one-closed star.circ-circ-mult star.circ-sup-2 star-involutive star-sub-one)
next
  show $(x \sqcap -1)^* \leq x^*$
  by (simp add: star-isotone)
qed

lemma star-plus-loops:
  $x^* \sqcup 1 = x^+ \sqcup 1$
using star.circ-plus-one star-left-unfold-equal sup-commute by auto

lemma star-plus-without-loops:
  $x^* \sqcap -1 = x^+ \sqcap -1$
by (metis maddux-3-13 star-left-unfold-equal)

Theorem 4.2

lemma omit-redundant-points:
  assumes point p
  shows $p \sqcap x^* = (p \sqcap 1) \sqcup (p \sqcap x) \ast (\neg p \sqcap x)^*$
proof (rule antisym)
  let $?p = p \sqcap 1$
  have $?p \ast x \ast (\neg p \sqcap x)^* \ast ?p \leq ?p \ast \top \ast ?p$
    by (metis comp-associative mult-left-isotone mult-right-isotone top.extremum)
also have $\ldots \leq ?p$
  by (simp add: assms injective-codomain vector-inf-one-comp)
finally have $?p \ast x \ast (\neg p \sqcap x)^* \ast ?p \ast x \leq ?p \ast x$
  using mult-left-isotone by blast
hence $?p \ast x \ast (\neg p \sqcap x)^* \ast (p \sqcap x) \leq ?p \ast x$
  by (simp add: assms comp-associative vector-inf-one-comp)
also have $1: \ldots \leq ?p \ast x \ast (\neg p \sqcap x)^*$
  using mult-right-isotone star.circ-reflexive by fastforce
finally have $?p \ast x \ast (\neg p \sqcap x)^* \ast (p \sqcap x) \sqcup ?p \ast x \ast (\neg p \sqcap x)^* \ast (\neg p \sqcap x)
\leq ?p \ast x \ast (\neg p \sqcap x)^*$
  by (simp add: mult-right-isotone star.circ-plus-same star.left-plus-below-circ mult-associ)
  hence $?p \ast x \ast (\neg p \sqcap x)^* \ast ((p \sqcup \neg p) \sqcap x) \leq ?p \ast x \ast (\neg p \sqcap x)^*$
    by (simp add: comp-inf.mult-right-dist-sup mult-left-dist-sup)
  hence $?p \ast x \ast (\neg p \sqcap x)^* \ast x \leq ?p \ast x \ast (\neg p \sqcap x)^*$
    by (metis assms bijective-regular inf.absorb2 inf.cobounded1 inf.sup-monoid.add-commute shunting-p)
  hence $?p \ast x \ast (\neg p \sqcap x)^* \ast x \sqcup ?p \ast x \ast (\neg p \sqcap x)^*$
    using 1 by simp
  hence $?p \ast (1 \sqcup x) \ast (\neg p \sqcap x)^* \ast x \leq ?p \ast x \ast (\neg p \sqcap x)^*$
by (simp add: comp-associative mult-left-dist-sup mult-right-dist-sup)
also have ... \leq p \ast (I \sqcup x \ast (\neg p \sqcap x)*)
by (simp add: comp-associative mult-right-isotone)
finally have \ ?p \ast x^* \leq p \ast (I \sqcup x \ast (\neg 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 \ ... = ?p \sqcup ?p \ast x \ast (\neg p \sqcap x)*
by (simp add: comp-associative semiring.
distrib-left)
finally show p \sqcap x^* \leq ?p \sqcup (p \sqcap x) \ast (\neg p \sqcap x)*
by (simp add: assms vector-inf-one-comp)
show \ ?p \sqcup (p \sqcap x) \ast (\neg p \sqcap x)* \leq p \sqcap x^*
by (metis assms comp-isotone inf.
boundedI inf.coboundedI inf.coboundedI2 inf.sup-monoid.add.
commute le-supI star.circ-increasing star.circ-transitive-equal
star-isotone star-left-unfold-equal sup.coboundedI vector-export-comp)
qed

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

Theorem 6.1
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 6.2
lemma \ wcc-increasing:
x \leq wcc x
by (simp add: star.
circ-sub-dist-1)

lemma \ wcc-isotone:
x \leq y \Rightarrow 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 6.3
lemma \ wcc-below-wcc:
x \leq wcc y \Rightarrow wcc x \leq wcc y
using wcc-idempotent wcc-isotone by fastforce

Theorem 6.4
lemma \ wcc-bot:
wcc bot = 1
by (simp add: star.
circ-zero)
lemma wcc-one:
\[ wcc \ 1 = 1 \]
by (simp add: star-one)

Theorem 6.5

lemma wcc-top:
\[ wcc \ top = top \]
by (simp add: star.circ-top)

Theorem 6.6

lemma wcc-with-loops:
\[ wcc \ x = wcc \ (x \sqcup 1) \]
using conv-dist-sup star-decompose-1 star-sup-one sup-commute
symmetric-one-closed by presburger

lemma wcc-without-loops:
\[ wcc \ x = wcc \ (x \sqcap -1) \]
by (metis conv-star-commute star-sum reachable-without-loops)

lemma forest-components-wcc:
\[ \text{inj} \ x \Rightarrow wcc \ x = \text{forest-components} \ x \]
by (simp add: cancel-separate-1)

abbreviation fc x ≡ x⋆∗xT⋆

Theorem 2.1

lemma fc-equivalence:
\[ \text{univ} \ x \Rightarrow \text{equivalence} \ (fc \ x) \]
apply (intro conjI)
subgoal by (simp add: reflexive-mult-closed star.circ-reflexive)
subgoal by (metis cancel-separate-1 eq-iff star.circ-transitive-equal)
subgoal by (simp add: conv-dist-comp conv-star-commute)
done

Theorem 2.2

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

Theorem 2.3

lemma fc-isotone:
\[ x \leq y \Rightarrow fc \ x \leq fc \ y \]
by (simp add: comp-isotone conv-isotone star-isotone)

Theorem 2.4

lemma fc-idempotent:
\[ \text{univ} \ x \Rightarrow fc \ (fc \ x) = fc \ x \]
by (metis fc-equivalence cancel-separate-1 star.circ-transitive-equal star-involutive)

Theorem 2.5

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

lemma fc-plus:
univalent x ⇒ (fc x)^+ = fc x
by (metis fc-star star.circ-decompose-9)

Theorem 2.6

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

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

Theorem 2.7

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

Theorem 6.7

lemma fc-wcc:
univalent x ⇒ wcc x = fc x
by (simp add: fc-star star-decompose-1)

Theorem 4.1

lemma update-acyclic-1:
assumes acyclic (p \sqcap -1)
and point y
and point w
and y ≤ p^* ∗ w
shows acyclic ((p[w\rightarrow]y) \sqcap -1)

proof –
let ?p = p[w\rightarrow]y
have w ≤ p^* ∗ y
using assms(2-4) by (metis (no-types, lifting) bijective-reverse conv-star-commute)

hence w ∗ y^T ≤ p^*
using assms(2) shunt-bijective by blast
hence w ∗ y^T ≤ (p \sqcap -1)^*
using reachable-without-loops by auto

hence w ∗ y^T ∩ -1 ≤ (p \sqcap -1)^* ∩ -1
by (simp add: inf.coboundedI2 inf.sup-monoid.add-commute)
also have ... ≤ \((p \cap -1)^+\)
  by (simp add: star-plus-without-loops)

finally have 1: \(w \cap y^T \cap -1 \leq (p \cap -1)^+\)
  using assms(2,3) vector-covector by auto

have \(?p \cap -1 = (w \cap y^T \cap -1) \sqcup (-w \cap p \cap -1)\)
  by (simp add: inf-sup-distrib2)
also have ... ≤ \((p \cap -1)^+ \sqcup (-w \cap p \cap -1)\)
  using 1 sup-left-isotone by blast
also have ... ≤ \((p \cap -1)^+ \sqcup (p \cap -1)\)
  using comp-inf mult-semi-associative sup-right-isotone by auto
also have ... = \((p \cap -1)^+\)
  by (metis star.circ-back-loop-fixpoint sup.right-idem)

finally have \((?p \cap -1)^+ \leq (p \cap -1)^+\)
  by (metis comp-associative comp-isotone star.circ-transitive-equal star.left-plus-circ star-isotone)
also have ... ≤ -1
  using assms(1) by blast
finally show \(?thesis\)
  by simp
qed

abbreviation rectangle :: 'a ⇒ bool
  where rectangle x ≡ x * top * x = x

lemma arc-rectangle:
  arc x ⇒ rectangle x
  using arc-top-arc by blast

lemma rectangle-star-rectangle:
  rectangle a ⇒ a * x^* * a ≤ a
  by (metis mult-left-isotone mult-right-isotone top.extremum)

