Relational Disjoint-Set Forests

Walter Guttmann

August 26, 2020

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 with a naive union operation and a find operation with path compression.

Contents

1 Overview ........................................... 1

2 Relation-Algebraic Semantics of Associative Array Access ........................................... 3

3 Relation-Algebraic Semantics of Disjoint-Set Forests ........................................... 7

4 Verifying Operations on Disjoint-Set Forests ........................................... 17
   4.1 Make-Set ........................................... 17
   4.2 Find-Set ........................................... 18
   4.3 Path Compression ........................................... 23
   4.4 Find-Set with Path Compression ........................................... 33
   4.5 Union-Sets ........................................... 35

1 Overview

Relation algebras and Kleene algebras have previously been used to reason about graphs and graph algorithms [2, 3, 4, 5, 9, 12, 15]. 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 [13, 17]. We implement and verify the correctness of disjoint-set forests with path compression and naive union [6, 8, 16].

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, find-set and union-sets operations are implemented and verified in Section 4.

This Isabelle/HOL theory formally verifies results in [11]. Theorem numbers from this paper are mentioned in the theory 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 [14]. Improved automation of this proof is considered in Archive of Formal Proofs entry [18]. These approaches are based on logical specifications whereas the present theory uses relation algebras and Kleene algebras.

theory Disjoint-Set-Forests

imports
  Aggregation-Algebras.Hoare-Logic

begin

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

context stone-relation-algebra

begin

  We start with a few basic properties of arcs, points and rectangles.

  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].

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

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

lemma injective-codomain:
assumes injective $x$
shows $x \ast (x \cap 1) = x \cap 1$
proof (rule antisym)
  show $x \ast (x \cap 1) \leq x \cap 1$
    by (metis assms comp-right-one dual-order.trans inf.bounded1 inf.cobounded1
      inf.sup-monoid.add-commute mult-right-isotone one-inf-conv)
  next
  show $x \cap 1 \leq x \ast (x \cap 1)$
    by (metis coreflexive-idempotent inf.ca.longer
      cobounded1 inf.ca.longer2 mult-left-isotone)
qed

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

lemma arc-rectangle:
arc $x \to rectangle x$
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
where $x[y\leadsto z] \equiv (y \cap z^T) \cup (-y \cap x)$

abbreviation rel-access :: 'a ⇒ 'a ⇒ 'a
where $x[(y)] \equiv x^T \ast y$

Theorem 1.1

lemma update-univalent:
assumes univalent $x$
and vector $y$
and injective $z$
s shows univalent $(x[y\leadsto z])$
proof

have 1: univalent \((y \cap z^T)\)
  using assms(3) inf-commute univalent-inf-closed by force
have \((y \cap z^T)^T \ast (-y \cap x) = (y^T \cap z) \ast (-y \cap x)\)
  by (simp add: conv-dist-inf)
also have \(\ldots = z \ast (y \cap -y \cap x)\)
  by (metis assms(2) covector-inf-comp-3 inf_sup-monoid.add-assoc
inf.sup-monoid.add-commute)
finally have 2: \((y \cap z^T)^T \ast (-y \cap x) = \text{bot}\)
  by simp
have 3: \((x[y\mapsto\rightarrow z]^T \ast (y \cap z^T)\)
  using assms\((1)\) covector-inf-comp by simp
have \((x[y\mapsto\rightarrow z]^T \ast (y \cap z^T)\)
  by (simp add: conv-complement)
also have \(\ldots = (y \cap z^T) \ast (-y \cap x) \ast (y \cap z^T)\)
  by (simp add: mult-left-dist-sup sup-assoc
inf.sup-monoid.add-commute)
finally show \(?\text{thesis}\)
  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 \(z\)
sshows total \((x[y\mapsto\rightarrow z])\)
proof

have \((x[y\mapsto\rightarrow z])^T \ast \text{top} = x^T \ast \text{top}\)
  by (simp add: assms(2) semiring.distrib-right vector-complement-closed
vector-inf-comp conv-dist-comp)
also have \(\ldots = \text{top}\)
  using assms(1) assms(4) by simp
also have \(\ldots = \text{top}\)
  using assms(3) regular-complement-top by auto
finally show \(?\text{thesis}\)
  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→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-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

Theorem 1.8

lemma update-postcondition:
assumes point x point y
shows x ∩ p = x * y^T ←→ 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 antisym)
subgoal by (metis assms conv-dist-comp conv-involutive inf.boundedI
ing.coboundedI vector-covector vector-restrict-comp-conv)
subgoal by (smt assms conv-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:
assumes vector y surjective y vector z
shows (x[y↦z])[y] = z
proof –
have (x[y↦z])[y] = (y ⊓ z) * y ∪ (−yT ∩ xT) * y
by (simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)
also have ... = z * y
proof –
have (−yT ∩ xT) * y = bot
by (metis assms(1) covector-inf-comp-3 inf-commute conv-complement
mult-right-zero p-inf vector-complement-closed)
thus ?thesis
by (simp add: assms covector-inf-comp-3 inf-commute)
qed
also have ... = z
by (metis assms(2,3) mult-assoc)
finally show ?thesis
.
qed

lemma put-put:
(x[y↦z])[y↦w] = x[y↦w]
by (metis inf-absorb2 inf-commute inf-le1 inf-sup-distrib1 maddux-3-13
sup-inf-absorb)

lemma get-put:
assumes point y
shows x[y↦x][y] = x
proof –
have x[y↦x][y] = (y ⊓ yT * x) ∪ (−y ∩ x)
by (simp add: conv-dist-comp)
also have ... = (y ∩ x) ∪ (−y ∩ x)
proof –
have y ∩ yT * x = y ∩ x
proof (rule antisym)
have y ∩ yT * x = (y ∩ yT) * x
by (simp add: assms vector-inf-comp)
also have (y ∩ yT) * x = y * yT * x
by (simp add: assms vector-covector)
also have ... ≤ x
using assms comp-isotone by fastforce
finally show $y \cap y^T \ast x \leq y \cap x$
  by simp
have $y \cap x \leq y^T \ast x$
  by (simp add: assms vector-restrict-comp-conv)
thus $y \cap x \leq y \cap y^T \ast x$
  by simp
qed
thus ?thesis
  by simp
qed

also have 
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

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.

