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

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
  Stone-Kleene-Relation-Algebras.Kleene-Relation-Algebras
begin

no-notation
  trancl $((\text{-}^{+})$ $[1000]$ $999)$

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 $\Longrightarrow$ *point y* $\Longrightarrow$ *arc* $(x * y^T)$
by (*metis comp-associative conv-dist-comp conv-involutive equivalence-top-closed*)

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

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

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

lemma *arc-rectangle*:
arc x $\Longrightarrow$ *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* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ ((-[-$\longmapsto$-]) [*70*, *65*, *65*] *61*)
where $x[y \longmapsto z] \equiv (y \sqcap z^T) \sqcup (-y \sqcap x)$

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

Theorem 1.1

lemma *update-univalent*:
assumes *univalent x*
and *vector y*
and *injective z*
shows *univalent* $(x[y \longmapsto z])$

proof −
have *1*: *univalent* $(y \sqcap z^T)$
using *assms*(*3*) *inf-commute univalent-inf-closed* **by** *force*
have $(y \sqcap z^T)^T * (-y \sqcap x) = (y^T \sqcap z) * (-y \sqcap x)$
by (*simp add*: *conv-dist-inf*)
also have ... $= z * (y \sqcap -y \sqcap x)$
by (*metis assms*(*2*) *covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute*)
finally have *2*: $(y \sqcap z^T)^T * (-y \sqcap x) = bot$
by *simp*
have *3*: *vector* $(-y)$
using *assms*(*2*) *vector-complement-closed* **by** *simp*
have $(-y \sqcap x)^T * (y \sqcap z^T) = (-y^T \sqcap x^T) * (y \sqcap z^T)$
by (*simp add*: *conv-complement conv-dist-inf*)
also have ... $= x^T * (-y \sqcap y \sqcap z^T)$
using *3* **by** (*metis* (*mono-tags*, *hide-lams*) *conv-complement covector-inf-comp-3 inf.sup-monoid.add-assoc inf.sup-monoid.add-commute*)
finally have *4*: $(-y \sqcap x)^T * (y \sqcap z^T) = bot$
by *simp*
have *5*: *univalent* $(-y \sqcap x)$
using *assms*(*1*) *inf-commute univalent-inf-closed* **by** *fastforce*
have $(x[y \longmapsto z])^T * (x[y \longmapsto z]) = (y \sqcap z^T)^T * (x[y \longmapsto z]) \sqcup (-y \sqcap x)^T * (x[y \longmapsto z])$
by (*simp add*: *conv-dist-sup mult-right-dist-sup*)
also have ... $= (y \sqcap z^T)^T * (y \sqcap z^T) \sqcup (y \sqcap z^T)^T * (-y \sqcap x) \sqcup (-y \sqcap x)^T * (y \sqcap z^T) \sqcup (-y \sqcap x)^T * (-y \sqcap x)$
by (*simp add*: *mult-left-dist-sup sup-assoc*)
finally show *?thesis*
using *1 2 4 5* **by** *simp*
qed

Theorem 1.2

lemma *update-total*:
assumes *total x*
and *vector y*
and *regular y*
and *surjective z*
shows *total* $(x[y \longmapsto z])$
proof −
have $(x[y \longmapsto z]) * top = x*top[y \longmapsto top*z]$
by (*simp add*: *assms*(*2*) *semiring.distrib-right vector-complement-closed vector-inf-comp conv-dist-comp*)
also have ... $= top[y \longmapsto top]$
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* $(x[y \longmapsto z])$
  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 $x \sqcap p = x * y^T \longleftrightarrow p[[x]] = y$
  apply (*rule iffI*)
  subgoal by (*metis assms comp-associative conv-dist-comp conv-involutive covector-inf-comp-3 equivalence-top-closed vector-covector*)
  subgoal
    apply (*rule antisym*)
    subgoal by (*metis assms conv-dist-comp conv-involutive inf.boundedI inf.cobounded1 vector-covector vector-restrict-comp-conv*)

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

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

lemma *put-get*:
assumes *vector* y *surjective* y *vector* z
shows $(x[y \longmapsto z])[[y]] = z$
proof –
have $(x[y \longmapsto z])[[y]] = (y^T \sqcap z) * y \sqcup (-y^T \sqcap x^T) * y$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $= z * y$
proof –
have $(-y^T \sqcap x^T) * y = bot$
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*:
$(x[y \longmapsto z])[y \longmapsto w] = x[y \longmapsto w]$
by (*metis inf-absorb2 inf-commute inf-le1 inf-sup-distrib1 maddux-3-13 sup-inf-absorb*)

lemma *get-put*:
assumes *point* y
shows $x[y \longmapsto x[[y]]] = x$
proof –
have $x[y \longmapsto x[[y]]] = (y \sqcap y^T * x) \sqcup (-y \sqcap x)$
by (*simp add*: *conv-dist-comp*)
also have ... $= (y \sqcap x) \sqcup (-y \sqcap x)$
proof –
have $y \sqcap y^T * x = y \sqcap x$
proof (*rule antisym*)
have $y \sqcap y^T * x = (y \sqcap y^T) * x$
by (*simp add*: *assms vector-inf-comp*)
also have $(y \sqcap y^T) * x = y * y^T * x$
by (*simp add*: *assms vector-covector*)
also have ... $\leq x$
using *assms comp-isotone* **by** *fastforce*

finally show $y \sqcap y^T * x \leq y \sqcap x$
by *simp*
have $y \sqcap x \leq y^T * x$
by (*simp add*: *assms vector-restrict-comp-conv*)
thus $y \sqcap x \leq y \sqcap y^T * x$
by *simp*
qed
thus *?thesis*
by *simp*
qed
also have ... $= x$
proof −
have *regular* y
using *assms bijective-regular* **by** *blast*
thus *?thesis*
by (*metis inf.sup-monoid.add-commute maddux-3-11-pp*)
qed
finally show *?thesis*
.
qed

end

3 Relation-Algebraic Semantics of Disjoint-Set Forests

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

context *stone-kleene-relation-algebra*
begin

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

lemma *equivalence-plus-closed*:
equivalence $x \Longrightarrow$ *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*:
$x^\star = (x \sqcap -1)^\star$

proof (*rule antisym*)
have $x * (x \sqcap -1)^\star = (x \sqcap 1) * (x \sqcap -1)^\star \sqcup (x \sqcap -1) * (x \sqcap -1)^\star$
by (*metis maddux-3-11-pp mult-right-dist-sup regular-one-closed*)
also have ... $\leq (x \sqcap -1)^\star$
by (*metis inf.cobounded2 le-supI mult-left-isotone star.circ-circ-mult star.left-plus-below-circ star-involutive star-one*)
finally show $x^\star \leq (x \sqcap -1)^\star$
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 $(x \sqcap -1)^\star \leq x^\star$
by (*simp add*: *star-isotone*)
qed

