Relational Characterisations of Paths

Walter Guttmann and Peter Höfner

July 28, 2020

Abstract

Binary relations are one of the standard ways to encode, characterise and reason about graphs. Relation algebras provide equational axioms for a large fragment of the calculus of binary relations. Although relations are standard tools in many areas of mathematics and computing, researchers usually fall back to point-wise reasoning when it comes to arguments about paths in a graph. We present a purely algebraic way to specify different kinds of paths in Kleene relation algebras, which are relation algebras equipped with an operation for reflexive transitive closure. We study the relationship between paths with a designated root vertex and paths without such a vertex. Since we stay in first-order logic this development helps with mechanising proofs. To demonstrate the applicability of the algebraic framework we verify the correctness of three basic graph algorithms.

Contents

Overview

A path in a graph can be defined as a connected subgraph of edges where each vertex has at most one incoming edge and at most one outgoing edge [3, 12]. We develop a theory of paths based on this representation and use it for algorithm verification. All reasoning is done in variants of relation algebras and Kleene algebras [8, 9, 11].

Section 1 presents fundamental results that hold in relation algebras. Relation-algebraic characterisations of various kinds of paths are introduced and compared in Section 2. We extend this to paths with a designated root in Section 3. Section 4 verifies the correctness of a few basic graph algorithms.

These Isabelle/HOL theories formally verify results in [2]. See this paper for further details and related work.

1 (More) Relation Algebra

This theory presents fundamental properties of relation algebras, which are not present in the AFP entry on relation algebras but could be integrated there [1]. Many theorems concern vectors and points.

theory *More-Relation-Algebra*

imports *Relation-Algebra.Relation-Algebra-RTC*
Relation-Algebra.Relation-Algebra-Functions

begin

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

context *relation-algebra*
begin

notation
  converse $((\text{-}^{T})$ $[102]$ $101)$

abbreviation *bijective*
  where $\mathit{bijective}\ x \equiv \mathit{is\text{-}inj}\ x \wedge \mathit{is\text{-}sur}\ x$

abbreviation *reflexive*
  where $\mathit{reflexive}\ R \equiv 1' \leq R$

abbreviation *symmetric*
  where $\mathit{symmetric}\ R \equiv R = R^T$

abbreviation *transitive*

where *transitive* $R \equiv R;R \leq R$

General theorems

lemma *x-leq-triple-x*:
$x \leq x;x^T;x$
proof –
have $x = x;1' \cdot 1$
by *simp*
also have ... $\leq (x \cdot 1;1'^T);(1' \cdot x^T;1)$
by (*rule dedekind*)
also have ... $= x;(x^T;1 \cdot 1')$
by (*simp add*: *inf.commute*)
also have ... $\leq x;(x^T \cdot 1';1^T);(1 \cdot (x^T)^T;1')$
by (*metis comp-assoc dedekind mult-isol*)
also have ... $\leq x;x^T;x$
by *simp*
finally show *?thesis* **.**
qed

lemma *inj-triple*:
assumes *is-inj x*
shows $x = x;x^T;x$
by (*metis assms eq-iff inf-absorb2 is-inj-def mult-1-left mult-subdistr x-leq-triple-x*)

lemma *p-fun-triple*:
assumes *is-p-fun x*
shows $x = x;x^T;x$
by (*metis assms comp-assoc eq-iff is-p-fun-def mult-isol mult-oner x-leq-triple-x*)

lemma *loop-backward-forward*:
$x^T \leq -(1') + x$
by (*metis conv-e conv-times inf.cobounded2 test-dom test-domain test-eq-conv galois-2 inf.commute*
sup.commute)

lemma *inj-sur-semi-swap*:
assumes *is-sur z*
and *is-inj x*
shows $z \leq y;x \Longrightarrow x \leq y^T;z$
proof –
assume $z \leq y;x$
hence $z;x^T \leq y;(x;x^T)$
by (*metis mult-isor mult-assoc*)
hence $z;x^T \leq y$
using ⟨*is-inj x*⟩ **unfolding** *is-inj-def*
by (*metis mult-isol order.trans mult-1-right*)
hence $(z^T;z);x^T \leq z^T;y$
by (*metis mult-isol mult-assoc*)
hence $x^T \leq z^T;y$

using ‹*is-sur z*› **unfolding** *is-sur-def*
by (*metis mult-isor order.trans mult-1-left*)
thus *?thesis*
using *conv-iso* **by** *fastforce*
qed

lemma *inj-sur-semi-swap-short*:
assumes *is-sur z*
and *is-inj x*
shows $z \leq y^T;x \Longrightarrow x \leq y;z$
proof –
assume *as*: $z \leq y^T;x$
hence $z;x^T \leq y^T$
using ‹$z \leq y^T;x$› ‹*is-inj x*› **unfolding** *is-inj-def*
by (*metis assms*(*2*) *conv-invol inf.orderI inf-absorb1 inj-p-fun ss-422iii*)
hence $x^T \leq z^T;y^T$
using ‹*is-sur z*› **unfolding** *is-sur-def*
by (*metis as assms inj-sur-semi-swap conv-contrav conv-invol conv-iso*)
thus $x \leq y;z$
using *conv-iso* **by** *fastforce*
qed

lemma *bij-swap*:
assumes *bijective z*
and *bijective x*
shows $z \leq y^T;x \longleftrightarrow x \leq y;z$
by (*metis assms inj-sur-semi-swap conv-invol*)

The following result is [10, Proposition 4.2.2(iv)].

lemma *ss422iv*:
assumes *is-p-fun y*
and $x \leq y$
and $y;1 \leq x;1$
shows $x = y$
proof –
have $y \leq (x;1) \cdot y$
using *assms*(*3*) *le-infI maddux-20 order-trans* **by** *blast*
also have ... $\leq x;x^T;y$
by (*metis inf-top-left modular-1-var comp-assoc*)
also have ... $\leq x;y^T;y$
using *assms*(*2*) *conv-iso mult-double-iso* **by** *blast*
also have ... $\leq x$
using *assms*(*1*) *comp-assoc is-p-fun-def mult-isol mult-1-right*
by *fastforce*
finally show *?thesis*
by (*simp add*: *assms*(*2*) *antisym*)
qed

The following results are variants of [10, Proposition 4.2.3].

lemma *ss423conv*:
  assumes *bijective* x
    shows $x \; ; \; y \leq z \longleftrightarrow y \leq x^T \; ; \; z$
by (*metis assms conv-contrav conv-iso inj-p-fun is-map-def ss423 sur-total*)

lemma *ss423bij*:
  assumes *bijective* x
    shows $y \; ; \; x^T \leq z \longleftrightarrow y \leq z \; ; \; x$
by (*simp add*: *assms is-map-def p-fun-inj ss423 total-sur*)

lemma *inj-distr*:
  assumes *is-inj* z
    shows $(x \cdot y);z = (x;z) \cdot (y;z)$
apply (*rule antisym*)
 using *mult-subdistr-var* **apply** *blast*
using *assms conv-iso inj-p-fun p-fun-distl* **by** *fastforce*

lemma *test-converse*:
  $x \cdot 1' = x^T \cdot 1'$
by (*metis conv-e conv-times inf-le2 is-test-def test-eq-conv*)

lemma *injective-down-closed*:
  assumes *is-inj* x
    and $y \leq x$
    shows *is-inj* y
by (*meson assms conv-iso dual-order.trans is-inj-def mult-isol-var*)

lemma *injective-sup*:
  assumes *is-inj* t
    and $e;t^T \leq 1'$
    and *is-inj* e
    shows *is-inj* $(t + e)$
proof –
  have *1*: $t;e^T \leq 1'$
    using *assms*(*2*) *conv-contrav conv-e conv-invol conv-iso* **by** *fastforce*
  have $(t + e);(t + e)^T = t;t^T + t;e^T + e;t^T + e;e^T$
    by (*metis conv-add distrib-left distrib-right′ sup-assoc*)
  also have ... $\leq 1'$
    using *1 assms* **by** (*simp add*: *is-inj-def le-supI*)
  finally show *?thesis*
    unfolding *is-inj-def* **.**
qed

Some (more) results about vectors

lemma *vector-meet-comp*:
  assumes *is-vector* v
    and *is-vector* w
    shows $v;w^T = v \cdot w^T$
by (*metis assms conv-contrav conv-one inf-top-right is-vector-def vector-1*)

lemma *vector-meet-comp'*:
  assumes *is-vector v*
    shows $v;v^T = v \cdot v^T$
using *assms vector-meet-comp* **by** *blast*

lemma *vector-meet-comp-x*:
  $x;1;x^T = x;1 \cdot 1;x^T$
by (*metis comp-assoc inf-top.right-neutral is-vector-def one-idem-mult vector-1*)

lemma *vector-meet-comp-x'*:
  $x;1;x = x;1 \cdot 1;x$
by (*metis inf-commute inf-top.right-neutral ra-1*)

lemma *vector-prop1*:
  assumes *is-vector v*
    shows $-v^T;v = 0$
by (*metis assms compl-inf-bot inf-top.right-neutral one-compl one-idem-mult vector-2*)

The following results and a number of others in this theory are from [5].

lemma *ee*:
  assumes *is-vector v*
    and $e \leq v;-v^T$
    shows $e;e = 0$
proof −
  have $e;v \leq 0$
    by (*metis assms annir mult-isor vector-prop1 comp-assoc*)
  thus *?thesis*
    by (*metis assms(2) annil antisym bot-least comp-assoc mult-isol*)
qed

lemma *et*:
  assumes *is-vector v*
    and $e \leq v;-v^T$
    and $t \leq v;v^T$
    shows $e;t = 0$
    and $e;t^T = 0$
proof −
  have $e;t \leq v;-v^T;v;v^T$
    by (*metis assms(2−3) mult-isol-var comp-assoc*)
  thus $e;t = 0$
    by (*simp add: assms(1) comp-assoc le-bot vector-prop1*)
next
  have $t^T \leq v;v^T$
    using *assms(3) conv-iso* **by** *fastforce*
  hence $e;t^T \leq v;-v^T;v;v^T$
    by (*metis assms(2) mult-isol-var comp-assoc*)
  thus $e;t^T = 0$

by (*simp add*: *assms*(*1*) *comp-assoc le-bot vector-prop1*)
qed

Some (more) results about points

definition *point*
where *point x* ≡ *is-vector x* ∧ *bijective x*

lemma *point-swap*:
assumes *point p*
and *point q*
shows $p \leq x;q \longleftrightarrow q \leq x^T;p$
by (*metis assms conv-invol inj-sur-semi-swap point-def*)

Some (more) results about singletons

abbreviation *singleton*
where *singleton x* ≡ *bijective* (*x*;*1*) ∧ *bijective* (x^T;*1*)

lemma *singleton-injective*:
assumes *singleton x*
shows *is-inj x*
using *assms injective-down-closed maddux-20* **by** *blast*

lemma *injective-inv*:
assumes *is-vector v*
and *singleton e*
and $e \leq v;-v^T$
and $t \leq v;v^T$
and *is-inj t*
shows *is-inj* (*t* + *e*)
by (*metis assms singleton-injective injective-sup bot-least et*(*2*))

lemma *singleton-is-point*:
assumes *singleton p*
shows *point* (*p*;*1*)
by (*simp add*: *assms comp-assoc is-vector-def point-def*)

lemma *singleton-transp*:
assumes *singleton p*
shows *singleton* (p^T)
by (*simp add*: *assms*)

lemma *point-to-singleton*:
assumes *singleton p*
shows *singleton* ($1' \cdot p;p^T$)
using *assms dom-def-aux-var dom-one is-vector-def point-def* **by** *fastforce*

lemma *singleton-singletonT*:
assumes *singleton p*
shows $p;p^T \leq 1'$

using *assms singleton-injective is-inj-def* **by** *blast*

Minimality

abbreviation *minimum*
  where *minimum x v* ≡ *v* · −(x^T;*v*)

Regressively finite

abbreviation *regressively-finite*
  where *regressively-finite x* ≡ ∀ *v* . *is-vector v* ∧ *v* ≤ x^T;*v* ⟶ *v* = *0*

lemma *regressively-finite-minimum*:
  regressively-finite R ⟹ *is-vector v* ⟹ *v* ≠ *0* ⟹ *minimum R v* ≠ *0*
using *galois-aux2* **by** *blast*

lemma *regressively-finite-irreflexive*:
  assumes *regressively-finite x*
    shows *x* ≤ −*1'*
proof −
  have *1*: *is-vector* ((x^T · *1'*);*1*)
    by (*simp add*: *is-vector-def mult-assoc*)
  have (x^T · *1'*);*1* = (x^T · *1'*);(x^T · *1'*);*1*
    by (*simp add*: *is-test-def test-comp-eq-mult*)
  with *1* **have** (x^T · *1'*);*1* = *0*
    by (*metis assms comp-assoc mult-subdistr*)
  thus *?thesis*
    by (*metis conv-e conv-invol conv-times conv-zero galois-aux ss-p18*)
qed

end

1.1 Relation algebras satisfying the Tarski rule

class *relation-algebra-tarski* = *relation-algebra* +
  assumes *tarski*: *x* ≠ *0* ⟷ *1*;*x*;*1* = *1*
begin

Some (more) results about points

lemma *point-equations*:
  assumes *is-point p*
  shows *p*;*1*=*p*
    and *1*;*p*=*1*
    and p^T;*1*=*1*
    and *1*;p^T=p^T
   apply (*metis assms is-point-def is-vector-def*)
  using *assms is-point-def is-vector-def tarski vector-comp* **apply** *fastforce*
 apply (*metis assms conv-contrav conv-one conv-zero is-point-def is-vector-def tarski*)
by (*metis assms conv-contrav conv-one is-point-def is-vector-def*)

The following result is [10, Proposition 2.4.5(i)].

```
lemma point-singleton:
  assumes is-point p
      and is-vector v
      and v ≠ 0
      and v ≤ p
    shows v = p
proof −
  have 1;v = 1
    using assms(2,3) comp-assoc is-vector-def tarski by fastforce
  hence p = 1;v · p
    by simp
  also have ... ≤ (1 · p;v^T);(v · 1^T;p)
    using dedekind by blast
  also have ... ≤ p;v^T;v
    by (simp add: mult-subdistl)
  also have ... ≤ p;p^T;v
    using assms(4) conv-iso mult-double-iso by blast
  also have ... ≤ v
    by (metis assms(1) is-inj-def is-point-def mult-isor mult-onel)
  finally show ?thesis
    using assms(4) by simp
qed

lemma point-not-equal-aux:
  assumes is-point p
      and is-point q
    shows p≠q ⟷ p · −q ≠ 0
proof
  show p ≠ q ⟹ p · − q ≠ 0
  proof (rule contrapos-nn)
    assume p · −q = 0
    thus p = q
     using assms galois-aux2 is-point-def point-singleton by fastforce
  qed
next
  show p · − q ≠ 0 ⟹ p ≠ q
    using inf-compl-bot by blast
qed
```

The following result is part of [10, Proposition 2.4.5(ii)].

```
lemma point-not-equal:
  assumes is-point p
      and is-point q
    shows p≠q ⟷ p≤−q
      and p≤−q ⟷ p;q^T ≤ −1'
      and p;q^T ≤ −1' ⟷ p^T;q ≤ 0
proof −
  have p ≠ q ⟹ p ≤ − q
    by (metis assms point-not-equal-aux is-point-def vector-compl vector-mult
```

```
point-singleton
          inf.orderI inf.cobounded1)
  thus p≠q ⟷ p≤−q
    by (metis assms(1) galois-aux inf.orderE is-point-def order.refl)
next
  show (p ≤ − q) = (p ; q^T ≤ − 1')
    by (simp add: conv-galois-2)
next
  show (p ; q^T ≤ − 1') = (p^T ; q ≤ 0)
    by (metis assms(2) compl-bot-eq conv-galois-2 galois-aux maddux-141
mult-1-right
          point-equations(4))
qed

lemma point-is-point:
  point x ⟷ is-point x
apply (rule iffI)
 apply (simp add: is-point-def point-def surj-one tarski)
using is-point-def is-vector-def mult-assoc point-def sur-def-var1 tarski by
fastforce

lemma point-in-vector-or-complement:
  assumes point p
      and is-vector v
    shows p ≤ v ∨ p ≤ −v
proof (cases p ≤ −v)
  assume p ≤ −v
  thus ?thesis
    by simp
next
  assume ¬(p ≤ −v)
  hence p·v ≠ 0
    by (simp add: galois-aux)
  hence 1;(p·v) = 1
    using assms comp-assoc is-vector-def point-def tarski vector-mult by fastforce
  hence p ≤ p;(p·v)^T;(p·v)
    by (metis inf-top.left-neutral modular-2-var)
  also have ... ≤ p;p^T;v
    by (simp add: mult-isol-var)
  also have ... ≤ v
    using assms(1) comp-assoc point-def ss423conv by fastforce
  finally show ?thesis ..
qed

lemma point-in-vector-or-complement-iff:
  assumes point p
      and is-vector v
    shows p ≤ v ⟷ ¬(p ≤ −v)
by (metis assms annir compl-top-eq galois-aux inf.orderE one-compl point-def
```

ss423conv tarski
top-greatest point-in-vector-or-complement)

lemma *different-points-consequences*:
assumes *point p*
and *point q*
and $p \neq q$
shows $p^T;-q=1$
and $-q^T;p=1$
and $-(p^T;-q)=0$
and $-(-q^T;p)=0$
proof −
have $p \leq -q$
by (*metis assms compl-le-swap1 inf.absorb1 inf.absorb2 point-def point-in-vector-or-complement*)
thus *1*: $p^T;-q=1$
using *assms*(*1*) **by** (*metis is-vector-def point-def ss423conv top-le*)
thus *2*: $-q^T;p=1$
using *conv-compl conv-one* **by** *force*
from *1* **show** $-(p^T;-q)=0$
by *simp*
from *2* **show** $-(-q^T;p)=0$
by *simp*
qed

Some (more) results about singletons

lemma *singleton-pq*:
assumes *point p*
and *point q*
shows *singleton* $(p;q^T)$
using *assms comp-assoc point-def point-equations*(*1*,*3*) *point-is-point* **by** *fastforce*

lemma *singleton-equal-aux*:
assumes *singleton p*
and *singleton q*
and $q \leq p$
shows $p \leq q;1$
proof −
have *pLp*: $p;1;p^T \leq 1'$
by (*simp add*: *assms*(*1*) *maddux-21 ss423conv*)

have $p = 1;(q^T;q;1) \cdot p$
using *tarski*
by (*metis assms*(*2*) *annir singleton-injective inf.commute inf-top.right-neutral inj-triple*
mult-assoc surj-one)
also have ... $\leq (1 \cdot p;(q^T;q;1)^T);(q^T;q;1 \cdot 1;p)$
using *dedekind* **by** (*metis conv-one*)
also have ... $\leq p;1;q^T;q;q^T;q;1$

```
    by (simp add: comp-assoc mult-isol)
  also have ... ≤ p;1;p^T;q;q^T;q;1
    using assms(3) by (metis comp-assoc conv-iso mult-double-iso)
  also have ... ≤ 1';q;q^T;q;1
    using pLp using mult-isor by blast
  also have ... ≤ q;1
    using assms(2) singleton-singletonT by (simp add: comp-assoc mult-isol)
  finally show ?thesis .
qed

lemma singleton-equal:
 assumes singleton p
     and singleton q
     and q≤p
   shows q=p
proof −
  have p1: p ≤ q;1
    using assms by (rule singleton-equal-aux)
  have p^T ≤ q^T;1
    using assms singleton-equal-aux singleton-transp conv-iso by fastforce
  hence p2: p ≤ 1;q
    using conv-iso by force

  have p ≤ q;1 · 1;q
    using p1 p2 inf.boundedI by blast
  also have ... ≤ (q · 1;q;1);(1 · q^T;1;q)
    using dedekind by (metis comp-assoc conv-one)
  also have ... ≤ q;q^T;1;q
    by (simp add: mult-isor comp-assoc)
  also have ... ≤ q;1'
    by (metis assms(2) conv-contrav conv-invol conv-one is-inj-def mult-assoc
mult-isol
              one-idem-mult)
  also have ... ≤ q
    by simp
  finally have p ≤ q .
  thus q=p
  using assms(3) by simp
qed

lemma singleton-nonsplit:
  assumes singleton p
      and x≤p
    shows x=0 ∨ x=p
proof (cases x=0)
  assume x=0
  thus ?thesis ..
next
  assume 1: x≠0
```

```
  have singleton x
  proof (safe)
    show is-inj (x;1)
      using assms injective-down-closed mult-isor by blast
    show is-inj (x^T;1)
      using assms conv-iso injective-down-closed mult-isol-var by blast
    show is-sur (x;1)
      using 1 comp-assoc sur-def-var1 tarski by fastforce
    thus is-sur (x^T;1)
      by (metis conv-contrav conv-one mult.semigroup-axioms sur-def-var1
semigroup.assoc)
  qed
  thus ?thesis
    using assms singleton-equal by blast
qed

lemma singleton-nonzero:
  assumes singleton p
    shows p≠0
proof
  assume p = 0
  hence point 0
    using assms singleton-is-point by fastforce
  thus False
    by (simp add: is-point-def point-is-point)
qed

lemma singleton-sum:
  assumes singleton p
    shows p ≤ x+y ⟷ (p≤x ∨ p≤y)
proof
  show p ≤ x + y ⟹ p ≤ x ∨ p ≤ y
  proof −
    assume as: p ≤ x + y
    show p ≤ x ∨ p ≤ y
    proof (cases p≤x)
      assume p≤x
      thus ?thesis ..
    next
      assume a:¬(p≤x)
      hence p·x ≠ p
        using a inf.orderI by fastforce
      hence p ≤ −x
        using assms singleton-nonsplit galois-aux inf-le1 by blast
      hence p≤y
        using as by (metis galois-1 inf.orderE)
      thus ?thesis
        by simp
    qed
```

```
  qed
next
  show p ≤ x ∨ p ≤ y ⟹ p ≤ x + y
    using sup.coboundedI1 sup.coboundedI2 by blast
qed

lemma singleton-iff:
 singleton x ⟷ x ≠ 0 ∧ x^T;1;x + x;1;x^T ≤ 1'
by (smt comp-assoc conv-contrav conv-invol conv-one is-inj-def le-sup-iff
one-idem-mult
        sur-def-var1 tarski)

lemma singleton-not-atom-in-relation-algebra-tarski:
 assumes p≠0
     and ∀ x . x≤p ⟶ x=0 ∨ x=p
   shows singleton p
nitpick [expect=genuine] oops

end
```