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

lemma star-arc-decompose:
\[
\text{arc } a \Rightarrow (a \sqcup x)^* = x^+ \sqcup x^* \star a \star x^*
\]
using arc-top-arc star-rectangle-decompose by blast

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

Theorem 6.1

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

Theorem 6.2

lemma update-acyclic-2:
assumes acyclic \((p \sqcap -1)\)
and point \(y\)
and point \(w\)
and \(y \sqcap p^* \star w = \bot\)
shows acyclic \(((p[\!\!\!w \mapsto y]) \sqcap -1)\)
proof –
let \(?p = p[\!\!\!w \mapsto y]\)
have \(y^\top \star p^* \star w \leq -1\)
using assms(4) comp-associative pseudo-complement schroeder-3-p by auto

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

by (metis comp-associative comp-commute-below-diversity
star.circ-transitive-equal)

have \( ?p \cap -1 \leq (w \cap y^T) \sqcup (p \cap -1) \)

by (metis comp-inf .mult-right-dist-sup dual-order .trans inf .cobounded1
inf .cobounded12 inf .sup-monoid .add-assoc le-sup1 sup .cobounded1 sup-ge2)

also have \( ... = w \cdot y^T \sqcup (p \cap -1) \)

using assms(2,3) by (simp add: vector-covector)

finally have \( (?p \cap -1)^* \leq (w \cdot y^T \sqcup (p \cap -1))^+ \)

by (simp add: comp-isotone star-isotone)

also have \( ... = (p \cap -1)^+ \sqcup (p \cap -1)^* \cdot w \cdot y^T \cdot (p \cap -1)^* \)

using assms(2,3) plus-arc-decompose points-arc by (simp add: comp-associative)

also have \( ... \leq (p \cap -1)^+ \sqcup p^* \cdot w \cdot y^T \cdot p^* \)

using reachable-without-loops by auto

also have \( ... \leq -1 \)

using 1 assms(1) by simp

finally show \(?thesis \)

by simp

qed


lemma acyclic-down-closed:
\( x \leq y \Rightarrow \text{acyclic } y \Rightarrow \text{acyclic } x \)

using comp-isotone star-isotone by fastforce


Lemma 6.3

lemma update-acyclic-3:
assumes acyclic \( (p \cap -1) \)

and point \( w \)

shows acyclic \( (\{w[w\rightarrow w]\} \cap -1) \)

proof –

let \( ?p = p[w\rightarrow w] \)

have \( ?p \cap -1 \leq (w \cap y^T \cap -1) \sqcup (p \cap -1) \)

using comp-inf .mult-right-dist-sup inf .cobounded2 inf .sup-monoid .add-assoc
sup-right-isotone by presburger

also have \( ... = p \cap -1 \)

using assms(2) by (metis comp-inf .covektor-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

end

custom stonetion-algebra-tarski

begin
lemma point-in-vector-partition:
assumes point x and vector y
shows $x \leq -y \lor x \leq -y$
proof (cases $x \cdot x^T \leq -y$)
case True
have $x \leq x \cdot x^T \cdot x$
  by (simp add: ex231c)
also have $... \leq -y \cdot x$
  by (simp add: True mult-left-isotone)
also have $... \leq -y$
  by (metis assms (2) mult-right-isotone top.extremum
  vector-complement-closed)
finally show ?thesis
  by simp
next
case False
have $x \leq x \cdot x^T \cdot x$
  by (simp add: ex231c)
also have $... \leq -y \cdot x$
  using False assms (1) arc-in-partition mult-left-isotone point-arc by blast
also have $... \leq -y$
  by (metis assms (2) mult-right-isotone top.extremum
  vector-complement-closed)
finally show ?thesis
  by simp
qed

lemma point-atomic-vector:
assumes point x and vector y and regular y and $y \leq x$
shows $y = x \lor y = \bot$
proof (cases $x \leq -y$)
case True
thus ?thesis
  using assms (4) inf.absorb2 pseudo-complement by force
next
case False
thus ?thesis
  using assms point-in-vector-partition by fastforce
qed

Theorem 4.3

lemma distinct-points:
assumes point x and point y and $x \neq y$
\[
\text{shows } x \cap y = \text{bot}
\]
\begin{itemize}
\item by (metis asssms antisym comp-bijective-complement inf.sap-monoid.add-commute mult-left-one pseudo-complement regular-one-closed point-in-vector-partition)
\end{itemize}

end

3 Verifying Operations on Disjoint-Set Forests

syntax
\[
\text{-rel-update :: idt } \Rightarrow 'a \Rightarrow 'a \Rightarrow 'b \text{ com } ((2\cdot[]) := -) [70, 65, 65] 61
\]

translations
\[
x[y] := z \Rightarrow (x := (y \cap z^T) \sqcup (\text{CONST } \text{uminus} y \cap x))
\]

class finite-regular-p-algebra = p-algebra +
\begin{itemize}
\item assumes finite-regular: finite \{ x . regular x \}
\end{itemize}

class stone-kleene-relation-algebra-tarski = stone-kleene-relation-algebra + stone-relation-algebra-tarski

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

begin

abbreviation root p x \equiv p^{T*} \ast x \cap (p \cap 1) \ast \text{top}

Theorem 3.1

lemma root-var:
\[
\text{root } p x = (p \cap 1) \ast p^{T*} \ast x
\]
\begin{itemize}
\item by (simp add: coreflexive-comp-top-inf inf-commute mult-assoc)
\end{itemize}

Theorem 3.2

lemma root-successor-loop:
\[
\text{univalent } p \implies \text{root } p x = p[\text{root } p x]
\]
\begin{itemize}
\item by (metis root-var injective-codomain comp-associative conv-dist-inf coreflexive-symmetric equivalence-one-closed inf.cobounded2 univalent-conv-injective)
\end{itemize}

lemma root-transitive-successor-loop:
\[
\text{univalent } p \implies \text{root } p x = p^{T*} \ast (\text{root } p x)
\]
\begin{itemize}
\item by (metis mult-1-right star-one star-simulation-right-equal root-successor-loop)
\end{itemize}

Theorem 1.8

lemma update-postcondition:
\begin{itemize}
\item assumes point x point y
\item shows \( x \cap p = x \ast y^T \longleftrightarrow p[[x]] = y \)
\item apply (rule iffI)
\end{itemize}
subgoal by (metis assms comp-associative conv-dist-comp conv-involute
covector-inf-comp-3 equivalence-top-closed vector-covector)

subgoal
  apply (rule antisym)
subgoal by (metis assms conv-dist-comp conv-involute inf.boundedI
inf.coboundedI vector-covector vector-restrict-comp-conv)

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

done

3.1 Make-set

definition make-set-postcondition p x p0 \equiv\ x \sqcap p = x \sqcap p0 = x \sqcap p = -x \sqcap p0

theorem make-set:
  VARS p
  [ point x \land 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])