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

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

Theorem 4.2

lemma *omit-redundant-points*:
assumes *point p*
shows $p \sqcap x^\star = (p \sqcap 1) \sqcup (p \sqcap x) * (-p \sqcap x)^\star$
proof (*rule antisym*)
let $?p = p \sqcap 1$
have $?p * x * (-p \sqcap x)^\star * ?p \leq ?p * top * ?p$
by (*metis comp-associative mult-left-isotone mult-right-isotone top.extremum*)
also have ... $\leq ?p$
by (*simp add*: *assms injective-codomain vector-inf-one-comp*)
finally have $?p * x * (-p \sqcap x)^\star * ?p * x \leq ?p * x$
using *mult-left-isotone* **by** *blast*
hence $?p * x * (-p \sqcap x)^\star * (p \sqcap x) \leq ?p * x$
by (*simp add*: *assms comp-associative vector-inf-one-comp*)
also have *1*: ... $\leq ?p * x * (-p \sqcap x)^\star$
using *mult-right-isotone star.circ-reflexive* **by** *fastforce*
finally have $?p * x * (-p \sqcap x)^\star * (p \sqcap x) \sqcup ?p * x * (-p \sqcap x)^\star * (-p \sqcap x) \leq ?p * x * (-p \sqcap x)^\star$
by (*simp add*: *mult-right-isotone star.circ-plus-same star.left-plus-below-circ mult-assoc*)
hence $?p * x * (-p \sqcap x)^\star * ((p \sqcup -p) \sqcap x) \leq ?p * x * (-p \sqcap x)^\star$
by (*simp add*: *comp-inf.mult-right-dist-sup mult-left-dist-sup*)
hence $?p * x * (-p \sqcap x)^\star * x \leq ?p * x * (-p \sqcap x)^\star$
by (*metis assms bijective-regular inf.absorb2 inf.cobounded1 inf.sup-monoid.add-commute shunting-p*)
hence $?p * x * (-p \sqcap x)^\star * x \sqcup ?p * x \leq ?p * x * (-p \sqcap x)^\star$
using *1* **by** *simp*

hence $?p * (1 \sqcup x * (-p \sqcap x)^\star) * x \le ?p * x * (-p \sqcap x)^\star$
by (*simp add*: *comp-associative mult-left-dist-sup mult-right-dist-sup*)
also have ... $\le ?p * (1 \sqcup x * (-p \sqcap x)^\star)$
by (*simp add*: *comp-associative mult-right-isotone*)
finally have $?p * x^\star \le ?p * (1 \sqcup x * (-p \sqcap x)^\star)$
using *star-right-induct* **by** (*meson dual-order.trans le-supI mult-left-sub-dist-sup-left mult-sub-right-one*)
also have ... $= ?p \sqcup ?p * x * (-p \sqcap x)^\star$
by (*simp add*: *comp-associative semiring.distrib-left*)
finally show $p \sqcap x^\star \le ?p \sqcup (p \sqcap x) * (-p \sqcap x)^\star$
by (*simp add*: *assms vector-inf-one-comp*)
show $?p \sqcup (p \sqcap x) * (-p \sqcap x)^\star \le p \sqcap x^\star$
by (*metis assms comp-isotone inf.boundedI inf.coboundedI1 inf.coboundedI2 inf.sup-monoid.add-commute le-supI star.circ-increasing star.circ-transitive-equal star-isotone star-left-unfold-equal sup.coboundedI1 vector-export-comp*)
qed

Weakly connected components

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

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*:
$x \le wcc\ x$
by (*simp add*: *star.circ-sub-dist-1*)

lemma *wcc-isotone*:
$x \le y \Longrightarrow wcc\ x \le wcc\ y$
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*:
$x \le wcc\ y \Longrightarrow wcc\ x \le wcc\ y$
using *wcc-idempotent wcc-isotone* **by** *fastforce*

Theorem 5.4

lemma *wcc-bot*:

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

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

Theorem 5.5

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

Theorem 5.6

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

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

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

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

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

Theorem 2.1

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

Theorem 2.3

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

Theorem 2.4

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

Theorem 2.5

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

lemma *fc-plus*:
univalent $x \Longrightarrow (fc\ x)^+ = fc\ x$
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 5.7

lemma *fc-wcc*:
univalent $x \Longrightarrow wcc\ x = fc\ x$
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 $y \le p^{T\star} * w$
shows *acyclic* $((p[w \longmapsto y]) \sqcap -1)$
proof –
let $?p = p[w \longmapsto y]$
have $w \le p^\star * y$
using *assms*(2–4) **by** (*metis* (*no-types*, *lifting*) *bijective-reverse conv-star-commute*)
hence $w * y^T \le p^\star$
using *assms*(2) *shunt-bijective* **by** *blast*
hence $w * y^T \le (p \sqcap -1)^\star$

using *reachable-without-loops* **by** *auto*
hence $w * y^T \sqcap -1 \leq (p \sqcap -1)^\star \sqcap -1$
by (*simp add*: *inf.coboundedI2 inf.sup-monoid.add-commute*)
also have ... $\leq (p \sqcap -1)^+$
by (*simp add*: *star-plus-without-loops*)
finally have *1*: $w \sqcap y^T \sqcap -1 \leq (p \sqcap -1)^+$
using *assms*(*2*,*3*) *vector-covector* **by** *auto*
have $?p \sqcap -1 = (w \sqcap y^T \sqcap -1) \sqcup (-w \sqcap p \sqcap -1)$
by (*simp add*: *inf-sup-distrib2*)
also have ... $\leq (p \sqcap -1)^+ \sqcup (-w \sqcap p \sqcap -1)$
using *1 sup-left-isotone* **by** *blast*
also have ... $\leq (p \sqcap -1)^+ \sqcup (p \sqcap -1)$
using *comp-inf.mult-semi-associative sup-right-isotone* **by** *auto*
also have ... $= (p \sqcap -1)^+$
by (*metis star.circ-back-loop-fixpoint sup.right-idem*)
finally have $(?p \sqcap -1)^+ \leq (p \sqcap -1)^+$
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

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

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

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

finally show $(a \sqcup x)^\star \leq x^\star \sqcup x^\star * a * x^\star$
using *1* **by** (*metis comp-right-one le-supI star-left-induct*)
next
show $x^\star \sqcup x^\star * a * x^\star \leq (a \sqcup x)^\star$
by (*metis comp-isotone le-supE le-supI star.circ-increasing star.circ-transitive-equal star-isotone sup-ge2*)
qed

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

lemma *plus-rectangle-decompose*:
assumes *rectangle a*
shows $(a \sqcup x)^+ = x^+ \sqcup x^\star * a * x^\star$
proof –
have $(a \sqcup x)^+ = (a \sqcup x) * (x^\star \sqcup x^\star * a * x^\star)$
by (*simp add*: *assms star-rectangle-decompose*)
also have ... $= a * x^\star \sqcup a * x^\star * a * x^\star \sqcup x^+ \sqcup x^+ * a * x^\star$
by (*metis comp-associative semiring.combine-common-factor semiring.distrib-left sup-commute*)
also have ... $= a * x^\star \sqcup x^+ \sqcup x^+ * a * x^\star$
using *assms rectangle-star-rectangle* **by** (*simp add*: *mult-left-isotone sup-absorb1*)
also have ... $= x^+ \sqcup x^\star * a * x^\star$
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*:
arc a $\Longrightarrow (a \sqcup x)^+ = x^+ \sqcup x^\star * a * x^\star$
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* $((p[w \longmapsto y]) \sqcap -1)$
proof –
let $?p = p[w \longmapsto y]$
have $y^T * p^\star * w \leq -1$
using *assms*(*4*) *comp-associative pseudo-complement schroeder-3-p* **by** *auto*
hence *1*: $p^\star * w * y^T * p^\star \leq -1$
by (*metis comp-associative comp-commute-below-diversity star.circ-transitive-equal*)

```
  have ?p ⊓ −1 ≤ (w ⊓ y^T) ⊔ (p ⊓ −1)
    by (metis comp-inf.mult-right-dist-sup dual-order.trans inf.cobounded1
inf.coboundedI2 inf.sup-monoid.add-assoc le-supI sup.cobounded1 sup-ge2)
  also have ... = w * y^T ⊔ (p ⊓ −1)
    using assms(2,3) by (simp add: vector-covector)
  finally have (?p ⊓ −1)^+ ≤ (w * y^T ⊔ (p ⊓ −1))^+
    by (simp add: comp-isotone star-isotone)
  also have ... = (p ⊓ −1)^+ ⊔ (p ⊓ −1)^⋆ * w * y^T * (p ⊓ −1)^⋆
    using assms(2,3) plus-arc-decompose points-arc by (simp add:
comp-associative)
  also have ... ≤ (p ⊓ −1)^+ ⊔ p^⋆ * w * y^T * p^⋆
    using reachable-without-loops by auto
  also have ... ≤ −1
    using 1 assms(1) by simp
  finally show ?thesis
    by simp
qed

lemma acyclic-down-closed:
  x ≤ y ⟹ acyclic y ⟹ acyclic x
  using comp-isotone star-isotone by fastforce
```