1.2 Relation algebras satisfying the point axiom

```
class relation-algebra-point = relation-algebra +
  assumes point-axiom: x ≠ 0 ⟶ (∃ y z . point y ∧ point z ∧ y;z^T ≤ x)
begin
```

Some (more) results about points

```
lemma point-exists:
  ∃ x . point x
by (metis (full-types) eq-iff is-inj-def is-sur-def is-vector-def point-axiom
point-def)

lemma point-below-vector:
  assumes is-vector v
      and v ≠ 0
    shows ∃ x . point x ∧ x ≤ v
proof –
  from assms(2) obtain y and z where 1: point y ∧ point z ∧ y;z^T ≤ v
    using point-axiom by blast
  have z^T;1 = (1;z)^T
    using conv-contrav conv-one by simp
  hence y;(1;z)^T ≤ v
    using 1 by (metis assms(1) comp-assoc is-vector-def mult-isor)
  thus ?thesis
    using 1 by (metis conv-one is-vector-def point-def sur-def-var1)
qed

end
```

class *relation-algebra-tarski-point* = *relation-algebra-tarski* +
relation-algebra-point
begin

lemma *atom-is-singleton*:
  assumes $p{\neq}0$
    and $\forall x . x{\leq}p \longrightarrow x{=}0 \vee x{=}p$
   shows *singleton p*
by (*metis assms singleton-nonzero singleton-pq point-axiom*)

lemma *singleton-iff-atom*:
  singleton p $\longleftrightarrow p{\neq}0 \wedge (\forall x . x{\leq}p \longrightarrow x{=}0 \vee x{=}p)$
using *singleton-nonsplit singleton-nonzero atom-is-singleton* **by** *blast*

lemma *maddux-tarski*:
  assumes $x{\neq}0$
  shows $\exists y . y{\neq}0 \wedge y{\leq}x \wedge$ *is-p-fun* y
proof –
  obtain *p q* **where** *1*: *point p* ∧ *point q* ∧ $p;q^T \leq x$
   using *assms point-axiom* **by** *blast*
  hence *2*: $p;q^T{\neq}0$
   by (*simp add*: *singleton-nonzero singleton-pq*)
  have *is-p-fun* $(p;q^T)$
   using *1* **by** (*meson singleton-singletonT singleton-pq singleton-transp*
is-inj-def p-fun-inj)
  thus *?thesis*
   using *1 2* **by** *force*
qed

Intermediate Point Theorem [10, Proposition 2.4.8]

lemma *intermediate-point-theorem*:
  assumes *point p*
    and *point r*
   shows $p \leq x;y;r \longleftrightarrow (\exists q .$ *point q* $\wedge p \leq x;q \wedge q \leq y;r)$
proof
  assume *1*: $p \leq x;y;r$
  let $?v = x^T;p \cdot y;r$
  have *2*: *is-vector ?v*
   using *assms comp-assoc is-vector-def point-def vector-mult* **by** *fastforce*
  have $?v \neq 0$
   using *1* **by** (*metis assms*(*1*) *inf.absorb2 is-point-def maddux-141*
point-is-point mult.assoc)
  hence $\exists q .$ *point q* $\wedge q \leq ?v$
   using *2 point-below-vector* **by** *blast*
  thus $\exists q .$ *point q* $\wedge p \leq x;q \wedge q \leq y;r$
   using *assms*(*1*) *point-swap* **by** *auto*
next
  assume $\exists q .$ *point q* $\wedge p \leq x;q \wedge q \leq y;r$
  thus $p \leq x;y;r$

using *comp-assoc mult-isol order-trans* **by** *fastforce*
qed

end

context *relation-algebra*
begin

lemma *unfoldl-inductl-implies-unfoldr*:
assumes $\bigwedge x.\ 1' + x;(rtc\ x) \leq rtc\ x$
and $\bigwedge x\ y\ z.\ x+y;z \leq z \Longrightarrow rtc(y);x \leq z$
shows $1' + rtc(x);x \leq rtc\ x$
by (*metis assms le-sup-iff mult-oner order.trans subdistl-eq sup-absorb2 sup-ge1*)

lemma *star-transpose-swap*:
assumes $\bigwedge x.\ 1' + x;(rtc\ x) \leq rtc\ x$
and $\bigwedge x\ y\ z.\ x+y;z \leq z \Longrightarrow rtc(y);x \leq z$
shows $rtc(x^T) = (rtc\ x)^T$
apply(*simp only*: *eq-iff*; *rule conjI*)
apply (*metis assms conv-add conv-contrav conv-e conv-iso mult-1-right unfoldl-inductl-implies-unfoldr*)
by (*metis assms conv-add conv-contrav conv-e conv-invol conv-iso mult-1-right unfoldl-inductl-implies-unfoldr*)

lemma *unfoldl-inductl-implies-inductr*:
assumes $\bigwedge x.\ 1' + x;(rtc\ x) \leq rtc\ x$
and $\bigwedge x\ y\ z.\ x+y;z \leq z \Longrightarrow rtc(y);x \leq z$
shows $x+z;y \leq z \Longrightarrow x;rtc(y) \leq z$
by (*metis assms conv-add conv-contrav conv-iso star-transpose-swap*)

end

context *relation-algebra-rtc*
begin

abbreviation *tc* ((-$^+$) [*101*] *100*) **where** $tc\ x \equiv x;x^\star$

abbreviation *is-acyclic*
where $is\text{-}acyclic\ x \equiv x^+ \leq -1'$

General theorems

lemma *star-denest-10*:
assumes $x;y=0$
shows $(x+y)^\star = y;y^\star;x^\star+x^\star$
using *assms bubble-sort sup.commute* **by** *auto*

lemma *star-star-plus*:

$x^\star + y^\star = x^+ + y^\star$
by (*metis* (*full-types*) *sup.left-commute star-plus-one star-unfoldl-eq sup.commute*)

The following two lemmas are from [6].

lemma *cancel-separate*:
assumes $x ; y \leq 1'$
shows $x^\star ; y^\star \leq x^\star + y^\star$
proof −
have $x ; y^\star = x + x ; y ; y^\star$
by (*metis comp-assoc conway.dagger-unfoldl-distr distrib-left mult-oner*)
also have ... $\leq x + y^\star$
by (*metis assms join-isol star-invol star-plus-one star-subdist-var-2 sup.absorb2 sup.assoc*)
also have ... $\leq x^\star + y^\star$
using *join-iso* **by** *fastforce*
finally have $x ; (x^\star + y^\star) \leq x^\star + y^\star$
by (*simp add*: *distrib-left le-supI1*)
thus *?thesis*
by (*simp add*: *rtc-inductl*)
qed

lemma *cancel-separate-inj-converse*:
assumes *is-inj x*
shows $x^\star ; x^{T\star} = x^\star + x^{T\star}$
apply (*rule antisym*)
using *assms cancel-separate is-inj-def* **apply** *blast*
by (*metis conway.dagger-unfoldl-distr le-supI mult-1-right mult-isol sup.cobounded1*)

lemma *cancel-separate-p-fun-converse*:
assumes *is-p-fun x*
shows $x^{T\star} ; x^\star = x^\star + x^{T\star}$
using *sup-commute assms cancel-separate-inj-converse p-fun-inj* **by** *fastforce*

lemma *cancel-separate-converse-idempotent*:
assumes *is-inj x*
and *is-p-fun x*
shows $(x^\star + x^{T\star});(x^\star + x^{T\star}) = x^\star + x^{T\star}$
by (*metis assms cancel-separate cancel-separate-p-fun-converse church-rosser-equiv is-inj-def star-denest-var-6*)

lemma *triple-star*:
assumes *is-inj x*
and *is-p-fun x*
shows $x^\star;x^{T\star};x^\star = x^\star + x^{T\star}$
by (*simp add*: *assms cancel-separate-inj-converse cancel-separate-p-fun-converse*)

lemma *inj-xxts*:
assumes *is-inj x*
shows $x;x^{T\star} \leq x^\star + x^{T\star}$
by (*metis assms cancel-separate-inj-converse distrib-right less-eq-def star-ext*)

lemma *plus-top*:
$x^+;1 = x;1$
by (*metis comp-assoc conway.dagger-unfoldr-distr sup-top-left*)

lemma *top-plus*:
$1;x^+ = 1;x$
by (*metis comp-assoc conway.dagger-unfoldr-distr star-denest-var-2 star-ext star-slide-var*
sup-top-left top-unique)

lemma *plus-conv*:
$(x^+)^T = x^{T+}$
by (*simp add*: *star-conv star-slide-var*)

lemma *inj-implies-step-forwards-backwards*:
assumes *is-inj x*
shows $x^\star;(x^+ \cdot 1');1 \leq x^T;1$
proof –
have $(x^+ \cdot 1');1 \leq (x^\star \cdot x^T);(x \cdot (x^\star)^T);1$
by (*metis conv-contrav conv-e dedekind mult-1-right mult-isor star-slide-var*)
also have ... $\leq (x^\star \cdot x^T);1$
by (*simp add*: *comp-assoc mult-isol*)
finally have *1*: $(x^+ \cdot 1');1 \leq (x^\star \cdot x^T);1$.

have $x;(x^\star \cdot x^T);1 \leq (x^+ \cdot x;x^T);1$
by (*metis inf-idem meet-interchange mult-isor*)
also have ... $\leq (x^+ \cdot 1');1$
using *assms is-inj-def meet-isor mult-isor* **by** *fastforce*
finally have $x;(x^\star \cdot x^T);1 \leq (x^\star \cdot x^T);1$
using *1* **by** *fastforce*
hence $x^\star;(x^+ \cdot 1');1 \leq (x^\star \cdot x^T);1$
using *1* **by** (*simp add*: *comp-assoc rtc-inductl*)
thus $x^\star;(x^+ \cdot 1');1 \leq x^T;1$
using *inf.cobounded2 mult-isor order-trans* **by** *blast*
qed

Acyclic relations

The following result is from [4].

lemma *acyclic-inv*:
assumes *is-acyclic t*
and *is-vector v*
and $e \leq v;-v^T$
and $t \leq v;v^T$
shows *is-acyclic* $(t + e)$

proof −
  have $t^+;e \leq t^+;v;-v^T$
    by (*simp add*: *assms*(*3*) *mult-assoc mult-isol*)
  also have ... $\leq v;v^T;t^\star;v;-v^T$
    by (*simp add*: *assms*(*4*) *mult-isor*)
  also have ... $\leq v;-v^T$
    by (*metis assms*(*2*) *mult-double-iso top-greatest is-vector-def mult-assoc*)
  also have ... $\leq -1'$
    by (*simp add*: *conv-galois-1*)
  finally have *1*: $t^+;e \leq -1'$ **.**
  have $e \leq v;-v^T$
    using *assms*(*3*) **by** *simp*
  also have ... $\leq -1'$
    by (*simp add*: *conv-galois-1*)
  finally have *2*: $t^+;e + e \leq -1'$
    using *1* **by** *simp*
  have *3*: $e;t^\star = e$
    by (*metis assms*(*2*−*4*) *et*(*1*) *independence2*)
  have *4*: $e^\star = 1' + e$
    using *assms*(*2*−*3*) *ee boffa-var bot-least* **by** *blast*
  have $(t + e)^+ = (t + e);t^\star;(e;t^\star)^\star$
    by (*simp add*: *comp-assoc*)
  also have ... $= (t + e);t^\star;(1' + e)$
    using *3 4* **by** *simp*
  also have ... $= t^+;(1' + e) + e;t^\star;(1' + e)$
    by *simp*
  also have ... $= t^+;(1' + e) + e;(1' + e)$
    using *3* **by** *simp*
  also have ... $= t^+;(1' + e) + e$
    using *4 assms*(*2*−*3*) *ee independence2* **by** *fastforce*
  also have ... $= t^+ + t^+;e + e$
    by (*simp add*: *distrib-left*)
  also have ... $\leq -1'$
    using *assms*(*1*) *2* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *acyclic-single-step*:
  assumes *is-acyclic x*
    shows $x \leq -1'$
by (*metis assms dual-order.trans mult-isol mult-oner star-ref*)

lemma *acyclic-reachable-points*:
  assumes *is-point p*
      and *is-point q*
      and $p \leq x;q$
      and *is-acyclic x*
    shows $p \neq q$
proof

assume $p=q$
hence $p \leq x;q \cdot q$
by (*simp add*: *assms*(*3*) *eq-iff inf.absorb2*)
also have ... $= (x \cdot 1');q$
using *assms*(*2*) *inj-distr is-point-def* **by** *simp*
also have ... $\leq (-1' \cdot 1');q$
using *acyclic-single-step assms*(*4*) **by** (*metis abel-semigroup.commute inf.abel-semigroup-axioms*
meet-isor mult-isor)
also have ... $= 0$
by *simp*
finally have $p \leq 0$ **.**
thus *False*
using *assms*(*1*) *bot-unique is-point-def* **by** *blast*
qed

lemma *acyclic-trans*:
assumes *is-acyclic x*
shows $x \leq -(x^{T+})$
proof −
have $\exists\, c \geq x.\ c \leq -\ (x^+)^T$
by (*metis assms compl-mono conv-galois-2 conv-iso double-compl mult-onel star-1l*)
thus *?thesis*
by (*metis dual-order.trans plus-conv*)
qed

lemma *acyclic-trans'*:
assumes *is-acyclic x*
shows $x^\star \leq -(x^{T+})$
proof −
have $x^\star \leq -\ (-\ (-\ (x^T\ ;\ -\ (-\ 1')))\ ;\ (x^\star)^T)$
by (*metis assms conv-galois-1 conv-galois-2 order-trans star-trans*)
then show *?thesis*
by (*simp add*: *star-conv*)
qed

Regressively finite

lemma *regressively-finite-acyclic*:
assumes *regressively-finite x*
shows *is-acyclic x*
proof −
have *1*: *is-vector* $((x^+ \cdot 1');1)$
by (*simp add*: *is-vector-def mult-assoc*)
have $(x^+ \cdot 1');1 = (x^{T+} \cdot 1');1$
by (*metis plus-conv test-converse*)
also have ... $\leq x^T;(1';x^{T\star} \cdot x);1$
by (*metis conv-invol modular-1-var mult-isor mult-oner mult-onel*)
also have ... $\leq x^T;(1' \cdot x^+);x^{T\star};1$

by (*metis comp-assoc conv-invol modular-2-var mult-isol mult-isor star-conv*)
also have ... = x^T;($x^+ \cdot 1'$);*1*
by (*metis comp-assoc conway.dagger-unfoldr-distr inf.commute sup.cobounded1 top-le*)
finally have ($x^+ \cdot 1'$);*1* = *0*
using *1 assms* **by** (*simp add*: *comp-assoc*)
thus *?thesis*
by (*simp add*: *galois-aux ss-p18*)
qed

notation *power* (**infixr** ↑ *80*)

lemma *power-suc-below-plus*:
x ↑ *Suc n* ≤ x^+
apply (*induct n*)
using *mult-isol star-ref* **apply** *fastforce*
by (*simp add*: *mult-isol-var order-trans*)

end

class *relation-algebra-rtc-tarski* = *relation-algebra-rtc* + *relation-algebra-tarski*
begin

lemma *point-loop-not-acyclic*:
assumes *is-point p*
and p ≤ x ↑ *Suc n* ; p
shows ¬ *is-acyclic x*
proof −
have $p \leq x^+$; p
by (*meson assms dual-order.trans point-def point-is-point ss423bij power-suc-below-plus*)
hence p ; $p^T \leq x^+$
using *assms*(*1*) *point-def point-is-point ss423bij* **by** *blast*
thus *?thesis*
using *assms*(*1*) *order.trans point-not-equal*(*1*) *point-not-equal*(*2*) **by** *blast*
qed

end

class *relation-algebra-rtc-point* = *relation-algebra-rtc* + *relation-algebra-point*

class *relation-algebra-rtc-tarski-point* = *relation-algebra-rtc-tarski* + *relation-algebra-rtc-point* +
relation-algebra-tarski-point

Finite graphs: the axiom says the algebra has finitely many elements. This means the relations have a finite base set.

class *relation-algebra-rtc-tarski-point-finite* = *relation-algebra-rtc-tarski-point* + *finite*

begin

For a finite acyclic relation, the powers eventually vanish.

lemma *acyclic-power-vanishes*:
assumes *is-acyclic x*
shows $\exists\, n\ .\ x \uparrow Suc\ n = 0$
proof –
let $?n = card\ \{\ p\ .\ is\text{-}point\ p\ \}$
let $?p = x \uparrow ?n$
have $?p = 0$
proof (*rule ccontr*)
assume $?p \neq 0$
from *this* **obtain** *p q* **where** *1*: $point\ p \wedge point\ q \wedge p;q^T \leq ?p$
using *point-axiom* **by** *blast*
hence *2*: $p \leq ?p;q$
using *point-def ss423bij* **by** *blast*
have $\forall\, n{\leq}?n\ .\ (\exists f.\ \forall\, i{\leq}n\ .\ is\text{-}point\ (f\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x{\uparrow}(?n{-}i)\ ;\ f\ i \wedge f\ i \leq x{\uparrow}(i{-}j)\ ;\ f\ j))$
proof
fix *n*
show $n{\leq}?n \longrightarrow (\exists f.\ \forall\, i{\leq}n\ .\ is\text{-}point\ (f\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x{\uparrow}(?n{-}i)\ ;\ f\ i \wedge f\ i \leq x{\uparrow}(i{-}j)\ ;\ f\ j))$
proof (*induct n*)
case *0*
thus *?case*
using *1 2 point-is-point* **by** *fastforce*
next
case (*Suc n*)
fix *n*
assume *3*: $n{\leq}?n \longrightarrow (\exists f\ .\ \forall\, i{\leq}n\ .\ is\text{-}point\ (f\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x \uparrow (?n{-}i)\ ;\ f\ i \wedge f\ i \leq x \uparrow (i{-}j)\ ;\ f\ j))$
show $Suc\ n{\leq}?n \longrightarrow (\exists f\ .\ \forall\, i{\leq}Suc\ n\ .\ is\text{-}point\ (f\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x \uparrow (?n{-}i)\ ;\ f\ i \wedge f\ i \leq x \uparrow (i{-}j)\ ;\ f\ j))$
proof
assume *4*: $Suc\ n{\leq}?n$
from *this* **obtain** *f* **where** *5*: $\forall\, i{\leq}n\ .\ is\text{-}point\ (f\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x \uparrow (?n{-}i)\ ;\ f\ i \wedge f\ i \leq x \uparrow (i{-}j)\ ;\ f\ j)$
using *3* **by** *auto*
have $p \leq x \uparrow (?n{-}n)\ ;\ f\ n$
using *5* **by** *blast*
also have ... $= x \uparrow (?n{-}n{-}one\text{-}class.one)\ ;\ x\ ;\ f\ n$
using *4* **by** (*metis* (*no-types*) *Suc-diff-le diff-Suc-1 diff-Suc-Suc power-Suc2*)
finally obtain *r* **where** *6*: $point\ r \wedge p \leq x \uparrow (?n{-}Suc\ n)\ ;\ r \wedge r \leq x\ ;\ f\ n$
using *1 5 intermediate-point-theorem point-is-point* **by** *fastforce*
let $?g = \lambda m\ .\ if\ m = Suc\ n\ then\ r\ else\ f\ m$
have $\forall\, i{\leq}Suc\ n\ .\ is\text{-}point\ (?g\ i) \wedge (\forall\, j{\leq}i\ .\ p \leq x \uparrow (?n{-}i)\ ;\ ?g\ i \wedge ?g\ i \leq x \uparrow (i{-}j)\ ;\ ?g\ j)$

```
proof
  fix i
  show i≤Suc n ⟶ is-point (?g i) ∧ (∀ j≤i . p ≤ x ↑ (?n−i) ; ?g i ∧ ?g i ≤ x ↑ (i−j) ; ?g j)
  proof (cases i≤n)
    case True
    thus ?thesis
      using 5 by simp
  next
    case False
    have is-point (?g (Suc n)) ∧ (∀ j≤Suc n . p ≤ x ↑ (?n−Suc n) ; ?g (Suc n) ∧ ?g (Suc n) ≤ x ↑ (Suc n−j) ; ?g j)
    proof
      show is-point (?g (Suc n))
        using 6 point-is-point by fastforce
    next
      show ∀ j≤Suc n . p ≤ x ↑ (?n−Suc n) ; ?g (Suc n) ∧ ?g (Suc n) ≤ x ↑ (Suc n−j) ; ?g j
      proof
        fix j
        show j≤Suc n ⟶ p ≤ x ↑ (?n−Suc n) ; ?g (Suc n) ∧ ?g (Suc n) ≤ x ↑ (Suc n−j) ; ?g j
        proof
          assume 7: j≤Suc n
          show p ≤ x ↑ (?n−Suc n) ; ?g (Suc n) ∧ ?g (Suc n) ≤ x ↑ (Suc n−j) ; ?g j
          proof
            show p ≤ x ↑ (?n−Suc n) ; ?g (Suc n)
              using 6 by simp
          next
            show ?g (Suc n) ≤ x ↑ (Suc n−j) ; ?g j
            proof (cases j = Suc n)
              case True
              thus ?thesis
                by simp
            next
              case False
              hence f n ≤ x ↑ (n−j) ; f j
                using 5 7 by fastforce
              hence x ; f n ≤ x ↑ (Suc n−j) ; f j
                using 7 False Suc-diff-le comp-assoc mult-isol by fastforce
              thus ?thesis
                using 6 False by fastforce
            qed
          qed
        qed
      qed
    qed
    thus ?thesis
```

```
            by (simp add: False le-Suc-eq)
          qed
        qed
        thus ∃f . ∀i≤Suc n . is-point (f i) ∧ (∀j≤i . p ≤ x ↑ (?n−i) ; f i ∧ f i
≤ x ↑ (i−j) ; f j)
          by auto
      qed
    qed
  qed
  from this obtain f where 8: ∀i≤?n . is-point (f i) ∧ (∀j≤i . p ≤ x ↑
(?n−i) ; f i ∧ f i ≤ x ↑ (i−j) ; f j)
    by fastforce
  let ?A = { k . k≤?n }
  have f ‘ ?A ⊆ { p . is-point p }
    using 8 by blast
  hence card (f ‘ ?A) ≤ ?n
    by (simp add: card-mono)
  hence ¬ inj-on f ?A
    by (simp add: pigeonhole)
  from this obtain i j where 9: i ≤ ?n ∧ j ≤ ?n ∧ i ≠ j ∧ f i = f j
    by (metis (no-types, lifting) inj-on-def mem-Collect-eq)
  show False
    apply (cases i < j)
   using 8 9 apply (metis Suc-diff-le Suc-leI assms diff-Suc-Suc
order-less-imp-le
                      point-loop-not-acyclic)
  using 8 9 by (metis assms neqE point-loop-not-acyclic Suc-diff-le Suc-leI
assms diff-Suc-Suc
                    order-less-imp-le)
  qed
  thus ?thesis
    by (metis annir power.simps(2))
qed
```