case 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 \( x \star (x \sqcap -1)^* = (x \sqcap 1) \star (x \sqcap -1)^* \sqcup (x \sqcap -1) \star (x \sqcap -1)^* \)
  by (metis maddux-3-11-pp mult-right-dist-sup regular-one-closed)
also have ... \( \leq (x \sqcap -1)^* \)
  by (metis inf.cobounded2 le-supI mult-left-isotone star.circ-circ-mult star.left-plus-below-circ star-involutive star-one)
finally have \( x^* \leq (x \sqcap -1)^* \)
  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 \( p \sqcap x^* = (p \sqcap 1) \sqcup (p \sqcap x) \star (-p \sqcap x)^* \)
proof (rule antisym)
let \( \tilde{p} = p \sqcap 1 \)
have \( \tilde{p} \star x \star (-p \sqcap x)^* \star p \leq \tilde{p} \star \top \star \tilde{p} \)
  by (metis comp-associative mult-left-isotone mult-right-isotone top.extremum)
also have ... \( \leq \tilde{p} \)
  by (simp add: assms injective-codomain vector-inf-one-comp)
finally have \( \tilde{p} \star x \star (-p \sqcap x)^* \star \tilde{p} \star x \leq \tilde{p} \star x \)
  using mult-left-isotone by blast
hence \( \tilde{p} \star x \star (-p \sqcap x)^* \star (p \sqcap x) \leq \tilde{p} \star x \)
  by (simp add: assms comp-associative vector-inf-one-comp)
also have \( 1: ... \leq \tilde{p} \star x \star (-p \sqcap x)^* \)
  using mult-right-isotone star.circ-reflexive by fastforce
finally have \( \tilde{p} \star x \star (-p \sqcap x)^* \star (p \sqcap x) \sqcup \tilde{p} \star x \star (-p \sqcap x)^* \star (p \sqcap x) \)
  \( \leq \tilde{p} \star x \star (-p \sqcap x)^* \)
  by (simp add: mult-right-isotone star.circ-plus-same star.left-plus-below-circ mult-assoc)
  hence \( \tilde{p} \star x \star (-p \sqcap x)^* \star ((p \sqcup -p) \sqcap x) \leq \tilde{p} \star x \star (-p \sqcap x)^* \)
  by (simp add: comp-inf.mult-right-dist-sup mult-left-dist-sup)
  hence \( \tilde{p} \star x \star (-p \sqcap x)^* \star x \leq \tilde{p} \star x \star (-p \sqcap x)^* \)
  by (metis assms bijective-regular inf.absorb2 inf.cobounded1 inf.sup-monoid.add-commute shunting-p)
  hence \( \tilde{p} \star x \star (-p \sqcap x)^* \star x \sqcup \tilde{p} \star x \leq \tilde{p} \star x \star (-p \sqcap x)^* \)
  using 1 by simp
hence \( ?p \ast (I \sqcup x \ast (-p \sqcap x)^*) \ast x \leq ?p \ast x \ast (-p \sqcap x)^* \)
   by (simp add: comp-associative mult-left-dist-sup mult-right-dist-sup)
also have ... \(\leq ?p \ast (I \sqcup x \ast (-p \sqcap x)^*) \)
   by (simp add: comp-associative mult-right-isotone)
finally have \(?p \ast x^* \leq ?p \ast (I \sqcup x \ast (-p \sqcap x)^*) \)
   using star-right-induct by (meson dual-order.trans le-supI
mult-left-sub-dist-sup-left mult-sub-right-one)
also have ... \(\leq ?p \sqcup ?p \ast x \ast (-p \sqcap x)^* \)
   by (simp add: comp-associative semiring.distrib-left)
finally show \(p \sqcap x^* \leq \?p \sqcup (p \sqcap x) \ast (-p \sqcap x)^* \)
   by (simp add: assms vector-inf-one-comp)
show \(?p \sqcup (p \sqcap x) \ast (-p \sqcap x)^* \leq p \sqcap x^* \)
   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

Weakly connected components

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

Theorem 5.1

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 5.2

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 5.3

lemma wcc-below-wcc:
  \(x \leq wcc y \implies wcc x \leq wcc y \)
  using wcc-idempotent wcc-isotone by fastforce

Theorem 5.4

lemma wcc-bot:
\[ \text{wcc} \text{ bot} = 1 \]
\[ \text{by (simp add: star.circ-zero)} \]

**Lemma wcc-one:**
\[ \text{wcc} \ 1 = 1 \]
\[ \text{by (simp add: star-one)} \]

**Theorem 5.5**

**Lemma wcc-top:**
\[ \text{wcc} \ \text{top} = \text{top} \]
\[ \text{by (simp add: star.circ-top)} \]

**Theorem 5.6**

**Lemma wcc-with-loops:**
\[ \text{wcc} \ x = \text{wcc} \ (x \uplus 1) \]
\[ \text{by (metis conv-dist-sup star-decompose-1 star-sup-one sup-commute symmetric-one-closed)} \]

**Lemma wcc-without-loops:**
\[ \text{wcc} \ x = \text{wcc} \ (x \ominus 1) \]
\[ \text{by (metis conv-star-commute star-sum reachable-without-loops)} \]

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

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

**Abbreviation** \[ \text{fc} \ x \equiv x^* \ast x^T^* \]

**Theorem 2.1**

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

**Theorem 2.2**

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

**Theorem 2.3**

**Lemma fc-isotone:**
\[ x \leq y = \Rightarrow \text{fc} \ x \leq \text{fc} \ y \]
\[ \text{by (simp add: comp-isotone conv-isotone star-isotone)} \]
Theorem 2.4

**lemma fc-idempotent:**

univalent \( x \) \( \implies \) \( fc(\text{fc} x) = \text{fc} x \)

by (metis fc-equivalence cancel-separate-1 star.circ-transitive-equal star-involutive)

Theorem 2.5

**lemma fc-star:**

univalent \( x \) \( \implies \) \( (\text{fc} x)^* = \text{fc} x \)

using fc-equivalence fc-idempotent star.circ-transitive-equal by simp

**lemma fc-plus:**

univalent \( x \) \( \implies \) \( (\text{fc} x)^+ = \text{fc} x \)

by (metis fc-star star.circ-decompose-9)

Theorem 2.6

**lemma fc-bot:**

\( \text{fc bot} = 1 \)

by (simp add: star.circ-zero)

**lemma fc-one:**

\( \text{fc 1} = 1 \)

by (simp add: star-one)

Theorem 2.7

**lemma fc-top:**

\( \text{fc top} = \text{top} \)

by (simp add: star.circ-top)

Theorem 5.7

**lemma fc-wcc:**

univalent \( x \) \( \implies \) \( \text{wcc} x = \text{fc} x \)

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

Theorem 4.1

**lemma update-acyclic-1:**

assumes acyclic \( (p \cap -1) \)

and point \( y \)

and point \( w \)

and \( y \leq p^{T*} \ast w \)

shows acyclic \( (\langle p[w\mapsto y]\rangle) \cap -1) \)

**proof** –

let \( \hat{p} = p[w\mapsto y] \)

have \( w \leq p^* \ast y \)

using assms(2-4) by (metis (no-types, lifting) bijective-reverse cone-star-commute)

hence \( w \ast y^T \leq p^* \)

using assms(2) shunt-bijective by blast

hence \( w \ast y^T \leq (p \cap -1)^* \)
using reachable-without-loops by auto
hence $w \ast y^T \cap -1 \leq (p \cap -1)^* \cap -1$
by (simp add: inf.coboundedI2 inf.sup-monoid.add-commute)
also have ... $\leq (p \cap -1)^+$
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 ... $\leq (p \cap -1)^+ \sqcup (-w \cap p \cap -1)$
using 1 sup-left-isotone by blast
also have ... $\leq (p \cap -1)^+$
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 ... $\leq -1$
using assms(1) by blast
finally show ?thesis
by simp
qed

lemma rectangle-star-rectangle:
rectangle $a \Rightarrow a \ast x^* \ast a \leq a$
by (metis mult-left-isotone mult-right-isotone top.extremum)

lemma arc-star-arc:
arc $a \Rightarrow a \ast x^* \ast a \leq a$
using arc-top-arc rectangle-star-rectangle by blast

lemma star-rectangle-decompose:
assumes rectangle $a$
shows $(a \sqcup x)^* = x^* \sqcup x^* \ast a \ast x^*$
proof (rule antisym)
have 1: $1 \leq x^* \sqcup x^* \ast a \ast x^*$
by (simp add: star.circ-reflexive sup.coboundedI1)
have $(a \sqcup x) \ast (x^* \sqcup x^* \ast a \ast x^*) = a \ast x^* \sqcup a \ast x^* \ast a \ast x^* \sqcup x^+ \sqcup x^+ \ast a 
\ast x^*$
by (metis comp-associative semiring.combine-common-factor
semiring.distrib-left sup-commute)
also have ... $= a \ast x^* \sqcup x^+ \sqcup x^+ \ast a \ast x^*$
using assms rectangle-star-rectangle by (simp add: mult-left-isotone
sup-absorb1)
also have ... $= x^+ \sqcup x^* \ast a \ast x^*$
by (metis comp-associative star.circ-loop-fixpoint sup-assoc sup-commute)
also have ... $\leq x^* \sqcup x^* \ast a \ast x^*$
using star.left-plus-below-circ sup-left-isotone by auto
finally show \((a \sqcup x)^* \leq x^* \sqcup x^* \ast a \ast x^*)
using 1 by (metis comp-right-one le-supI star-left-induct)