Theorem 6.3

```
lemma update-acyclic-3:
  assumes acyclic (p ⊓ −1)
    and point w
  shows acyclic ((p[w⟼w]) ⊓ −1)
proof −
  let ?p = p[w⟼w]
  have ?p ⊓ −1 ≤ (w ⊓ w^T ⊓ −1) ⊔ (p ⊓ −1)
    by (metis comp-inf.mult-right-dist-sup inf.cobounded2
inf.sup-monoid.add-assoc sup-right-isotone)
  also have ... = p ⊓ −1
    using assms(2) by (metis comp-inf.covector-complement-closed
equivalence-top-closed inf-top.right-neutral maddux-3-13 pseudo-complement
regular-closed-top regular-one-closed vector-covector vector-top-closed)
  finally show ?thesis
    using assms(1) acyclic-down-closed by blast
qed
```

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

```
abbreviation root p x ≡ p^T⋆ * x ⊓ (p ⊓ 1) * top
```

Theorem 3.1

```
lemma root-var:
  root p x = (p ⊓ 1) * p^T⋆ * x
  by (simp add: coreflexive-comp-top-inf inf-commute mult-assoc)
```

Theorem 3.2

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

lemma *root-transitive-successor-loop*:
  univalent p $\Longrightarrow$ *root p x* $= p^{T\star} *$ (*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*:
  assumes *point x*
    and *vector y*
  shows $x \le -y \vee x \le --y$
proof (*cases* $x * x^T \le -y$)
  case *True*
  have $x \le x * x^T * x$
    by (*simp add*: *ex231c*)
  also have ... $\le -y * x$
    by (*simp add*: *True mult-left-isotone*)
  also have ... $\le -y$
    by (*metis assms*(*2*) *mult-right-isotone top.extremum vector-complement-closed*)
  finally show *?thesis*
    by *simp*
next
  case *False*
  have $x \le x * x^T * x$
    by (*simp add*: *ex231c*)
  also have ... $\le --y * x$
    using *False assms*(*1*) *arc-in-partition mult-left-isotone point-arc* **by** *blast*
  also have ... $\le --y$
    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 $y \le x$

shows $y = x \vee y = bot$
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 $x \sqcap y = bot$
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 $(x[y\longmapsto z])[[w]] = x[[w]]$
proof –
have $(x[y\longmapsto z])[[w]] = (y^T \sqcap z) * w \sqcup (-y^T \sqcap x^T) * w$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $= z * (w \sqcap y) \sqcup x^T * (w \sqcap -y)$
by (*metis assms*(*1*) *conv-complement covector-inf-comp-3 inf-commute vector-complement-closed*)
also have ... $= x^T * w$
proof –
have *1*: $w \sqcap y = bot$
using *assms distinct-points* **by** *simp*
hence $w \leq -y$
using *pseudo-complement* **by** *simp*
thus *?thesis*
using *1* **by** (*simp add*: *inf.absorb1*)
qed
finally show *?thesis*
.
qed

lemma *put-put-different*:
assumes *point* y *point* v $v \neq y$
shows $(x[y\longmapsto z])[v\longmapsto w] = (x[v\longmapsto w])[y\longmapsto z]$
proof –
have $(x[y\longmapsto z])[v\longmapsto w] = (v \sqcap w^T) \sqcup (-v \sqcap y \sqcap z^T) \sqcup (-v \sqcap -y \sqcap x)$

by $(simp\ add{:}\ comp\text{-}inf.semiring.distrib\text{-}left\ inf\text{-}assoc\ sup\text{-}assoc)$
also have ... $= (v \sqcap w^T) \sqcup (y \sqcap z^T) \sqcup (-v \sqcap -y \sqcap x)$
using *assms distinct-points pseudo-complement inf.absorb2* **by** *simp*
also have ... $= (y \sqcap z^T) \sqcup (v \sqcap w^T) \sqcup (-y \sqcap -v \sqcap x)$
by $(simp\ add{:}\ inf\text{-}commute\ sup\text{-}commute)$
also have ... $= (y \sqcap z^T) \sqcup (-y \sqcap v \sqcap w^T) \sqcup (-y \sqcap -v \sqcap x)$
using *assms distinct-points pseudo-complement inf.absorb2* **by** *simp*
also have ... $= (x[v \longmapsto w])[y \longmapsto z]$
by $(simp\ add{:}\ comp\text{-}inf.semiring.distrib\text{-}left\ inf\text{-}assoc\ sup\text{-}assoc)$
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 :: $idt \Rightarrow {'a} \Rightarrow {'a} \Rightarrow {'b}\ com$ ((2-[-] :=/ -) [70, 65, 65] 61)

translations
$x[y] := z$ => $(x := (y \sqcap z^T) \sqcup (CONST\ uminus\ y \sqcap x))$

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* $p\ x\ p0 \equiv x \sqcap p = x * x^T \wedge -x \sqcap p = -x \sqcap p0$

theorem *make-set*:

VARS p
[*point x* ∧ *p0* = *p*]
p[*x*] := *x*
[*make-set-postcondition p x p0*]
apply *vcg-tc-simp*
by (*simp add*: *make-set-postcondition-def inf-sup-distrib1 inf-assoc*[*THEN sym*] *vector-covector*[*THEN sym*])

theorem *make-set-2*:
VARS p
[*point x* ∧ *p0* = *p* ∧ *p* ≤ *1*]
p[*x*] := *x*
[*make-set-postcondition p x p0* ∧ *p* ≤ *1*]
proof *vcg-tc*
fix *p*
assume *1*: *point x* ∧ *p0* = *p* ∧ *p* ≤ *1*
show *make-set-postcondition* (*p*[*x*⟼*x*]) *x p0* ∧ *p*[*x*⟼*x*] ≤ *1*
proof (*rule conjI*)
show *make-set-postcondition* (*p*[*x*⟼*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*⟼*x*] ≤ *1*
using *1* **by** (*metis coreflexive-sup-closed dual-order.trans inf.cobounded2 vector-covector*)
qed
qed

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

lemma *make-set-exists*:
point x ⟹ ∃ *p′* . *make-set-postcondition p′ x p*
using *tc-extract-function make-set* **by** *blast*

definition *make-set p x* ≡ (*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* = λ*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

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* $\equiv$ *mapping p* $\wedge$ *acyclic* $(p \sqcap -1)$

definition *find-set-precondition p x* $\equiv$ *disjoint-set-forest p* $\wedge$ *point x*
definition *find-set-invariant p x y* $\equiv$ *find-set-precondition p x* $\wedge$ *point y* $\wedge$ $y \leq p^{T\star} * x$
definition *find-set-postcondition p x y* $\equiv$ *point y* $\wedge$ $y =$ *root p x*