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

3.2 Find-set

abbreviation disjoint-set-forest p \equiv mapping p \land acyclic (p \sqcap -1)

definition find-set-precondition p x \equiv disjoint-set-forest p \land point x

definition find-set-invariant p x y \equiv find-set-precondition p x \land point y \land y \leq p^Ty \sqcap x
definition find-set-postcondition \( p \ x \ y \equiv \text{point} \ y \land y = \text{root} \ p \ x \)

lemma find-set-1:
find-set-precondition \( p \ x \implies \text{find-set-invariant} \ p \ x \ x \)
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 \( p \ x \ y \land y \neq p[[y]] \land \text{card} \ \{ z . \ \text{regular} \ z \land z \leq p^{T^*} \ast y \} = n \implies \text{find-set-invariant} \ p \ x (p[[y]]) \land \text{card} \ \{ z . \ \text{regular} \ z \land z \leq p^{T^*} \ast (p[[y]]) \} < n \)

proof —
let ?s = \{ z . \ \text{regular} \ z \land z \leq p^{T^*} \ast y \}
let ?t = \{ z . \ \text{regular} \ z \land z \leq p^{T^*} \ast (p[[y]]) \}
assume 1: find-set-invariant \( p \ x \ y \land y \neq p[[y]] \land \text{card} \ ?s = n \)
hence 2: point \( (p[[y]]) \)
using read-point find-set-invariant-def find-set-precondition-def by simp
show \( \text{find-set-invariant} \ p \ x (p[[y]]) \land \text{card} \ ?t < n \)
proof (unfold find-set-invariant-def, intro conjI)
show \( \text{find-set-precondition} \ p \ x \)
using 1 find-set-invariant-def by simp
show vector \( (p[[y]]) \)
using 2 by simp
show surjective \( (p[[y]]) \)
using 2 by simp
show \( \text{p}[[y]] \leq p^{T^*} \ast x \)
using 1 by (metis (hide-lams) find-set-invariant-def comp-associative comp-isotone star.circ-increasing star.circ-transitive-equal)
show \( \text{card} \ ?t < n \)
proof —
have 3: \( (p^T \cap -1) \ast (p^T \cap -1)^+ \ast y \leq (p^T \cap -1)^+ \ast y \)
by (simp add: mult-left-isotone mult-right-isotone star.left-plus-below-circ)
have \( p[[y]] = (p^T \cap 1) \ast y \lor (p^T \cap -1) \ast y \)
by (metis maddux-3-11-pp mult-right-dist-sup regular-one-closed)
also have \... \leq ((p[[y]]) \cap y) \lor (p^T \cap -1) \ast y \)
by (metis comp-left-subdist-inf mult-1-left semiring.add-right-mono)
also have \... = (p^T \cap -1) \ast y \)
using 1 2 find-set-invariant-def distinct-points by auto
finally have 4: \( (p^T \cap -1)^+ \ast (p[[y]]) \leq (p^T \cap -1)^+ \ast y \)
using 3 by (metis inf.antisym-conv inf.eq-refl inf-le1 mult-left-isotone star-plus mult-assoc)
hence \( p^{T^*} \ast (p[[y]]) \leq p^{T^*} \ast y \)
by (metis mult-isotone order-refl star.left-plus-below-circ star-plus mult-assoc)
hence 5: \( ?t \subseteq ?s \)
using order-trans by auto
have 6: \( y \in ?s \)
using 1 find-set-invariant-def bijective-regular mult-left-isotone
star.circ-reflexive by fastforce
have 7: ~ y ∈ ?t
proof
  assume y ∈ ?t
  hence y ≤ (p⁺T ∩ -1)⁺ * y
    using 4 by (metis reachable-without-loops mem-Collect-eq order-trans)
  hence y * y⁺T ≤ (p⁺T ∩ -1)⁺
    using 1 find-set-invariant-def shunt-bijective by simp
  also have ... ≤ -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 ≤ -y
  using Schroeder-4-p by auto
thus False
  using 1 by (metis find-set-invariant-def shunt-bijective shunt-mapping top-right-mult-increasing
pseudo-complement surjective-conv-total top.extremum vector-top-closed
regular-closed-top)
qed
have card ?t < card ?s
  apply (rule psubset-card-mono)
subgoal using finite-regular by simp
subgoal using 5 6 7 by auto
done
thus ?thesis
  using 1 by simp
qed
qed

lemma find-set-3:
  find-set-invariant p x y ∧ y = p[[y]] → find-set-postcondition p x y
proof –
  assume 1: find-set-invariant p x y ∧ 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 = root p x
      using antisym
      have y * y⁺T ≤ 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 ≤ p ∩ 1
        using 1 find-set-invariant-def le-infI by blast
      hence y ≤ (p ∩ 1) * top
using 1 by (metis find-set-invariant-def order-lesseq-imp shunt-bijective)
thus \( y \leq \text{root } p \ x \n
using 1 find-set-invariant-def by simp
next
have 2: \( x \leq p^* \ast y \)
using 1 find-set-invariant-def find-set-precondition-def bijective-reverse
conv-star-commute by auto
have \( p^T \ast p^* \ast y = p^T \ast p \ast p^* \ast y \uplus (p[[y]]) \)
by (metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint)
also have \( \ldots \leq p^* \ast y \uplus y \)
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 \( \ldots = p^* \ast y \)
by (metis star.circ-loop-fixpoint sup.left-idem sup-commute)
finally have 3: \( p^{T^*} \ast x \leq p^* \ast y \)
using 2 by (simp add: comp-associative star-left-induct)
have \( p \ast y \cap (p \cap 1) \ast top = (p \cap 1) \ast p \ast y \)
using comp-associative coreflexive-comp-top-inf inf-commute by auto
also have \( \ldots \leq p^T \ast p \ast y \)
by (metis inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone
one-inf-conv)
also have \( \ldots \leq y \)
using 1 find-set-invariant-def find-set-precondition-def mult-left-isotone by fastforce
finally have 4: \( p \ast y \leq y \uplus -((p \cap 1) \ast top) \)
using 1 by (metis find-set-invariant-def shunting-p bijective-regular)
have \( p^T \ast p \ast (p \cap 1) \leq p^T \ast 1 \)
using 1 by (metis find-set-invariant-def find-set-precondition-def N-top
comp-isotone coreflexive-idempotent inf.cobounded2 inf.sup-monoid.add-commute
inf-assoc one-inf-conv shunt-mapping)
hence \( p^T \ast (p \cap 1) \ast top \leq (p \cap 1) \ast top \)
using inf-commute mult-isotone one-inf-conv by auto
hence \( p \ast -((p \cap 1) \ast top) \leq -((p \cap 1) \ast top) \)
by (metis comp-associative inf.sup-monoid.add-commute p-antitone
p-antitone-iff schroeder-3-p)
hence \( p \ast y \cap p \ast -((p \cap 1) \ast top) \leq y \uplus -((p \cap 1) \ast top) \)
using 4 dual-order.trans le-supl sup-ge2 by blast
hence \( p \ast (y \uplus -((p \cap 1) \ast top)) \leq y \uplus -((p \cap 1) \ast top) \)
by (simp add: mult-left-dist-sup)
hence \( p^* \ast y \leq y \uplus -((p \cap 1) \ast top) \)
by (simp add: star-left-induct)
hence \( p^{T^*} \ast x \leq y \uplus -((p \cap 1) \ast top) \)
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
theorem find-set:
VARS y
[find-set-precondition p x ]
y := x;
WHILE y \neq p[[y]]
{find-set-invariant p x y }
VAR {card \{z . regular z \land z \leq p^T \ast y \}}
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 . find-set-postcondition p x y
using tc-extract-function find-set by blast