Hence finite acyclic relations are regressively finite.

```
lemma acyclic-regressively-finite:
 assumes is-acyclic x
   shows regressively-finite x
proof
 have is-acyclic (x^T)
   using assms acyclic-trans' compl-le-swap1 order-trans star-ref by blast
 from this obtain n where 1: x^T ↑ Suc n = 0
   using acyclic-power-vanishes by fastforce
 fix v
 show is-vector v ∧ v ≤ x^T;v ⟶ v = 0
 proof
   assume 2: is-vector v ∧ v ≤ x^T;v
   have v ≤ x^T ↑ Suc n ; v
   proof (induct n)
```

```
      case 0
      thus ?case
        using 2 by simp
    next
      case (Suc n)
      hence x^T ; v ≤ x^T ↑ Suc (Suc n) ; v
        by (simp add: comp-assoc mult-isol)
      thus ?case
        using 2 dual-order.trans by blast
    qed
    thus v = 0
      using 1 by (simp add: le-bot)
  qed
qed

lemma acyclic-is-regressively-finite:
  is-acyclic x ⟷ regressively-finite x
using acyclic-regressively-finite regressively-finite-acyclic by blast

end

end
```

2 Relational Characterisation of Paths

This theory provides the relation-algebraic characterisations of paths, as defined in Sections 3–5 of [2].

```
theory Paths

imports More-Relation-Algebra

begin

context relation-algebra-tarski
begin

lemma path-concat-aux-0:
  assumes is-vector v
      and v ≠ 0
      and w;v^T ≤ x
      and v;z ≤ y
    shows w;1;z ≤ x;y
proof -
  from tarski assms(1,2) have 1 = 1;v^T;v;1
    by (metis conv-contrav conv-one eq-refl inf-absorb1 inf-top-left is-vector-def
ra-2)
  hence w;1;z = w;1;v^T;v;1;z
    by (simp add: mult-isor mult-isol mult-assoc)
```

also from *assms*(*1*) **have** ... $= w;v^T;v;z$
  by (*metis is-vector-def comp-assoc conv-contrav conv-one*)
also from *assms*(*3*) **have** ... $\leq x;v;z$
  by (*simp add*: *mult-isor*)
also from *assms*(*4*) **have** ... $\leq x;y$
  by (*simp add*: *mult-isol mult-assoc*)
finally show *?thesis* **.**
qed

end

2.1 Consequences without the Tarski rule

context *relation-algebra-rtc*
begin

Definitions for path classifications

abbreviation *connected*
  where *connected* $x \equiv x;1;x \leq x^\star + x^{T\star}$

abbreviation *many-strongly-connected*
  where *many-strongly-connected* $x \equiv x^\star = x^{T\star}$

abbreviation *one-strongly-connected*
  where *one-strongly-connected* $x \equiv x^T;1;x^T \leq x^\star$

definition *path*
  where *path* $x \equiv$ *connected* $x \wedge$ *is-p-fun* $x \wedge$ *is-inj* x

abbreviation *cycle*
  where *cycle* $x \equiv$ *path* $x \wedge$ *many-strongly-connected* x

abbreviation *start-points*
  where *start-points* $x \equiv x;1 \cdot -(x^T;1)$

abbreviation *end-points*
  where *end-points* $x \equiv x^T;1 \cdot -(x;1)$

abbreviation *no-start-points*
  where *no-start-points* $x \equiv x;1 \leq x^T;1$

abbreviation *no-end-points*
  where *no-end-points* $x \equiv x^T;1 \leq x;1$

abbreviation *no-start-end-points*
  where *no-start-end-points* $x \equiv x;1 = x^T;1$

abbreviation *has-start-points*
  where *has-start-points* $x \equiv 1 = -(1;x);x;1$

abbreviation *has-end-points*
where $\mathit{has\text{-}end\text{-}points}\ x \equiv 1 = 1;x;-(x;1)$

abbreviation *has-start-end-points*
where $\mathit{has\text{-}start\text{-}end\text{-}points}\ x \equiv 1 = -(1;x);x;1 \cdot 1;x;-(x;1)$

abbreviation *backward-terminating*
where $\mathit{backward\text{-}terminating}\ x \equiv x \leq -(1;x);x;1$

abbreviation *forward-terminating*
where $\mathit{forward\text{-}terminating}\ x \equiv x \leq 1;x;-(x;1)$

abbreviation *terminating*
where $\mathit{terminating}\ x \equiv x \leq -(1;x);x;1 \cdot 1;x;-(x;1)$

abbreviation *backward-finite*
where $\mathit{backward\text{-}finite}\ x \equiv x \leq x^{T\star} + -(1;x);x;1$

abbreviation *forward-finite*
where $\mathit{forward\text{-}finite}\ x \equiv x \leq x^{T\star} + 1;x;-(x;1)$

abbreviation *finite*
where $\mathit{finite}\ x \equiv x \leq x^{T\star} + (-(1;x);x;1 \cdot 1;x;-(x;1))$

abbreviation *no-start-points-path*
where $\mathit{no\text{-}start\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{no\text{-}start\text{-}points}\ x$

abbreviation *no-end-points-path*
where $\mathit{no\text{-}end\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{no\text{-}end\text{-}points}\ x$

abbreviation *no-start-end-points-path*
where $\mathit{no\text{-}start\text{-}end\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{no\text{-}start\text{-}end\text{-}points}\ x$

abbreviation *has-start-points-path*
where $\mathit{has\text{-}start\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{has\text{-}start\text{-}points}\ x$

abbreviation *has-end-points-path*
where $\mathit{has\text{-}end\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{has\text{-}end\text{-}points}\ x$

abbreviation *has-start-end-points-path*
where $\mathit{has\text{-}start\text{-}end\text{-}points\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{has\text{-}start\text{-}end\text{-}points}\ x$

abbreviation *backward-terminating-path*
where $\mathit{backward\text{-}terminating\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{backward\text{-}terminating}\ x$

abbreviation *forward-terminating-path*
where $\mathit{forward\text{-}terminating\text{-}path}\ x \equiv \mathit{path}\ x \wedge \mathit{forward\text{-}terminating}\ x$

abbreviation *terminating-path*

where *terminating-path x* ≡ *path x* ∧ *terminating x*

abbreviation *backward-finite-path*
where *backward-finite-path x* ≡ *path x* ∧ *backward-finite x*

abbreviation *forward-finite-path*
where *forward-finite-path x* ≡ *path x* ∧ *forward-finite x*

abbreviation *finite-path*
where *finite-path x* ≡ *path x* ∧ *finite x*

General properties

lemma *reachability-from-z-in-y*:
assumes $x \leq y^\star;z$
and $x \cdot z = 0$
shows $x \leq y^+;z$
by (*metis assms conway.dagger-unfoldl-distr galois-1 galois-aux inf.orderE*)

lemma *reachable-imp*:
assumes *point p*
and *point q*
and $p^\star;q \leq p^{T\star};p$
shows $p \leq p^\star;q$
by (*metis assms conway.dagger-unfoldr-distr le-supE point-swap star-conv*)

Basic equivalences

lemma *no-start-end-points-iff*:
no-start-end-points x ⟷ *no-start-points x* ∧ *no-end-points x*
by *fastforce*

lemma *has-start-end-points-iff*:
has-start-end-points x ⟷ *has-start-points x* ∧ *has-end-points x*
by (*metis inf-eq-top-iff*)

lemma *terminating-iff*:
terminating x ⟷ *backward-terminating x* ∧ *forward-terminating x*
by *simp*

lemma *finite-iff*:
finite x ⟷ *backward-finite x* ∧ *forward-finite x*
by (*simp add*: *sup-inf-distrib1 inf.boundedI*)

lemma *no-start-end-points-path-iff*:
no-start-end-points-path x ⟷ *no-start-points-path x* ∧ *no-end-points-path x*
by *fastforce*

lemma *has-start-end-points-path-iff*:
has-start-end-points-path x ⟷ *has-start-points-path x* ∧ *has-end-points-path x*
using *has-start-end-points-iff* **by** *blast*

lemma *terminating-path-iff*:
terminating-path x $\longleftrightarrow$ *backward-terminating-path x* $\wedge$ *forward-terminating-path x*
by *fastforce*

lemma *finite-path-iff*:
finite-path x $\longleftrightarrow$ *backward-finite-path x* $\wedge$ *forward-finite-path x*
using *finite-iff* **by** *fastforce*

Closure under converse

lemma *connected-conv*:
connected x $\longleftrightarrow$ *connected* (x^T)
by (*metis comp-assoc conv-add conv-contrav conv-iso conv-one star-conv*)

lemma *conv-many-strongly-connected*:
many-strongly-connected x $\longleftrightarrow$ *many-strongly-connected* (x^T)
by *fastforce*

lemma *conv-one-strongly-connected*:
one-strongly-connected x $\longleftrightarrow$ *one-strongly-connected* (x^T)
by (*metis comp-assoc conv-contrav conv-iso conv-one star-conv*)

lemma *conv-path*:
path x $\longleftrightarrow$ *path* (x^T)
using *connected-conv inj-p-fun path-def* **by** *fastforce*

lemma *conv-cycle*:
cycle x $\longleftrightarrow$ *cycle* (x^T)
using *conv-path* **by** *fastforce*

lemma *conv-no-start-points*:
no-start-points x $\longleftrightarrow$ *no-end-points* (x^T)
by *simp*

lemma *conv-no-start-end-points*:
no-start-end-points x $\longleftrightarrow$ *no-start-end-points* (x^T)
by *fastforce*

lemma *conv-has-start-points*:
has-start-points x $\longleftrightarrow$ *has-end-points* (x^T)
by (*metis comp-assoc conv-compl conv-contrav conv-invol conv-one*)

lemma *conv-has-start-end-points*:
has-start-end-points x $\longleftrightarrow$ *has-start-end-points* (x^T)
by (*metis comp-assoc conv-compl conv-contrav conv-invol conv-one inf-eq-top-iff*)

lemma *conv-backward-terminating*:
backward-terminating x $\longleftrightarrow$ *forward-terminating* (x^T)

by (*metis comp-assoc conv-compl conv-contrav conv-iso conv-one*)

lemma *conv-terminating*:
terminating x ⟷ *terminating* (x^T)
apply (*rule iffI*)
apply (*metis conv-compl conv-contrav conv-one conv-times inf.commute le-iff-inf mult-assoc*)
by (*metis conv-compl conv-contrav conv-invol conv-one conv-times inf.commute le-iff-inf mult-assoc*)

lemma *conv-backward-finite*:
backward-finite x ⟷ *forward-finite* (x^T)
by (*metis comp-assoc conv-add conv-compl conv-contrav conv-iso conv-one star-conv*)

lemma *conv-finite*:
finite x ⟷ *finite* (x^T)
by (*metis finite-iff conv-backward-finite conv-invol*)

lemma *conv-no-start-points-path*:
no-start-points-path x ⟷ *no-end-points-path* (x^T)
using *conv-path* **by** *fastforce*

lemma *conv-no-start-end-points-path*:
no-start-end-points-path x ⟷ *no-start-end-points-path* (x^T)
using *conv-path* **by** *fastforce*

lemma *conv-has-start-points-path*:
has-start-points-path x ⟷ *has-end-points-path* (x^T)
using *conv-has-start-points conv-path* **by** *fastforce*

lemma *conv-has-start-end-points-path*:
has-start-end-points-path x ⟷ *has-start-end-points-path* (x^T)
using *conv-has-start-end-points conv-path* **by** *fastforce*

lemma *conv-backward-terminating-path*:
backward-terminating-path x ⟷ *forward-terminating-path* (x^T)
using *conv-backward-terminating conv-path* **by** *fastforce*

lemma *conv-terminating-path*:
terminating-path x ⟷ *terminating-path* (x^T)
using *conv-path conv-terminating* **by** *fastforce*

lemma *conv-backward-finite-path*:
backward-finite-path x ⟷ *forward-finite-path* (x^T)
using *conv-backward-finite conv-path* **by** *fastforce*

lemma *conv-finite-path*:
finite-path x ⟷ *finite-path* (x^T)

using *conv-finite conv-path* **by** *blast*

Equivalences for *connected*

lemma *connected-iff2*:
  assumes *is-inj x*
    and *is-p-fun x*
   shows *connected x* $\longleftrightarrow$ $x;1;x^T \leq x^\star + x^{T\star}$
proof
  assume *1*: *connected x*
  have $x;1;x^T \leq x;1;x;x^T$
   by (*metis conv-invol modular-var-3 vector-meet-comp-x'*)
  also have ... $\leq (x^+ + x^{T\star});x^T$
   using *1 mult-isor star-star-plus* **by** *fastforce*
  also have ... $\leq x^\star;x;x^T + x^{T\star}$
   using *join-isol star-slide-var* **by** *simp*
  also from *assms*(*1*) **have** ... $\leq x^\star + x^{T\star}$
   by (*metis is-inj-def comp-assoc join-iso mult-1-right mult-isol*)
  finally show $x;1;x^T \leq x^\star + x^{T\star}$ **.**
next
  assume *2*: $x;1;x^T \leq x^\star + x^{T\star}$
  have $x;1;x \leq x;1;x^T;x$
   by (*simp add*: *modular-var-3 vector-meet-comp-x*)
  also have ... $\leq (x^\star + x^{T+});x$
   using *2* **by** (*metis mult-isor star-star-plus sup-commute*)
  also have ... $\leq x^\star + x^{T\star};x^T;x$
   using *join-iso star-slide-var* **by** *simp*
  also from *assms*(*2*) **have** ... $\leq x^\star + x^{T\star}$
   by (*metis comp-assoc is-p-fun-def join-isol mult-1-right mult-isol*)
  finally show *connected x* **.**
qed

lemma *connected-iff3*:
  assumes *is-inj x*
    and *is-p-fun x*
   shows *connected x* $\longleftrightarrow$ $x^T;1;x \leq x^\star + x^{T\star}$
by (*metis assms connected-conv connected-iff2 inj-p-fun p-fun-inj conv-invol add-commute*)

lemma *connected-iff4*:
  connected x $\longleftrightarrow$ $x^T;1;x^T \leq x^\star + x^{T\star}$
by (*metis connected-conv conv-invol add-commute*)

lemma *connected-iff5*:
  connected x $\longleftrightarrow$ $x^+;1;x^+ \leq x^\star + x^{T\star}$
using *comp-assoc plus-top top-plus* **by** *fastforce*

lemma *connected-iff6*:
  assumes *is-inj x*
    and *is-p-fun x*

shows *connected* $x \longleftrightarrow x^+;1;(x^+)^T \le x^\star + x^{T\star}$
using *assms connected-iff2 comp-assoc plus-conv plus-top top-plus* **by** *fastforce*

lemma *connected-iff7*:
assumes *is-inj* x
and *is-p-fun* x
shows *connected* $x \longleftrightarrow (x^+)^T;1;x^+ \le x^\star + x^{T\star}$
by (*metis assms connected-iff3 conv-contrav conv-invol conv-one top-plus vector-meet-comp-x*)

lemma *connected-iff8*:
connected $x \longleftrightarrow (x^+)^T;1;(x^+)^T \le x^\star + x^{T\star}$
by (*metis connected-iff4 comp-assoc conv-contrav conv-invol conv-one plus-conv star-conv top-plus*)

Equivalences and implications for *many-strongly-connected*

lemma *many-strongly-connected-iff-1*:
many-strongly-connected $x \longleftrightarrow x^T \le x^\star$
apply (*rule iffI,simp*)
by (*metis conv-invol conv-iso eq-iff star-conv star-invol star-iso*)

lemma *many-strongly-connected-iff-2*:
many-strongly-connected $x \longleftrightarrow x^T \le x^+$
proof
assume *as*: *many-strongly-connected* x
hence $x^T \le x^\star \cdot (-(1') + x)$
by (*metis many-strongly-connected-iff-1 loop-backward-forward inf-greatest*)
also have ... $\le (x^\star \cdot -(1')) + (x^\star \cdot x)$
by (*simp add*: *inf-sup-distrib1*)
also have ... $\le x^+$
by (*metis as eq-iff mult-1-right mult-isol star-ref sup.absorb1 conv-invol eq-refl galois-1*
inf.absorb-iff1 inf.commute star-unfoldl-eq sup-mono many-strongly-connected-iff-1)
finally show $x^T \le x^+$ **.**
next
show $x^T \le x^+ \Longrightarrow$ *many-strongly-connected* x
using *order-trans star-1l many-strongly-connected-iff-1* **by** *blast*
qed

lemma *many-strongly-connected-iff-3*:
many-strongly-connected $x \longleftrightarrow x \le x^{T\star}$
by (*metis conv-invol many-strongly-connected-iff-1*)

lemma *many-strongly-connected-iff-4*:
many-strongly-connected $x \longleftrightarrow x \le x^{T+}$
by (*metis conv-invol many-strongly-connected-iff-2*)

lemma *many-strongly-connected-iff-5*:

many-strongly-connected x $\longleftrightarrow x^\star;x^T \le x^+$
by (*metis comp-assoc conv-contrav conway.dagger-unfoldr-distr star-conv star-denest-var-2 star-invol star-trans-eq star-unfoldl-eq sup.boundedE many-strongly-connected-iff-2*)

lemma *many-strongly-connected-iff-6*:
many-strongly-connected x $\longleftrightarrow x^T;x^\star \le x^+$
by (*metis dual-order.trans star-1l star-conv star-inductl-star star-invol star-slide-var many-strongly-connected-iff-1 many-strongly-connected-iff-5*)

lemma *many-strongly-connected-iff-7*:
many-strongly-connected x $\longleftrightarrow x^{T+} = x^+$
by (*metis antisym conv-invol star-slide-var star-unfoldl-eq many-strongly-connected-iff-5*)

lemma *many-strongly-connected-iff-5-eq*:
many-strongly-connected x $\longleftrightarrow x^\star;x^T = x^+$
by (*metis order.refl star-slide-var many-strongly-connected-iff-5 many-strongly-connected-iff-7*)

lemma *many-strongly-connected-iff-6-eq*:
many-strongly-connected x $\longleftrightarrow x^T;x^\star = x^+$
using *many-strongly-connected-iff-6 many-strongly-connected-iff-7* **by** *force*

lemma *many-strongly-connected-implies-no-start-end-points*:
assumes *many-strongly-connected x*
shows *no-start-end-points x*
by (*metis assms conway.dagger-unfoldl-distr mult-assoc sup-top-left conv-invol many-strongly-connected-iff-7*)

lemma *many-strongly-connected-implies-8*:
assumes *many-strongly-connected x*
shows $x;x^T \le x^+$
by (*simp add*: *assms mult-isol*)

lemma *many-strongly-connected-implies-9*:
assumes *many-strongly-connected x*
shows $x^T;x \le x^+$
by (*metis assms eq-refl phl-cons1 star-ext star-slide-var*)

lemma *many-strongly-connected-implies-10*:
assumes *many-strongly-connected x*
shows $x;x^T;x^\star \le x^+$
by (*simp add*: *assms comp-assoc mult-isol*)

lemma *many-strongly-connected-implies-10-eq*:
assumes *many-strongly-connected x*

shows $x;x^T;x^\star = x^+$
proof (*rule antisym*)
show $x;x^T;x^\star \leq x^+$
by (*simp add*: *assms comp-assoc mult-isol*)
next
have $x^+ \leq x;x^T;x;x^\star$
using *mult-isor x-leq-triple-x* **by** *blast*
thus $x^+ \leq x;x^T;x^\star$
by (*simp add*: *comp-assoc mult-isol order-trans*)
qed

lemma *many-strongly-connected-implies-11*:
assumes *many-strongly-connected x*
shows $x^\star;x^T;x \leq x^+$
by (*metis assms conv-contrav conv-iso mult-isol star-1l star-slide-var*)

lemma *many-strongly-connected-implies-11-eq*:
assumes *many-strongly-connected x*
shows $x^\star;x^T;x = x^+$
by (*metis assms comp-assoc conv-invol many-strongly-connected-iff-5-eq many-strongly-connected-implies-10-eq*)

lemma *many-strongly-connected-implies-12*:
assumes *many-strongly-connected x*
shows $x^\star;x;x^T \leq x^+$
by (*metis assms comp-assoc mult-isol star-1l star-slide-var*)

lemma *many-strongly-connected-implies-12-eq*:
assumes *many-strongly-connected x*
shows $x^\star;x;x^T = x^+$
by (*metis assms comp-assoc star-slide-var many-strongly-connected-implies-10-eq*)

lemma *many-strongly-connected-implies-13*:
assumes *many-strongly-connected x*
shows $x^T;x;x^\star \leq x^+$
by (*metis assms star-slide-var many-strongly-connected-implies-11 mult.assoc*)

lemma *many-strongly-connected-implies-13-eq*:
assumes *many-strongly-connected x*
shows $x^T;x;x^\star = x^+$
by (*metis assms conv-invol many-strongly-connected-iff-7 many-strongly-connected-implies-10-eq*)

lemma *many-strongly-connected-iff-8*:
assumes *is-p-fun x*
shows *many-strongly-connected x* $\longleftrightarrow x;x^T \leq x^+$
apply (*rule iffI*)
apply (*simp add*: *mult-isol*)
apply (*simp add*: *many-strongly-connected-iff-1*)

by (*metis comp-assoc conv-invol dual-order.trans mult-isol x-leq-triple-x assms comp-assoc*
dual-order.trans is-p-fun-def order.refl prod-star-closure star-ref)

lemma *many-strongly-connected-iff-9*:
assumes *is-inj x*
shows *many-strongly-connected x* $\longleftrightarrow x^T;x \le x^+$
by (*metis assms conv-contrav conv-iso inj-p-fun star-conv star-slide-var*
many-strongly-connected-iff-1 many-strongly-connected-iff-8)

lemma *many-strongly-connected-iff-10*:
assumes *is-p-fun x*
shows *many-strongly-connected x* $\longleftrightarrow x;x^T;x^\star \le x^+$
apply (*rule iffI*)
apply (*simp add*: *comp-assoc mult-isol*)
by (*metis assms mult-isol mult-oner order-trans star-ref many-strongly-connected-iff-8*)

lemma *many-strongly-connected-iff-10-eq*:
assumes *is-p-fun x*
shows *many-strongly-connected x* $\longleftrightarrow x;x^T;x^\star = x^+$
using *assms many-strongly-connected-iff-10 many-strongly-connected-implies-10-eq* **by** *fastforce*

lemma *many-strongly-connected-iff-11*:
assumes *is-inj x*
shows *many-strongly-connected x* $\longleftrightarrow x^\star;x^T;x \le x^+$
by (*metis assms comp-assoc conv-contrav conv-iso inj-p-fun plus-conv star-conv*
many-strongly-connected-iff-10 many-strongly-connected-iff-2)

lemma *many-strongly-connected-iff-11-eq*:
assumes *is-inj x*
shows *many-strongly-connected x* $\longleftrightarrow x^\star;x^T;x = x^+$
using *assms many-strongly-connected-iff-11 many-strongly-connected-implies-11-eq* **by** *fastforce*

lemma *many-strongly-connected-iff-12*:
assumes *is-p-fun x*
shows *many-strongly-connected x* $\longleftrightarrow x^\star;x;x^T \le x^+$
by (*metis assms dual-order.trans mult-double-iso mult-oner star-ref star-slide-var*
many-strongly-connected-iff-8 many-strongly-connected-implies-12)

lemma *many-strongly-connected-iff-12-eq*:
assumes *is-p-fun x*
shows *many-strongly-connected x* $\longleftrightarrow x^\star;x;x^T = x^+$
using *assms many-strongly-connected-iff-12 many-strongly-connected-implies-12-eq* **by** *fastforce*

lemma *many-strongly-connected-iff-13*:

assumes *is-inj x*
shows *many-strongly-connected x* $\longleftrightarrow$ $x^T;x;x^\star \leq x^+$
by (*metis assms comp-assoc conv-contrav conv-iso inj-p-fun star-conv star-slide-var many-strongly-connected-iff-1 many-strongly-connected-iff-12*)

lemma *many-strongly-connected-iff-13-eq*:
assumes *is-inj x*
shows *many-strongly-connected x* $\longleftrightarrow$ $x^T;x;x^\star = x^+$
using *assms many-strongly-connected-iff-13 many-strongly-connected-implies-13-eq* **by** *fastforce*