lemma *find-set-1*:
  find-set-precondition p x $\Longrightarrow$ *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 $\wedge$ $y \neq p[[y]]$ $\wedge$ *card* $\{\ z\ .\ \mathit{regular}\ z \wedge z \leq p^{T\star} * y\ \} = n$
$\Longrightarrow$ *find-set-invariant p x* $(p[[y]])$ $\wedge$ *card* $\{\ z\ .\ \mathit{regular}\ z \wedge z \leq p^{T\star} * (p[[y]])\ \} < n$
proof –
  let *?s* $= \{\ z\ .\ \mathit{regular}\ z \wedge z \leq p^{T\star} * y\ \}$
  let *?t* $= \{\ z\ .\ \mathit{regular}\ z \wedge z \leq p^{T\star} * (p[[y]])\ \}$
  assume *1*: *find-set-invariant p x y* $\wedge$ $y \neq p[[y]]$ $\wedge$ *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* $(p[[y]])$ $\wedge$ *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 $p[[y]] \leq p^{T\star} * x$
      using *1* **by** (*metis* (*hide-lams*) *find-set-invariant-def comp-associative comp-isotone star.circ-increasing star.circ-transitive-equal*)
    show *card ?t* $< n$
    proof –
      have *3*: $(p^T \sqcap -1) * (p^T \sqcap -1)^+ * y \leq (p^T \sqcap -1)^+ * y$
        by (*simp add*: *mult-left-isotone mult-right-isotone star.left-plus-below-circ*)
      have $p[[y]] = (p^T \sqcap 1) * y \sqcup (p^T \sqcap -1) * y$
        by (*metis maddux-3-11-pp mult-right-dist-sup regular-one-closed*)
      also have ... $\leq ((p[[y]]) \sqcap y) \sqcup (p^T \sqcap -1) * y$
        by (*metis comp-left-subdist-inf mult-1-left semiring.add-right-mono*)
      also have ... $= (p^T \sqcap -1) * y$
        using *1 2 find-set-invariant-def distinct-points* **by** *auto*

finally have *4*: $(p^T \sqcap -1)^\star * (p[[y]]) \leq (p^T \sqcap -1)^+ * y$
using *3* **by** (*metis inf.antisym-conv inf.eq-refl inf-le1 mult-left-isotone star-plus mult-assoc*)
hence $p^{T\star} * (p[[y]]) \leq p^{T\star} * y$
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 $y \leq (p^T \sqcap -1)^+ * y$
using *4* **by** (*metis reachable-without-loops mem-Collect-eq order-trans*)
hence $y * y^T \leq (p^T \sqcap -1)^+$
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 $y \leq -y$
using *schroeder-4-p* **by** *auto*
thus *False*
using *1* **by** (*metis find-set-invariant-def comp-inf.coreflexive-idempotent conv-complement covector-vector-comp inf.absorb1 inf.sup-monoid.add-commute pseudo-complement surjective-conv-total top.extremum vector-top-closed regular-closed-top*)
qed
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
qed

lemma *find-set-3*:
find-set-invariant p x y $\wedge$ $y = p[[y]] \Longrightarrow$ *find-set-postcondition p x y*
proof −
assume *1*: *find-set-invariant p x y* $\wedge$ $y = p[[y]]$
show *find-set-postcondition p x y*
proof (*unfold find-set-postcondition-def*, *rule conjI*)
show *point y*

using *1 find-set-invariant-def* **by** *simp*
show $y = root\ p\ x$
proof (*rule antisym*)
have $y * y^T \leq p$
using *1* **by** (*metis find-set-invariant-def find-set-precondition-def shunt-bijective shunt-mapping top-right-mult-increasing*)
hence $y * y^T \leq p \sqcap 1$
using *1 find-set-invariant-def le-infI* **by** *blast*
hence $y \leq (p \sqcap 1) * top$
using *1* **by** (*metis find-set-invariant-def order-lesseq-imp shunt-bijective top-right-mult-increasing mult-assoc*)
thus $y \leq root\ p\ x$
using *1 find-set-invariant-def* **by** *simp*
next
have *2*: $x \leq p^\star * y$
using *1 find-set-invariant-def find-set-precondition-def bijective-reverse conv-star-commute* **by** *auto*
have $p^T * p^\star * y = p^T * p * p^\star * y \sqcup (p[[y]])$
by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint*)
also have ... $\leq p^\star * y \sqcup y$
using *1* **by** (*metis find-set-invariant-def find-set-precondition-def comp-isotone mult-left-sub-dist-sup semiring.add-right-mono star.circ-back-loop-fixpoint star.circ-circ-mult star.circ-top star.circ-transitive-equal star-involutive star-one*)
also have ... $= p^\star * y$
by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)
finally have *3*: $p^{T\star} * x \leq p^\star * y$
using *2* **by** (*simp add*: *comp-associative star-left-induct*)
have $p * y \sqcap (p \sqcap 1) * top = (p \sqcap 1) * p * y$
using *comp-associative coreflexive-comp-top-inf inf-commute* **by** *auto*
also have ... $\leq p^T * p * y$
by (*metis inf.cobounded2 inf.sup-monoid.add-commute mult-left-isotone one-inf-conv*)
also have ... $\leq y$
using *1 find-set-invariant-def find-set-precondition-def mult-left-isotone* **by** *fastforce*
finally have *4*: $p * y \leq y \sqcup -((p \sqcap 1) * top)$
using *1* **by** (*metis find-set-invariant-def shunting-p bijective-regular*)
have $p^T * (p \sqcap 1) \leq p^T \sqcap 1$
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 $p^T * (p \sqcap 1) * top \leq (p \sqcap 1) * top$
using *inf-commute mult-isotone one-inf-conv* **by** *auto*
hence $p * -((p \sqcap 1) * top) \leq -((p \sqcap 1) * top)$
by (*metis comp-associative inf.sup-monoid.add-commute p-antitone p-antitone-iff schroeder-3-p*)
hence $p * y \sqcup p * -((p \sqcap 1) * top) \leq y \sqcup -((p \sqcap 1) * top)$
using *4 dual-order.trans le-supI sup-ge2* **by** *blast*

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

theorem *find-set*:
VARS y
[*find-set-precondition p x*]
$y := x$;
WHILE $y \neq p[[y]]$
INV { *find-set-invariant p x y* }
VAR { *card* { *z* . *regular z* $\wedge z \leq p^{T\star} * y$ } }
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 $\Longrightarrow \exists y$. *find-set-postcondition p x y*
using *tc-extract-function find-set* **by** *blast*

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

Theorem 3.3

lemma *root-point*:
disjoint-set-forest p $\Longrightarrow$ *point x* $\Longrightarrow$ *point* (*root p x*)
using *find-set-exists find-set-precondition-def find-set-postcondition-def* **by** *simp*

definition *find-set p x* $\equiv$ (*SOME y* . *find-set-postcondition p x y*)

lemma *find-set-function*:
assumes *find-set-precondition p x*
and $y =$ *find-set p x*
shows *find-set-postcondition p x y*

by (*metis assms find-set-def find-set-exists someI*)

4.3 Path Compression

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

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

lemma *path-compression-1*:
path-compression-precondition $p\ x\ y \wedge p0 = p \Longrightarrow$ *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 $p\ x\ y\ p0\ w \wedge y \neq p[[w]] \wedge$ *card* $\{ z\ .$ *regular* $z \wedge z$ $\leq p^{T\star} * w \} = n$
$\Longrightarrow$ *path-compression-invariant* $(p[w \longmapsto y])\ x\ y\ p0\ (p[[w]]) \wedge$ *card* $\{ z\ .$ *regular* $z \wedge z \leq (p[w \longmapsto y])^{T\star} * (p[[w]]) \} < n$
proof –
let $?p = p[w \longmapsto y]$
let $?s = \{ z\ .$ *regular* $z \wedge z \leq p^{T\star} * w \}$
let $?t = \{ z\ .$ *regular* $z \wedge z \leq ?p^{T\star} * (p[[w]]) \}$
assume *1*: *path-compression-invariant* $p\ x\ y\ p0\ w \wedge y \neq p[[w]] \wedge$ *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* $?p\ x\ y\ p0\ (p[[w]]) \wedge$ *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*: $w \neq y$
using *1* **by** (*metis* (*no-types*, *hide-lams*) *path-compression-invariant-def path-compression-precondition-def root-successor-loop*)
hence *5*: $w \sqcap y = bot$