next

show \(x^* \sqcup x^* \ast a \ast x^* \leq (a \sqcup x)^*\)
by (metis comp-isotone le-supE le-supI star.circ-increasing
star.circ-transitive-equal star-isotone sup-ge2)

qed

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

lemma \(\text{plus-rectangle-decompose}\):
assumes \(\text{rectangle } a\)
shows \((a \sqcup x)^+ = x^+ \sqcup x^* \ast a \ast x^*)\)
proof –

have \((a \sqcup x)^+ = (a \sqcup x) \ast (x^* \sqcup x^* \ast a \ast x^*)\)
by (simp add: assms star-rectangle-decompose)

also have \(... = a \ast x^* \sqcup a \ast x^* \ast a \ast x^* \sqcup x^+ \sqcup x^* \ast a \ast x^*\)
by (metis comp-associative semiring.combine-common-factor
semiring.distrib-left sup-commute)

also have \(... = a \ast x^* \sqcup x^+ \sqcup x^* \ast a \ast x^*\)
using assms rectangle-star-rectangle by (simp add: mult-left-isotone
sup-absorb1)

also have \(... = x^+ \sqcup x^* \ast a \ast x^*\)
by (metis comp-associative star.circ-loop-fixpoint sup-assoc
sup-commute)

finally show \(?\text{thesis}\)
by simp

qed

Theorem 6.1

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

Theorem 6.2

lemma \(\text{update-acyclic-2}\):
assumes \(\text{acyclic } (p \sqcap \neg 1)\)
and \(\text{point } y\)
and \(\text{point } w\)
and \(y \sqcap p^* \ast w = \text{bot}\)
shows \(\text{acyclic } ((p[w\mapsto y]) \sqcap \neg 1)\)
proof –

let \(?p = p[w\mapsto y]\)

have \(y^T \ast p^* \ast w \leq \neg 1\)
using assms(4) comp-associative pseudo-complement schroeder-3-p
by auto

hence \(1: p^* \ast w \ast y^T \ast p^* \leq \neg 1\)
by (metis comp-associative comp-commute-below-diversity
star.circ-transitive-equal)
have \( ?p \cap -1 \leq (w \cap y^T) \cup (p \cap -1) \)
  by (metis comp-inf.mult-right-dist-sup dual-order.trans inf.cobounded1
  inf.cobounded2 inf.sup-monoid.add-assoc le-sup1 sup.cobounded1 sup.ge2)
also have \( \ldots = w \ast y^T \cup (p \cap -1) \)
  using assms(2,3) by (simp add: vector-covector)
finally have \( (\bar{?p} \cap -1)^+ \leq (w \ast y^T \cup (p \cap -1))^+ \)
  by (simp add: comp-isotone star-isotone)
also have \( \ldots = (p \cap -1)^+ \cup (p \cap -1)^+ \ast w \ast y^T \ast (p \cap -1)^+ \)
  using assms(2,3) plus-arc-decompose points-arc by (simp add: comp-associative)
also have \( \ldots \leq (p \cap -1)^+ \cup (p \cap -1)^+ \ast w \ast y^T \ast (p \cap -1)^+ \)
  using reachable-without-loops by auto
also have \( \ldots \leq -1 \)
  using 1 assms by simp
finally show \(?thesis \)
  by simp
qed

lemma acyclic-down-closed:
\( x \leq y \Longrightarrow \text{acyclic } y \Longrightarrow \text{acyclic } x \)
  using comp-isotone star-isotone by fastforce

Theorem 6.3

lemma update-acyclic-3:
assumes \( \text{acyclic } (p \cap -1) \)
  and point \( w \)
shows \( \text{acyclic } ((p[w\mapsto w]) \cap -1) \)
proof –
  let \( ?p = p[w\mapsto w] \)
  have \( ?p \cap -1 \leq (w \cap w^T \cap -1) \cup (p \cap -1) \)
  by (metis comp-inf.mult-right-dist-sup inf.cobounded2
  inf.sup-monoid.add-assoc sup-right-isotone)
also have \( \ldots = p \cap -1 \)
  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 root \( p \) \( x \equiv p^{T^*} \ast x \cap (p \cap 1) \ast \text{top} \)

Theorem 3.1

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

Theorem 3.2
lemma root-successor-loop:
  \textit{univalent }p \implies \text{root }p \ x = p[[\text{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:
  \textit{univalent }p \implies \text{root }p \ x = p^T \ast (\text{root }p \ x)
by (metis mult-1-right star-one star-simulation-right-equal root-successor-loop)
end

context stone-relation-algebra-tarski
begin

Two basic results about points using the Tarski rule of relation algebras

lemma point-in-vector-partition:
  \textit{assumes }point \ x
  \textit{and }vector \ y
  \textit{shows }x \leq -y \lor x \leq --y
proof (cases \ x \ast x \ T \leq -y)
  case True
  have \ x \leq \ x \ast x \ T \ast x
  by (simp add: ex231c)
  also have ... \leq -y \ast \ 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 \ast x \ T \ast x
  by (simp add: ex231c)
  also have ... \leq --y \ast \ 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:
  \textit{assumes }point \ x
  \textit{and }vector \ y
  \textit{and regular }y
  \textit{and }y \leq x

qed
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 \( x \neq y \)
  shows \( x \cap y = \bot \)
  by (metis assms antisym comp-bijective-complement inf.sup-monoid.add-commute mult-left-one pseudo-complement regular-one-closed point-in-vector-partition)

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

lemma put-get-different:
  assumes point y point w w \( \neq y \)
  shows \( (x[y\rightarrow z])[w] = x[w] \)
proof –
  have \( (x[y\rightarrow z])[w] = (y^T \cap z) \cup (-y^T \cap x^T) \cup w \)
    by (simp add: conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)
  also have \( ... = z \cup (w \cap y) \cup x^T \cup (w \cap -y) \)
    by (metis assms inf.q3 conv-complement covector-inf-comp-3 inf-commute vector-complement-closed)
  also have \( ... = x^T \cup w \)
proof –
  have I: \( w \cap y = \bot \)
    using assms distinct-points by simp
  hence \( w \leq -y \)
    using pseudo-complement by simp
  thus ?thesis
    using I by (simp add: inf.absorb1)
qed

finally show ?thesis
.

qed

lemma put-put-different:
  assumes point y point v v \( \neq y \)
  shows \( (x[y\rightarrow z])[v\rightarrow w] = (x[v\rightarrow w])[y\rightarrow z] \)
proof –
  have \( (x[y\rightarrow z])[v\rightarrow w] = (v \cap w^T) \cup (-v \cap y \cap z^T) \cup (-v \cap -y \cap x) \)

16
by (simp add: comp-inf semiring distrib-left inf-assoc sup-assoc)
also have ... = (v ∩ w^T) ∪ (y ∩ z^T) ∪ (−v ∩ −y ∩ x)
  using assms distinct-points pseudo-complement inf.absorb2 by simp
also have ... = (y ∩ z^T) ∪ (v ∩ w^T) ∪ (−y ∩ v ∩ w)
  by (simp add: inf-commute sup-commute)
also have ... = (y ∩ z^T) ∪ (−y ∩ v ∩ w^T) ∪ (−y ∩ −v ∩ x)
  using assms distinct-points pseudo-complement inf.absorb2 by simp
also have ... = (x[v↦→w][y↦→z])
  by (simp add: comp-inf semiring distrib-left inf-assoc sup-assoc)
finally show ?thesis
.
qed

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 ∩ z^T) ∪ (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 }
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