3.3 Path Compression

definition path-compression-precondition p x y \equiv disjoint-set-forest p \land point x
\land point y \land y = root p x
definition path-compression-invariant p x y p0 w \equiv
path-compression-precondition p x y \land point w \land y \leq p^T \ast w \land (w \neq x \implies
p[[x]] = y \land y \neq x \land p^T \ast w \leq -x) \land p \cap 1 = p0 \cap 1 \land fc p = fc p0
definition path-compression-postcondition p x y p0 \equiv
path-compression-precondition p x y \land p \cap 1 = p0 \cap 1 \land fc p = fc p0

lemma path-compression-1:
path-compression-precondition p x y \land p0 = p \implies path-compression-invariant p
x y p x
using path-compression-invariant-def path-compression-precondition-def by auto
lemma path-compression-2:
path-compression-invariant p x y p0 w ∧ y ≠ p[[w]] ∧ card { z . regular z ∧ z ≤ pT* ∗ w } = n ⇒ path-compression-invariant (p[w→y]) x y p0 (p[[w]]) ∧ card { z . regular z ∧ z ≤ (p[w→y])T* ∗ (p[[w]]) } ≤ n
proof –
  let ?p = p[w→y]
  let ?s = { z . regular z ∧ z ≤ pT* ∗ w }
  let ?t = { z . regular z ∧ z ≤ ?pT* ∗ (p[[w]]) }
  assume 1: path-compression-invariant p x y p0 w ∧ y ≠ p[[w]] ∧ card ?s = n
  hence 2: point (p[[w]])
  by (simp add: path-compression-invariant-def
path-compression-precondition-def read-point)
  show path-compression-invariant ?p x y p0 (p[[w]]) ∧ card ?t < n
  proof (unfold path-compression-invariant-def, intro conjI)
    have 3: mapping ?p
      using 1 by (meson path-compression-invariant-def
path-compression-precondition-def update-mapping bijective-regular)
    have 4: w ≠ y
      using 1 by (metis (no-types, hide-lams) path-compression-invariant-def
path-compression-precondition-def root-successor-loop)
    hence 5: w ⊓ y = bot
      using 1 distinct-points path-compression-invariant-def
path-compression-precondition-def by auto
    hence y * wT ≤ −1
      using pseudo-complement schroeder-4-p by auto
    hence y * wT ≤ pT* ⊓ −1
      using 1 shunt-bijective path-compression-invariant-def by auto
    also have ... ≤ pT+
      by (simp add: star-plus-without-loops)
    finally have 6: y ≤ pT* ∗ w
      using 1 shunt-bijective path-compression-invariant-def by blast
    have 7: w * wT ≤ −pT+
      proof (rule cocontr)
        assume − w * wT ≤ −pT+
        hence w * wT ≤ −pT+
          using 1 path-compression-invariant-def point-arc arc-in-partition by blast
        hence w * wT ≤ pT* ⊓ 1
          using 1 path-compression-invariant-def path-compression-precondition-def
mapping-regular regular-conv-closed regular-conv-closed-star regular-mult-closed by simp
        also have ... = ((pT ⊓ 1) ∗ pT* ⊓ 1) ∪ ((pT ⊓ −1) ∗ pT* ⊓ 1)
          by (metis comp-inf mult-right-dist-sup maddux-3-11-pp mult-right-dist-sup
regular-one-closed)
        also have ... = ((pT ⊓ 1) ∗ pT* ⊓ 1) ∪ ((p ⊓ −1)+ ⊓ 1)T
          by (metis cone-complement conv-dist-inf cone-plus-commute
equivalence-one-closed reachable-without-loops)
        also have ... ≤ ((pT ⊓ 1) ∗ pT* ⊓ 1) ∪ (−1 ⊓ 1)T
          using 1 by (metis (no-types, hide-lams) path-compression-invariant-def
... ...
path-compression-precondition-def sup-right-isotone inf.sup-left-isotone
conv-isotone)
also have ... = (pT ⨿ 1) * pT* ⨿ 1
by simp
also have ... ≤ (pT ⨿ 1) * top ⨿ 1
by (metis comp-inf.comp-isotone coreflexive-comp-top-inf
equivalence-one-closed inf.cobounded1 inf.cobounded2)
also have ... ≤ pT
by (simp add: coreflexive-comp-top-inf-one)
finally have w * wT ≤ pT
by simp
hence w ≤ p[[w]]
using 1 path-compression-invariant-def shunt-bijective by blast
hence w = p[[w]]
using 1 2 path-compression-invariant-def epm-3 by fastforce
hence w = pT+ * w
using 2 by (metis comp-associative star.circ-top star-simulation-right-equal)
thus False
using 1 4 6 epm-3 path-compression-invariant-def
path-compression-precondition-def by fastforce
qed
hence 8: w ⨿ pT+ * w = bot
using p-antitone-iff pseudo-complement Schroeder-4-p by blast
show y ≤ ?pT* * (p[[w]])
proof –
  have (w ⨿ yT)T * (¬w ⨿ p)T* * pT * w ≤ wT * (¬w ⨿ p)T* * pT * w
  by (simp add: conv-isotone mult-left-isotone)
  also have ... ≤ wT * pT* * pT * w
  by (simp add: conv-isotone mult-left-isotone star-isotone mult-right-isotone)
  also have ... = wT * pT+ * w
  by (simp add: star-plus mult-assoc)
  also have ... = bot
  using 1 8 by (metis (no-types, hide-lams) path-compression-invariant-def
covector-inf-comp-3 mult-associative conv-dist-comp conv-star-commute
covector-bot-closed equivalence-top-closed inf.le-iff-sup mult-left-isotone)
finally have (w ⨿ yT)T ⨿ ((¬w ⨿ p)T* * pT * w ≤ (¬w ⨿ p)T * (¬w ⨿ p)T* * pT * w
by (simp add: bot-unique mult-right-dist-sup)
also have ... ≤ (¬w ⨿ p)T* * pT * w
by (simp add: mult-left-isotone star.left-plus-below-circ)
finally have ?pT* * (¬w ⨿ p)T* * pT * w ≤ (¬w ⨿ p)T* * pT * w
by (simp add: conv-dist-sup)
hence ?pT* * pT* * w ≤ (¬w ⨿ p)T* * pT * w
by (metis comp-associative star.circ-loop-fixpoint star-left-induct
sup-commute sup-left-sup-sup-left-divisibility)
hence w ⨿ ?pT* * pT* * w ≤ w ⨿ ((¬w ⨿ p)T* * pT * w
using inf.sup-right-isotone by blast
also have ... ≤ w ⨿ pT* * pT * w
using conv-isotone mult-left-isotone star-isotone inf.sup-right-isotone by simp
also have ... = bot
using 8 by (simp add: star-plus)
finally have 9: \( w^T \otimes p^T \otimes p^T \otimes w = bot \)
using 1 by (metis (no-types, hide-lams) path-compression-invariant-def
covector-inf-comp-3 mult-assoc cone-dist-comp covector-bot-closed
equivalence-top-closed inf.le-iff-sup mult-left-isotone bot-least inf.absorb1)
also have \( p^T \otimes p^T \otimes p^T \otimes w \leq ((w \cap p)^T \cup (w \cap p)^T) \otimes p^T \otimes p^T \otimes w \)
using 1 by (metis (no-types, lifting) bijective-regular conv-dist-sup
inf-commute maddux-3-11-pp path-compression-invariant-def)
also have ... = (w \cap p)^T \otimes p^T \otimes p^T \otimes w \cup (w \cap p)^T \otimes p^T \otimes p^T \otimes w
by (simp add: conv-isotone mult-left-isotone)
also have ... \leq p^T \otimes p^T \otimes w
by (simp add: comp-isotone star.left-plus-below-circ)
finally have \( p^T \otimes p^T \otimes w \leq p^T \otimes p^T \otimes w
by (metis comp-associative star.circ-loop-fixpoint star-left-induct
sup-commute sup-bot sup-left-divisibility)
thus \( y \leq p^T \otimes (p[[w]]) \)
using 6 by (simp add: star-simulation-right-equal mult-assoc)
Qed
have 10: acyclic \( \forall p \neq -1 \)
using 1 update-acyclic-1 path-compression-invariant-def
path-compression-precondition-def by auto
have \( p[[p^T \otimes w]] \leq p^T \otimes w \)
proof
have \( (w^T \cap y) \otimes p^T \otimes w = y \cap w^T \otimes p^T \otimes w \)
using 1 by (metis (no-types, hide-lams) path-compression-invariant-def
path-compression-precondition-def inf-commute vector-inf-comp)
also have ... \leq y \cup (w^T \cap p)^T \otimes p^T \otimes w
by (simp add: conv-isotone sup-right-isotone conv-dist-sup
mult-left-isotone by auto)
also have ... \leq y \cup p^T \otimes p^T \otimes w
using mult-left-isotone sup-right-isotone by auto
finally show \( ?thesis \)
by simp