Equivalences and implications for *one-strongly-connected*

lemma *one-strongly-connected-iff*:
one-strongly-connected x $\longleftrightarrow$ *connected x* $\wedge$ *many-strongly-connected x*
apply (*rule iffI*)
apply (*metis top-greatest x-leq-triple-x mult-double-iso top-greatest dual-order.trans many-strongly-connected-iff-1 comp-assoc conv-contrav conv-invol conv-iso le-supI2 star-conv*)
by (*metis comp-assoc conv-contrav conv-iso conv-one conway.dagger-denest star-conv star-invol star-sum-unfold star-trans-eq*)

lemma *one-strongly-connected-iff-1*:
one-strongly-connected x $\longleftrightarrow$ $x^T;1;x^T \leq x^+$
proof
assume *1*: *one-strongly-connected x*
have $x^T;1;x^T \leq x^T;x;x^T;1;x^T$
by (*metis conv-invol mult-isor x-leq-triple-x*)
also from *1* **have** ... $\leq x^T;x;x^\star$
by (*metis distrib-left mult-assoc sup.absorb-iff1*)
also from *1* **have** ... $\leq x^+$
using *many-strongly-connected-implies-13 one-strongly-connected-iff* **by** *blast*
finally show $x^T;1;x^T \leq x^+$
.
next
assume $x^T;1;x^T \leq x^+$
thus *one-strongly-connected x*
using *dual-order.trans star-1l* **by** *blast*
qed

lemma *one-strongly-connected-iff-1-eq*:
one-strongly-connected x $\longleftrightarrow$ $x^T;1;x^T = x^+$
apply (*rule iffI, simp-all*)
by (*metis comp-assoc conv-contrav conv-invol mult-double-iso plus-conv star-slide-var top-greatest top-plus many-strongly-connected-implies-10-eq one-strongly-connected-iff*

eq-iff
one-strongly-connected-iff-1)

lemma *one-strongly-connected-iff-2*:
one-strongly-connected $x \longleftrightarrow x;1;x \leq x^{T\star}$
by (*metis conv-invol eq-refl less-eq-def one-strongly-connected-iff*)

lemma *one-strongly-connected-iff-3*:
one-strongly-connected $x \longleftrightarrow x;1;x \leq x^{T+}$
by (*metis comp-assoc conv-contrav conv-invol conv-iso conv-one star-conv*
one-strongly-connected-iff-1)

lemma *one-strongly-connected-iff-3-eq*:
one-strongly-connected $x \longleftrightarrow x;1;x = x^{T+}$
by (*metis conv-invol one-strongly-connected-iff-1-eq one-strongly-connected-iff-2*)

lemma *one-strongly-connected-iff-4-eq*:
one-strongly-connected $x \longleftrightarrow x^T;1;x = x^+$
apply (*rule iffI*)
apply (*metis comp-assoc top-plus many-strongly-connected-iff-7*
one-strongly-connected-iff
one-strongly-connected-iff-1-eq)
by (*metis comp-assoc conv-contrav conv-invol conv-one plus-conv top-plus*
one-strongly-connected-iff-1-eq)

lemma *one-strongly-connected-iff-5-eq*:
one-strongly-connected $x \longleftrightarrow x;1;x^T = x^+$
using *comp-assoc conv-contrav conv-invol conv-one plus-conv top-plus*
many-strongly-connected-iff-7
one-strongly-connected-iff one-strongly-connected-iff-3-eq **by** *metis*

lemma *one-strongly-connected-iff-6-aux*:
$x;x^+ \leq x;1;x$
by (*metis comp-assoc maddux-21 mult-isol top-plus*)

lemma *one-strongly-connected-implies-6-eq*:
assumes *one-strongly-connected* x
shows $x;1;x = x;x^+$
by (*metis assms comp-assoc many-strongly-connected-iff-7*
many-strongly-connected-implies-10-eq
one-strongly-connected-iff one-strongly-connected-iff-3-eq)

lemma *one-strongly-connected-iff-7-aux*:
$x^+ \leq x;1;x$
by (*metis le-infI maddux-20 maddux-21 plus-top top-plus vector-meet-comp-x'*)

lemma *one-strongly-connected-implies-7-eq*:
assumes *one-strongly-connected* x
shows $x;1;x = x^+$

using *assms many-strongly-connected-iff-7 one-strongly-connected-iff one-strongly-connected-iff-3-eq*
by *force*

lemma *one-strongly-connected-implies-8*:
  assumes *one-strongly-connected x*
    shows $x;1;x \leq x^\star$
using *assms one-strongly-connected-iff* **by** *fastforce*

lemma *one-strongly-connected-iff-4*:
  assumes *is-inj x*
    shows *one-strongly-connected x* $\longleftrightarrow x^T;1;x \leq x^+$
proof
  assume *one-strongly-connected x*
  thus $x^T;1;x \leq x^+$
    by (*simp add*: *one-strongly-connected-iff-4-eq*)
next
  assume *1*: $x^T;1;x \leq x^+$
  hence $x^T;1;x^T \leq x^\star;x;x^T$
    by (*metis mult-isor star-slide-var comp-assoc conv-invol modular-var-3 vector-meet-comp-x*
            order.trans)
  also from *assms* **have** ... $\leq x^\star$
    using *comp-assoc is-inj-def mult-isol mult-oner* **by** *fastforce*
  finally show *one-strongly-connected x*
    using *dual-order.trans star-1l* **by** *fastforce*
qed

lemma *one-strongly-connected-iff-5*:
  assumes *is-p-fun x*
    shows *one-strongly-connected x* $\longleftrightarrow x;1;x^T \leq x^+$
  apply (*rule iffI*)
 using *one-strongly-connected-iff-5-eq* **apply** *simp*
by (*metis assms comp-assoc mult-double-iso order.trans star-slide-var top-greatest top-plus*
          many-strongly-connected-iff-12 many-strongly-connected-iff-7 one-strongly-connected-iff-3)

lemma *one-strongly-connected-iff-6*:
  assumes *is-p-fun x*
      and *is-inj x*
    shows *one-strongly-connected x* $\longleftrightarrow x;1;x \leq x;x^+$
proof
  assume *one-strongly-connected x*
  thus $x;1;x \leq x;x^+$
   by (*simp add*: *one-strongly-connected-implies-6-eq*)
next
  assume *1*: $x;1;x \leq x;x^+$
  have $x^T;1;x \leq x^T;x;x^T;1;x$

by (*metis conv-invol mult-isor x-leq-triple-x*)
also have ... $\leq x^T;x;1;x$
by (*metis comp-assoc mult-double-iso top-greatest*)
also from *1* **have** ... $\leq x^T;x;x^+$
by (*simp add*: *comp-assoc mult-isol*)
also from *assms*(*1*) **have** ... $\leq x^+$
by (*metis comp-assoc is-p-fun-def mult-isor mult-onel*)
finally show *one-strongly-connected x*
using *assms*(*2*) *one-strongly-connected-iff-4* **by** *blast*
qed

lemma *one-strongly-connected-iff-6-eq*:
assumes *is-p-fun x*
and *is-inj x*
shows *one-strongly-connected x* $\longleftrightarrow x;1;x = x;x^+$
apply (*rule iffI*)
using *one-strongly-connected-implies-6-eq* **apply** *blast*
by (*simp add*: *assms one-strongly-connected-iff-6*)

Start points and end points

lemma *start-end-implies-terminating*:
assumes *has-start-points x*
and *has-end-points x*
shows *terminating x*
using *assms* **by** *simp*

lemma *start-points-end-points-conv*:
start-points x = *end-points* (x^T)
by *simp*

lemma *start-point-at-most-one*:
assumes *path x*
shows *is-inj* (*start-points x*)
proof −
have *isvec*: *is-vector* $(x;1 \cdot -(x^T;1))$
by (*simp add*: *comp-assoc is-vector-def one-compl vector-1*)

have $x;1 \cdot 1;x^T \leq x;1;x;x^T$
by (*metis comp-assoc conv-contrav conv-one inf.cobounded2 mult-1-right mult-isol one-conv ra-2*)
also have ... $\leq (x^\star + x^{T\star});x^T$
using ‹*path x*› **by** (*metis path-def mult-isor*)
also have ... $= x^T + x^+;x^T + x^{T+}$
by (*simp add*: *star-slide-var*)
also have ... $\leq x^{T+} + x^+;x^T + x^{T+}$
by (*metis add-iso mult-1-right star-unfoldl-eq subdistl*)
also have ... $\leq x^\star;x;x^T + x^{T+}$
by (*simp add*: *star-slide-var add-comm*)
also have ... $\leq x^\star;1' + x^{T+}$

using ‹*path x*› **by** (*metis path-def is-inj-def comp-assoc distrib-left join-iso less-eq-def*)
also have ... $= 1' + x^\star;x + x^T;x^{T\star}$
by *simp*
also have ... $\leq 1' + 1;x + x^T;1$
by (*metis join-isol mult-isol mult-isor sup.mono top-greatest*)
finally have *aux*: $x;1 \cdot 1;x^T \leq 1' + 1;x + x^T;1$.

from *aux* **have** $x;1 \cdot 1;x^T \cdot -(x^T;1) \cdot -(1;x) \leq 1'$
by (*simp add*: *galois-1 sup-commute*)
hence $(x;1 \cdot -(x^T;1)) \cdot (x;1 \cdot -(x^T;1))^T \leq 1'$
by (*simp add*: *conv-compl inf.assoc inf.left-commute*)
with *isvec* **have** $(x;1 \cdot -(x^T;1)) ; (x;1 \cdot -(x^T;1))^T \leq 1'$
by (*metis vector-meet-comp'*)
thus *is-inj* (*start-points x*)
by (*simp add*: *conv-compl is-inj-def*)
qed

lemma *start-point-zero-point*:
assumes *path x*
shows *start-points x* = *0* ∨ *is-point* (*start-points x*)
using *assms start-point-at-most-one comp-assoc is-point-def is-vector-def vector-compl vector-mult*
by *simp*

lemma *start-point-iff1*:
assumes *path x*
shows *is-point* (*start-points x*) ⟷ ¬(*no-start-points x*)
using *assms start-point-zero-point galois-aux2 is-point-def* **by** *blast*

lemma *end-point-at-most-one*:
assumes *path x*
shows *is-inj* (*end-points x*)
by (*metis assms conv-path compl-bot-eq conv-invol inj-def-var1 is-point-def top-greatest*
start-point-zero-point)

lemma *end-point-zero-point*:
assumes *path x*
shows *end-points x* = *0* ∨ *is-point* (*end-points x*)
using *assms conv-path start-point-zero-point* **by** *fastforce*

lemma *end-point-iff1*:
assumes *path x*
shows *is-point* (*end-points x*) ⟷ ¬(*no-end-points x*)
using *assms end-point-zero-point galois-aux2 is-point-def* **by** *blast*

lemma *predecessor-point'*:
assumes *path x*

and *point s*
and *point e*
and $e;s^T \leq x$
shows $x;s = e$
proof (*rule antisym*)
show *1*: $e \leq x ; s$
using *assms*(*2*,*4*) *point-def ss423bij* **by** *blast*
show $x ; s \leq e$
proof −
have $e^T ; (x ; s) = 1$
using *1* **by** (*metis assms*(*3*) *eq-iff is-vector-def point-def ss423conv top-greatest*)
thus *?thesis*
by (*metis assms*(*1*−*3*) *comp-assoc conv-contrav conv-invol eq-iff inj-compose is-vector-def*
mult-isol path-def point-def ss423conv sur-def-var1 top-greatest)
qed
qed

lemma *predecessor-point*:
assumes *path x*
and *point s*
and *point e*
and $e;s^T \leq x$
shows *point*($x;s$)
using *predecessor-point′ assms* **by** *blast*

lemma *points-of-path-iff*:
shows $(x + x^T);1 = x^T;1 +$ *start-points*(x)
and $(x + x^T);1 = x;1 +$ *end-points*(x)
using *aux9 inf.commute sup.commute* **by** *auto*

Path concatenation preliminaries

lemma *path-concat-aux-1*:
assumes $x;1 \cdot y;1 \cdot y^T;1 = 0$
and *end-points x* = *start-points y*
shows $x;1 \cdot y;1 = 0$
proof −
have $x;1 \cdot y;1 = (x;1 \cdot y;1 \cdot y^T;1) + (x;1 \cdot y;1 \cdot -(y^T;1))$
by *simp*
also from *assms*(*1*) **have** ... $= x;1 \cdot y;1 \cdot -(y^T;1)$
by (*metis aux6-var de-morgan-3 inf.left-commute inf-compl-bot inf-sup-absorb*)
also from *assms*(*2*) **have** ... $= x;1 \cdot x^T;1 \cdot -(x;1)$
by (*simp add*: *inf.assoc*)
also have ... $= 0$
by (*simp add*: *inf.commute inf.assoc*)
finally show *?thesis* **.**
qed

lemma *path-concat-aux-2*:
  assumes $x;1 \cdot x^T;1 \cdot y^T;1 = 0$
    and *end-points* x = *start-points* y
  shows $x^T;1 \cdot y^T;1 = 0$
proof –
  have $y^T;1 \cdot x^T;1 \cdot (x^T)^T;1 = 0$
    using *assms*(1) *inf.assoc inf.commute* **by** *force*
  thus *?thesis*
    by (*metis assms*(2) *conv-invol inf.commute path-concat-aux-1*)
qed

lemma *path-concat-aux3-1*:
  assumes *path* x
  shows $x;1;x^T \leq x^\star + x^{T\star}$
proof –
  have $x;1;x^T \leq x;1;x^T;x;x^T$
    by (*metis comp-assoc conv-invol mult-isol x-leq-triple-x*)
  also have ... $\leq x;1;x;x^T$
    by (*metis mult-isol mult-isor mult-assoc top-greatest*)
  also from *assms* **have** ... $\leq (x^\star + x^{T\star});x^T$
    using *path-def comp-assoc mult-isor* **by** *blast*
  also have ... $= x^\star;x;x^T + x^{T\star};x^T$
    by (*simp add*: *star-slide-var star-star-plus*)
  also have ... $\leq x^\star;1' + x^{T\star};x^T$
    by (*metis assms path-def is-inj-def join-iso mult-isol mult-assoc*)
  also have ... $\leq x^\star + x^{T\star}$
    using *join-isol* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *path-concat-aux3-2*:
  assumes *path* x
  shows $x^T;1;x \leq x^\star + x^{T\star}$
proof –
  have $x^T;1;x \leq x^T;x;x^T;1;x$
    by (*metis comp-assoc conv-invol mult-isor x-leq-triple-x*)
  also have ... $\leq x^T;x;1;x$
    by (*metis mult-isol mult-isor mult-assoc top-greatest*)
  also from *assms* **have** ... $\leq x^T;(x^\star + x^{T\star})$
    by (*simp add*: *comp-assoc mult-isol path-def*)
  also have ... $= x^T;x;x^\star + x^T;x^{T\star}$
    by (*simp add*: *comp-assoc distrib-left star-star-plus*)
  also have ... $\leq 1';x^\star + x^T;x^{T\star}$
    by (*metis assms path-def is-p-fun-def join-iso mult-isor mult-assoc*)
  also have ... $\leq x^\star + x^{T\star}$
    using *join-isol* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *path-concat-aux3-3*:
  assumes *path x*
    shows $x^T;1;x^T \leq x^\star + x^{T\star}$
proof –
  have $x^T;1;x^T \leq x^T;x;x^T;1;x^T$
    by (*metis comp-assoc conv-invol mult-isor x-leq-triple-x*)
  also have ... $\leq x^T;x;1;x^T$
    by (*metis mult-isol mult-isor mult-assoc top-greatest*)
  also from *assms* **have** ... $\leq x^T;(x^\star + x^{T\star})$
    using *path-concat-aux3-1* **by** (*simp add*: *mult-assoc mult-isol*)
  also have ... $= x^T;x;x^\star + x^T;x^{T\star}$
    by (*simp add*: *comp-assoc distrib-left star-star-plus*)
  also have ... $\leq 1';x^\star + x^T;x^{T\star}$
    by (*metis assms path-def is-p-fun-def join-iso mult-isor mult-assoc*)
  also have ... $\leq x^\star + x^{T\star}$
    using *join-isol* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *path-concat-aux-3*:
  assumes *path x*
      and $y \leq x^+ + x^{T+}$
      and $z \leq x^+ + x^{T+}$
    shows $y;1;z \leq x^\star + x^{T\star}$
proof –
  from *assms*(*2*,*3*) **have** $y;1;z \leq (x^+ + x^{T+});1;(x^+ + x^{T+})$
    using *mult-isol-var mult-isor* **by** *blast*
  also have ... $= x^+;1;x^+ + x^+;1;x^{T+} + x^{T+};1;x^+ + x^{T+};1;x^{T+}$
    by (*simp add*: *distrib-left sup-commute sup-left-commute*)
  also have ... $= x;x^\star;1;x^\star;x + x;x^\star;1;x^{T\star};x^T + x^T;x^{T\star};1;x^\star;x +$
$x^T;x^{T\star};1;x^{T\star};x^T$
    by (*simp add*: *comp-assoc star-slide-var*)
  also have ... $\leq x;1;x + x;x^\star;1;x^{T\star};x^T + x^T;x^{T\star};1;x^\star;x + x^T;x^{T\star};1;x^{T\star};x^T$
    by (*metis comp-assoc mult-double-iso top-greatest join-iso*)
  also have ... $\leq x;1;x + x;1;x^T + x^T;x^{T\star};1;x^\star;x + x^T;x^{T\star};1;x^{T\star};x^T$
    by (*metis comp-assoc mult-double-iso top-greatest join-iso join-isol*)
  also have ... $\leq x;1;x + x;1;x^T + x^T;1;x + x^T;x^{T\star};1;x^{T\star};x^T$
    by (*metis comp-assoc mult-double-iso top-greatest join-iso join-isol*)
  also have ... $\leq x;1;x + x;1;x^T + x^T;1;x + x^T;1;x^T$
    by (*metis comp-assoc mult-double-iso top-greatest join-isol*)
  also have ... $\leq x^\star + x^{T\star}$
    using *assms*(*1*) *path-def path-concat-aux3-1 path-concat-aux3-2*
path-concat-aux3-3 join-iso join-isol
    by *simp*
  finally show *?thesis* **.**
qed

lemma *path-concat-aux-4*:
  $x^\star + x^{T\star} \leq x^\star + x^T;1$

by (*metis star-star-plus add-comm join-isol mult-isol top-greatest*)

lemma *path-concat-aux-5*:
assumes *path* x
and $y \leq$ *start-points* x
and $z \leq x + x^T$
shows $y;1;z \leq x^\star$
proof −
from *assms*(*1*) **have** $x;1;x \leq x^\star + x^T;1$
using *path-def path-concat-aux-4 dual-order.trans* **by** *blast*
hence *aux1*: $x;1;x \cdot -(x^T;1) \leq x^\star$
by (*simp add*: *galois-1 sup-commute*)

from *assms*(*1*) **have** $x;1;x^T \leq x^\star + x^T;1$
using *dual-order.trans path-concat-aux3-1 path-concat-aux-4* **by** *blast*
hence *aux2*: $x;1;x^T \cdot -(x^T;1) \leq x^\star$
by (*simp add*: *galois-1 sup-commute*)

from *assms*(*2*,*3*) **have** $y;1;z \leq (x;1 \cdot -(x^T;1));1;(x + x^T)$
by (*simp add*: *mult-isol-var mult-isor*)
also have ... $= (x;1 \cdot -(x^T;1));1;x + (x;1 \cdot -(x^T;1));1;x^T$
using *distrib-left* **by** *blast*
also have ... $= (x;1 \cdot -(x^T;1) \cdot 1;x) + (x;1 \cdot -(x^T;1));1;x^T$
by (*metis comp-assoc inf-top-right is-vector-def one-idem-mult vector-1 vector-compl*)
also have ... $= (x;1 \cdot -(x^T;1) \cdot 1;x) + (x;1 \cdot -(x^T;1) \cdot 1;x^T)$
by (*metis comp-assoc inf-top-right is-vector-def one-idem-mult vector-1 vector-compl*)
also have ... $= (x;1;x \cdot -(x^T;1)) + (x;1;x^T \ -(x^T;1))$
using *vector-meet-comp-x vector-meet-comp-x' diff-eq inf.assoc inf.commute* **by** *simp*
also from *aux1 aux2* **have** ... $\leq x^\star$
by (*simp add*: *diff-eq join-iso*)
finally show *?thesis* **.**
qed

lemma *path-conditions-disjoint-points-iff*:
$x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0 \wedge$ *start-points* $x \cdot$ *end-points* $y = 0 \longleftrightarrow x;1 \cdot y^T;1 = 0$
proof
assume *1*: $x ; 1 \cdot y^T ; 1 = 0$
hence *g1*: $x ; 1 \cdot (x^T ; 1 + y ; 1) \cdot y^T ; 1 = 0$
by (*metis inf.left-commute inf-bot-right inf-commute*)
have *g2*: *start-points* $x \cdot$ *end-points* $y = 0$
using *1* **by** (*metis compl-inf-bot inf.assoc inf.commute inf.left-idem*)
show $x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0 \wedge$ *start-points* $x \cdot$ *end-points* $y = 0$
using *g1* **and** *g2* **by** *simp*
next
assume *a*: $x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0 \wedge$ *start-points* $x \cdot$ *end-points* $y = 0$

```
  from a have a1: x;1 · x^T;1 · y^T;1 = 0
    by (simp add: inf.commute inf-sup-distrib1)
  from a have a2: x;1 · y;1 · y^T;1 = 0
    by (simp add: inf.commute inf-sup-distrib1)
  from a have a3: start-points x · end-points y = 0
    by blast

  have x;1 · y^T;1 = x;1 · x^T;1 · y^T;1 + x;1 · −(x^T;1) · y^T;1
    by (metis aux4 inf-sup-distrib2)
  also from a1 have ... = x;1 · −(x^T;1) · y^T;1
    using sup-bot-left by blast
  also have ... = x;1 · −(x^T;1) · y;1 · y^T;1 + x;1 · −(x^T;1) · −(y;1) · y^T;1
    by (metis aux4 inf-sup-distrib2)
  also have ... ≤ x;1 · y;1 · y^T;1 + x;1 · −(x^T;1) · −(y;1) · y^T;1
    using join-iso meet-iso by simp
  also from a2 have ... = start-points x · end-points y
    using sup-bot-left inf.commute inf.left-commute by simp
  also from a3 have ... = 0
    by blast
  finally show x;1 · y^T;1 = 0
    using le-bot by blast
qed

end
```