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 \( p \)
\[
\begin{array}{l}
\text{[ point } x \land p0 = p \text{ ]}
\end{array}
\]
\[
\begin{array}{l}
p[x] := x
\end{array}
\]
\[
\begin{array}{l}
\text{[ make-set-postcondition } p x p0 \text{ ]}
\end{array}
\]
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:
\[
\begin{array}{l}
\text{VARS } p
\end{array}
\]
\[
\begin{array}{l}
\text{[ point } x \land p0 = p \land p \leq 1 \text{ ]}
\end{array}
\]
\[
\begin{array}{l}
p[x] := x
\end{array}
\]
\[
\begin{array}{l}
\text{[ make-set-postcondition } p x p0 \land p \leq 1 \text{ ]}
\end{array}
\]
proof vcg-tc
fix \( p \)
assume \( 1: \text{ point } x \land p0 = p \land p \leq 1 \)
show make-set-postcondition \((p[x \mapsto x]) x p0 \land p[x \mapsto x] \leq 1 \)
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

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:
\[
\begin{array}{l}
\text{point } x \Rightarrow \exists p'. \text{ make-set-postcondition } p' x p
\end{array}
\]
using tc-extract-function make-set by blast

definition make-set \( p x \equiv (\text{SOME } p'. \text{ make-set-postcondition } p' x p) \)

**Lemma** make-set-function:
asumes \( \text{point } x \)
and \( p' = \text{make-set } p x \)
sshows make-set-postcondition \( p' x p \)
proof
let \( ?P = \lambda p'. \text{ make-set-postcondition } p' x p \)
have \( ?P (\text{SOME } z . ?P z) \)
using assms(1) make-set-exists by (meson someI)
thus \( \text{thesis} \)
using assms(2) make-set-def by auto
qed

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.

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

**definition** find-set-precondition \( p x \equiv \text{disjoint-set-forest } p \land \text{point } x \)

**definition** find-set-invariant \( p x y \equiv \text{find-set-precondition } p x \land y \leq p^{T^*} x \)

**definition** find-set-postcondition \( p x y \equiv \text{point } y \land y = \text{root } p x \)

**lemma** find-set-1:
\[
\text{find-set-precondition } p x \implies \text{find-set-invariant } p x x
\]
apply (unfold find-set-invariant-def)
using mult-left-isotone star.circ-reflexive find-set-precondition-def by fastforce

**lemma** find-set-2:
\[
\text{find-set-invariant } p x y \land y \neq p[y] \land \text{card } \{z . \text{regular } z \land z \leq p^{T^*} y \} = n \implies \text{find-set-invariant } p x (p[y]) \land \text{card } \{z . \text{regular } z \land z \leq p^{T^*} (p[y]) \} < n
\]

**proof** –
let \( ?s = \{z . \text{regular } z \land z \leq p^{T^*} y \} \)
let \( ?t = \{z . \text{regular } z \land z \leq p^{T^*} (p[y]) \} \)
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 find-set-invariant \( p x (p[y]) \land \text{card } ?t < n \)
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 (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: \( (p^T \sqcap -1) \ast (p^T \sqcap -1)^+ \ast y \leq (p^T \sqcap -1)^+ \ast y \)
by (simp add: mult-left-isotone mult-right-isotone star.left-plus-below-circ)
have \( p[y] = (p^T \sqcap 1) \ast y \sqcup (p^T \sqcap -1) \ast y \)
by (metis maddux-3-11-pp mult-right-dist-sup regular-one-closed)
also have \( \ldots \leq ((p[y]) \sqcap y) \sqcup (p^T \sqcap -1) \ast y \)
by (metis comp-left-subdist-inf mult-1-left semiring.add-right-mono)
also have \( \ldots = (p^T \sqcap -1) \ast y \)
using 1 2 find-set-invariant-def distinct-points by auto
finally have 4: \((p^T \cap -1)^* (p[y]) \leq (p^T \cap -1)^+ y\)
using 3 by (metis inf.antisym-conv inf.eq-refl inf-le1 mult-left-isotone star-plus mult-assoc)
hence \(p^T^* (p[y]) \leq p^T^* y\)
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: \(\neg y \in ?t\)
proof
assume \(y \in ?t\)
hence \(y \leq (p^T \cap -1)^+ y\)
using 4 by (metis reachable-without-loops mem-Collect-eq order-trans)
hence \(y^T y \leq (p^T \cap -1)^+\)
using 1 find-set-invariant-def shunt-bijective by simp
also have \(... \leq -1\)
finally have \(y \leq -y\)
using schroeder-4-p by auto
thus \(\neg\)
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 \(p x y \land y = p[[y]] \Rightarrow\) find-set-postcondition \(p x y\)
proof
assume 1: find-set-invariant \(p x y \land y = p[[y]]\)
show find-set-postcondition \(p x y\)
proof (unfold find-set-postcondition-def, rule conj)
show point \(y\)
using 1 find-set-invariant-def by simp
show y = root p x
proof (rule 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-infl by blast
  hence y ≤ (p ∩ 1) * top
    using 1 by (metis find-set-invariant-def order-lesseq-imp shunt-bijective
    top-right-mult-increasing mult-assoc)
  thus y ≤ root p x
    using 1 find-set-invariant-def by simp
next
  have 2: x ≤ 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 ∪ (p[[y]])
    by (metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint)
  also have ... ≤ p^* * y ∪ 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-mul star.circ-top
    star.circ-transitive-equal star-involutive star-one)
  also have ... = p^* * y
    by (metis star.circ-loop-fixpoint sup.left-idem sup-commute)
  finally have 3: p^T * x ≤ p^* * y
    using 2 by (simp add: comp-associative star-left-induct)
  have p * y ∩ (p ∩ 1) * top = (p ∩ 1) * p * y
    using comp-associative coreflexive-comp-top-inf inf-commute by auto
  also have ... ≤ p^T * p * y
    by (metis inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone
    one-inf-conv)
  also have ... ≤ y
    using 1 find-set-invariant-def find-set-precondition-def mult-left-isotone by simp
  finally have 4: p * y ≤ y ∪ −((p ∩ 1) * top)
    using 1 by (metis find-set-invariant-def shunting-p bijective-regular)
  have p^T * (p ∩ 1) ≤ p^T ∩ 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 * (p ∩ 1) * top ≤ (p ∩ 1) * top
    using inf-commute mult-isotone one-inf-conv by auto
  hence p * −((p ∩ 1) * top) ≤ −((p ∩ 1) * top)
    by (metis comp-associative inf.sup-monoid.add-commute p-antitone
    p-antitone-iff schroeder-3-p)
  hence p * y ∪ p * −((p ∩ 1) * top) ≤ y ∪ −((p ∩ 1) * top)
    using 4 dual-order.trans le-supI sup-ge2 by blast
hence \( p \ast (y \sqcup -((p \cap 1) \ast \text{top})) \leq y \sqcup -((p \cap 1) \ast \text{top}) \)

by \((\text{simp add: \text{mult-left-dist-sup}})\)

hence \( p' \ast y \leq y \sqcup -((p \cap 1) \ast \text{top}) \)

by \((\text{simp add: \text{star-left-induct}})\)

hence \( p'' \ast x \leq y \sqcup -((p \cap 1) \ast \text{top}) \)

using 3 dual-order.trans by blast

thus root \( p x \leq y \)

using 1 by \((\text{metis \text{find-set-invariant-def shunting-p bijective-regular}})\)

qed

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

lemma \text{find-set-function}:
assumes \text{find-set-precondition p x}
and \( y = \text{find-set p x} \)
shows \text{find-set-postcondition p x y}
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.