21
qed

hence 11: \( ?pT^* * (p[[w]]) \leq pT^+ * w \)
using star-left-induct by (simp add: mult-left-isotone star.circ-mult-increasing)

hence 12: \( ?pT^+ * (p[[w]]) \leq pT^+ * w \)
using dual-order.trans mult-left-isotone star.left-plus-below-circ by blast

have 13: \( ?p[[x]] = y \land y \neq x \land ?pT^* * (p[[w]]) \leq -x \)
proof (cases \( w = x \))

  case True
  hence \( ?p[[x]] = (wT \cap y) * w \cup (-wT \cap pT) * w \)
  by (simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)
  also have ... = \( (wT \cap y) * w \cup -w \cap w \)
  using 1 by (metis (no-types, lifting) conv-complement inf.sup-monoid.add-commute path-compression-invariant-def covector-inf-comp-3 vector-complement-closed)
  also have ... = \( y * w \)
  using 1 inf.sup-monoid.add-commute path-compression-invariant-def covector-inf-comp-3 by simp
  also have ... = \( pT * (-w \cap x) \)
  using 1 False path-compression-invariant-def
path-compression-precondition-def distinct-points by auto
  also have ... = \( y \)
  using 1 False path-compression-invariant-def path-compression-precondition-def distinct-points inf.absorb2 pseudo-complement by auto
finally show \(?thesis\)
  using 1 12 False path-compression-invariant-def by auto

next
  case False
  have \( ?p[[x]] = (wT \cap y) * x \cup (-wT \cap pT) * x \)
  by (simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)
  also have ... = \( y * (w \cap x) \cup pT * (-w \cap x) \)
  using 1 by (metis (no-types, lifting) conv-complement inf.sup-monoid.add-commute path-compression-invariant-def covector-inf-comp-3 vector-complement-closed)
  also have ... = \( pT * (-w \cap x) \)
  using 1 False path-compression-invariant-def
path-compression-precondition-def distinct-points by auto
  also have ... = \( y \)
  using 1 False path-compression-invariant-def path-compression-precondition-def distinct-points inf.absorb2 pseudo-complement by auto
finally show \(?thesis\)
  using 1 12 False path-compression-invariant-def by auto

qed

thus \( p[[w]] \neq x \rightarrow ?p[[x]] = y \land y \neq x \land ?pT^* * (p[[w]]) \leq -x \)
by simp

have 14: \( ?pT^* * x = x \cup y \)
proof (rule antisym)
  have \(?p^T * (x \sqcup y) = y \sqcup {?p^T * y}
  using 13 by (simp add: mult-left-dist-sup)
  also have \(\ldots = y \sqcup (w^T \sqcap y) * y \sqcup (-w^T \sqcap p^T) * y\)
    by (simp add: cone-complement cone-dist-inf cone-dist-sup)
  mult-right-dist-sup sup-assoc
  also have \(\ldots \leq y \sqcup (w^T \sqcap y) * y \sqcup p^T * y\)
    using mult-left-isotone sup-right-isotone by auto
  also have \(\ldots = y \sqcup (w^T \sqcap y) * y\)
    using 1 by (simp add: cone-complement cone-dist-inf cone-dist-sup)
path-compression-invariant-def path-compression-precondition-def
root-successor-loop
  also have \(\ldots \leq y \sqcup y * y\)
    using left by (simp add: mult-left-isotone sup-right-isotone by auto)
  also have \(\ldots = y\)
    using 1 by (metis mult-semi-associative sup-absorb1)
finally have 16: \((?p \sqcap 1) * y = y\)
  using 1 by (metis mult-assoc root-var)
path-compression-precondition-def root-var
  also have \(\ldots \leq (?p^T * x \sqcup x\)).
    by (simp add: star-left-induct)
next
  show \(x \sqcup y \leq (?p^T * x\).
    using 13 by (metis mult-semi-associative sup-absorb1)
star.circ-loop-fixpoint sup.boundedI sup-ge2
qtd
  have 15: \(y = \text{root} ?p x\)
proof
  have \((?p \sqcap 1) * y = (p \sqcap 1) * (p \sqcap 1) * p^T * x\)
    using 1 path-compression-invariant-def path-compression-precondition-def
root-var mult-assoc by auto
  also have \(\ldots = (p \sqcap 1) * p^T * x\)
    using coreflexive-idempotent by auto
finally have 16: \((?p \sqcap 1) * y = y\)
  using 1 path-compression-invariant-def path-compression-precondition-def
root-var by auto
  have 17: \((?p \sqcap 1) * x \leq y\)
    using 1 by (metis mult-left-isotone star.circ-reflexive path-compression-invariant-def
path-compression-precondition-def root-var)
  have root \(?p x = (?p \sqcap 1) * (x \sqcup y)\)
    using 14 by (metis mult-assoc root-var)
  also have \(\ldots = (w \sqcap y^T \sqcap 1) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)\)
    by (simp add: inf-sup-distrib2 semiring.distrib-right)
  also have \(\ldots = (w \sqcap 1 \sqcap y^T) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)\)
    by (simp add: inf.left-commute inf.sup-monoid.add-commute)
  also have \(\ldots = (w \sqcap 1) * (y \sqcap (x \sqcup y)) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)\)
    using 1 by (metis mult-left-isotone path-compression-invariant-def
path-compression-precondition-def covector-inf-comp-3)

also have \( (w \cap I) \ast y \sqcup (\neg w \cap p \cap I) \ast (x \sqcup y) \)
by (simp add: inf.absorb1)
also have \( (w \cap I \ast y) \sqcup (\neg w \cap (p \cap I) \ast (x \sqcup y)) \).
using 1 by (metis (no-types, lifting) inf-associative vector-complement-closed path-compression-invariant-def vector-inf-comb)
also have \( (w \cap y) \sqcup (\neg w \cap (p \cap 1) \ast (x \sqcup y)) \)
using 16 by (simp add: mult-left-dist-sup)
also have \( (w \cap y) \sqcup (\neg w \cap y) \)
using 17 by (simp add: sup.absorb2)
also have \( y \)
using 1 by (metis id-apply bijective-regular comp-inf.right-neutral regular-complement-top path-compression-invariant-def)
finally show \(?thesis\)
by simp
done.