using *1 distinct-points path-compression-invariant-def path-compression-precondition-def* **by** *auto*
hence $y * w^T \leq -1$
using *pseudo-complement schroeder-4-p* **by** *auto*
hence $y * w^T \leq p^{T\star} \sqcap -1$
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*: $w * w^T \leq -p^{T+}$
proof (*rule ccontr*)
assume $\neg\ w * w^T \leq -p^{T+}$
hence $w * w^T \leq --p^{T+}$
using *1 path-compression-invariant-def point-arc arc-in-partition* **by** *blast*
hence $w * w^T \leq p^{T+} \sqcap 1$
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 ... $= ((p^T \sqcap 1) * p^{T\star} \sqcap 1) \sqcup ((p^T \sqcap -1) * p^{T\star} \sqcap 1)$
by (*metis comp-inf.mult-right-dist-sup maddux-3-11-pp mult-right-dist-sup regular-one-closed*)
also have ... $= ((p^T \sqcap 1) * p^{T\star} \sqcap 1) \sqcup ((p \sqcap -1)^+ \sqcap 1)^T$
by (*metis conv-complement conv-dist-inf conv-plus-commute equivalence-one-closed reachable-without-loops*)
also have ... $\leq ((p^T \sqcap 1) * p^{T\star} \sqcap 1) \sqcup (-1 \sqcap 1)^T$
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 ... $= (p^T \sqcap 1) * p^{T\star} \sqcap 1$
by *simp*
also have ... $\leq (p^T \sqcap 1) * top \sqcap 1$
by (*metis comp-inf.comp-isotone coreflexive-comp-top-inf equivalence-one-closed inf.cobounded1 inf.cobounded2*)
also have ... $\leq p^T$
by (*simp add*: *coreflexive-comp-top-inf-one*)
finally have $w * w^T \leq p^T$
by *simp*
hence $w \leq p[[w]]$
using *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*: $w \sqcap p^{T+} * w = bot$
using *p-antitone-iff pseudo-complement schroeder-4-p* **by** *blast*
show $y \leq ?p^{T\star} * (p[[w]])$
proof −
have $(w \sqcap y^T)^T * (-w \sqcap p)^{T\star} * p^T * w \leq w^T * (-w \sqcap p)^{T\star} * p^T * w$
by (*simp add*: *conv-isotone mult-left-isotone*)
also have ... $\leq w^T * p^{T\star} * p^T * w$
by (*simp add*: *conv-isotone mult-left-isotone star-isotone mult-right-isotone*)
also have ... $= w^T * p^{T+} * w$
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-assoc conv-dist-comp conv-star-commute covector-bot-closed equivalence-top-closed inf.le-iff-sup mult-left-isotone*)
finally have $((w \sqcap y^T)^T \sqcup (-w \sqcap p)^T) * (-w \sqcap p)^{T\star} * p^T * w \leq (-w \sqcap p)^T * (-w \sqcap p)^{T\star} * p^T * w$
by (*simp add*: *bot-unique mult-right-dist-sup*)
also have ... $\leq (-w \sqcap p)^{T\star} * p^T * w$
by (*simp add*: *mult-left-isotone star.left-plus-below-circ*)
finally have $?p^T * (-w \sqcap p)^{T\star} * p^T * w \leq (-w \sqcap p)^{T\star} * p^T * w$
by (*simp add*: *conv-dist-sup*)
hence $?p^{T\star} * p^T * w \leq (-w \sqcap p)^{T\star} * p^T * w$
by (*metis comp-associative star.circ-loop-fixpoint star-left-induct sup-commute sup-least sup-left-divisibility*)
hence $w \sqcap ?p^{T\star} * p^T * w \leq w \sqcap (-w \sqcap p)^{T\star} * p^T * w$
using *inf.sup-right-isotone* **by** *blast*
also have ... $\leq w \sqcap p^{T\star} * p^T * w$
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*: $w^T * ?p^{T\star} * p^T * w = bot$
using *1* **by** (*metis* (*no-types*, *hide-lams*) *path-compression-invariant-def covector-inf-comp-3 mult-assoc conv-dist-comp covector-bot-closed equivalence-top-closed inf.le-iff-sup mult-left-isotone bot-least inf.absorb1*)
have $p^T * ?p^{T\star} * p^T * w = ((w \sqcap p)^T \sqcup (-w \sqcap p)^T) * ?p^{T\star} * p^T * w$
using *1* **by** (*metis* (*no-types*, *lifting*) *bijective-regular conv-dist-sup inf-commute maddux-3-11-pp path-compression-invariant-def*)
also have ... $= (w \sqcap p)^T * ?p^{T\star} * p^T * w \sqcup (-w \sqcap p)^T * ?p^{T\star} * p^T * w$
by (*simp add*: *mult-right-dist-sup*)
also have ... $\leq w^T * ?p^{T\star} * p^T * w \sqcup (-w \sqcap p)^T * ?p^{T\star} * p^T * w$
using *semiring.add-right-mono comp-isotone conv-isotone* **by** *auto*
also have ... $= (-w \sqcap p)^T * ?p^{T\star} * p^T * w$
using *9* **by** *simp*
also have ... $\leq ?p^{T+} * p^T * w$
by (*simp add*: *conv-isotone mult-left-isotone*)
also have ... $\leq ?p^{T\star} * p^T * w$
by (*simp add*: *comp-isotone star.left-plus-below-circ*)
finally have $p^{T\star} * p^T * w \leq ?p^{T\star} * p^T * w$

by (*metis comp-associative star.circ-loop-fixpoint star-left-induct sup-commute sup-least sup-left-divisibility*)
thus $y \leq ?p^{T\star} * (p[[w]])$
using *6* **by** (*simp add*: *star-simulation-right-equal mult-assoc*)
qed
have *10*: *acyclic* ($?p \sqcap -1$)
using *1 update-acyclic-1 path-compression-invariant-def path-compression-precondition-def* **by** *auto*
have $?p[[p^{T+} * w]] \leq p^{T+} * w$
proof $-$
have $(w^T \sqcap y) * p^{T+} * w = y \sqcap w^T * p^{T+} * w$
using *1* **by** (*metis* (*no-types*, *hide-lams*) *path-compression-invariant-def path-compression-precondition-def inf-commute vector-inf-comp*)
hence $?p[[p^{T+} * w]] = (y \sqcap w^T * p^{T+} * w) \sqcup (-w^T \sqcap p^T) * p^{T+} * w$
by (*simp add*: *comp-associative conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $\leq y \sqcup (-w^T \sqcap p^T) * p^{T+} * w$
using *sup-left-isotone* **by** *auto*
also have ... $\leq y \sqcup p^T * p^{T+} * w$
using *mult-left-isotone sup-right-isotone* **by** *auto*
also have ... $\leq y \sqcup p^{T+} * w$
using *semiring.add-left-mono mult-left-isotone mult-right-isotone star.left-plus-below-circ* **by** *auto*
also have ... $= p^{T+} * w$
using *6* **by** (*simp add*: *sup-absorb2*)
finally show *?thesis*
by *simp*
qed
hence *11*: $?p^{T\star} * (p[[w]]) \leq p^{T+} * w$
using *star-left-induct* **by** (*simp add*: *mult-left-isotone star.circ-mult-increasing*)
hence *12*: $?p^{T+} * (p[[w]]) \leq p^{T+} * w$
using *dual-order.trans mult-left-isotone star.left-plus-below-circ* **by** *blast*
have *13*: $?p[[x]] = y \wedge y \neq x \wedge ?p^{T+} * (p[[w]]) \leq -x$
proof (*cases* $w = x$)
case *True*
hence $?p[[x]] = (w^T \sqcap y) * w \sqcup (-w^T \sqcap p^T) * w$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $= (w^T \sqcap y) * w \sqcup p^T * (-w \sqcap w)$
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 ... $= (w^T \sqcap y) * w$
by *simp*
also have ... $= y * w$
using *1 inf.sup-monoid.add-commute path-compression-invariant-def covector-inf-comp-3* **by** *simp*
also have ... $= y$