**definition** path-compression-precondition p x y ≡ disjoint-set-forest p ∧ point x ∧ point y ∧ y = root p x

**definition** path-compression-invariant p x y p0 w ≡

(path-compression-precondition p x y ∧ point w ∧ y ≤ p̂ ⋆ w ∧ (w ≠ x ⇒ p[[x]] = y ∧ x ≠ x ∧ p̂ ⋆ w ≤ ¬x) ∧ p ∩ 1 = p0 ∩ 1 ∧ fc p = fc p0)

**definition** path-compression-postcondition p x y p0 ≡

(path-compression-precondition p x y ∧ p ∩ 1 = p0 ∩ 1 ∧ fc p = fc p0)

**lemma** path-compression-1:

path-compression-precondition p x y ∧ p0 = p =⇒ path-compression-invariant p x y p0 x y p x

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

**lemma** path-compression-2:

path-compression-invariant p x y p0 w ∧ y ≠ p[[w]] ∧ card { z . regular z ∧ z ≤ p̂ ⋆ w } = n

⇒ path-compression-invariant (p[w→y]) x y p0 (p[[w]]) ∧ card { z . regular z ∧ z ≤ (p[w→y])̂ ⋆ (p[[w]]) } < n

**proof**

- let ?p = p[w→y]
- let ?s = { z . regular z ∧ z ≤ p̂ ⋆ w }
- let ?t = { z . regular z ∧ z ≤ ?p̂ ⋆ (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 conj1)

**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 ∩ 1 = bot
using 1 distinct-points path-compression-invariant-def
path-compression-precondition-def by auto
hence $y \ast w^T \leq -1$
using pseudo-complement schroeder-4-p by auto
hence $y \ast w^T \leq p^T \ast \cap -1$
using 1 shunt-bijective path-compression-invariant-def by auto
also have ... $\leq p^T$
by (simp add: star-plus-without-loops)
finally have 6: $y \leq p^T + w$
using 1 shunt-bijective path-compression-invariant-def by blast
have 7: $w \ast w^T \leq -p^T$
proof (rule ccontr)
assume $\neg w \ast w^T \leq -p^T$
hence $w \ast w^T \leq -p^T$
using 1 path-compression-invariant-def point-arc arc-in-partition by blast
hence $w \ast w^T \leq p^T \ast \cap 1$
using 1 path-compression-invariant-def path-compression-precondition-def
mapping-regular regular-conv-closed regular-closed-star regular-mult-closed by simp
also have ...
= $((p^T \cap 1) * p^T \ast \cap 1) \sqcup ((p^T \cap -1) * p^T \ast \cap 1)$
by (metis comp-inf.multiplication-right-dist-sup maddux-3-11-pp
multiplication-right-dist-sup regular-one-closed)
also have ... $\leq ((p^T \cap 1) * p^T \ast \cap 1) \sqcup ((p \cap -1) * p^T \ast \cap 1)$
by (metis conv-complement conv-contr dist-inf conv-plus-commute
equivalence-one-closed reachable-without-loops)
also have ...
= $((p^T \cap 1) * p^T \ast \cap 1) \sqcup ((-1 \cap 1)^T)$
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 ...
= $(p^T \cap 1) * p^T \ast \cap 1$
by simp
also have ...
$\leq (p^T \cap 1) * top \cap 1$
by (metis comp-inf.comp-isotone coreflexive-comp-top-inf
equivalence-one-closed inf.cobounded1 inf.cobounded2)
also have ...
$\leq p^T$
by (simp add: coreflexive-comp-top-inf-one)
finally have $w \ast w^T \leq p^T$
by simp
hence $w \leq p[[w]]$
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 $w = p^T + w$
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 $8: w \cap p^T + w = \bot$
using $p$-antitone-iff pseudo-complement schroeder-4-p by blast
show $y \leq ?pT* * (p[[w]])$
proof

have $(w \cap y^T)^T \leq (w \cap p)^T + p^T * w \leq w^T + (w \cap p)^T * p^T * w$
by (simp add: cone-isotone mult-left-isotone)
also have \ldots \leq w^T \leq p^T * p^T * w$
by (simp add: cone-isotone mult-left-isotone star-isotone mult-right-isotone)
also have \ldots = w^T = p^T * p^T * w$
by (simp add: star-plus mult-assoc)
also have \ldots = \bot

using I 8 by (metis (no-types, hide-lams) path-compression-invariant-def covector-inf-comp-3 mult-assoc cone-dist-comp conv-star-commute covector-bot-closed equivalence-top-closed inf.le-iff-sup mult-left-isotone)
finally have \ldots = \bot

using I 8 by (simp add: star-plus)
also have \ldots = \bot

using 1 by (simp add: star-plus)
finally have 9: $w^T = ?pT * pT * \bot$
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)
have $p^T \leq p^T * pT * w = ((w \cap p)^T \cup (w \cap p)^T) + ?pT * pT * w$
by (simp add: cone-dist-sup)
also have \ldots \leq w^T \leq p^T * pT * w$
using semiring.add-right-mono conv-isotone conv-star-isotone by auto
also have \ldots \leq \bot

using 9 by simp
also have \ldots \leq \bot

by (simp add: cone-isotone mult-left-isotone)
also have \ldots \leq \bot

by (simp add: comp-isotone star.left-plus-below-circ)
finally have $pT * pT * w \leq ?pT * pT * w$

finally have $pT * pT * w \leq \bot$

by (metis comp-associative star.circ-loop-fixpoint star-left-induct
sup-commute sup-least sup-left-divisibility)
thus \( y \leq \hat{\varphi}^{T+} (p[[w]]) \)
  using 6 by (simp add: star-simulation-right-equal mult-assoc)
qed

have 10: acyclic \((\varphi \cap -1)\)
  using 1 update-acyclic-1 path-compression-invariant-def
path-compression-precondition-def by auto

proof
  have \( \{p[[p^{T+} \star w]] \leq p^{T+} \star w \}
  using 6 by (simp add: comp-associative conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup)
  also have \( \cdots \leq y \sqcup (w^T \cap y) * p^{T+} \star w \)
    using sup-left-isotone by auto
  also have \( \cdots \leq y \sqcup p^T * p^{T+} \star w \)
    using mult-left-isotone sup-right-isotone by auto
  also have \( \cdots \leq y \sqcup p^{T+} \star w \)
    using semiring.add-left-mono mult-left-isotone mult-right-isotone

star.left-plus-below-circ by auto
  also have \( \cdots = p^{T+} \star w \)
    using 6 by (simp add: sup-absorb2)
finally show \( \varphi \)thesis
  by simp

qed

hence 11: \( \hat{\varphi}^{T+} (p[[w]]) \leq p^{T+} \star w \)
  using star-left-induct by (simp add: mult-left-isotone
star.circ-mult-increasing)

hence 12: \( \hat{\varphi}^{T+} (p[[w]]) \leq p^{T+} \star w \)
  using dual-order.trans mult-left-isotone star.left-plus-below-circ by blast

have 13: \( \varphi[[x]] = y \land y \neq x \land \hat{\varphi}^{T+} (p[[w]]) \leq -x \)
  proof (cases w = x)
  case True
    hence \( \varphi[[x]] = (w^T \cap y) * w \sqcup (-w^T \cap p^T) * w \)
      by (simp add: conv-complement conv-dist-inf conv-dist-sup
mult-right-dist-sup)
  also have \( \cdots = (w^T \cap y) \star w \sqcup (-w^T \cap p^T) \star w \)
    using 1 by (metis (no-types, lifting) conv-complement
inf.sup-monoid.add-commute path-compression-invariant-def convector-inf-comp-3
vector-complementation-closed)
  also have \( \cdots = y \star w \)
    using 1 inf.sup-monoid.add-commute path-compression-invariant-def
convector-inf-comp-3 by simp
  also have \( \cdots = y \)