show path-compression-precondition \(?p x y\)
using 1 3 10 15 path-compression-invariant-def
path-compression-precondition-def by auto
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 \( w \cap p \cap I \leq w \cap w^T \cap p \)
by (metis inf.boundedE inf.boundedI inf.cobounded1 inf.cobounded2 one-inf-conv)
also have \( w \ast w^T \cap p \)
using 1 vector-convector path-compression-invariant-def by auto
also have \( (\neg p^T \cap p) \)
using 7 by (simp add: inf.cobounded12 inf.sup-monoid.add-commute)
finally have \( w \cap p \cap I = bot \)
by (metis (no-types, hide-lams) conv-dist-inf coreflexive-symmetric inf.absorb1 inf.boundedE inf.inf.cobounded2 pseudo-complement star.circ-mult-increasing)
also have \( w \cap y^T \cap I = bot \)
using 5 antisymmetric-bot-closed asymmetric-bot-closed comp-inf.schroeder2 inf.absorb1 one-inf-conv by fastforce
finally have \( w \cap p \cap I = w \cap y^T \cap I \)
by simp
thus \(?p \cap I = p0 \cap I\)
using 1 by (metis bijective-regular comp-inf.semiring.distrib-left inf.sup-monoid.add-commute maddux-3-11-pp path-compression-invariant-def)
show fc \(?p = fc p0\)
proof -
have \( p[[w]] = p^T \ast (w \cap p^T \ast y) \)
using 1 by (metis (no-types, lifting) bijective-reverse conv-star-commute inf.absorb1 path-compression-invariant-def path-compression-precondition-def)
also have ... = p^T * (w \cap p^*) * y
  using 1 vector-inf-comp path-compression-invariant-def mult-assoc by auto
also have ... = p^T * ((w \cap 1) \cup (w \cap p) * (\neg w \cap p^*)) * y
  using 1 omit-redundant-points path-compression-invariant-def by auto
also have ... = p^T * (w \cap 1) * y \cup p^T * (w \cap p) * (\neg w \cap p^*) * y
  by (simp add: comp-associative mult-left-dist-sup mult-right-dist-sup)
also have ... \leq p^T * y \cup p^T * (w \cap p) * (\neg w \cap p^*) * y
  by (metis semiring.add-right-mono comp-isotone eq-iff inf.cobounded1
inf.sup-monoid.add-commute mult-1-right)
also have ... = y \cup p^T * (w \cap p) * (\neg w \cap p)^* * y
  using 1 path-compression-invariant-def path-compression-precondition-def
root-successor-loop by fastforce
also have ... = (\neg w \cap p)^* * y
  by (metis star.circ-loop-fixpoint sup.left-idem sup-commute)
finally have 18: p[[w]] \leq (\neg w \cap p)^* * y
  by simp
have p^T * (\neg w \cap p)^* * y = p^T * y \cup p^T * (\neg w \cap p) * (\neg w \cap p)^* * y
  by (metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint
sup-commute)
also have ... = y \cup p^T * (\neg w \cap p) * (\neg w \cap p)^* * y
  using 1 path-compression-invariant-def path-compression-precondition-def
root-successor-loop by fastforce
also have ... \leq y \cup p^T * p * (\neg w \cap p)^* * y
  using comp-isotope sup-right-isotone by auto
also have ... \leq y \cup (\neg w \cap p)^* * y
  using 1 by (metis (no-types, lifting) mult-left-isotone star.circ-circ-mult
star-involutive star-one sup-right-isotone path-compression-invariant-def
path-compression-precondition-def)
also have ... = (\neg w \cap p)^* * y
  by (metis star.circ-loop-fixpoint sup.left-idem sup-commute)
finally have 18: p[[w]] \leq (\neg w \cap p)^* * y
  by simp
have p^T * (\neg w \cap p)^* * y = p^T * y \cup p^T * (\neg w \cap p) * (\neg w \cap p)^* * y
  by (metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint
sup-commute)}
by (metis conv-dist-comp conv-dist-inf conv-involutive conv-isotone
conv-star-commute)

hence \( w \sqcap p \leq (w \sqcap y^T) \ast (-w \sqcap p)^T \ast \)

using 1 by (metis inf.absorb1 inf.left-commute inf.left-idem inf.orderI
vector-inf-comp path-compression-invariant-def)

also have ... \( \leq (w \sqcap y^T) \ast ?p^T \ast \)
by (simp add: conv-isotone mult-right-isotone star-isotone)

also have ... \( \leq ?p \ast ?p^T \ast \)
by (simp add: mult-left-isotone)

also have ... \( \leq fc ?p \)
by (simp add: mult-left-isotone star.circ-increasing)

finally have 20: \( w \sqcap p \leq fc ?p \)
by simp

have \( -w \sqcap p \leq ?p \)
by simp

also have ... \( \leq fc ?p \)
by (simp add: fc-increasing)

finally have \( (w \sqcup -w) \sqcap p \leq fc ?p \)
using 20 by (simp add: comp-inf.semiring.distrib-left
inf.sup-monoid.add-commute)

hence \( p \leq fc ?p \)

using 1 by (metis (no-types, hide-lams) bijective-regular
comp-inf.semiring.distrib-left inf.sup-monoid.add-commute maddux-3-11-pp
path-compression-invariant-def)

hence 21: \( fc p \leq fc ?p \)

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 ... \( \leq w \ast y^T \sqcup p \)
using 1 path-compression-invariant-def path-compression-precondition-def
vector-covector by auto

also have ... \( \leq p^T \sqcup p \)
using 1 by (metis (no-types, lifting) conv-dist-comp conv-involutive
conv-isotone conv-star-commute le-sup1 shunt-bijective star.circ-increasing
sup-absorb1 path-compression-invariant-def)

also have ... \( \leq fc p \)
using fc-increasing star.circ-back-loop-prefixpoint by auto

finally have \( fc ?p \leq fc p \)
using 1 by (metis (no-types, lifting) path-compression-invariant-def path-compression-precondition-def fc-idempotent fc-isotone)

thus \( \text{thesis} \)

using 1 21 path-compression-invariant-def by simp

qed

show \( \text{card} \ ?t < n \)

proof -

\begin{align*}
\text{have} \quad & \ ?p^T \ast p^T \ast \ast w = (w^T \sqcap y) \ast p^T \ast \ast w \sqcup (-w^T \sqcap p^T) \ast p^T \ast \ast w \\
& \text{by (simp add: cone-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)}
\end{align*}

also have ... \( \leq (w^T \sqcap y) \ast p^T \ast \ast w \sqcup p^T \ast p^T \ast \ast w \)
using mult-left-isotone sup-right-isotone by auto
also have ... ≤ (w \ T \ y) \ p^{T*} \ w \ p^{T*} \ w
using mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast
also have ... ≤ y \ p^{T*} \ w \ p^{T*} \ w
using semiring.add-right-mono mult-left-isotone by auto
also have ... ≤ y \ top \ p^{T*} \ w
by (simp add: comp-associative le-supI1 mult-right-isotone)
also have ... = p^{T*} \ w
using 1 path-compression-invariant-def path-compression-precondition-def
sup-absorb2 by auto
finally have ?p^{T*} \ p^{T*} \ w ≤ p^{T*} \ w
using 11 by (metis dual-order.trans star.circ-loop-fixpoint sup-commute
sup-ge2 mult-assoc)
hence 22: ?t ⊆ ?s
using order-lesseq-imp mult-assoc by auto
have 23: w ∈ ?s
using 1 bijective-regular path-compression-invariant-def eq-iff
star.circ-loop-fixpoint by auto
have 24: ¬ w ∈ ?t
proof
assume w ∈ ?t
hence 25: w ≤ (?p^{T} \ - 1)^{T} \ (p[[w]])
using reachable-without-loops by auto
hence p[[w]] ≤ (?p \ - 1)^{T} \ w
using 1 2 by (metis (no-types, hide-lams) bijective-reverse
cone-star-commute reachable-without-loops path-compression-invariant-def)
also have ... ≤ p^{T*} \ w
proof
have p^{T*} \ y = y
using 1 path-compression-invariant-def
path-compression-precondition-def root-transitive-successor-loop by fastforce
hence y^{T} \ p^{*} \ w = y^{T} \ p^{T}
by (metis conv-dist-comp conv-involutive conv-star-commute)
also have ... = bot
using 1 5 by (metis (no-types, hide-lams) conv-dist-comp conv-dist-inf
equivalence-top-closed inf-top.right-neutral Schroeder-2 symmetric-bot-closed
path-compression-invariant-def)
finally have 26: y^{T} \ p^{*} \ w = bot
by simp
have (?p \ - 1)^{T} \ p^{*} \ w = (w \ y^{T} \ - 1)^{T} \ p^{*} \ w \ p^{T} \ ?w \ (w \ p \ - 1)^{T}
* p^{*} \ w
by (simp add: comp-inf.mult-right-dist-sup mult-right-dist-sup)
also have ... ≤ (w \ y^{T} \ - 1)^{T} \ p^{*} \ w \ p^{*} \ w
by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-right-isotone)
also have ... ≤ (w \ y^{T} \ - 1)^{T} \ p^{*} \ w \ p^{*} \ w
using mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast
also have ... ≤ y^{T} \ p^{*} \ w \ p^{*} \ w
by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-left-isotone)
also have ... = p^{*} \ w
using 26 by simp 

finally show \texttt{thesis} 
by (metis comp-associative le-supI star.circ-loop-fixpoint sup-ge2 star-left-induct)

qed 

finally have \( w \leq p^T \star p^T \star w \)
using 11 25 reachable-without-loops star-plus by auto 

thus False 
using 1 7 by (metis inf.le-iff-sup le-bot pseudo-complement schroeder-4-p semiring.mult-zero-right star.circ-plus-same path-compression-invariant-def)

qed 

have \( \text{card } ?t < \text{card } ?s \)
apply (rule psubset-card-mono)

subgoal using finite-regular by simp 
subgoal using 22 23 24 by auto

done 
thus \texttt{thesis} 
using 1 by simp

qed 

lemma \texttt{path-compression-3}: 
path-compression-invariant \( p \ x \ y \ p0 \ w \wedge y = p[[w]] \implies \)

path-compression-postcondition \( p \ x \ (p[[w]]) \ p0 \)
using path-compression-invariant-def path-compression-postcondition-def path-compression-precondition-def by auto

theorem \texttt{path-compression}: 
VARS \( p \ t \ w \)

\[ \text{path-compression-precondition } p \ x \ y \wedge p0 \ = \ p \]