2.2 Consequences with the Tarski rule

```
context relation-algebra-rtc-tarski
begin
```

General theorems

```
lemma reachable-implies-predecessor:
  assumes p ≠ q
      and point p
      and point q
      and x⋆;q ≤ x^T⋆;p
    shows x;q ≠ 0
proof
  assume contra: x;q=0
  with assms(4) have q ≤ x^T⋆;p
    by (simp add: independence1)
  hence p ≤ x⋆;q
    by (metis assms(2,3) point-swap star-conv)
  with contra assms(2,3) have p=q
    by (simp add: independence1 is-point-def point-singleton point-is-point)
  with assms(1) show False
    by simp
qed

lemma acyclic-imp-one-step-different-points:
```

assumes *is-acyclic x*
and *point p*
and *point q*
and $p \leq x;q$
shows $p \leq -q$ **and** $p \neq q$
using *acyclic-reachable-points assms point-is-point point-not-equal*(*1*) **by** *auto*

Start points and end points

lemma *start-point-iff2*:
assumes *path x*
shows *is-point* (*start-points x*) $\longleftrightarrow$ *has-start-points x*
proof −
have *has-start-points x* $\longleftrightarrow 1 \leq -(1;x);x;1$
by (*simp add*: *eq-iff*)
also have ... $\longleftrightarrow 1 \leq 1;x^T;-(x^T;1)$
by (*metis comp-assoc conv-compl conv-contrav conv-iso conv-one*)
also have ... $\longleftrightarrow 1 \leq 1;(x;1 \cdot -(x^T;1))$
by (*metis* (*no-types*) *conv-contrav conv-one inf.commute is-vector-def one-idem-mult ra-2 vector-1*
vector-meet-comp-x)
also have ... $\longleftrightarrow 1 = 1;(x;1 \cdot -(x^T;1))$
by (*simp add*: *eq-iff*)
also have ... $\longleftrightarrow x;1 \cdot -(x^T;1) \neq 0$
by (*metis tarski comp-assoc one-compl ra-1 ss-p18*)
also have ... $\longleftrightarrow$ *is-point* (*start-points x*)
using *assms is-point-def start-point-zero-point* **by** *blast*
finally show *?thesis* **..**
qed

lemma *end-point-iff2*:
assumes *path x*
shows *is-point* (*end-points x*) $\longleftrightarrow$ *has-end-points x*
by (*metis assms conv-invol conv-has-start-points conv-path start-point-iff2*)

lemma *edge-is-path*:
assumes *is-point p*
and *is-point q*
shows *path* $(p;q^T)$
apply (*unfold path-def*; *intro conjI*)
apply (*metis assms comp-assoc is-point-def le-supI1 star-ext vector-rectangle point-equations*(*3*))
apply (*metis is-p-fun-def assms comp-assoc conv-contrav conv-invol is-inj-def is-point-def*
vector-2-var vector-meet-comp-x' point-equations)
by (*metis is-inj-def assms conv-invol conv-times is-point-def p-fun-mult-var vector-meet-comp*)

lemma *edge-start*:
assumes *is-point p*

    and *is-point q*
    and $p \neq q$
  shows *start-points* $(p;q^T) = p$
using *assms* **by** (*simp add*: *comp-assoc point-equations*(*1*,*3*) *point-not-equal inf.absorb1*)

lemma *edge-end*:
 assumes *is-point p*
    and *is-point q*
    and $p \neq q$
  shows *end-points* $(p;q^T) = q$
using *assms edge-start* **by** *simp*

lemma *loop-no-start*:
 assumes *is-point p*
  shows *start-points* $(p;p^T) = 0$
by *simp*

lemma *loop-no-end*:
 assumes *is-point p*
  shows *end-points* $(p;p^T) = 0$
by *simp*

lemma *start-point-no-predecessor*:
 $x;$*start-points*$(x) = 0$
by (*metis inf-top.right-neutral modular-1-aux′*)

lemma *end-point-no-successor*:
 $x^T;$*end-points*$(x) = 0$
by (*metis conv-invol start-point-no-predecessor*)

lemma *start-to-end*:
 assumes *path x*
  shows *start-points*$(x);$*end-points*$(x)^T \leq x^\star$
proof (*cases end-points*$(x) = 0$)
 assume *end-points*$(x) = 0$
 thus *?thesis*
  by *simp*
next
 assume *ass*: *end-points*$(x) \neq 0$
 hence *nz*: $x;$*end-points*$(x) \neq 0$
  by (*metis comp-res-aux compl-bot-eq inf.left-idem*)
 have *a*: $x;$*end-points*$(x);$*end-points*$(x)^T \leq x + x^T$
  by (*metis end-point-at-most-one assms*(*1*) *is-inj-def comp-assoc mult-isol mult-oner le-supI1*)

 have *start-points*$(x);$*end-points*$(x)^T =$ *start-points*$(x);1;$*end-points*$(x)^T$
  using *ass* **by** (*simp add*: *comp-assoc is-vector-def one-compl vector-1*)
 also have ... = *start-points*$(x);1;x;$*end-points*$(x);1;$*end-points*$(x)^T$

using *nz tarski* **by** (*simp add*: *comp-assoc*)
also have ... = *start-points*(*x*);*1*;*x*;*end-points*(*x*);*end-points*(*x*)^T
using *ass* **by** (*simp add*: *comp-assoc is-vector-def one-compl vector-1*)
also with *a assms*(*1*) **have** ... ≤ $x^\star$
using *path-concat-aux-5 comp-assoc eq-refl* **by** *simp*
finally show *?thesis* **.**
qed

lemma *path-acyclic*:
assumes *has-start-points-path x*
shows *is-acyclic x*
proof −
let *?r* = *start-points*(*x*)
have *pt*: *point*(*?r*)
using *assms point-is-point start-point-iff2* **by** *blast*
have $x^+ \cdot 1' = (x^+)^T \cdot x^+ \cdot 1'$
by (*metis conv-e conv-times inf.assoc inf.left-idem inf-le2 many-strongly-connected-iff-7*
mult-oner star-subid)
also have ... ≤ $x^T;1 \cdot x^+ \cdot 1'$
by (*metis conv-contrav inf.commute maddux-20 meet-double-iso plus-top star-conv star-slide-var*)
finally have $?r;(x^+ \cdot 1') \leq ?r;(x^T;1 \cdot x^+ \cdot 1')$
using *mult-isol* **by** *blast*
also have ... = $(?r \cdot 1;x);(x^+ \cdot 1')$
by (*metis* (*no-types*, *lifting*) *comp-assoc conv-contrav conv-invol conv-one inf.assoc*
is-vector-def one-idem-mult vector-2)
also have ... = $?r;x;(x^+ \cdot 1')$
by (*metis comp-assoc inf-top.right-neutral is-vector-def one-compl one-idem-mult vector-1*)
also have ... ≤ $(x^\star + x^{T\star});(x^+ \cdot 1')$
using *assms*(*1*) *mult-isor*
by (*meson connected-iff4 dual-order.trans mult-subdistr path-concat-aux3-3*)
also have ... = $x^\star;(x^+ \cdot 1') + x^{T+};(x^+ \cdot 1')$
by (*metis distrib-right star-star-plus sup.commute*)
also have ... ≤ $x^\star;(x^+ \cdot 1') + x^T;1$
by (*metis join-isol mult-isol plus-top top-greatest*)
finally have $?r;(x^+ \cdot 1');1 \leq x^\star;(x^+ \cdot 1');1 + x^T;1$
by (*metis distrib-right inf-absorb2 mult-assoc mult-subdistr one-idem-mult*)
hence *1*: $?r;(x^+ \cdot 1');1 \leq x^T;1$
using *assms*(*1*) *path-def inj-implies-step-forwards-backwards sup-absorb2* **by** *simp*
have $x^+ \cdot 1' \leq (x^+ \cdot 1');1$
by (*simp add*: *maddux-20*)
also have ... ≤ $?r^T;?r;(x^+ \cdot 1');1$
using *pt comp-assoc point-def ss423conv* **by** *fastforce*
also have ... ≤ $?r^T;x^T;1$
using *1* **by** (*simp add*: *comp-assoc mult-isol*)

also have ... = *0*
by (*metis start-point-no-predecessor annil conv-contrav conv-zero*)
finally show *?thesis*
using *galois-aux le-bot* **by** *blast*
qed

Equivalences for *terminating*

lemma *backward-terminating-iff1*:
assumes *path x*
shows *backward-terminating x* $\longleftrightarrow$ *has-start-points x* $\vee$ $x = 0$
proof
assume *backward-terminating x*
hence $1;x;1 \le 1;-(1;x);x;1;1$
by (*metis mult-isor mult-isol comp-assoc*)
also have ... $= -(1;x);x;1$
by (*metis conv-compl conv-contrav conv-invol conv-one mult-assoc one-compl one-idem-mult*)
finally have $1;x;1 \le -(1;x);x;1$ **.**

with *tarski* **show** *has-start-points x* $\vee$ $x = 0$
by (*metis top-le*)
next
show *has-start-points x* $\vee$ $x = 0 \Longrightarrow$ *backward-terminating x*
by *fastforce*
qed

lemma *backward-terminating-iff2-aux*:
assumes *path x*
shows $x;1 \cdot 1;x^T \cdot -(1;x) \le x^{T\star}$
proof −
have $x;1 \cdot 1;x^T \le x;1;x;x^T$
by (*metis conv-invol modular-var-3 vector-meet-comp-x vector-meet-comp-x'*)
also from *assms* **have** ... $\le (x^\star + x^{T\star});x^T$
using *path-def mult-isor* **by** *blast*
also have ... $\le x^\star;x;x^T + x^{T\star};x^T$
by (*simp add*: *star-star-plus star-slide-var add-comm*)
also from *assms* **have** ... $\le x^\star;1' + x^{T\star};x^T$
by (*metis path-def is-inj-def join-iso mult-assoc mult-isol*)
also have ... $= x^+ + x^{T\star}$
by (*metis mult-1-right star-slide-var star-star-plus sup.commute*)
also have ... $\le x^{T\star} + 1;x$
by (*metis join-iso mult-isor star-slide-var top-greatest add-comm*)
finally have $x;1 \cdot 1;x^T \le x^{T\star} + 1;x$ **.**
thus *?thesis*
by (*simp add*: *galois-1 sup.commute*)
qed

lemma *backward-terminating-iff2*:
assumes *path x*

shows *backward-terminating* $x \longleftrightarrow x \leq x^{T\star};-(x^T;1)$
proof
assume *backward-terminating* x
with *assms* **have** *has-start-points* $x \vee x = 0$
by (*simp add*: *backward-terminating-iff1*)
thus $x \leq x^{T\star};-(x^T;1)$
proof
assume $x = 0$
thus *?thesis*
by *simp*
next
assume *has-start-points* x
hence *aux1*: $1 = 1;x^T;-(x^T;1)$
by (*metis comp-assoc conv-compl conv-contrav conv-one*)
have $x = x \cdot 1$
by *simp*
also have ... $\leq (x;-(1;x) \cdot 1;x^T);-(x^T;1)$
by (*metis inf.commute aux1 conv-compl conv-contrav conv-invol conv-one modular-2-var*)
also have ... $= (x;1 \cdot -(1;x) \cdot 1;x^T);-(x^T;1)$
by (*metis comp-assoc conv-compl conv-contrav conv-invol conv-one inf.commute inf-top-left one-compl ra-1*)
also from *assms* **have** ... $\leq x^{T\star};-(x^T;1)$
using *backward-terminating-iff2-aux inf.commute inf.assoc mult-isor* **by** *fastforce*
finally show $x \leq x^{T\star};-(x^T;1)$ **.**
qed
next
assume $x \leq x^{T\star};-(x^T;1)$
hence $x \leq x^{T\star};-(x^T;1) \cdot x$
by *simp*
also have ... $= (x^{T\star} \cdot -(1;x));1 \cdot x$
by (*metis one-compl conv-compl conv-contrav conv-invol conv-one inf-top-left ra-2*)
also have ... $\leq (x^{T\star} \cdot -(1;x))\ ;\ (1 \cdot (x^\star \cdot -(1;x)^T);x)$
by (*metis* (*mono-tags*) *conv-compl conv-invol conv-times modular-1-var star-conv*)
also have ... $\leq -(1;x);x^\star;x$
by (*simp add*: *mult-assoc mult-isol-var*)
also have ... $\leq -(1;x);x;1$
by (*simp add*: *mult-assoc mult-isol star-slide-var*)
finally show *backward-terminating* x **.**
qed

lemma *backward-terminating-iff3-aux*:
assumes *path* x
shows $x^T;1 \cdot 1;x^T \cdot -(1;x) \leq x^{T\star}$
proof −

have $x^T;1 \cdot 1;x^T \leq x^T;1;x;x^T$
by (*metis conv-invol modular-var-3 vector-meet-comp-x vector-meet-comp-x'*)
also from *assms* **have** ... $\leq (x^\star + x^{T\star});x^T$
using *mult-isor path-concat-aux3-2* **by** *blast*
also have ... $\leq x^\star;x;x^T + x^{T\star};x^T$
by (*simp add*: *star-star-plus star-slide-var add-comm*)
also from *assms* **have** ... $\leq x^\star;1' + x^{T\star};x^T$
by (*metis path-def is-inj-def join-iso mult-assoc mult-isol*)
also have ... $= x^+ + x^{T\star}$
by (*metis mult-1-right star-slide-var star-star-plus sup.commute*)
also have ... $\leq x^{T\star} + 1;x$
by (*metis join-iso mult-isor star-slide-var top-greatest add-comm*)
finally have $x^T;1 \cdot 1;x^T \leq x^{T\star} + 1;x$ **.**
thus *?thesis*
by (*simp add*: *galois-1 sup.commute*)
qed

lemma *backward-terminating-iff3*:
assumes *path x*
shows *backward-terminating x* $\longleftrightarrow x^T \leq x^{T\star};-(x^T;1)$
proof
assume *backward-terminating x*
with *assms* **have** *has-start-points x* $\vee x = 0$
by (*simp add*: *backward-terminating-iff1*)
thus $x^T \leq x^{T\star};-(x^T;1)$
proof
assume $x = 0$
thus *?thesis*
by *simp*
next
assume *has-start-points x*
hence *aux1*: $1 = 1;x^T;-(x^T;1)$
by (*metis comp-assoc conv-compl conv-contrav conv-one*)
have $x^T = x^T \cdot 1$
by *simp*
also have ... $\leq (x^T;-(1;x) \cdot 1;x^T);-(x^T;1)$
by (*metis inf.commute aux1 conv-compl conv-contrav conv-invol conv-one modular-2-var*)
also have ... $= (x^T;1 \cdot -(1;x) \cdot 1;x^T);-(x^T;1)$
by (*metis comp-assoc conv-compl conv-contrav conv-invol conv-one inf.commute inf-top-left one-compl ra-1*)
also from *assms* **have** ... $\leq x^{T\star};-(x^T;1)$
using *backward-terminating-iff3-aux inf.commute inf.assoc mult-isor* **by** *fastforce*
finally show $x^T \leq x^{T\star};-(x^T;1)$ **.**
qed
next
have *1*: $-(1;x) \cdot x = 0$
by (*simp add*: *galois-aux2 inf.commute maddux-21*)

assume $x^T \leq x^{T\star};-(x^T;1)$
hence $x = -(1;x);x^\star \cdot x$
by (*metis* (*mono-tags*, *lifting*) *conv-compl conv-contrav conv-iso conv-one inf.absorb2 star-conv*)
also have ... $= (-(1;x);x^+ + -(1;x);1') \cdot x$
by (*metis distrib-left star-unfoldl-eq sup-commute*)
also have ... $= -(1;x);x^+ \cdot x + -(1;x) \cdot x$
by (*simp add*: *inf-sup-distrib2*)
also have ... $\leq -(1;x);x^+$
using *1* **by** *simp*
also have ... $\leq -(1;x);x;1$
by (*simp add*: *mult-assoc mult-isol star-slide-var*)
finally show *backward-terminating x* **.**
qed

lemma *backward-terminating-iff4*:
assumes *path x*
shows *backward-terminating x* $\longleftrightarrow x \leq -(1;x);x^\star$
apply (*subst backward-terminating-iff3*)
apply (*rule assms*)
by (*metis* (*mono-tags*, *lifting*) *conv-compl conv-iso star-conv conv-contrav conv-one*)

lemma *forward-terminating-iff1*:
assumes *path x*
shows *forward-terminating x* $\longleftrightarrow$ *has-end-points x* $\vee x = 0$
by (*metis comp-assoc eq-refl le-bot one-compl tarski top-greatest*)

lemma *forward-terminating-iff2*:
assumes *path x*
shows *forward-terminating x* $\longleftrightarrow x^T \leq x^\star;-(x;1)$
by (*metis assms backward-terminating-iff1 backward-terminating-iff2 end-point-iff2*
forward-terminating-iff1 compl-bot-eq conv-compl conv-invol conv-one conv-path
double-compl start-point-iff2)

lemma *forward-terminating-iff3*:
assumes *path x*
shows *forward-terminating x* $\longleftrightarrow x \leq x^\star;-(x;1)$
by (*metis assms backward-terminating-iff1 backward-terminating-iff3 end-point-iff2*
forward-terminating-iff1 compl-bot-eq conv-compl conv-invol conv-one conv-path
double-compl start-point-iff2)

lemma *forward-terminating-iff4*:
assumes *path x*
shows *forward-terminating x* $\longleftrightarrow x \leq -(1;x^T);x^{T\star}$

using *forward-terminating-iff2 conv-contrav conv-iso star-conv assms conv-compl*
by *force*

lemma *terminating-iff1*:
assumes *path x*
shows *terminating x* $\longleftrightarrow$ *has-start-end-points x* $\vee$ $x = 0$
using *assms backward-terminating-iff1 forward-terminating-iff1* **by** *fastforce*

lemma *terminating-iff2*:
assumes *path x*
shows *terminating x* $\longleftrightarrow x \leq x^{T\star};-(x^T;1) \cdot -(1;x^T);x^{T\star}$
using *assms backward-terminating-iff2 forward-terminating-iff2 conv-compl conv-iso star-conv*
by *force*

lemma *terminating-iff3*:
assumes *path x*
shows *terminating x* $\longleftrightarrow x \leq x^\star;-(x;1) \cdot -(1;x);x^\star$
using *assms backward-terminating-iff4 forward-terminating-iff3* **by** *fastforce*

lemma *backward-terminating-path-irreflexive*:
assumes *backward-terminating-path x*
shows $x \leq -1'$
proof −
have *1*: $x;x^T \leq 1'$
using *assms is-inj-def path-def* **by** *blast*
have $x;(x^T \cdot 1') \leq x;x^T \cdot x$
by (*metis inf.bounded-iff inf.commute mult-1-right mult-subdistl*)
also have ... $\leq 1' \cdot x$
using *1 meet-iso* **by** *blast*
also have ... $= 1' \cdot x^T$
by (*metis conv-e conv-times inf.cobounded1 is-test-def test-eq-conv*)
finally have *2*: $x^T;-(x^T \cdot 1') \leq -(x^T \cdot 1')$
by (*metis compl-le-swap1 conv-galois-1 inf.commute*)
have $x^T \cdot 1' \leq x^T;1$
by (*simp add*: *le-infI1 maddux-20*)
hence $-(x^T;1) \leq -(x^T \cdot 1')$
using *compl-mono* **by** *blast*
hence $x^T;-(x^T \cdot 1') + -(x^T;1) \leq -(x^T \cdot 1')$
using *2* **by** (*simp add*: *le-supI*)
hence $x^{T\star};-(x^T;1) \leq -(x^T \cdot 1')$
by (*simp add*: *rtc-inductl*)
hence $x^T \cdot 1' \cdot x^{T\star};-(x^T;1) = 0$
by (*simp add*: *compl-le-swap1 galois-aux*)
hence $x^T \cdot 1' = 0$
using *assms backward-terminating-iff3 inf.order-iff le-infI1* **by** *blast*
hence $x \cdot 1' = 0$
by (*simp add*: *conv-self-conjugate*)
thus *?thesis*