26
using 1 by (metis comp-associative path-compression-precondition-def path-compression-invariant-def)
finally show ?thesis
using 4 8 12 True pseudo-complement inf.sup-monoid.add-commute
order.trans by blast
next
case False
have \(?p[[x]] = (w^T \cap y) \sqcup (-w^T \cap p^T) \times x\)
  by (simp add: conv-complement conv-dist-inf conv-dist-sup
mult-right-dist-sup)
also have ... = y \times (w \cap x) \sqcup p^T \times (-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-complementation)
also have ... = p^T \times (-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 \(p[[w]] \neq x \rightarrow ?p[[x]] = y \wedge y \neq x \wedge ?p^{T*} \times (p[[w]]) \leq -x\)
by simp
have 14: \(?p^{T*} \times x = x \sqcup y\)
proof (rule antisym)
  have \(?p^{T*} \times (x \sqcup y) = y \sqcup ?p^{T*} \times y\)
    using 13 by (simp add: mult-left-dist-sup)
  also have ... = y \sqcup (w^T \cap y) \times y \sqcup (-w^T \cap p^T) \times y
    by (simp add: conv-complement conv-dist-inf conv-dist-sup
mult-right-dist-sup sup-assoc)
  also have ... \leq y \sqcup (w^T \cap y) \times y \sqcup p^T \times y
    using mult-left-isotone sup-right-isotone by auto
  also have ... = y \sqcup (w^T \cap y) \times y
    using 1 by (smt sup.cobounded1 sup-absorb1
path-compression-invariant-def path-compression-precondition-def
root-successor-loop)
  also have ... \leq y \sqcup y \times y
    using mult-left-isotone sup-right-isotone by auto
  also have ... = y
    using 1 by (metis mult-semi-associative sup-absorb1
path-compression-invariant-def path-compression-precondition-def)
  also have ... \leq x \sqcup y
    by simp
finally show \(?p^{T*} \times x \leq x \sqcup y\)
    by (simp add: star-left-induct)
next
show \( x \sqcup y \leq \top \) 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 

proof

have \( (p \cap 1) \ast y = (p \cap 1) \ast (p \cap 1) \ast \top \ast x \)

using 1 path-compression-invariant-def path-compression-precondition-def root-var mult-assoc by auto

also have \( \ldots = (p \cap 1) \ast \top \ast x \)

using coreflexive-idempotent by auto

finally have 16: \( (p \cap 1) \ast y = y \)

using 1 path-compression-invariant-def path-compression-precondition-def root-var by auto

have 17: \( (p \cap 1) \ast x \leq y \)

using 1 by (metis (no-types, lifting) comp-right-one mult-left-isotone mult-right-isotone star.circ-reflexive path-compression-invariant-def path-compression-precondition-def root-var)

have \( \text{root} \ ?p \ x = (\ ?p \cap 1) \ast (x \sqcup y) \)

using 14 by (metis mult-assoc root-var)

also have \( \ldots = (w \sqcap y \top \cap 1) \ast (x \sqcup y) \sqcup (w \sqcap p \cap 1) \ast (x \sqcup y) \)

by (simp add: inf-sup-distrib2 semiring.distrib-right)

also have \( \ldots = (w \sqcap 1 \cap y \top) \ast (x \sqcup y) \sqcup (w \sqcap p \cap 1) \ast (x \sqcup y) \)

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

also have \( \ldots = (w \sqcap 1) \ast (y \sqcap (x \sqcup y)) \sqcup (w \sqcap p \sqcap 1) \ast (x \sqcup y) \)

using 1 by (metis (no-types, lifting) path-compression-invariant-def path-compression-precondition-def covector-inf-comp3)

also have \( \ldots = (w \sqcap 1) \ast y \sqcup (w \sqcap (p \sqcap 1) \ast (x \sqcup y)) \)

by (simp add: inf.absorb1)

also have \( \ldots = (w \sqcap 1 \ast y) \sqcup (w \sqcap (p \sqcap 1) \ast (x \sqcup y)) \)

using 1 by (metis (no-types, lifting) inf-associative vector-complement-closed path-compression-invariant-def covector-inf-comp)

also have \( \ldots = (w \sqcap y) \sqcup (w \sqcap ((p \sqcap 1) \ast x \sqcup y)) \)

using 16 by (simp add: mult-left-dist-sup)

also have \( \ldots = (w \sqcap y) \sqcup (w \sqcap y) \)

using 17 by (simp add: sup.absorb2)

also have \( \ldots = y \)

using 1 by (metis id-apply bijective-regular comp-inf.mult-right-dist-sup comp-inf.vector-conv-covector inf-top.right-neutral regular-complement-top path-compression-invariant-def)

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

by simp

qed

show path-compression-precondition \( ?p \ x \ y \)

using 13 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 ∩ p ∩ 1 ≤ w ∩ w^T ∩ p
  by (metis inf.boundedE inf.boundedI inf.coboundedI inf.cobounded2
one-inf-conv)
  also have ... = w * w^T ∩ p
    using 1 vector-covector path-compression-invariant-def by auto
also have ... ≤ −p^T + p
  using 7 by (simp add: inf.coboundedI2 inf.sup-monoid.add-commute)
finally have w ∩ p ∩ 1 = bot
  by (metis (no-types, hide-lams) cone-dist-inf coreflexive-symmetric
inf.absorb1 inf.boundedE inf.cobounded2 pseudo-complement
star.circ-mult-increasing)
  also have w ∩ y^T ∩ 1 = bot
  using 5 antisymmetric-bot-closed asymmetric-bot-closed comp-inf.schroeder-2
inf.absorb1 one-inf-conv by fastforce
  finally have w ∩ p ∩ 1 = w ∩ y^T ∩ 1
  by simp
thus ?p ∩ 1 = p0 ∩ 1
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 * (w ∩ p^* * 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 ∩ p^*) * y
    using 1 vector-inf-comp path-compression-invariant-def mult-assoc by auto
also have ... = p^T * ((w ∩ 1) ⊔ (w ∩ p) * (−w ∩ p)^*) * y
  using 1 omit-redundant-points path-compression-invariant-def by auto
also have ... = p^T * (w ∩ 1) * y ⊔ p^T * (w ∩ p) * (−w ∩ p)^* * y
  by (simp add: comp-associative mult-left-dist-sup mult-right-dist-sup)
also have ... ≤ p^T * y ⊔ p^T * (w ∩ p) * (−w ∩ p)^* * y
  by (metis semiring.add-right-mono comp-isotone eq-iff inf.cobounded1
inf.sup-monoid.add-commute mult-I-right)
  also have ... = y ⊔ p^T * (w ∩ p) * (−w ∩ p)^* * y
    using 1 path-compression-invariant-def path-compression-precondition-def
root-successor-loop by fastforce
  also have ... ≤ y ⊔ p^T * p * (−w ∩ p)^* * y
    using comp-isotone sup-right-isotone by auto
also have ... ≤ y ⊔ (−w ∩ 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 ... = (−w ∩ p)^* * y
  by (metis star.circ-loop-fixpoint sup.left-idem sup-commute)
finally have 18: p[[w]] ≤ (−w ∩ p)^* * y
  by simp
have \( p^T \ast (-w \cap p)^* \ast y = p^T \ast y \sqcup p^T \ast (-w \cap p) \ast (-w \cap p)^* \ast y \)
by (metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint sup-commute)
also have \( \ldots = y \sqcup p^T \ast (-w \cap p) \ast (-w \cap p)^* \ast y \)
using 1 path-compression-invariant-def path-compression-precondition-def root-successor-loop by fastforce
also have \( \ldots \leq y \sqcup (-w \cap p)^* \ast y \)
using comp-isotone sup-right-isotone by auto
also have \( \ldots \leq y \sqcup (-w \cap p)^* \ast y \)
using 1 by (metis (no-types, lifting) mult-left-isotone star star-involutive star-one sup-right-isotone path-compression-invariant-def path-compression-precondition-def)
also have \( \ldots = (-w \cap p)^* \ast y \)
by (metis star.circ-loop-fixpoint sup-commute)
finally have \( 19: p^T \ast (w^T \cap 1) \leq (-w \cap p)^* \ast y \)
by (simp add: mult-left-isotone star-star-left-induct)
also have \( \ldots \leq (w \cap p) \ast (-w \cap p)^T \ast y \)
by (simp add: mult-right-isotone star-star-left-induct)
also have \( \ldots \leq fc \ast p \)
by (simp add: mult-left-isotone star-star-left-induct)
also have \( \ldots \leq fc \ast p \)
by (simp add: mult-left-isotone star-star-left-induct)
finally have \( 20: w \cap p \leq fc \ast p \)
by simp
have \( -w \cap p \leq \ast p \)
by simp
also have \( \ldots \leq fc \ast p \)
by (simp add: fc-increasing)
finally have \( (w \sqcup -w) \cap p \leq fc \ast p \)
using 20 by (simp add: comp-inf.semiring.distrib-left inf.sup-monoid.add-commute)
also have \( \ldots \leq fc \ast p \)
by (simp add: mult-left-isotone star-star-left-induct)
also have \( \ldots \leq fc \ast p \)
by (simp add: fc-increasing)
finally have \( (w \sqcup -w) \cap p \leq fc \ast p \)
using 20 by (simp add: comp-inf.semiring.distrib-left inf.sup-monoid.add-commute)
also have \( \ldots \leq fc \ast p \)
by (simp add: mult-left-isotone star-star-left-induct)
also have \( \ldots \leq fc \ast p \)
by (simp add: fc-increasing)
finally have \( (w \sqcup -w) \cap p \leq fc \ast p \)
using 20 by (simp add: comp-inf.semiring.distrib-left inf.sup-monoid.add-commute)
hence \( p \leq fc \ast p \)
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 \( ?p \leq (w \cap y^T) \cup p \)
using sup-right-isotone by auto
also have \( ... = w \ast y^T \cup p \)
using 1 path-compression-invariant-def path-compression-precondition-def
vector-covector by auto
also have \( ... \leq p^* \cup p \)
using 1 by (metis (no-types, lifting) cone-dist-comp conv-involutive conv-isotone conv-star-commute le-supI 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 \( ?thesis \)
using 1 21 path-compression-invariant-def by simp
qued