\( w := x; \)

WHILE \( y \neq p[[w]] \)

INV \{ \text{path-compression-invariant } p \ x \ y \ p0 \ w \}

VAR \{ \text{card } \{ z . \text{regular } z \wedge z \leq p^T \star w \} \}

DO \( t := w; \)

\( w := p[[w]]; \)

\( p[t] := y \)

OD 

\[ \text{path-compression-postcondition } p \ x \ y \ p0 \]

apply vcg-tc-simp 
apply (fact path-compression-1)
apply (fact path-compression-2)
by (fact path-compression-3)

lemma \texttt{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 \( p \) \( x \) \( y \) \( \equiv \) (SOME \( p' \). path-compression-postcondition \( p' \) \( x \) \( y \) \( p \))

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

3.4 Find-set with Path Compression

theorem find-set-path-compression:
  VARS \( p \) \( y \)
  [\ find-set-precondition \( p \) \( x \) \( \land \) \( p0 = p \) \] 
  \( y := \) 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 \( p \) \( x \) \( \land \) \( p0 = p \) \]
  \( y := \) find-set \( p \) \( x \);
  \( w := x \);
  WHILE \( y \neq p[[w]] \]
  INV \{ path-compression-invariant \( p \) \( x \) \( y \) \( p0 \) \( w \) \}
  VAR \{ card \{ z . regular \( z \) \( \land \) \( z \leq p^\tau \ast w \) \} \}
  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 (fact path-compression-2)
by (fact path-compression-3)

theorem find-set-path-compression-2:
  VARS \( p \) \( y \)
  [\ find-set-precondition \( p \) \( x \) \( \land \) \( p0 = p \) \]
  \( y := x \);
  WHILE \( y \neq p[[y]] \]
  INV \{ find-set-invariant \( p \) \( x \) \( y \) \( p0 = p \) \}
  VAR \{ card \{ z . regular \( z \) \( \land \) \( z \leq p^\tau \ast y \) \} \}
  DO \( y := p[[y]] \)
\[ \text{OD; } \]
\[ p := \text{path-compression} \quad x \quad y \]
\[ \text{[ path-compression-postcondition} \quad x \quad y \quad p0 \text{ ]} \]
\text{apply vcg-tc-simp}
\text{apply (simp add: find-set-1)}
\text{using find-set-2 apply blast}
\text{by (smt find-set-3 find-set-invariant-def find-set-postcondition-def find-set-precondition-def path-compression-function path-compression-precondition-def)}

\text{theorem find-set-path-compression-3:}
\text{VARS} \; p \; t \; w \; y
\text{[ find-set-precondition} \quad p \; x \quad p0 \; = \; p \text{ ]}
\text{y} := \; x;
\text{WHILE} \; y \; \neq \; p[[y]]
\text{INV} \; \{ \text{find-set-invariant} \quad p \; x \; y \; p0 \; = \; p \}
\text{VAR} \; \{ \text{card} \; \{ z \; . \; \text{regular} \quad z \; \leq \; p^{\star} \quad \ast \quad y \; \} \}
\text{DO} \; y := \; p[[y]]
\text{OD;}
\text{w} := \; x;
\text{WHILE} \; y \; \neq \; p[[w]]
\text{INV} \; \{ \text{path-compression-invariant} \quad p \; x \; y \; p0 \; w \}
\text{VAR} \; \{ \text{card} \; \{ z \; . \; \text{regular} \quad z \; \leq \; p^{\star} \quad \ast \quad w \; \} \}
\text{DO} \; t := \; w;
\text{w} := \; p[[w]];
\text{p}[t] := \; y
\text{OD}
\text{[ path-compression-postcondition} \quad x \; y \; p0 \; ]
\text{apply vcg-tc-simp}
\text{apply (simp add: find-set-1)}
\text{using find-set-2 apply blast}
\text{using find-set-3 find-set-invariant-def find-set-postcondition-def find-set-precondition-def path-compression-invariant-def path-compression-precondition-def apply blast}
\text{apply (fact path-compression-2)}
\text{by (fact path-compression-3)}

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

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

\text{lemma find-set-path-compression-function:}
\text{assumes find-set-precondition} \; p \; x
\text{and} \; (p',y) = \text{find-set-path-compression} \; p \; x
\text{shows path-compression-postcondition} \; p' \; x \; y \; p
\text{proof} \; –
let \( P = \lambda (p', y). \) 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

3.5 Union-sets

definition union-sets-precondition \( p x y \equiv \) disjoint-set-forest \( p \) \( \land \) point \( x \) \( \land \) point \( y \)
definition union-sets-postcondition \( p x y p0 \equiv \) union-sets-precondition \( p x y \) \( \land \) \( fc p = wcc (p0 \sqcup x \ast y^2) \)

theorem union-sets:
  VARS \( p r s t \)
  \( [ \) union-sets-precondition \( p x y \land p0 = p \) \( ] \)
  \( t \) := find-set-path-compression \( p x \);\n  \( p := \) fst \( t \);\n  \( r := \) snd \( t \);\n  \( t \) := find-set-path-compression \( p y \);\n  \( p := \) fst \( t \);\n  \( s := \) snd \( t \);\n  \( p[r] := s \)
  \( [ \) union-sets-postcondition \( p x y p0 \) \( ] \)
proof vcg-te-simp
  fix \( p \)
  let \( ?t1 = \) find-set-path-compression \( p x \)
  let \( ?p1 = \) fst \( ?t1 \)
  let \( ?r = \) snd \( ?t1 \)
  let \( ?t2 = \) find-set-path-compression \( ?p1 y \)
  let \( ?p2 = \) fst \( ?t2 \)
  let \( ?s = \) snd \( ?t2 \)
  let \( ?p = \) \( ?p2[?r\mapsto?s]\)
assume 1: union-sets-precondition \( p x y \land p0 = p \)
show union-sets-postcondition \( ?p x y p \)
proof (unfold union-sets-postcondition-def union-sets-precondition-def, intro conjI)
  have path-compression-postcondition \( ?p1 x ?r p \)
    using 1 by (simp add: find-set-precondition-def union-sets-precondition-def find-set-path-compression-function)
  hence 2: disjoint-set-forest \( ?p1 \land \) point \( ?r \land ?r = \) root \( ?p1 x \land \) \( ?p1 \sqcap 1 = p \)
    \( \sqcap 1 \land \) \( fc \) \( ?p1 = fc p \)
    using path-compression-precondition-def path-compression-postcondition-def
by auto
  hence path-compression-postcondition \( ?p2 y ?s ?p1 \)
    using 1 by (simp add: find-set-precondition-def union-sets-precondition-def find-set-path-compression-function)
hence 3: disjoint-set-forest \( ?p2 \land \text{point} ?s \land ?s = \text{root} ?p2 y \land ?p2 \cap 1 = ?p1 \cap 1 \land fc ?p2 = fc ?p1 \)