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 $?p[[x]] = (w^T \sqcap y) * x \sqcup (-w^T \sqcap p^T) * x$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $= y * (w \sqcap x) \sqcup p^T * (-w \sqcap x)$
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 ... $= p^T * (-w \sqcap x)$
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 $p[[w]] \neq x \longrightarrow ?p[[x]] = y \wedge y \neq x \wedge ?p^{T+} * (p[[w]]) \leq -x$
by *simp*
have *14*: $?p^{T\star} * x = x \sqcup y$
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 ... $= y \sqcup (w^T \sqcap y) * y \sqcup (-w^T \sqcap p^T) * y$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup sup-assoc*)
also have ... $\leq y \sqcup (w^T \sqcap y) * y \sqcup p^T * y$
using *mult-left-isotone sup-right-isotone* **by** *auto*
also have ... $= y \sqcup (w^T \sqcap y) * y$
using *1* **by** (*smt sup.cobounded1 sup-absorb1 path-compression-invariant-def path-compression-precondition-def root-successor-loop*)
also have ... $\leq y \sqcup y * y$
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 $?p^{T\star} * x \leq x \sqcup y$
by (*simp add*: *star-left-induct*)
next

show $x \sqcup y \leq ?p^{T\star} * x$
using *13* **by** (*metis mult-left-isotone star.circ-increasing star.circ-loop-fixpoint sup.boundedI sup-ge2*)
qed
have *15*: $y = root\ ?p\ x$
proof −
have $(p \sqcap 1) * y = (p \sqcap 1) * (p \sqcap 1) * p^{T\star} * x$
using *1 path-compression-invariant-def path-compression-precondition-def root-var mult-assoc* **by** *auto*
also have ... $= (p \sqcap 1) * p^{T\star} * x$
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* (*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 $root\ ?p\ x = (?p \sqcap 1) * (x \sqcup y)$
using *14* **by** (*metis mult-assoc root-var*)
also have ... $= (w \sqcap y^T \sqcap 1) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$
by (*simp add*: *inf-sup-distrib2 semiring.distrib-right*)
also have ... $= (w \sqcap 1 \sqcap y^T) * (x \sqcup y) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$
by (*simp add*: *inf.left-commute inf.sup-monoid.add-commute*)
also have ... $= (w \sqcap 1) * (y \sqcap (x \sqcup y)) \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$
using *1* **by** (*metis* (*no-types*, *lifting*) *path-compression-invariant-def path-compression-precondition-def covector-inf-comp-3*)
also have ... $= (w \sqcap 1) * y \sqcup (-w \sqcap p \sqcap 1) * (x \sqcup y)$
by (*simp add*: *inf.absorb1*)
also have ... $= (w \sqcap 1 * y) \sqcup (-w \sqcap (p \sqcap 1) * (x \sqcup y))$
using *1* **by** (*metis* (*no-types*, *lifting*) *inf-assoc vector-complement-closed path-compression-invariant-def vector-inf-comp*)
also have ... $= (w \sqcap y) \sqcup (-w \sqcap ((p \sqcap 1) * x \sqcup y))$
using *16* **by** (*simp add*: *mult-left-dist-sup*)
also have ... $= (w \sqcap y) \sqcup (-w \sqcap y)$
using *17* **by** (*simp add*: *sup.absorb2*)
also have ... $= y$
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 *?thesis*
by *simp*
qed
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 $w \sqcap p \sqcap 1 \leq w \sqcap w^T \sqcap p$
by (*metis inf.boundedE inf.boundedI inf.cobounded1 inf.cobounded2 one-inf-conv*)
also have ... $= w * w^T \sqcap p$
using *1 vector-covector path-compression-invariant-def* **by** *auto*
also have ... $\leq -p^{T+} \sqcap p$
using *7* **by** (*simp add*: *inf.coboundedI2 inf.sup-monoid.add-commute*)
finally have $w \sqcap p \sqcap 1 = bot$
by (*metis* (*no-types*, *hide-lams*) *conv-dist-inf coreflexive-symmetric inf.absorb1 inf.boundedE inf.cobounded2 pseudo-complement star.circ-mult-increasing*)
also have $w \sqcap y^T \sqcap 1 = bot$
using *5 antisymmetric-bot-closed asymmetric-bot-closed comp-inf.schroeder-2 inf.absorb1 one-inf-conv* **by** *fastforce*
finally have $w \sqcap p \sqcap 1 = w \sqcap y^T \sqcap 1$
by *simp*
thus $?p \sqcap 1 = p0 \sqcap 1$
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 $p[[w]] = p^T * (w \sqcap p^\star * y)$
using *1* **by** (*metis* (*no-types*, *lifting*) *bijective-reverse conv-star-commute inf.absorb1 path-compression-invariant-def path-compression-precondition-def*)
also have ... $= p^T * (w \sqcap p^\star) * y$
using *1 vector-inf-comp path-compression-invariant-def mult-assoc* **by** *auto*
also have ... $= p^T * ((w \sqcap 1) \sqcup (w \sqcap p) * (-w \sqcap p)^\star) * y$
using *1 omit-redundant-points path-compression-invariant-def* **by** *auto*
also have ... $= p^T * (w \sqcap 1) * y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^\star * y$
by (*simp add*: *comp-associative mult-left-dist-sup mult-right-dist-sup*)
also have ... $\leq p^T * y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^\star * y$
by (*metis semiring.add-right-mono comp-isotone eq-iff inf.cobounded1 inf.sup-monoid.add-commute mult-1-right*)
also have ... $= y \sqcup p^T * (w \sqcap p) * (-w \sqcap p)^\star * y$
using *1 path-compression-invariant-def path-compression-precondition-def root-successor-loop* **by** *fastforce*
also have ... $\leq y \sqcup p^T * p * (-w \sqcap p)^\star * y$
using *comp-isotone sup-right-isotone* **by** *auto*
also have ... $\leq y \sqcup (-w \sqcap p)^\star * y$
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 ... $= (-w \sqcap p)^\star * y$
by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)
finally have *18*: $p[[w]] \leq (-w \sqcap p)^\star * y$
by *simp*