show card \(?t < n\)
proof –
have \( \?p^T \ast p^{T*} \ast w = (w^T \cap y) \ast p^{T*} \ast w \cup \(-w^T \cap p^{T}) \ast p^{T*} \ast w \)
by (simp add: cone-complement conv-dist-inf conv-dist-sup
mult-right-dist-sup)
also have \( ... \leq (w^T \cap y) \ast p^{T*} \ast w \cup p^T \ast p^{T*} \ast w \)
using mult-left-isotone sup-right-isotone by auto
also have \( ... \leq (w^T \cap y) \ast p^{T*} \ast w \cup p^T \ast w \)
using mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast
also have \( ... \leq y \ast p^{T*} \ast w \cup p^T \ast w \)
using semiring.add-right-mono mult-left-isotone by auto
also have \( ... \leq y \ast top \cup p^{T*} \ast w \)
by (simp add: comp-associative le-supI1 mult-right-isotone)
also have \( ... = p^{T*} \ast w \)
using 1 path-compression-invariant-def path-compression-precondition-def
sup-absorb2 by auto
finally have \( ?p^{T*} \ast p^T \ast w \leq p^{T*} \ast w \)
using 11 by (metis dual-order.trans star_circ-loop-fixpoint sup-commute
sup-ge2 mult-assoc)

hence 22: \(?t \leq \?s\)
using order-lesseq-imp mult-assoc by auto
have \( \?s \in \?s\)
using 1 bijective-regular path-compression-invariant-def eq-iff
star_circ-loop-fixpoint by auto
have \( \?s \in \?s\)
proof
assume \( w \in \?t \)
hence \( \exists w \leq (\?p^T \cap -1)^* \ast (p[w]) \)
using reachable-without-loops by auto
hence p[[w]] \leq (\exists p \cap -1)^* w
using 1 2 by (metis (no-types, hide-lams) bijective-reverse
conve-star-commute reachable-without-loops path-compression-invariant-def)
also have ... \leq p^* \cdot w
proof -
  have p^T \cdot y = y
  using 1 path-compression-invariant-def
path-compression-precondition-def root-transitive-successor-loop by fastforce
  hence y^T \cdot p^* \cdot w = y^T \cdot w
  by (metis conv-dist-comp conv-involute 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 \cdot p^* \cdot w = bot
by simp
have (\exists p \cap -1)^* \cdot p^* \cdot w \geq (\exists w \cap y^T \cap -1) \cdot p^* \cdot w \sqcup (\neg w \cap p \cap -1) ^* p^* \cdot w
by (simp add: comp-inf mult-right-dist-sup mult-right-dist-sup)
also have ... \leq (\exists w \cap y^T \cap -1) \cdot p^* \cdot w \sqcup p \cdot p^* \cdot w
by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-right-isotone)
also have ... \leq (\exists w \cap y^T \cap -1) \cdot p^* \cdot w \sqcup p^* \cdot w
using mult-left-isotone star.left-plus-below-circ sup-right-isotone by blast
also have ... \leq y^T \cdot p^* \cdot w \sqcup p^* \cdot w
by (meson inf-le1 inf-le2 mult-left-isotone order-trans sup-left-isotone)
also have ... = p^* \cdot w
using 26 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 \cdot p^T \cdot 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 card \?t < card \?s
apply (rule psbset-card-mono)
subgoal using finite-regular by simp
subgoal using 22 23 24 by auto
done
thus \?thesis
using 1 by simp
qed
qed
qed
lemma path-compression-3:
path-compression-invariant \( p \ x \ y \ p0 \ w \land y = p[[w]] \implies \)
path-compression-postcondition \( p \ x \ (p[[w]]) \ p0 \)
using path-compression-invariant-def path-compression-postcondition-def
path-compression-precondition-def by auto

theorem path-compression:
VARS \( p \ t \ w \)
\[
\begin{align*}
& [ \text{path-compression-precondition} \ p \ x \ y \ \land \ p0 = p ] \\
& w := x; \\
& \text{WHILE} \ y \neq p[[w]] \\
& \quad \text{INV} \{ \text{path-compression-invariant} \ p \ x \ y \ p0 \ w \} \\
& \quad \text{VAR} \{ \text{card} \{ z \mid \text{regular} \ z \land z \leq p^* \ast w \} \} \\
& \quad \text{DO} \ t := w; \\
& \quad \hspace{1em} w := p[[w]]; \\
& \quad \hspace{2em} p[t] := y \\
& \quad \text{OD} \\
& [ \text{path-compression-postcondition} \ p \ x \ y \ p0 ]
\end{align*}
\]
apply vcg-tc-simp
apply (fact path-compression-1)
apply (fact path-compression-2)
by (fact path-compression-3)

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 imple-
mentations which use the previously derived functions and implementations
which unfold their definitions.

theorem find-set-path-compression:
VARS \( p \ y \)
\[
\begin{align*}
& [ \text{find-set-precondition} \ p \ x \ \land \ p0 = p ] \\
& y := \text{find-set} \ p \ x; \\
& p := \text{path-compression} \ p \ x \ y \\
& [ \text{path-compression-postcondition} \ p \ x \ y \ p0 ]
\end{align*}
\]
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;
\textbf{WHILE} y \neq p[[w]]
  INV \{ path-compression-invariant p x y p0 w \}
  VAR \{ card \{ z . \ regular z \land z \leq p^T \star w \} \}
  DO t := w;
  \quad w := p[[w]];
  \quad 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 t w y
 [ find-set-precondition p x \land p0 = p ]
y := x;
\textbf{WHILE} y \neq p[[y]]
  INV \{ find-set-invariant p x y \land p0 = p \}
  VAR \{ card \{ z . \ regular z \land z \leq p^T \star y \} \}
  DO y := p[[y]]
  OD;
p := path-compression p x y
 [ path-compression-postcondition p x y p0 ]
apply vcg-tc-simp
apply (simp add: find-set-1)
using find-set-2 apply blast
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 \land p0 = p ]
y := x;
\textbf{WHILE} y \neq p[[y]]
  INV \{ find-set-invariant p x y \land p0 = p \}
  VAR \{ card \{ z . \ regular z \land z \leq p^T \star y \} \}
  DO y := p[[y]]
OD;
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^T * 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)
using find-set-2 apply blast
using find-set-3 find-set-invariant-def find-set-postcondition-def
find-set-precondition-def path-compression-invariant-def
path-compression-postcondition-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 . 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 (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 \( \lambda p' y \cdot \) path-compression-postcondition p' x y p
have \( \lambda p . \) (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