by (*simp add*: *galois-aux*)
qed

lemma *forward-terminating-path-end-points-1*:
assumes *forward-terminating-path x*
shows $x \leq x^+$;*end-points x*
proof −
have *1*: $-(x;1) \cdot x = 0$
by (*simp add*: *galois-aux maddux-20*)
have $x = x^\star;-(x;1) \cdot x$
using *assms forward-terminating-iff3 inf.absorb2* **by** *fastforce*
also have ... $= (x^+;-(x;1) + 1';-(x;1)) \cdot x$
by (*simp add*: *sup.commute*)
also have ... $= x^+;-(x;1) \cdot x + -(x;1) \cdot x$
using *inf-sup-distrib2* **by** *fastforce*
also have ... $= x^+;-(x;1) \cdot x$
using *1* **by** *simp*
also have ... $\leq x^+;(-(x;1) \cdot (x^+)^T;x)$
using *modular-1-var* **by** *blast*
also have ... $= x^+;(-(x;1) \cdot x^{T+};x)$
using *plus-conv* **by** *fastforce*
also have ... $\leq x^+$;*end-points x*
by (*metis inf-commute inf-top-right modular-1′ mult-subdistl plus-conv plus-top*)
finally show *?thesis* **.**
qed

lemma *forward-terminating-path-end-points-2*:
assumes *forward-terminating-path x*
shows $x^T \leq x^\star$;*end-points x*
proof −
have $x^T \leq x^T;x;x^T$
by (*metis conv-invol x-leq-triple-x*)
also have ... $\leq x^T;x;1$
using *mult-isol top-greatest* **by** *blast*
also have ... $\leq x^T;x^+$;*end-points x*;*1*
by (*metis assms forward-terminating-path-end-points-1 comp-assoc mult-isol mult-isor*)
also have ... $= x^T;x^+$;*end-points x*
by (*metis inf-commute mult-assoc one-compl ra-1*)
also have ... $\leq x^\star$;*end-points x*
by (*metis assms comp-assoc compl-le-swap1 conv-galois-1 conv-invol p-fun-compl path-def*)
finally show *?thesis* **.**
qed

lemma *forward-terminating-path-end-points-3*:
assumes *forward-terminating-path x*
shows *start-points* $x \leq x^+$;*end-points x*

proof −
  have *start-points x* $\leq x^+$;*end-points x*;*1*
    using *assms forward-terminating-path-end-points-1 comp-assoc mult-isor inf.coboundedI1*
    by *blast*
  also have ... = x^+;*end-points x*
    by (*metis inf-commute mult-assoc one-compl ra-1*)
  finally show *?thesis* **.**
qed

lemma *backward-terminating-path-start-points-1*:
  assumes *backward-terminating-path x*
    shows $x^T \leq x^{T+}$;*start-points x*
using *assms forward-terminating-path-end-points-1 conv-backward-terminating-path* **by** *fastforce*

lemma *backward-terminating-path-start-points-2*:
  assumes *backward-terminating-path x*
    shows $x \leq x^{T\star}$;*start-points x*
using *assms forward-terminating-path-end-points-2 conv-backward-terminating-path* **by** *fastforce*

lemma *backward-terminating-path-start-points-3*:
  assumes *backward-terminating-path x*
    shows *end-points x* $\leq x^{T+}$;*start-points x*
using *assms forward-terminating-path-end-points-3 conv-backward-terminating-path* **by** *fastforce*

lemma *path-aux1a*:
 assumes *forward-terminating-path x*
   shows $x \neq 0 \longleftrightarrow$ *end-points x* $\neq 0$
using *assms end-point-iff2 forward-terminating-iff1 end-point-iff1 galois-aux2* **by** *force*

lemma *path-aux1b*:
  assumes *backward-terminating-path y*
    shows $y \neq 0 \longleftrightarrow$ *start-points y* $\neq 0$
using *assms start-point-iff2 backward-terminating-iff1 start-point-iff1 galois-aux2* **by** *force*

lemma *path-aux1*:
  assumes *forward-terminating-path x*
      and *backward-terminating-path y*
    shows $x \neq 0 \lor y \neq 0 \longleftrightarrow$ *end-points x* $\neq 0 \lor$ *start-points y* $\neq 0$
using *assms path-aux1a path-aux1b* **by** *blast*

Equivalences for *finite*

lemma *backward-finite-iff-msc*:
backward-finite x ⟷ *many-strongly-connected x* ∨ *backward-terminating x*
proof
assume *1*: *backward-finite x*
thus *many-strongly-connected x* ∨ *backward-terminating x*
proof (*cases* −(*1*;*x*);*x*;*1* = *0*)
assume −(*1*;*x*);*x*;*1* = *0*
thus *many-strongly-connected x* ∨ *backward-terminating x*
using *1* **by** (*metis conv-invol many-strongly-connected-iff-1 sup-bot-right*)
next
assume −(*1*;*x*);*x*;*1* ≠ *0*
hence *1*;−(*1*;*x*);*x*;*1* = *1*
by (*simp add*: *comp-assoc tarski*)
hence −(*1*;*x*);*x*;*1* = *1*
by (*metis comp-assoc conv-compl conv-contrav conv-invol conv-one one-compl*)
thus *many-strongly-connected x* ∨ *backward-terminating x*
using *1* **by** *simp*
qed
next
assume *many-strongly-connected x* ∨ *backward-terminating x*
thus *backward-finite x*
by (*metis star-ext sup.coboundedI1 sup.coboundedI2*)
qed

lemma *forward-finite-iff-msc*:
forward-finite x ⟷ *many-strongly-connected x* ∨ *forward-terminating x*
by (*metis backward-finite-iff-msc conv-backward-finite conv-backward-terminating conv-invol*)

lemma *finite-iff-msc*:
finite x ⟷ *many-strongly-connected x* ∨ *terminating x*
using *backward-finite-iff-msc forward-finite-iff-msc finite-iff* **by** *fastforce*

Path concatenation

lemma *path-concatenation*:
assumes *forward-terminating-path x*
and *backward-terminating-path y*
and *end-points x* = *start-points y*
and $x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0$
shows *path* (*x*+*y*)
proof (*cases y* = *0*)
assume *y* = *0*
thus *?thesis*
using *assms*(*1*) **by** *fastforce*
next
assume *as*: *y* ≠ *0*
show *?thesis*
proof (*unfold path-def*; *intro conjI*)

from *assms(4)* **have** *a*: $x;1 \cdot x^T;1 \cdot y^T;1 + x;1 \cdot y;1 \cdot y^T;1 = 0$
by (*simp add*: *inf-sup-distrib1 inf-sup-distrib2*)
hence *aux1*: $x;1 \cdot x^T;1 \cdot y^T;1 = 0$
using *sup-eq-bot-iff* **by** *blast*
from *a* **have** *aux2*: $x;1 \cdot y;1 \cdot y^T;1 = 0$
using *sup-eq-bot-iff* **by** *blast*

show *is-inj* $(x + y)$
proof (*unfold is-inj-def*; *auto simp add*: *distrib-left*)
show $x;x^T \leq 1'$
using *assms(1) path-def is-inj-def* **by** *blast*
show $y;y^T \leq 1'$
using *assms(2) path-def is-inj-def* **by** *blast*
have $y;x^T = 0$
by (*metis assms(3) aux1 annir comp-assoc conv-one le-bot modular-var-2 one-idem-mult*
path-concat-aux-2 schroeder-2)
thus $y;x^T \leq 1'$
using *bot-least le-bot* **by** *blast*
thus $x;y^T \leq 1'$
using *conv-iso* **by** *force*
qed

show *is-p-fun* $(x + y)$
proof (*unfold is-p-fun-def*; *auto simp add*: *distrib-left*)
show $x^T;x \leq 1'$
using *assms(1) path-def is-p-fun-def* **by** *blast*
show $y^T;y \leq 1'$
using *assms(2) path-def is-p-fun-def* **by** *blast*
have $y^T;x \leq y^T;(y;1 \cdot x;1)$
by (*metis conjugation-prop2 inf.commute inf-top.left-neutral maddux-20 mult-isol order-trans*
schroeder-1-var)
also have $... = 0$
using *assms(3) aux2 annir inf-commute path-concat-aux-1* **by** *fastforce*
finally show $y^T;x \leq 1'$
using *bot-least le-bot* **by** *blast*
thus $x^T;y \leq 1'$
using *conv-iso* **by** *force*
qed

show *connected* $(x + y)$
proof (*auto simp add*: *distrib-left*)
have $x;1;x \leq x^\star + x^{T\star}$
using *assms(1) path-def* **by** *simp*
also have $... \leq (x^\star;y^\star)^\star + (x^{T\star};y^{T\star})^\star$
using *join-iso join-isol star-subdist* **by** *simp*
finally show $x;1;x \leq (x^\star;y^\star)^\star + (x^{T\star};y^{T\star})^\star$ **.**
have $y;1;y \leq y^\star + y^{T\star}$

```
    using assms(2) path-def by simp
  also have ... ≤ (x⋆;y⋆)⋆ + (x^T⋆;y^T⋆)⋆
    by (metis star-denest star-subdist sup.mono sup-commute)
  finally show y;1;y ≤ (x⋆;y⋆)⋆ + (x^T⋆;y^T⋆)⋆ .

  show y;1;x ≤ (x⋆;y⋆)⋆ + (x^T⋆;y^T⋆)⋆
  proof −
    have (y;1);1;(1;x) ≤ y^T⋆;x^T⋆
    proof (rule-tac v=start-points y in path-concat-aux-0)
      show is-vector (start-points y)
        by (metis is-vector-def comp-assoc one-compl one-idem-mult ra-1)
      show start-points y ≠ 0
        using as
        by (metis assms(2) conv-compl conv-contrav conv-one inf.orderE
inf-bot-right
                inf-top.right-neutral maddux-141)
      have (start-points y);1;y^T ≤ y⋆
        by (rule path-concat-aux-5) (simp-all add: assms(2))
      thus y;1;(start-points y)^T ≤ y^T⋆
        by (metis (mono-tags, lifting) conv-iso comp-assoc conv-contrav
conv-invol conv-one
                star-conv)
      have end-points x;1;x ≤ x^T⋆
        apply (rule path-concat-aux-5)
        using assms(1) conv-path by simp-all
      thus start-points y;(1;x) ≤ x^T⋆
        by (metis assms(3) mult-assoc)
    qed
    thus ?thesis
      by (metis comp-assoc le-supI2 less-eq-def one-idem-mult star-denest
star-subdist-var-1
              sup.commute)
  qed

  show x;1;y ≤ (x⋆;y⋆)⋆ + (x^T⋆;y^T⋆)⋆
  proof −
    have (x;1);1;(1;y) ≤ x⋆;y⋆
    proof (rule-tac v=start-points y in path-concat-aux-0)
      show is-vector (start-points y)
        by (simp add: comp-assoc is-vector-def one-compl vector-1-comm)
      show start-points y ≠ 0
        using as assms(2,4) backward-terminating-iff1 galois-aux2
start-point-iff1 start-point-iff2
        by blast
      have end-points x;1;x^T ≤ x^T⋆
        apply (rule path-concat-aux-5)
        using assms(1) conv-path by simp-all
      hence (end-points x;1;x^T)^T ≤ (x^T⋆)^T
        using conv-iso by blast
```

```
        thus x;1;(start-points y)^T ≤ x^⋆
          by (simp add: assms(3) comp-assoc star-conv)
        have start-points y;1;y ≤ y^⋆
          by (rule path-concat-aux-5) (simp-all add: assms(2))
        thus start-points y;(1;y) ≤ y^⋆
          by (simp add: mult-assoc)
      qed
      thus ?thesis
        by (metis comp-assoc dual-order.trans le-supI1 one-idem-mult star-ext)
    qed
  qed
 qed
qed

lemma path-concatenation-with-edge:
   assumes x≠0
      and forward-terminating-path x
      and is-point q
      and q ≤ −(1;x)
    shows path (x+(end-points x);q^T)
proof (rule path-concatenation)
  from assms(1,2) have 1: is-point(end-points x)
    using end-point-zero-point path-aux1a by blast
  show 2: backward-terminating-path ((end-points x);q^T)
    apply (intro conjI)
    apply (metis edge-is-path 1 assms(3))
    by (metis assms(2−4) 1 bot-least comp-assoc compl-le-swap1 conv-galois-2
double-compl
              end-point-iff1 le-supE point-equations(1) tarski top-le)
  thus end-points x = start-points ((end-points x);q^T)
    by (metis assms(3) 1 edge-start comp-assoc compl-top-eq double-compl
inf.absorb-iff2 inf.commute
              inf-top-right modular-2-aux' point-equations(2))
  show x;1 · (x^T;1 + ((end-points x);q^T);1) · ((end-points x);q^T)^T;1 = 0
    using 2 by (metis assms(3,4) annir compl-le-swap1 compl-top-eq
conv-galois-2 double-compl
                   inf.absorb-iff2 inf.commute modular-1' modular-2-aux'
point-equations(2))
  show forward-terminating-path x
    by (simp add: assms(2))
qed

lemma path-concatenation-cycle-free:
  assumes forward-terminating-path x
      and backward-terminating-path y
      and end-points x = start-points y
      and x;1 · y^T;1 = 0
    shows path (x+y)
apply (rule path-concatenation,simp-all add: assms)
```

by (*metis assms*(*4*) *inf.left-commute inf-bot-right inf-commute*)

lemma *path-concatenation-start-points-approx*:
assumes *end-points x* = *start-points y*
shows *start-points* (*x*+*y*) ≤ *start-points x*
proof −
have *start-points* (*x*+*y*) = $x;1 \cdot -(x^T;1) \cdot -(y^T;1) + y;1 \cdot -(x^T;1) \cdot -(y^T;1)$
by (*simp add*: *inf.assoc inf-sup-distrib2*)
also with *assms*(*1*) **have** ... = $x;1 \cdot -(x^T;1) \cdot -(y^T;1) + x^T;1 \cdot -(x^T;1) \cdot -(x;1)$
by (*metis inf.assoc inf.left-commute*)
also have ... = $x;1 \cdot -(x^T;1) \cdot -(y^T;1)$
by *simp*
also have ... ≤ *start-points x*
using *inf-le1* **by** *blast*
finally show *?thesis* **.**
qed

lemma *path-concatenation-end-points-approx*:
assumes *end-points x* = *start-points y*
shows *end-points* (*x*+*y*) ≤ *end-points y*
proof −
have *end-points* (*x*+*y*) = $x^T;1 \cdot -(x;1) \cdot -(y;1) + y^T;1 \cdot -(x;1) \cdot -(y;1)$
by (*simp add*: *inf.assoc inf-sup-distrib2*)
also from *assms*(*1*) **have** ... = $y;1 \cdot -(y^T;1) \cdot -(y;1) + y^T;1 \cdot -(x;1) \cdot -(y;1)$
by *simp*
also have ... = $y^T;1 \cdot -(x;1) \cdot -(y;1)$
by (*simp add*: *inf.commute*)
also have ... ≤ *end-points y*
using *inf-le1 meet-iso* **by** *blast*
finally show *?thesis* **.**
qed

lemma *path-concatenation-start-points*:
assumes *end-points x* = *start-points y*
and $x;1 \cdot y^T;1 = 0$
shows *start-points* (*x*+*y*) = *start-points x*
proof −
from *assms*(*2*) **have** *aux*: $x;1 \cdot -(y^T;1) = x;1$
by (*simp add*: *galois-aux inf.absorb1*)

have *start-points* (*x*+*y*) = $(x;1 \cdot -(x^T;1) \cdot -(y^T;1)) + (y;1 \cdot -(x^T;1) \cdot -(y^T;1))$
by (*simp add*: *inf-sup-distrib2 inf.assoc*)
also from *assms*(*1*) **have** ... = $(x;1 \cdot -(x^T;1) \cdot -(y^T;1)) + (x^T;1 \cdot -(x;1) \cdot -(x^T;1))$
using *inf.assoc inf.commute* **by** *simp*
also have ... = $(x;1 \cdot -(x^T;1) \cdot -(y^T;1))$

by (*simp add*: *inf.assoc*)
also from *aux* **have** ... = $x;1 \cdot -(x^T;1)$
by (*metis inf.assoc inf.commute*)
finally show *?thesis* **.**
qed

lemma *path-concatenation-end-points*:
assumes *end-points x* = *start-points y*
and $x;1 \cdot y^T;1 = 0$
shows *end-points* $(x+y)$ = *end-points y*
proof –
from *assms*(2) **have** *aux*: $y^T;1 \cdot -(x;1) = y^T;1$
using *galois-aux inf.absorb1 inf-commute* **by** *blast*

have *end-points* $(x+y) = (x^T;1 + y^T;1) \cdot -(x;1) \cdot -(y;1)$
using *inf.assoc* **by** *simp*
also from *assms*(1) **have** ... = $(y;1 \cdot -(y^T;1) \cdot -(y;1)) + (y^T;1 \cdot -(x;1) \cdot -(y;1))$
by (*simp add*: *inf-sup-distrib2*)
also have ... = $y^T;1 \cdot -(x;1) \cdot -(y;1)$
by (*simp add*: *inf.assoc*)
also from *aux* **have** ... = $y^T;1 \cdot -(y;1)$
by (*metis inf.assoc inf.commute*)
finally show *?thesis* **.**
qed

lemma *path-concatenation-cycle-free-complete*:
assumes *forward-terminating-path x*
and *backward-terminating-path y*
and *end-points x* = *start-points y*
and $x;1 \cdot y^T;1 = 0$
shows *path* $(x+y)$ ∧ *start-points* $(x+y)$ = *start-points x* ∧ *end-points* $(x+y)$ = *end-points y*
using *assms path-concatenation-cycle-free path-concatenation-end-points path-concatenation-start-points*
by *blast*

Path restriction (path from a given point)

lemma *reachable-points-iff*:
assumes *point p*
shows $(x^{T\star};p \cdot x) = (x^{T\star};p \cdot 1');x$
proof (*rule antisym*)
show $(x^{T\star};p \cdot 1');x \leq x^{T\star};p \cdot x$
proof (*rule le-infI*)
show $(x^{T\star};p \cdot 1');x \leq x^{T\star};p$
proof –
have $(x^{T\star};p \cdot 1');x \leq x^{T\star};p;1$
by (*simp add*: *mult-isol-var*)
also have ... $\leq x^{T\star};p$

using *assms* **by** (*simp add*: *comp-assoc eq-iff point-equations*(*1*)
point-is-point)
finally show *?thesis* **.**
qed
show $(x^{T\star};p \cdot 1');x \leq x$
by (*metis inf-le2 mult-isor mult-onel*)
qed
show $x^{T\star};p \cdot x \leq (x^{T\star};p \cdot 1');x$
proof −
have $(x^{T\star};p);x^{T} \leq x^{T\star};p + -1'$
by (*metis assms comp-assoc is-vector-def mult-isol point-def sup.coboundedI1*
top-greatest)
hence *aux*: $(-(x^{T\star};p) \cdot 1');x \leq -(x^{T\star};p)$
using *compl-mono conv-galois-2* **by** *fastforce*
have $x = (x^{T\star};p \cdot 1');x + (-(x^{T\star};p) \cdot 1');x$
by (*metis aux4 distrib-right inf-commute mult-1-left*)
also with *aux* **have** ... $\leq (x^{T\star};p \cdot 1');x + -(x^{T\star};p)$
using *join-isol* **by** *blast*
finally have $x \leq (x^{T\star};p \cdot 1');x + -(x^{T\star};p)$ **.**
thus *?thesis*
using *galois-2 inf.commute* **by** *fastforce*
qed
qed

lemma *path-from-given-point*:
assumes *path x*
and *point p*
shows *path*$(x^{T\star};p \cdot x)$
and *start-points*$(x^{T\star};p \cdot x) \leq p$
and *end-points*$(x^{T\star};p \cdot x) \leq$ *end-points*(x)
proof (*unfold path-def*; *intro conjI*)
show *uni*: *is-p-fun* $(x^{T\star};p \cdot x)$
by (*metis assms*(*1*) *inf-commute is-p-fun-def p-fun-mult-var path-def*)
show *inj*: *is-inj* $(x^{T\star};p \cdot x)$
by (*metis abel-semigroup.commute assms*(*1*) *conv-times*
inf.abel-semigroup-axioms inj-p-fun
is-p-fun-def p-fun-mult-var path-def)
show *connected* $(x^{T\star};p \cdot x)$
proof −
let $?t = x^{T\star};p \cdot 1'$
let $?u = -(x^{T\star};p) \cdot 1'$