have $p^T * (-w \sqcap p)^\star * y = p^T * y \sqcup p^T * (-w \sqcap p) * (-w \sqcap p)^\star * y$
by (*metis comp-associative mult-left-dist-sup star.circ-loop-fixpoint sup-commute*)
also have ... $= y \sqcup p^T * (-w \sqcap p) * (-w \sqcap p)^\star * y$
using *1 path-compression-invariant-def path-compression-precondition-def root-successor-loop* **by** *fastforce*
also have ... $\leq y \sqcup p^T * p * (-w \sqcap p)^\star * y$
using *comp-isotone sup-right-isotone* **by** *auto*
also have ... $\leq y \sqcup (-w \sqcap p)^\star * y$
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 ... $= (-w \sqcap p)^\star * y$
by (*metis star.circ-loop-fixpoint sup.left-idem sup-commute*)
finally have *19*: $p^{T\star} * p^T * w \leq (-w \sqcap p)^\star * y$
using *18* **by** (*simp add*: *comp-associative star-left-induct*)
have $w^T \sqcap p^T = p^T * (w^T \sqcap 1)$
using *1* **by** (*metis conv-dist-comp conv-dist-inf equivalence-one-closed vector-inf-one-comp path-compression-invariant-def*)
also have ... $\leq p[[w]]$
by (*metis comp-right-subdist-inf inf.boundedE inf.sup-monoid.add-commute one-inf-conv*)
also have ... $\leq p^{T\star} * p^T * w$
by (*simp add*: *mult-left-isotone star.circ-mult-increasing-2*)
also have ... $\leq (-w \sqcap p)^\star * y$
using *19* **by** *simp*
finally have $w \sqcap p \leq y^T * (-w \sqcap p)^{T\star}$
by (*metis conv-dist-comp conv-dist-inf conv-involutive conv-isotone conv-star-commute*)
hence $w \sqcap p \leq (w \sqcap y^T) * (-w \sqcap p)^{T\star}$
using *1* **by** (*metis inf.absorb1 inf.left-commute inf.left-idem inf.orderI vector-inf-comp path-compression-invariant-def*)
also have ... $\leq (w \sqcap y^T) * ?p^{T\star}$
by (*simp add*: *conv-isotone mult-right-isotone star-isotone*)
also have ... $\leq ?p * ?p^{T\star}$
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 ... $= w * y^T \sqcup p$
using *1 path-compression-invariant-def path-compression-precondition-def vector-covector* **by** *auto*
also have ... $\leq p^\star \sqcup p$
using *1* **by** (*metis* (*no-types*, *lifting*) *conv-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*
qed
show *card ?t* $<$ *n*
proof $-$
have $?p^T * p^{T\star} * w = (w^T \sqcap y) * p^{T\star} * w \sqcup (-w^T \sqcap p^T) * p^{T\star} * w$
by (*simp add*: *conv-complement conv-dist-inf conv-dist-sup mult-right-dist-sup*)
also have ... $\leq (w^T \sqcap y) * p^{T\star} * w \sqcup p^T * p^{T\star} * w$
using *mult-left-isotone sup-right-isotone* **by** *auto*
also have ... $\leq (w^T \sqcap y) * p^{T\star} * w \sqcup p^{T\star} * w$
using *mult-left-isotone star.left-plus-below-circ sup-right-isotone* **by** *blast*
also have ... $\leq y * p^{T\star} * w \sqcup p^{T\star} * w$
using *semiring.add-right-mono mult-left-isotone* **by** *auto*
also have ... $\leq y * top \sqcup p^{T\star} * w$
by (*simp add*: *comp-associative le-supI1 mult-right-isotone*)
also have ... $= p^{T\star} * w$
using *1 path-compression-invariant-def path-compression-precondition-def sup-absorb2* **by** *auto*
finally have $?p^{T\star} * p^T * w \leq p^{T\star} * w$
using *11* **by** (*metis dual-order.trans star.circ-loop-fixpoint sup-commute sup-ge2 mult-assoc*)
hence *22*: *?t* $\subseteq$ *?s*
using *order-lesseq-imp mult-assoc* **by** *auto*
have *23*: $w \in$ *?s*
using *1 bijective-regular path-compression-invariant-def eq-iff star.circ-loop-fixpoint* **by** *auto*
have *24*: $\neg\ w \in$ *?t*
proof
assume $w \in$ *?t*
hence *25*: $w \leq (?p^T \sqcap -1)^\star * (p[[w]])$

using *reachable-without-loops* **by** *auto*
hence $p[[w]] \leq (?p \sqcap -1)^\star * w$
using *1 2* **by** (*metis* (*no-types*, *hide-lams*) *bijective-reverse conv-star-commute reachable-without-loops path-compression-invariant-def*)
also have ... $\leq p^\star * w$
proof −
have $p^{T\star} * y = y$
using *1 path-compression-invariant-def path-compression-precondition-def root-transitive-successor-loop* **by** *fastforce*
hence $y^T * p^\star * w = y^T * w$
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*: $y^T * p^\star * w = bot$
by *simp*
have $(?p \sqcap -1) * p^\star * w = (w \sqcap y^T \sqcap -1) * p^\star * w \sqcup (-w \sqcap p \sqcap -1) * p^\star * w$
by (*simp add*: *comp-inf.mult-right-dist-sup mult-right-dist-sup*)
also have ... $\leq (w \sqcap y^T \sqcap -1) * p^\star * w \sqcup p * p^\star * w$
by (*meson inf-le1 inf-le2 mult-left-isotone order-trans sup-right-isotone*)
also have ... $\leq (w \sqcap y^T \sqcap -1) * p^\star * w \sqcup p^\star * w$
using *mult-left-isotone star.left-plus-below-circ sup-right-isotone* **by** *blast*
also have ... $\leq y^T * p^\star * w \sqcup p^\star * w$
by (*meson inf-le1 inf-le2 mult-left-isotone order-trans sup-left-isotone*)
also have ... $= p^\star * w$
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 $w \leq p^{T\star} * p^T * w$
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 psubset-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 $p\ x\ y\ p0\ w \wedge y = p[[w]] \Longrightarrow$ *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 $p\ t\ w$
[*path-compression-precondition* $p\ x\ y \wedge p0 = p$]
$w := x$;
WHILE $y \neq p[[w]]$
INV { *path-compression-invariant* $p\ x\ y\ p0\ w$ }
VAR { *card* { z . *regular* $z \wedge z \leq p^{T\star} * w$ } }
DO $t := w$;
$w := p[[w]]$;
$p[t] := y$
OD
[*path-compression-postcondition* $p\ x\ y\ p0$]
apply *vcg-tc-simp*
apply (*fact path-compression-1*)
apply (*fact path-compression-2*)
by (*fact path-compression-3*)

lemma *path-compression-exists*:
path-compression-precondition $p\ x\ y \Longrightarrow \exists\, p'$. *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*)

4.4 Find-Set with Path Compression

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

theorem *find-set-path-compression*:
VARS $p\ y$
[*find-set-precondition* $p\ x \wedge p0 = p$]
$y :=$ *find-set* $p\ x$;
$p :=$ *path-compression* $p\ x\ y$
[*path-compression-postcondition* $p\ x\ y\ p0$]
apply *vcg-tc-simp*

using *find-set-function find-set-postcondition-def find-set-precondition-def path-compression-function path-compression-precondition-def* **by** *fastforce*

theorem *find-set-path-compression-1*:
  VARS p t w y
  [*find-set-precondition p x* $\wedge$ *p0* = *p*]
  y := *find-set p x*;
  w := *x*;
  WHILE y $\neq$ *p*[[*w*]]
    INV { *path-compression-invariant p x y p0 w* }
    VAR { *card* { *z* . *regular z* $\wedge$ *z* $\leq$ $p^{T\star}$ $*$ *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 *fastforce*
   apply (*fact path-compression-2*)
  by (*fact path-compression-3*)

theorem *find-set-path-compression-2*:
  VARS p y
  [*find-set-precondition p x* $\wedge$ *p0* = *p*]
  y := *x*;
  WHILE y $\neq$ *p*[[*y*]]
    INV { *find-set-invariant p x y* $\wedge$ *p0* = *p* }
    VAR { *card* { *z* . *regular z* $\wedge$ *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* $\wedge$ *p0* = *p*]
  y := *x*;
  WHILE y $\neq$ *p*[[*y*]]
    INV { *find-set-invariant p x y* $\wedge$ *p0* = *p* }
    VAR { *card* { *z* . *regular z* $\wedge$ *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* $z \wedge z \leq p^{T\star} * w$ } }
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-precondition-def* **apply** *blast*
apply (*fact path-compression-2*)
by (*fact path-compression-3*)

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

lemma *find-set-path-compression-exists*:
find-set-precondition p x $\Longrightarrow \exists p'\ y$. *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 *?P* = λ(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