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 ≡ disjoint-set-forest p ∧ point x ∧ point y

definition union-sets-postcondition p x y p0 ≡ union-sets-precondition p x y ∧ fc p = wcc (p0 ⊔ x * y*^T)

theorem union-sets:

VARS p r s t
[ union-sets-precondition p x y ∧ p0 = p ]

let t := find-set-path-compression p x;
p := fst t;
r := snd t;

let t := find-set-path-compression p y;
p := fst t;
s := snd t;
p[r] := s
[ union-sets-postcondition p x y p0 ]

proof vcg-tc-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↦−→?s]

assume 1: union-sets-precondition p x y ∧ p0 = p
show union-sets-postcondition ?p x y p

proof (unfold union-sets-postcondition-def union-sets-precondition-def, intro conj)

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)

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)

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

36
show acyclic (=?p ⊓−1)
proof (cases ?r = ?s)
case True
  thus ?thesis
    using 3 update-acyclic-3 by fastforce
next
case False
  hence bot = ?r ⊓ ?s
    using 2 3 distinct-points by blast
  also have ... = ?r ⊓ ?p2T* ⊓ ?s
    using 3 root-transitive-successor-loop by force
  finally have ?s ⊓ ?p2* ⊓ ?r = bot
    using schroeder-1 conv-star-commute inf.sup-monoid.add-commute by fastforce
  thus ?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 ⊓ x ⊓ y)
proof (rule antisym)
  have ?r = ?p1[[?r]]
    using 2 root-successor-loop by force
  hence ?r * ?rT ≤ ?p1T
    using 2 eq-refl shunt-bijective by blast
  hence ?r * ?rT ≤ ?p1
    using 2 conv-order coreflexive-symmetric by fastforce
  hence ?r * ?rT ≤ ?p1 ⊓ 1
  also have ... = ?p2 ⊓ 1
    using 3 by simp
  finally have ?r * ?rT ≤ ?p2
    by simp
  hence ?r ≤ ?p2 * ?r
    using 2 shunt-bijective by blast
  hence 6: ?p2[[?r]] ≤ ?r
    using 3 shunt-mapping by blast
  have ?r ⊓ ?p2 ≤ ?r * (top ⊓ ?rT * ?p2)
    using 2 by (metis dedekind-1)
also have ... = ?r * ?rT * ?p2
by (simp add: mult-assoc)
also have ... ≤ ?r * ?rT
using 6 by (metis comp-associative conv-dist-comp conv-involutive
cconv-order mult-right-isotone)
also have ...
finally have 7: ?r ∩ ?p2 ≤ 1
by simp
have p ≤ wcc p
by (simp add: star.circ-sub-dist-1)
also have ... = wcc ?p2
using 4 by (simp add: star-decompose-1)
also have 8: ...
by simp
proof –
have wcc ?p2 = wcc ((− ?r ∩ ?p2) ⊔ (?r ∩ ?p2))
using 2 by (metis bijective-regular inf.sup-monoid.add-commute
maddux-3-11-pp)
also have ...
using 7 wcc-isotone sup-right-isotone by simp
also have ...
using wcc-with-loops by simp
also have ...
by simp
finally show ?thesis
by simp
qed
finally have 9: p ≤ wcc ?p
by force
have ?r ≤ ?p1T * x
using 2 by simp
hence 10: ?r * xT ≤ ?p1T
using 1 shunt-bijective union-sets-precondition-def by blast
hence x * ?rT ≤ ?p1
using conv-dist-comp conv-order conv-star-commute by force
also have ...
by (simp add: star.circ-sub-dist)
also have ...
by simp
also have ...
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 ?p

using star-isotone sup-ge2 by blast
also have ... ≤ wcc ?p
using 8 by simp

finally have 14: ?s * y^T ≤ wcc ?p

by simp

have \( x \leq x * ?r^T * ?r \wedge y \leq y * ?s^T * ?s \)
using 2 3 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 ... = x * ?r^T * ?r * ?s^T * ?s * y^T
by (simp add: comp-associative conv-dist-comp)
also have ... ≤ wcc ?p * (?r * ?s^T) * (y * ?s^T)
using 11 by (metis mult-left-isotone mult-assoc)
also have ... ≤ wcc ?p * wcc ?p * (?s * y^T)
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 * y^T ≤ wcc ?p

using 9 by simp

hence wcc (p ⊔ x * y^T) ≤ wcc ?p
using wcc-below-wcc by simp

thus wcc (p ⊔ x * y^T) ≤ fc ?p
using 5 fc-wcc by simp

have −?r ⊓ ?p2 ≤ wcc ?p
by (simp add: inf.coboundedI2 star.circ-sub-dist-1)
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 15: −?r ⊓ ?p2 ≤ wcc (p ⊔ x * y^T)
by simp

have ?r * x^T ≤ wcc ?p
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 * y^T)
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 ?p
using star.circ-sub-dist sup-commute by fastforce
also have ... = wcc p
using \( \star \) by (simp add: star-decompose-1)

also have \( \ldots \leq \text{wcc} (p \sqcup x \ast y^T) \)

by (simp add: wcc-isotone)

finally have 18: \( y \star ?s^T \leq \text{wcc} (p \sqcup x \ast y^T) \)

by simp

have \( ?r \leq ?r \ast x^T \ast x \land ?s \leq ?s \ast y^T \ast y \)

using I shunt-bijective union-sets-precondition-def by blast

hence \( ?r \star ?s^T \leq ?r \ast x^T \ast x \ast (?s \ast y^T \ast y)^T \)

using comp-isotone conv-isotone by blast

also have \( \ldots = ?r \ast x^T \ast x \ast y^T \ast y \ast ?s^T \)

by (simp add: star-circ-transitive-equal)

finally have \( \ldots \leq \text{wcc} (p \sqcup x \ast y^T) \)

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

also have \( \ldots \leq \text{wcc} (p \sqcup x \ast y^T) \ast (x \ast y^T) \ast (y \ast ?s^T) \)

using 16 by (metis mult-left-isotone mult-assoc)

also have \( \ldots \leq \text{wcc} (p \sqcup x \ast y^T) \ast \text{wcc} (p \sqcup x \ast y^T) \ast (y \ast ?s^T) \)

using 17 by (metis mult-left-isotone mult-right-isotone)

also have \( \ldots \leq \text{wcc} (p \sqcup x \ast y^T) \ast \text{wcc} (p \sqcup x \ast y^T) \ast \text{wcc} (p \sqcup x \ast y^T) \)

using 18 by (metis mult-right-isotone)

also have \( \ldots = \text{wcc} (p \sqcup x \ast y^T) \)

by (simp add: star-circ-transitive-equal)

finally have \( ?p \leq \text{wcc} (p \sqcup x \ast y^T) \)

using wcc-below-wcc by blast

thus \( ?p \leq \text{wcc} (p \sqcup x \ast y^T) \)

using 5 fc-wcc by simp

qed

qed

qed

lemma union-sets-exists:

\( \text{union-sets-precondition} \; p \; x \; y \; \Longrightarrow \exists \; p' \). \( \text{union-sets-postcondition} \; p' \; x \; y \; p \)

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 \( \text{union-sets-precondition} \; p \; x \; y \)

and \( p' = \text{union-sets} \; p \; x \; y \)

shows \( \text{union-sets-postcondition} \; p' \; x \; y \; p \)

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

end

end
References