have *t-plus-u*: $?t + ?u = 1'$
by (*simp add*: *inf.commute*)
have *t-times-u*: $?t ; ?u \leq 0$
by (*simp add*: *inf.left-commute is-test-def test-comp-eq-mult*)
have *t-conv*: $?t^{T} = ?t$
using *inf.cobounded2 is-test-def test-eq-conv* **by** *blast*
have *txu-zero*: $?t;x;?u \leq 0$

```
    proof -
      have x^T;?t;1 ≤ -?u
      proof -
        have x^T;?t;1 ≤ x^T;x^T⋆;p
          using assms(2)
          by (simp add: is-vector-def mult.semigroup-axioms mult-isol-var
mult-subdistr order.refl
                        point-def semigroup.assoc)
        also have ... ≤ -?u
          by (simp add: le-supI1 mult-isor)
        finally show ?thesis .
      qed
      thus ?thesis
        by (metis compl-bot-eq compl-le-swap1 conv-contrav conv-galois-1 t-conv)
    qed
    hence txux-zero: ?t;x;?u;x ≤ 0
      using annil le-bot by fastforce


    have tx-leq: ?t;x⋆ ≤ (?t;x)⋆
    proof -
      have ?t;x⋆ = ?t;(?t;x + ?u;x)⋆
        using t-plus-u by (metis distrib-right' mult-onel)
      also have ... = ?t;(?u;x;(?u;x)⋆;(?t;x)⋆+(?t;x)⋆)
        using txux-zero star-denest-10 by (simp add: comp-assoc le-bot)
      also have ... = ?t;?u;x;(?u;x)⋆;(?t;x)⋆+?t;(?t;x)⋆
        by (simp add: comp-assoc distrib-left)
      also have ... ≤ 0;x;(?u;x)⋆;(?t;x)⋆+?t;(?t;x)⋆
        using le-bot t-times-u by blast
      also have ... ≤(?t;x)⋆
        by (metis annil inf.commute inf-bot-right le-supI mult-onel mult-subdistr)
      finally show ?thesis .
    qed

    hence aux: ?t;x⋆;?t ≤ (?t;x)⋆
      using inf.cobounded2 order.trans prod-star-closure star-ref by blast
    with t-conv have aux-trans: ?t;x^T⋆;?t ≤ (?t;x)^T⋆
      by (metis comp-assoc conv-contrav conv-self-conjugate-var g-iso star-conv)

    from aux aux-trans have ?t;(x⋆+x^T⋆);?t ≤ (?t;x)⋆ + (?t;x)^T⋆
      by (metis sup-mono distrib-right' distrib-left)
    with assms(1) path-concat-aux3-1 have ?t;(x;1;x^T);?t ≤ (?t;x)⋆ + (?t;x)^T⋆
      using dual-order.trans mult-double-iso by blast
    with t-conv have (?t;x);1;(?t;x)^T ≤ (?t;x)⋆ + (?t;x)^T⋆
      using comp-assoc conv-contrav by fastforce
    with connected-iff2 show ?thesis
      using assms(2) inj reachable-points-iff uni by fastforce
  qed
next
```

show *start-points* $(x^{T\star};p \cdot x) \leq p$
proof $-$
have *1*: *is-vector* $(x^{T\star};p)$
using *assms*(*2*) **by** (*simp add*: *is-vector-def mult-assoc point-def*)
hence $(x^{T\star};p \cdot x);1 \leq x^{T\star};p$
by (*simp add*: *inf.commute vector-1-comm*)
also have ... $= x^{T+};p + p$
by (*simp add*: *sup.commute*)
finally have *2*: $(x^{T\star};p \cdot x);1 \cdot -(x^{T+};p) \leq p$
using *galois-1* **by** *blast*
have $(x^{T\star};p \cdot x)^T;1 = (x^T \cdot (x^{T\star};p)^T);1$
by (*simp add*: *inf.commute*)
also have ... $= x^T;(x^{T\star};p \cdot 1)$
using *1 vector-2* **by** *blast*
also have ... $= x^{T+};p$
by (*simp add*: *comp-assoc*)
finally show *start-points* $(x^{T\star};p \cdot x) \leq p$
using *2* **by** *simp*
qed
next
show *end-points*$(x^{T\star};p \cdot x) \leq$ *end-points*(x)
proof $-$
have *1*: *is-vector* $(x^{T\star};p)$
using *assms*(*2*) **by** (*simp add*: *is-vector-def mult-assoc point-def*)
have $(x^{T\star};p \cdot x)^T;1 = ((x^{T\star};p)^T \cdot x^T);1$
by (*simp add*: *star-conv*)
also have ... $= x^T;(x^{T\star};p \cdot 1)$
using *1 vector-2 inf.commute* **by** *fastforce*
also have ... $\leq x^{T\star};p$
using *comp-assoc mult-isor* **by** *fastforce*
finally have *2*: $(x^{T\star};p \cdot x)^T;1 \cdot -(x^{T\star};p) = 0$
using *galois-aux2* **by** *blast*
have $(x^{T\star};p \cdot x)^T;1 \cdot -((x^{T\star};p \cdot x);1) = (x^{T\star};p \cdot x)^T;1 \cdot (-(x^{T\star};p) + -(x;1))$
using *1 vector-1* **by** *fastforce*
also have ... $= (x^{T\star};p \cdot x)^T;1 \cdot -(x^{T\star};p) + (x^{T\star};p \cdot x)^T;1 \cdot -(x;1)$
using *inf-sup-distrib1* **by** *blast*
also have ... $= (x^{T\star};p \cdot x)^T;1 \cdot -(x;1)$
using *2* **by** *simp*
also have ... $\leq x^T;1 \cdot -(x;1)$
using *meet-iso mult-subdistr-var* **by** *fastforce*
finally show *?thesis* **.**
qed
qed

lemma *path-from-given-point'*:
assumes *has-start-points-path x*
and *point p*
and $p \leq x;1$

shows *path*$(x^{T\star};p \cdot x)$
and *start-points*$(x^{T\star};p \cdot x) = p$
and *end-points*$(x^{T\star};p \cdot x) =$ *end-points*(x)
proof −
show *path*$(x^{T\star};p \cdot x)$
using *assms path-from-given-point*(*2*) **by** *blast*
next
show *start-points*$(x^{T\star};p \cdot x) = p$
proof (*simp only*: *eq-iff*; *rule conjI*)
show *start-points*$(x^{T\star};p \cdot x) \le p$
using *assms path-from-given-point*(*2*) **by** *blast*
show $p \le$ *start-points*$(x^{T\star};p \cdot x)$
proof −
have *1*: *is-vector*$(x^{T\star};p)$
using *assms*(*2*) *comp-assoc is-vector-def point-equations*(*1*) *point-is-point*
by *fastforce*
hence *a*: $p \le (x^{T\star};p \cdot x);1$
by (*metis vector-1 assms*(*3*) *conway.dagger-unfoldl-distr inf.orderI*
inf-greatest
inf-sup-absorb)

have $x^{T+};p \cdot p \le (x^{T+} \cdot 1');p$
using *assms*(*2*) *inj-distr point-def* **by** *fastforce*
also have ... $\le (-1'^T \cdot 1');p$
using *assms*(*1*) *path-acyclic*
by (*metis conv-contrav conv-e meet-iso mult-isor star-conv star-slide-var*
test-converse)
also have ... ≤ 0
by *simp*
finally have *2*: $x^{T+};p \cdot p \le 0$ **.**

have *b*: $p \le -((x^{T\star};p \cdot x)^T;1)$
proof −
have $(x^{T\star};p \cdot x)^T;1 = ((x^{T\star};p)^T \cdot x^T);1$
by (*simp add*: *star-conv*)
also have ... $= x^T;(x^{T\star};p \cdot 1)$
using *1 vector-2 inf.commute* **by** *fastforce*
also have ... $= x^T;x^{T\star};p$
by (*simp add*: *comp-assoc*)
also have ... $\le -p$
using *2 galois-aux le-bot* **by** *blast*
finally show *?thesis*
using *compl-le-swap1* **by** *blast*
qed
with *a* **show** *?thesis*
by *simp*
qed
qed
next

show *end-points*$(x^{T\star};p \cdot x)$ = *end-points*(x)
proof (*simp only*: *eq-iff*; *rule conjI*)
show *end-points*$(x^{T\star};p \cdot x) \leq$ *end-points*(x)
using *assms path-from-given-point*(*3*) **by** *blast*
show *end-points*$(x) \leq$ *end-points*$(x^{T\star};p \cdot x)$
proof −
have *1*: *is-vector*$(x^{T\star};p)$
using *assms*(*2*) *comp-assoc is-vector-def point-equations*(*1*) *point-is-point*
by *fastforce*
have *2*: *is-vector*(*end-points*(x))
by (*simp add*: *comp-assoc is-vector-def one-compl vector-1-comm*)
have *a*: *end-points*$(x) \leq (x^{T\star};p \cdot x)^T;1$
proof −
have $x^T;1 \cdot 1;x^T = x^T;1;x^T$
by (*simp add*: *vector-meet-comp-x'*)
also have ... $\leq x^{T\star} + x^\star$
using *assms*(*1*) *path-concat-aux3-3 sup.commute* **by** *fastforce*
also have ... $= x^{T\star} + x^+$
by (*simp add*: *star-star-plus sup.commute*)
also have ... $\leq x^{T\star} + x;1$
using *join-isol mult-isol* **by** *fastforce*
finally have *end-points*$(x) \cdot 1;x^T \leq x^{T\star}$
by (*metis galois-1 inf.assoc inf.commute sup-commute*)
hence *end-points*$(x) \cdot p^T \leq x^{T\star}$
using *assms*(*3*)
by (*metis conv-contrav conv-iso conv-one dual-order.trans inf.cobounded1*
inf.right-idem
inf-mono)
hence *end-points*(x) ; $p^T \leq x^{T\star}$
using *assms*(*2*) *2* **by** (*simp add*: *point-def vector-meet-comp*)
hence *end-points*$(x) \leq x^{T\star};p$
using *assms*(*2*) *point-def ss423bij* **by** *blast*
hence $x^T;1 \leq x^{T\star};p + x;1$
by (*simp add*: *galois-1 sup-commute*)
hence $x^T;1 \leq x^{T+};p + p + x;1$
by (*metis conway.dagger-unfoldl-distr sup-commute*)
hence $x^T;1 \leq x^{T+};p + x;1$
by (*simp add*: *assms*(*3*) *sup.absorb2 sup.assoc*)
hence *end-points*$(x) \leq x^{T+};p$
by (*simp add*: *galois-1 sup-commute*)
also have ... $= (x^{T\star};p \cdot x)^T;1$
using *1 inf-commute mult-assoc vector-2* **by** *fastforce*
finally show *?thesis* **.**
qed
have $x^T;1 \cdot (x^{T\star};p \cdot x);1 \leq x;1$
by (*simp add*: *le-infI2 mult-isor*)
hence *b*: *end-points*$(x) \leq -((x^{T\star};p \cdot x);1)$
using *galois-1 galois-2* **by** *blast*
with *a* **show** *?thesis*

```
      by simp
    qed
  qed
qed
```

Cycles

```
lemma selfloop-is-cycle:
  assumes is-point x
  shows cycle (x;x^T)
  by (simp add: assms edge-is-path)

lemma start-point-no-cycle:
  assumes has-start-points-path x
    shows ¬ cycle x
using assms many-strongly-connected-implies-no-start-end-points
no-start-end-points-iff
      start-point-iff1 start-point-iff2 by blast

lemma end-point-no-cycle:
  assumes has-end-points-path x
    shows ¬ cycle x
using assms end-point-iff2 end-point-iff1
many-strongly-connected-implies-no-start-end-points
      no-start-end-points-iff by blast

lemma cycle-no-points:
  assumes cycle x
    shows start-points x = 0
      and end-points x = 0
  by (metis assms inf-compl-bot
many-strongly-connected-implies-no-start-end-points)+
```

Path concatenation to cycle

```
lemma path-path-equals-cycle-aux:
  assumes has-start-end-points-path x
      and has-start-end-points-path y
      and start-points x = end-points y
      and end-points x = start-points y
shows x ≤ (x+y)^T⋆
proof−
  let ?e = end-points(x)
  let ?s = start-points(x)
  have sp: is-point(?s)
    using assms(1) start-point-iff2 has-start-end-points-path-iff by blast
  have ep: is-point(?e)
    using assms(1) end-point-iff2 has-start-end-points-path-iff by blast

  have x ≤ x^T⋆;?s;1 · 1;?e^T;x^T⋆
      by (metis assms(1) backward-terminating-path-start-points-2 end-point-iff2
ep
```

forward-terminating-iff1 forward-terminating-path-end-points-2
comp-assoc
conv-contrav conv-invol conv-iso inf.boundedI point-equations(1)
point-equations(4)
star-conv sp start-point-iff2)
also have ... $= x^{T\star};?s;1;?e^T;x^{T\star}$
by (*metis inf-commute inf-top-right ra-1*)
also have ... $= x^{T\star};?s;?e^T;x^{T\star}$
by (*metis ep comp-assoc point-equations(4)*)
also have ... $\leq x^{T\star};y^{T\star};x^{T\star}$
by (*metis (mono-tags, lifting) assms(2−4) start-to-end comp-assoc*
conv-contrav conv-invol
conv-iso mult-double-iso star-conv)
also have ... $= (x^\star;y^\star;x^\star)^T$
by (*simp add: comp-assoc star-conv*)
also have ... $\leq ((x+y)^\star;(x+y)^\star;(x+y)^\star)^T$
by (*metis conv-invol conv-iso prod-star-closure star-conv star-denest star-ext*
star-iso
star-trans-eq sup-ge1)
also have ... $= (x+y)^{T\star}$
by (*metis star-conv star-trans-eq*)
finally show *x*: $x \leq (x+y)^{T\star}$ **.**
qed

lemma *path-path-equals-cycle*:
assumes *has-start-end-points-path x*
and *has-start-end-points-path y*
and *start-points x = end-points y*
and *end-points x = start-points y*
and $x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0$
shows *cycle*$(x + y)$
proof (*intro conjI*)
show *path* $(x + y)$
apply (*rule path-concatenation*)
using *assms* **by**(*simp-all add:has-start-end-points-iff*)
show *many-strongly-connected* $(x + y)$
by (*metis path-path-equals-cycle-aux assms(1−4) sup.commute le-supI*
many-strongly-connected-iff-3)
qed

lemma *path-edge-equals-cycle*:
assumes *has-start-end-points-path x*
shows *cycle*$(x + \textit{end-points}(x);(\textit{start-points } x)^T)$
proof (*rule path-path-equals-cycle*)
let *?s = start-points x*
let *?e = end-points x*
let $?y = (?e;?s^T)$

have *sp*: *is-point(?s)*

```
    using start-point-iff2 assms has-start-end-points-path-iff by blast
  have ep: is-point(?e)
    using end-point-iff2 assms has-start-end-points-path-iff by blast

  show has-start-end-points-path x
    using assms by blast
  show has-start-end-points-path ?y
    using edge-is-path
    by (metis assms edge-end edge-start end-point-iff2 end-point-iff1 galois-aux2
          has-start-end-points-iff inf.left-idem inf-compl-bot-right start-point-iff2)
  show ?s = end-points ?y
    by (metis sp ep edge-end annil conv-zero inf.left-idem inf-compl-bot-right)
  thus ?e = start-points ?y
    by (metis edge-start ep conv-contrav conv-invol sp)
  show x;1 · (x^T;1 + ?e;?s^T;1) · (?e;?s^T)^T;1 = 0
  proof -
    have x;1 · (x^T;1 + ?e;?s^T;1) · (?e;?s^T)^T;1 = x;1 · (x^T;1 + ?e;1;?s^T;1) ·
(?s;?e^T);1
      using sp comp-assoc point-equations(3) by fastforce
    also have ... = x;1 · (x^T;1 + ?e;1) · ?s;1
      by (metis sp ep comp-assoc point-equations(1,3))
    also have ... ≤ 0
      by (simp add: sp ep inf.assoc point-equations(1))
    finally show ?thesis
      using bot-unique by blast
  qed
qed
```

Break cycles

```
lemma cycle-remove-edge:
  assumes cycle x
      and point s
      and point e
      and e;s^T ≤ x
    shows path(x · −(e;s^T))
      and start-points (x · −(e;s^T)) ≤ s
      and end-points (x · −(e;s^T)) ≤ e
proof -
  show path(x · −(e;s^T))
  proof (unfold path-def; intro conjI)
    show 1: is-p-fun(x · −(e;s^T))
      using assms(1) path-def is-p-fun-def p-fun-mult-var by blast
    show 2: is-inj(x · −(e;s^T))
      using assms(1) path-def inf.cobounded1 injective-down-closed by blast
    show connected (x · −(e;s^T))
    proof -
      have x⋆ = ((x · −(e;s^T)) + e;s^T)⋆
        by (metis assms(4) aux4-comm inf.absorb2)
      also have ... = (x · −(e;s^T))⋆ ; (e;s^T ; (x · −(e;s^T))⋆)⋆
```

by *simp*
also have ... $= (x \cdot -(e;s^T))^\star$; $(1' + e;s^T$; $(x \cdot -(e;s^T))^\star;(e;s^T$; $(x \cdot -(e;s^T))^\star)^\star)$
by *fastforce*
also have ... $= (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;s^T$; $(x \cdot -(e;s^T))^\star;(e;s^T$; $(x \cdot -(e;s^T))^\star)^\star$
by (*simp add*: *distrib-left mult-assoc*)
also have ... $= (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;(s^T$; $(x \cdot -(e;s^T))^\star;e)^\star;s^T$; $(x \cdot -(e;s^T))^\star$
by (*simp add*: *comp-assoc star-slide*)
also have ... $\leq (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;1;s^T$; $(x \cdot -(e;s^T))^\star$
using *top-greatest join-isol mult-double-iso* **by** (*metis mult-assoc*)
also have ... $= (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;s^T$; $(x \cdot -(e;s^T))^\star$
using *assms(3)* **by** (*simp add*: *comp-assoc is-vector-def point-def*)
finally have *3*: $x^\star \leq (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;s^T$; $(x \cdot -(e;s^T))^\star$
.

from *assms(4)* **have** $e;s^T \leq e;e^T;x$
using *assms(3) comp-assoc mult-isol point-def ss423conv* **by** *fastforce*
also have ... $\leq e;e^T;(x^\star)^T$
using *assms(1) many-strongly-connected-iff-3 mult-isol star-conv* **by** *fastforce*
also have ... $\leq e;e^T;((x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;s^T$; $(x \cdot -(e;s^T))^\star)^T$
using *3 conv-iso mult-isol* **by** *blast*
also have ... $\leq e;e^T;((x \cdot -(e;s^T))^{T\star} + (x \cdot -(e;s^T))^{T\star}$; $s;e^T$; $(x \cdot -(e;s^T))^{T\star})$
by (*simp add*: *star-conv comp-assoc*)
also have ... $\leq e;e^T;(x \cdot -(e;s^T))^{T\star} + e;e^T;(x \cdot -(e;s^T))^{T\star}$; $s;e^T$; $(x \cdot -(e;s^T))^{T\star}$
by (*simp add*: *comp-assoc distrib-left*)
also have ... $\leq e;e^T;(x \cdot -(e;s^T))^{T\star} + e;1;e^T$; $(x \cdot -(e;s^T))^{T\star}$
by (*metis comp-assoc join-isol mult-isol mult-isor top-greatest*)
also have ... $\leq e;e^T;(x \cdot -(e;s^T))^{T\star} + e;e^T;(x \cdot -(e;s^T))^{T\star}$
using *assms(3)* **by** (*simp add*: *point-equations(1) point-is-point*)
also have ... $= e;e^T;(x \cdot -(e;s^T))^{T\star}$
by *simp*
also have ... $\leq 1';(x \cdot -(e;s^T))^{T\star}$
using *assms(3) is-inj-def point-def join-iso mult-isor* **by** *blast*
finally have *4*: $e;s^T \leq (x \cdot -(e;s^T))^{T\star}$
by *simp*

have $(x \cdot -(e;s^T));1;(x \cdot -(e;s^T)) \leq x;1;x$
by (*simp add*: *mult-isol-var*)
also have ...$\leq x^\star$
using *assms(1) connected-iff4 one-strongly-connected-iff one-strongly-connected-implies-8 path-concat-aux3-3* **by** *blast*
also have ... $\leq (x \cdot -(e;s^T))^\star + (x \cdot -(e;s^T))^\star$; $e;s^T$; $(x \cdot -(e;s^T))^\star$

```
      by (rule 3)
    also have ... ≤ (x · −(e;s^T))^⋆ + (x · −(e;s^T))^⋆ ; (x · −(e;s^T))^T⋆ ; (x ·
−(e;s^T))^⋆
      using 4 by (metis comp-assoc join-isol mult-isol mult-isor)
    also have ... ≤ (x · −(e;s^T))^⋆ + (x · −(e;s^T))^T⋆
      using 1 2 triple-star by force
    finally show ?thesis .
  qed
 qed
next
 show start-points (x · −(e;s^T)) ≤ s
 proof −
  have 1: is-vector(−s)
   using assms(2) by (simp add: point-def vector-compl)
  have (x · −(e;s^T));1 · −s ≤ x;1 · −s
   using meet-iso mult-subdistr by blast
  also have ... ≤ x^T;1 · −s
   using assms(1) many-strongly-connected-implies-no-start-end-points meet-iso
      no-start-end-points-path-iff by blast
  also have ... ≤ (x^T · −s);1
   using 1 by (simp add: vector-1-comm)
  also have ... ≤ (x^T · −(s;e^T));1
   by (metis 1 galois-aux inf.boundedI inf.cobounded1 inf.commute mult-isor
schroeder-2
        vector-1-comm)
  also have ... = (x · −(e;s^T))^T;1
   by (simp add: conv-compl)
  finally show ?thesis
   by (simp add: galois-1 sup-commute)
 qed
next
 show end-points (x · −(e;s^T)) ≤ e
 proof −
  have 1: is-vector(−e)
   using assms(3) by (simp add: point-def vector-compl)
  have (x · −(e;s^T))^T;1 · −e ≤ x^T;1 · −e
   using meet-iso mult-subdistr by simp
  also have ... ≤ x;1 · −e
   using assms(1) many-strongly-connected-implies-no-start-end-points meet-iso
      no-start-end-points-path-iff by blast
  also have ... ≤ (x · −e);1
   using 1 by (simp add: vector-1-comm)
  also have ... ≤ (x · −(e;s^T));1
   by (metis 1 galois-aux inf.boundedI inf.cobounded1 inf.commute mult-isor
schroeder-2
        vector-1-comm)
  finally show ?thesis
   by (simp add: galois-1 sup-commute)
 qed
```