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*]
  t := *find-set-path-compression p x*;
  p := *fst t*;
  r := *snd t*;
  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 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* ∧ *point ?r* ∧ *?r* = *root ?p1 x* ∧ *?p1* ⊓ *1* = *p* ⊓ *1* ∧ *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* ∧ *point ?s* ∧ *?s* = *root ?p2 y* ∧ *?p2* ⊓ *1* = *?p1* ⊓ *1* ∧ *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* $(?p \sqcap -1)$
proof (*cases* $?r = ?s$)
case *True*
thus *?thesis*
using *3 update-acyclic-3* **by** *fastforce*
next
case *False*
hence $bot = ?r \sqcap ?s$
using *2 3 distinct-points* **by** *blast*
also have ... $= ?r \sqcap ?p2^{T\star} * ?s$
using *3 root-transitive-successor-loop* **by** *force*
finally have $?s \sqcap ?p2^{\star} * ?r = bot$
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 $fc\ ?p = wcc\ (p \sqcup x * y^T)$
proof (*rule antisym*)
have $?r = ?p1[[?r]]$
using *2 root-successor-loop* **by** *force*
hence $?r * ?r^T \le ?p1^T$
using *2 eq-refl shunt-bijective* **by** *blast*
hence $?r * ?r^T \le ?p1$
using *2 conv-order coreflexive-symmetric* **by** *fastforce*
hence $?r * ?r^T \le ?p1 \sqcap 1$
using *2 inf.boundedI* **by** *blast*
also have ... $= ?p2 \sqcap 1$
using *3* **by** *simp*
finally have $?r * ?r^T \le ?p2$
by *simp*
hence $?r \le ?p2 * ?r$
using *2 shunt-bijective* **by** *blast*
hence *6*: $?p2[[?r]] \le ?r$
using *3 shunt-mapping* **by** *blast*
have $?r \sqcap ?p2 \le ?r * (top \sqcap ?r^T * ?p2)$
using *2* **by** (*metis dedekind-1*)

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

also have ... $\leq$ *wcc ?p2*
  using *star-isotone sup-ge2* **by** *blast*
also have ... $\leq$ *wcc ?p*
  using *8* **by** *simp*
finally have *14*: $?s * y^T \leq wcc\ ?p$
  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 ... $\leq wcc\ ?p * (?r * ?s^T) * (?s * y^T)$
  using *11* **by** (*metis mult-left-isotone mult-assoc*)
also have ... $\leq wcc\ ?p * wcc\ ?p * (?s * y^T)$
  using *12* **by** (*metis mult-left-isotone mult-right-isotone*)
also have ... $\leq wcc\ ?p * wcc\ ?p * wcc\ ?p$
  using *14* **by** (*metis mult-right-isotone*)
also have ... $= wcc\ ?p$
  by (*simp add*: *star.circ-transitive-equal*)
finally have $p \sqcup x * y^T \leq wcc\ ?p$
  using *9* **by** *simp*
hence $wcc\ (p \sqcup x * y^T) \leq wcc\ ?p$
  using *wcc-below-wcc* **by** *simp*
thus $wcc\ (p \sqcup x * y^T) \leq fc\ ?p$
  using *5 fc-wcc* **by** *simp*
have $-?r \sqcap ?p2 \leq wcc\ ?p2$
  by (*simp add*: *inf.coboundedI2 star.circ-sub-dist-1*)
also have ... $= wcc\ p$
  using *4* **by** (*simp add*: *star-decompose-1*)
also have ... $\leq wcc\ (p \sqcup x * y^T)$
  by (*simp add*: *wcc-isotone*)
finally have *15*: $-?r \sqcap ?p2 \leq wcc\ (p \sqcup x * y^T)$
  by *simp*
have $?r * x^T \leq wcc\ ?p1$
  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 ... $\leq wcc\ (p \sqcup x * y^T)$
  by (*simp add*: *wcc-isotone*)
finally have *16*: $?r * x^T \leq wcc\ (p \sqcup x * y^T)$
  by *simp*
have *17*: $x * y^T \leq wcc\ (p \sqcup x * y^T)$
  using *le-supE star.circ-sub-dist-1* **by** *blast*
have $y * ?s^T \leq ?p2^\star$
  using *13 conv-dist-comp conv-order conv-star-commute* **by** *fastforce*
also have ... $\leq wcc\ ?p2$
  using *star.circ-sub-dist sup-commute* **by** *fastforce*
also have ... $= wcc\ p$

using *4* **by** (*simp add*: *star-decompose-1*)
also have ... $\leq$ *wcc* ($p \sqcup x * y^T$)
by (*simp add*: *wcc-isotone*)
finally have *18*: $y * ?s^T \leq$ *wcc* ($p \sqcup x * y^T$)
by *simp*
have $?r \leq ?r * x^T * x \wedge ?s \leq ?s * y^T * y$
using *1 shunt-bijective union-sets-precondition-def* **by** *blast*
hence $?r * ?s^T \leq ?r * x^T * x * (?s * y^T * y)^T$
using *comp-isotone conv-isotone* **by** *blast*
also have ... $= ?r * x^T * x * y^T * y * ?s^T$
by (*simp add*: *comp-associative conv-dist-comp*)
also have ... $\leq$ *wcc* ($p \sqcup x * y^T$) $* (x * y^T) * (y * ?s^T)$
using *16* **by** (*metis mult-left-isotone mult-assoc*)
also have ... $\leq$ *wcc* ($p \sqcup x * y^T$) $*$ *wcc* ($p \sqcup x * y^T$) $* (y * ?s^T)$
using *17* **by** (*metis mult-left-isotone mult-right-isotone*)
also have ... $\leq$ *wcc* ($p \sqcup x * y^T$) $*$ *wcc* ($p \sqcup x * y^T$) $*$ *wcc* ($p \sqcup x * y^T$)
using *18* **by** (*metis mult-right-isotone*)
also have ... $=$ *wcc* ($p \sqcup x * y^T$)
by (*simp add*: *star.circ-transitive-equal*)
finally have $?p \leq$ *wcc* ($p \sqcup x * y^T$)
using *2 3 15 vector-covector* **by** *auto*
hence *wcc* $?p \leq$ *wcc* ($p \sqcup x * y^T$)
using *wcc-below-wcc* **by** *blast*
thus *fc* $?p \leq$ *wcc* ($p \sqcup x * y^T$)
using *5 fc-wcc* **by** *simp*
qed
qed
qed

lemma *union-sets-exists*:
union-sets-precondition p x y $\Longrightarrow \exists p'$. *union-sets-postcondition p' x y p*
using *tc-extract-function union-sets* **by** *blast*

definition *union-sets p x y* $\equiv$ (*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*
by (*metis assms union-sets-def union-sets-exists someI*)

end

end

References

[1] R.-J. Back and J. von Wright. *Refinement Calculus.* Springer, New York, 1998.

[2] R. C. Backhouse and B. A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15(2):161–186, 1975.

[3] R. Berghammer. Combining relational calculus and the Dijkstra–Gries method for deriving relational programs. *Information Sciences*, 119(3–4):155–171, 1999.

[4] R. Berghammer and G. Struth. On automated program construction and verification. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction (MPC 2010)*, volume 6120 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2010.

[5] R. Berghammer, B. von Karger, and A. Wolf. Relation-algebraic derivation of spanning tree algorithms. In J. Jeuring, editor, *Mathematics of Program Construction (MPC 1998)*, volume 1422 of *Lecture Notes in Computer Science*, pages 23–43. Springer, 1998.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[7] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Trans. Prog. Lang. Syst.*, 29(3:17):1–65, 2007.

[8] B. A. Galler and M. J. Fisher. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, 1964.

[9] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings.* Springer, 2008.

[10] W. Guttmann. Verifying minimum spanning tree algorithms with Stone relation algebras. *Journal of Logical and Algebraic Methods in Programming*, 101:132–150, 2018.

[11] W. Guttmann. Verifying the correctness of disjoint-set forests with Kleene relation algebras. In U. Fahrenberg, P. Jipsen, and M. Winter, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2020)*, volume 12062 of *Lecture Notes in Computer Science*, pages 134–151. Springer, 2020.

[12] P. Höfner and B. Möller. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing*, 24(4):459–476, 2012.

[13] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[14] P. Lammich and R. Meis. A separation logic framework for Imperative HOL. *Archive of Formal Proofs*, 2012.

[15] B. Möller. Derivation of graph and pointer algorithms. In B. Möller, H. A. Partsch, and S. A. Schuman, editors, *Formal Program Development*, volume 755 of *Lecture Notes in Computer Science*, pages 123–160. Springer, 1993.

[16] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[17] A. Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

[18] B. Zhan. Verifying imperative programs using Auto2. *Archive of Formal Proofs*, 2018.