using path-compression-precondition-def path-compression-postcondition-def

by auto

hence 4: fc ?p2 = fc p

using 2 by simp

show 5: univalent ?p

using 2 3 update-univalent by blast

show total ?p

using 2 3 bijective-regular update-total by blast

show acyclic (?p \cap -1)

proof (cases ?r = ?s)

case True

thus \(?\text{thesis}\)

using 3 update-acyclic-3 by fastforce

next

case False

hence bot = ?r \cap ?s

using 2 3 distinct-points by blast

also have ... = ?r \cap \?p2T* \* ?s

using 3 root-transitive-successor-loop by force

finally have ?s \cap \?p2* \* ?r = bot

using Schroeder-1 conv-star-commute inf.sup-monoid.add-commute by

fastforce

thus \(?\text{thesis}\)

using 2 3 update-acyclic-2 by blast

qed

show vector x

using 1 by (simp add: union-sets-precondition-def)

show injective x

using 1 by (simp add: union-sets-precondition-def)

show surjective x

using 1 by (simp add: union-sets-precondition-def)

show vector y

using 1 by (simp add: union-sets-precondition-def)

show injective y

using 1 by (simp add: union-sets-precondition-def)

show surjective y

using 1 by (simp add: union-sets-precondition-def)

show fc ?p = wcc (p \sqcup x * y^T)

proof (rule antisym)

have \(?r = ?pI[?r]\)

using 2 root-successor-loop by force

hence \(?r \ast ?r^T \leq ?pI^T\)

using 2 eq-refl shunt-bijective by blast

hence \(?r \ast ?r^T \leq ?pI\)

using 2 conv-order coreflexive-symmetric by fastforce

hence \(?r \ast ?r^T \leq ?pI \cap 1\)

using 2 inf.boundedI by blast
also have \( \ast \leq \top \cap \bot \)
using 3 by simp
finally have \( \bot \ast \ast \leq \top \)
by simp
hence \( \bot \leq \top \ast \ast \)
using 2 shunt-bijective by blast
hence 6: \( \top \leq \bot \ast \ast \)
by (simp add: mult-assoc)
also have \( \ast \leq \bot \ast \ast \)
by (simp add: conv-order mult-right-isotone)
proof –
also have \( \ast \leq \top \leq \bot \ast \ast \)
proof
finally have \( \ast \leq \top \)
by force
have \( \bot \leq \bot \ast \ast \ast \)
using 2 by simp
hence 10: \( \bot \ast \ast \ast \leq \bot \ast \ast \ast \)
using 1 shunt-bijective union-sets-precondition-def by blast
hence \( x \ast \ast \ast \leq \bot \ast \ast \ast \)
using conv-dist-comp conv-order conv-star-commute by force
also have \( \ast \leq \top \)
by (simp add: star.circ-sub-dist)
also have \( \ast \leq \top \)
using 2 3 by (simp add: fc-wcc)
q.e.d.
also have ... ≤ wcc ?p
   using 8 by simp
finally have 11: x * ?rT ≤ wcc ?p
   by simp
have 12: ?r * ?sT ≤ wcc ?p
   using 2 3 star, circ-sub-dist-1 sup-assoc vector-covector by auto
have ?s ≤ ?p2T * * y
   using 3 by simp
hence 13: ?s * yT ≤ ?p2T *
   using 1 shunt-bijective union-sets-precondition-def by blast
also have ... ≤ wcc ?p2
   using star-isotone sup-ge2 by blast
also have ... ≤ wcc ?p
   using 8 by simp
finally have 14: ?s * yT ≤ wcc ?p
   by simp
have x ≤ x * ?rT * ?r ∧ y ≤ y * ?sT * ?s
   using 2 3 shunt-bijective by blast
hence x * yT ≤ x * ?rT * ?r * (y * ?sT * ?s)T
   using comp-isotone conv-isotone by blast
also have ... = x * ?rT * ?r * ?sT * ?s * yT
   by (simp add: comp-associative conv-dist-comp)
also have ... ≤ wcc ?p * (?r * ?sT) * (?s * yT)
   using 11 by (metis mult-left-isotone mult-assoc)
also have ... ≤ wcc ?p * wcc ?p * (?s * yT)
   using 12 by (metis mult-left-isotone mult-right-isotone)
also have ... ≤ wcc ?p * wcc ?p * wcc ?p
   using 14 by (metis mult-right-isotone)
also have ... = wcc ?p
   by (simp add: star, circ-transitive-equal)
finally have p ⊔ x * yT ≤ wcc ?p
   using 9 by simp
hence wcc (p ⊔ x * yT) ≤ wcc ?p
   using wcc-below-wcc by simp
thus wcc (p ⊔ x * yT) ≤ fc ?p
   using 5 fc-wcc by simp
have − ?r ⊔ ?p2 ≤ wcc ?p2
   by (simp add: inf, coboundedII2 star, circ-sub-dist-1)
also have ... = wcc p
   using 4 by (simp add: star-decompose-1)
also have ... ≤ wcc (p ⊔ x * yT)
   by (simp add: wcc-isotone)
finally have 15: − ?r ⊔ ?p2 ≤ wcc (p ⊔ x * yT)
   by simp
have ?r * xT ≤ wcc ?p1
   using 10 inf, order-trans star, circ-sub-dist sup-commute by fastforce
also have ... = wcc p
   using 2 by (simp add: star-decompose-1)
also have ... ≤ wcc (p ⊔ x * yT)
by (simp add: wcc-isotone)
finally have 16: ?r * x^T ≤ wcc (p ⊔ x * y^T)
  by simp
have 17: x * y^T ≤ wcc (p ⊔ x * y^T)
  using le-supE star.circ-sub-dist-1 by blast
have y * ?s^T ≤ ?p2^*
  using 13 conv-dist-comp conv-order conv-star-commute by fastforce
also have ... ≤ wcc ?p2
  using star.circ-sub-dist sap-commute by fastforce
also have ... = wcc p
  using 4 by (simp add: star-decompose-1)
also have ... ≤ wcc (p ⊔ x * y^T)
  by (simp add: wcc-isotone)
finally have 18: y * ?s^T ≤ wcc (p ⊔ x * y^T)
  by simp
have ?r ≤ ?r * x^T * x ∧ ?s ≤ ?s * y^T * y
  using 1 shunt-bijective union-sets-precondition-def by blast
hence ?r * ?s^T ≤ ?r * x^T * x * (?s * y^T * y)^T
  using comp-isotone conv-isotone by blast
also have ... = ?r * x^T * x * y^T * y * ?s^T
  by (simp add: comp-associative conv-dist-comp)
also have ... ≤ wcc (p ⊔ x * y^T) * (x * y^T) * (y * ?s^T)
  using 16 by (metis mult-left-isotone mult-assoc)
also have ... ≤ wcc (p ⊔ x * y^T) * wcc (p ⊔ x * y^T) * (y * ?s^T)
  using 17 by (metis mult-left-isotone mult-right-isotone)
also have ... ≤ wcc (p ⊔ x * y^T) * wcc (p ⊔ x * y^T) * wcc (p ⊔ x * y^T)
  using 18 by (metis mult-right-isotone)
also have ... = wcc (p ⊔ x * y^T)
  by (simp add: star.circ-transitive-equal)
finally have ?p ≤ wcc (p ⊔ x * y^T)
  using 2 3 15 vector-covector by auto
hence wcc ?p ≤ wcc (p ⊔ x * y^T)
  using wcc-below-wcc by blast
thus fc ?p ≤ wcc (p ⊔ x * y^T)
  using 5 fc-wcc by simp
qed
qed
qed

lemma union-sets-exists:
union-sets-precondition p x y =⇒ ∃ p', union-sets-postcondition p' x y p
using tc-extract-function union-sets by blast

definition union-sets p x y ≡ (SOME p'. union-sets-postcondition p' x y p)

lemma union-sets-function:
assumes union-sets-precondition p x y
  and p' = union-sets p x y
shows union-sets-postcondition p' x y p

35
by (metis assms union-sets-def union-sets-exists someI)

end

end