qed

lemma *cycle-remove-edge′*:
assumes *cycle x*
and *point s*
and *point e*
and $s \neq e$
and $e;s^T \leq x$
shows $path(x \cdot -(e;s^T))$
and $s = start\text{-}points\ (x \cdot -(e;s^T))$
and $e = end\text{-}points\ (x \cdot -(e;s^T))$
proof −
show $path\ (x \cdot -\ (e\ ;\ s^T))$
using *assms*(*1*,*2*,*3*,*5*) *cycle-remove-edge*(*1*) **by** *blast*
next
show $s = start\text{-}points\ (x \cdot -\ (e\ ;\ s^T))$
proof (*simp only*: *eq-iff*; *rule conjI*)
show $s \leq start\text{-}points\ (x \cdot -\ (e\ ;\ s^T))$
proof −
have *a*: $s \leq (x \cdot -\ (e\ ;\ s^T));1$
proof −
have *1*: *is-vector*$(-e)$
using *assms*(*3*) *point-def vector-compl* **by** *blast*
from *assms*(*2*−*4*) **have** $s = s \cdot -e$
using *comp-assoc edge-end point-equations*(*1*) *point-equations*(*3*)
point-is-point **by** *fastforce*
also have ... $\leq x^T;e \cdot -e$
using *assms*(*3*,*5*) *conv-iso meet-iso point-def ss423conv* **by** *fastforce*
also have ... $\leq x;1 \cdot -e$
by (*metis assms*(*1*) *many-strongly-connected-implies-no-start-end-points*
meet-iso mult-isol
top-greatest)
also have ... $\leq (x \cdot -e);1$
using *1* **by** (*simp add*: *vector-1-comm*)
also have ... $\leq (x \cdot -\ (e\ ;\ s^T));1$
by (*metis assms*(*3*) *comp-anti is-vector-def meet-isor mult-isol mult-isor*
point-def
top-greatest)
finally show *?thesis* **.**
qed
have *b*: $s \leq -((x \cdot -\ (e\ ;\ s^T))^T;1)$
proof −
have *1*: $x;s = e$
using *assms predecessor-point′* **by** *blast*
have $s \cdot x^T = s;(e^T + -(e^T)) \cdot x^T$
using *assms*(*2*) *point-equations*(*1*) *point-is-point* **by** *fastforce*
also have ... $= s;e^T \cdot x^T$
by (*metis 1 conv-contrav inf.commute inf-sup-absorb modular-1′*)
also have ... $\leq e^T$

```
        by (metis assms(3) inf.coboundedI1 mult-isor point-equations(4)
point-is-point
                top-greatest)
      finally have s · x^T ≤ s · e^T
        by simp
      also have ... ≤ s ; e^T
        using assms(2,3) by (simp add: point-def vector-meet-comp)
      finally have 2: s · x^T · −(s ; e^T) = 0
        using galois-aux2 by blast
      thus ?thesis
      proof −
        have s ; e^T = e^T · s
          using assms(2,3) inf-commute point-def vector-meet-comp by force
        thus ?thesis
          using 2
          by (metis assms(2,3) conv-compl conv-invol conv-one conv-times
galois-aux
                  inf.assoc point-def point-equations(1) point-is-point schroeder-2
                  vector-meet-comp)
      qed
    qed
    with a show ?thesis
      by simp
  qed
  show start-points (x · − (e ; s^T)) ≤ s
    using assms(1,2,3,5) cycle-remove-edge(2) by blast
 qed
next
 show e = end-points (x · − (e ; s^T))
 proof (simp only: eq-iff; rule conjI)
  show e ≤ end-points (x · − (e ; s^T))

  proof −
    have a: e ≤ (x · − (e ; s^T))^T;1
    proof −
      have 1: is-vector(−s)
        using assms(2) point-def vector-compl by blast
      from assms(2−4) have e = e · −s
        using comp-assoc edge-end point-equations(1) point-equations(3)
point-is-point by fastforce
      also have ... ≤ x;s · −s
        using assms(2,5) meet-iso point-def ss423bij by fastforce
      also have ... ≤ x^T;1 · −s
        by (metis assms(1) many-strongly-connected-implies-no-start-end-points
meet-iso mult-isol
                top-greatest)
      also have ... ≤ (x^T · −s);1
        using 1 by (simp add: vector-1-comm)
      also have ... ≤ (x^T · − (s ; e^T));1
```

```
        by (metis assms(2) comp-anti is-vector-def meet-isor mult-isol mult-isor
point-def
                  top-greatest)
      finally show ?thesis
        by (simp add: conv-compl)
    qed
    have b: e ≤ −((x · − (e ; s^T));1)
    proof −
      have 1: x^T;e =s
        using assms predecessor-point' by (metis conv-contrav conv-invol
conv-iso conv-path)
      have e · x = e;(s^T+−(s^T)) · x
        using assms(3) point-equations(1) point-is-point by fastforce
      also have ... = e;s^T · x
        by (metis 1 conv-contrav conv-invol inf.commute inf-sup-absorb
modular-1')
      also have ... ≤ s^T
        by (metis assms(2) inf.coboundedI1 mult-isor point-equations(4)
point-is-point top-greatest)
      finally have e · x ≤ e · s^T
        by simp
      also have ... ≤ e ; s^T
        using assms(2,3) by (simp add: point-def vector-meet-comp)
      finally have 2: e · x · −(e ; s^T) = 0
        using galois-aux2 by blast
      thus ?thesis
      proof −
        have e ; s^T = s^T · e
          using assms(2,3) inf-commute point-def vector-meet-comp by force
        thus ?thesis
          using 2
          by (metis assms(2,3) conv-one galois-aux inf.assoc point-def
point-equations(1)
                  point-is-point schroeder-2 vector-meet-comp)
      qed
    qed
    with a show ?thesis
      by simp
  qed
  show end-points (x · − (e ; s^T)) ≤ e
    using assms(1,2,3,5) cycle-remove-edge(3) by blast
 qed
qed

end

end
```

3 Relational Characterisation of Rooted Paths

We characterise paths together with a designated root. This is important as often algorithms start with a single vertex, and then build up a path, a tree or another structure. An example is Dijkstra's shortest path algorithm.

theory *Rooted-Paths*

imports *Paths*

begin

context *relation-algebra*
begin

General theorems

lemma *step-has-target*:
  assumes $x;r \neq 0$
    shows $x^T;1 \neq 0$
using *assms inf.commute inf-bot-right schroeder-1* **by** *fastforce*

lemma *end-point-char*:
  $x^T;p = 0 \longleftrightarrow p \leq -(x;1)$
using *antisym bot-least compl-bot-eq conv-galois-1* **by** *fastforce*

end

context *relation-algebra-tarski*
begin

General theorems concerning points

lemma *successor-point*:
  assumes *is-inj* x
    and *point* r
    and $x;r \neq 0$
  shows *point* $(x;r)$
using *assms*
by (*simp add*: *inj-compose is-point-def is-vector-def mult-assoc point-is-point*)

lemma *no-end-point-char*:
  assumes *point* p
    shows $x^T;p \neq 0 \longleftrightarrow p \leq x;1$
by (*simp add*: *assms comp-assoc end-point-char is-vector-def*
point-in-vector-or-complement-iff)

lemma *no-end-point-char-converse*:
  assumes *point* p
    shows $x;p \neq 0 \longleftrightarrow p \leq x^T;1$
using *assms no-end-point-char* **by** *force*

end

3.1 Consequences without the Tarski rule

context *relation-algebra-rtc*
begin

Definitions for path classifications

definition *path-root*
where *path-root* $r\ x \equiv r;x \leq x^{\star} + x^{T\star} \wedge$ *is-inj* $x \wedge$ *is-p-fun* $x \wedge$ *point* r

abbreviation *connected-root*
where *connected-root* $r\ x \equiv r;x \leq x^{+}$

definition *backward-finite-path-root*
where *backward-finite-path-root* $r\ x \equiv$ *connected-root* $r\ x \wedge$ *is-inj* $x \wedge$ *is-p-fun* $x \wedge$ *point* r

abbreviation *backward-terminating-path-root*
where *backward-terminating-path-root* $r\ x \equiv$ *backward-finite-path-root* $r\ x \wedge x;r = 0$

abbreviation *cycle-root*
where *cycle-root* $r\ x \equiv r;x \leq x^{+} \cdot x^{T};1 \wedge$ *is-inj* $x \wedge$ *is-p-fun* $x \wedge$ *point* r

abbreviation *non-empty-cycle-root*
where *non-empty-cycle-root* $r\ x \equiv$ *backward-finite-path-root* $r\ x \wedge r \leq x^{T};1$

abbreviation *finite-path-root-end*
where *finite-path-root-end* $r\ x\ e \equiv$ *backward-finite-path-root* $r\ x \wedge$ *point* $e \wedge r \leq x^{\star};e$

abbreviation *terminating-path-root-end*
where *terminating-path-root-end* $r\ x\ e \equiv$ *finite-path-root-end* $r\ x\ e \wedge x^{T};e = 0$

Equivalent formulations of *connected-root*

lemma *connected-root-iff1*:
assumes *point* r
shows *connected-root* $r\ x \longleftrightarrow 1;x \leq r^{T};x^{+}$
by (*metis assms comp-assoc is-vector-def point-def ss423conv*)

lemma *connected-root-iff2*:
assumes *point* r
shows *connected-root* $r\ x \longleftrightarrow x^{T};1 \leq x^{T+};r$
by (*metis assms conv-contrav conv-invol conv-iso conv-one star-conv star-slide-var connected-root-iff1*)

lemma *connected-root-aux*:
$x^{T+};r \leq x^{T};1$
by (*simp add*: *comp-assoc mult-isol*)

lemma *connected-root-iff3*:
  assumes *point r*
    shows *connected-root r x* $\longleftrightarrow$ $x^T;1 = x^{T+};r$
using *assms antisym connected-root-aux connected-root-iff2* **by** *fastforce*

lemma *connected-root-iff4*:
  assumes *point r*
    shows *connected-root r x* $\longleftrightarrow$ $1;x = r^T;x^+$
by (*metis assms conv-contrav conv-invol conv-one star-conv star-slide-var connected-root-iff3*)

Consequences of *connected-root*

lemma *has-root-contra*:
  assumes *connected-root r x*
    and *point r*
    and $x^T;r = 0$
  shows $x = 0$
using *assms comp-assoc independence1 conv-zero ss-p18 connected-root-iff3*
by *force*

lemma *has-root*:
  assumes *connected-root r x*
    and *point r*
    and $x \neq 0$
  shows $x^T;r \neq 0$
using *has-root-contra assms* **by** *blast*

lemma *connected-root-move-root*:
  assumes *connected-root r x*
    and $q \leq x^\star;r$
  shows *connected-root q x*
by (*metis assms comp-assoc mult-isol phl-cons1 star-slide-var star-trans-eq*)

lemma *root-cycle-converse*:
  assumes *connected-root r x*
    and *point r*
    and $x;r \neq 0$
  shows $x^T;r \neq 0$
using *assms conv-zero has-root* **by** *fastforce*

Rooted paths

lemma *path-iff-aux-1*:
  assumes *bijective r*
    shows $r;x \leq x^\star + x^{T\star}$ $\longleftrightarrow$ $x \leq r^T;(x^\star + x^{T\star})$
by (*simp add*: *assms ss423conv*)

lemma *path-iff-aux-2*:
  assumes *bijective r*

shows $r;x \leq x^\star + x^{T\star} \longleftrightarrow x^T \leq (x^\star + x^{T\star});r$
proof −
have $((x^\star + x^{T\star});r)^T = r^T;(x^\star + x^{T\star})$
by (*metis conv-add conv-contrav conv-invol star-conv sup.commute*)
thus *?thesis*
by (*metis assms conv-invol conv-iso path-iff-aux-1*)
qed

lemma *path-iff-backward*:
assumes *is-inj x*
and *is-p-fun x*
and *point r*
and $r;x \leq x^\star + x^{T\star}$
shows *connected x*
proof −
have $x^T;1;x^T \leq (x^\star + x^{T\star});r;1;x^T$
using *assms*(*3*,*4*) *path-iff-aux-2 mult-isor point-def* **by** *blast*
also have ... $= (x^\star + x^{T\star});r;1;x^T;x;x^T$
using *assms*(*1*) *comp-assoc inj-p-fun p-fun-triple* **by** *fastforce*
also have ... $\leq (x^\star + x^{T\star});r;x;x^T$
by (*metis assms*(*3*) *mult-double-iso top-greatest point-def is-vector-def comp-assoc*)
also have ... $\leq (x^\star + x^{T\star});(x^\star + x^{T\star});x^T$
by (*metis assms*(*4*) *comp-assoc mult-double-iso*)
also have ... $\leq (x^\star + x^{T\star});(x^\star + x^{T\star});(x^\star + x^{T\star})$
using *le-supI2 mult-isol star-ext* **by** *blast*
also have ... $= x^\star + x^{T\star}$
using *assms*(*1*,*2*) *cancel-separate-converse-idempotent* **by** *fastforce*
finally show *?thesis*
by (*metis conv-add conv-contrav conv-invol conv-one mult-assoc star-conv sup.orderE sup.orderI*
sup-commute)
qed

lemma *empty-path-root-end*:
assumes *terminating-path-root-end r x e*
shows $e = r \longleftrightarrow x = 0$
apply(*standard*)
using *assms has-root backward-finite-path-root-def* **apply** *blast*
by (*metis assms antisym conv-e conv-zero independence1 is-inj-def mult-oner point-swap*
backward-finite-path-root-def ss423conv sur-def-var1 x-leq-triple-x)

lemma *path-root-acyclic*:
assumes *path-root r x*
and $x;r = 0$
shows *is-acyclic x*
proof −
have $x^+ \cdot 1' = (x^+)^T \cdot x^+ \cdot 1'$

by (*metis conv-e conv-times inf.assoc inf.left-idem inf-le2 many-strongly-connected-iff-7 mult-oner star-subid*)
also have ... $\leq x^T;1 \cdot x^+ \cdot 1'$
by (*metis conv-contrav inf.commute maddux-20 meet-double-iso plus-top star-conv star-slide-var*)
finally have $r;(x^+ \cdot 1') \leq r;(x^T;1 \cdot x^+ \cdot 1')$
using *mult-isol* **by** *blast*
also have ... $= (r \cdot 1;x);(x^+ \cdot 1')$
by (*metis* (*no-types, lifting*) *comp-assoc conv-contrav conv-invol conv-one inf.assoc is-vector-def one-idem-mult vector-2*)
also have ... $= r;x;(x^+ \cdot 1')$
by (*metis assms*(*1*) *path-root-def point-def inf-top-right vector-1*)
also have ... $\leq (x^\star + x^{T\star});(x^+ \cdot 1')$
using *assms*(*1*) *mult-isor path-root-def* **by** *blast*
also have ... $= x^\star;(x^+ \cdot 1') + x^{T+};(x^+ \cdot 1')$
by (*metis distrib-right star-star-plus sup.commute*)
also have ... $\leq x^\star;(x^+ \cdot 1') + x^T;1$
by (*metis join-isol mult-isol plus-top top-greatest*)
finally have $r;(x^+ \cdot 1');1 \leq x^\star;(x^+ \cdot 1');1 + x^T;1$
by (*metis distrib-right inf-absorb2 mult-assoc mult-subdistr one-idem-mult*)
hence *1*: $r;(x^+ \cdot 1');1 \leq x^T;1$
by (*metis assms*(*1*) *inj-implies-step-forwards-backwards sup-absorb2 path-root-def*)
have $x^+ \cdot 1' \leq (x^+ \cdot 1');1$
by (*simp add*: *maddux-20*)
also have ... $\leq r^T;r;(x^+ \cdot 1');1$
by (*metis assms*(*1*) *comp-assoc order.refl point-def ss423conv path-root-def*)
also have ... $\leq r^T;x^T;1$
using *1* **by** (*simp add*: *comp-assoc mult-isol*)
also have ... $= 0$
using *assms*(*2*) *annil conv-contrav conv-zero* **by** *force*
finally show *?thesis*
using *galois-aux le-bot* **by** *blast*
qed

Start points and end points

lemma *start-points-in-root-aux*:
assumes *backward-finite-path-root r x*
shows $x;1 \leq x^{T\star};r$
proof −
have $x;1 \leq x;x^{T+};r$
by (*metis assms inf-top.left-neutral modular-var-2 mult-assoc connected-root-iff3 backward-finite-path-root-def*)
also have ... $\leq 1';x^{T\star};r$
by (*metis assms is-inj-def mult-assoc mult-isor backward-finite-path-root-def*)
finally show *?thesis*
by *simp*
qed

lemma *start-points-in-root*:
  assumes *backward-finite-path-root r x*
  shows *start-points x* $\leq$ *r*
using *assms galois-1 sup-commute connected-root-iff3 backward-finite-path-root-def*
    start-points-in-root-aux **by** *fastforce*

lemma *start-points-not-zero-contra*:
  assumes *connected-root r x*
    and *point r*
    and *start-points x = 0*
    and $x;r = 0$
  shows $x = 0$
proof –
  have $x;1 \leq x^T;1$
    using *assms*(3) *galois-aux* **by** *force*
  also have ... $\leq -r$
    using *assms*(4) *comp-res compl-bot-eq* **by** *blast*
  finally show *?thesis*
    using *assms*(1,2) *has-root-contra galois-aux schroeder-1* **by** *force*
qed

lemma *start-points-not-zero*:
  assumes *connected-root r x*
    and *point r*
    and $x \neq 0$
    and $x;r = 0$
  shows *start-points x* $\neq 0$
using *assms start-points-not-zero-contra* **by** *blast*

Backwards terminating and backwards finite

lemma *backward-terminating-path-root-aux*:
  assumes *backward-terminating-path-root r x*
  shows $x \leq x^{T\star};-(x^T;1)$
proof –
  have $x^{T\star};r \leq x^{T\star};-(x^T;1)$
    using *assms comp-res compl-bot-eq compl-le-swap1 mult-isol* **by** *blast*
  thus *?thesis*
    using *assms dual-order.trans maddux-20 start-points-in-root-aux* **by** *blast*
qed

lemma *backward-finite-path-connected-aux*:
  assumes *backward-finite-path-root r x*
  shows $x^T;r;x^T \leq x^\star + x^{T\star}$
proof –
  have $x^T;r;x^T \cdot r^T = x^T;r;(x^T \cdot r^T)$
    by (*metis conv-invol conv-times vector-1-comm comp-assoc conv-contrav assms*
      backward-finite-path-root-def point-def)

also have ... $\leq x^T;r;r^T$
by (*simp add*: *mult-isol*)
also have *1*: ... $\leq x^T$
by (*metis assms comp-assoc is-inj-def mult-1-right mult-isol point-def backward-finite-path-root-def*)
also have ... $\leq x^{T\star}$
by *simp*
finally have *2*: $x^T;r;x^T \cdot r^T \leq x^{T\star}$.
let $?v = x;1 \cdot -r$
have $?v \leq x^{T+};r$
by (*simp add*: *assms galois-1 start-points-in-root-aux*)
hence $r^T;x \cdot ?v \leq r^T;x \cdot x^{T+};r$
using *meet-isor* **by** *blast*
also have *3*: ... $= x^{T+};r \cdot 1;r^T;x$
by (*metis assms conv-contrav conv-one inf-commute is-vector-def point-def backward-finite-path-root-def*)
also have ... $= (x^{T+};r \cdot 1);r^T;x$
using *3* **by** (*metis comp-assoc inf-commute is-vector-def star-conv vector-1 assms*
backward-finite-path-root-def point-def)
also have ... $= x^{T+};r;r^T;x$
by *simp*
also have ... $\leq x^{T+};x$
using *1* **by** (*metis mult-assoc mult-isol mult-isor star-slide-var*)
also have ... $= x^{T\star};x^T;x$
by (*simp add*: *star-slide-var*)
also have ... $\leq x^{T\star}$
by (*metis assms backward-finite-path-root-def is-p-fun-def mult-1-right mult-assoc mult-isol-var*
star-1l star-inductl-star)
finally have *4*: $x^T;r \cdot ?v^T \leq x^\star$
using *conv-iso star-conv* **by** *force*
have $x^T;r;x^T \cdot -r^T = (x^T;r \cdot 1);x^T \cdot -r^T$
by *simp*
also have ... $= x^T;r \cdot 1;x^T \cdot -r^T$
by (*metis inf.commute is-vector-def comp-assoc vector-1 assms backward-finite-path-root-def*
point-def)
also have ... $\leq x^\star$
using *4* **by** (*simp add*: *conv-compl inf.assoc*)
finally have $(x^T;r;x^T \cdot -r^T) + (x^T;r;x^T \cdot r^T) \leq x^\star + x^{T\star}$
using *2 sup.mono* **by** *blast*
thus *?thesis*
by *fastforce*
qed

lemma *backward-finite-path-connected*:
assumes *backward-finite-path-root r x*
shows *connected x*

proof −
from *assms* **obtain** *r* **where** *1*: *backward-finite-path-root r x* **..**
have $x^T;(x^\star + x^{T\star}) = x^T;(1' + x^+) + x^{T+}$
by (*simp add*: *distrib-left*)
also have ... $= x^T;x^+ + x^{T+}$
using *calculation distrib-left star-star-plus* **by** *fastforce*
also have ... $\leq 1';x^\star + x^{T+}$
using *1* **by** (*metis add-iso comp-assoc is-p-fun-def mult-isor backward-finite-path-root-def*)
also have ... $\leq x^\star + x^{T\star}$
using *join-isol* **by** *fastforce*
finally have $x^T;r;x^T + x^T;(x^\star + x^{T\star}) \leq x^\star + x^{T\star}$
using *1 backward-finite-path-connected-aux* **by** *simp*
hence $x^{T\star};x^T;r;x^T \leq x^\star + x^{T\star}$
using *star-inductl comp-assoc* **by** *simp*
hence $x^T;1;x^T \leq x^\star + x^{T\star}$
using *1 backward-finite-path-root-def connected-root-iff3 star-slide-var* **by** *fastforce*
thus *?thesis*
by (*metis* (*mono-tags*, *lifting*) *sup.commute comp-assoc conv-add conv-contrav conv-invol conv-iso*
conv-one star-conv)
qed

lemma *backward-finite-path-root-path*:
assumes *backward-finite-path-root r x*
shows *path x*
using *assms path-def backward-finite-path-connected backward-finite-path-root-def* **by** *blast*

lemma *backward-finite-path-root-path-root*:
assumes *backward-finite-path-root r x*
shows *path-root r x*
using *assms backward-finite-path-root-def le-supI1 star-star-plus path-root-def* **by** *fastforce*

lemma *zero-backward-terminating-path-root*:
assumes *point r*
shows *backward-terminating-path-root r 0*
by (*simp add*: *assms is-inj-def is-p-fun-def backward-finite-path-root-def*)

lemma *backward-finite-path-root-move-root*:
assumes *backward-finite-path-root r x*
and *point q*
and $q \leq x^\star;r$
shows *backward-finite-path-root q x*
using *assms connected-root-move-root backward-finite-path-root-def* **by** *blast*

Cycle

lemma *non-empty-cycle-root-var-axioms-1*:
non-empty-cycle-root r x $\longleftrightarrow$ $x^T;1 \leq x^{T+};r$ $\wedge$ *is-inj x* $\wedge$ *is-p-fun x* $\wedge$ *point r* $\wedge$ $r \leq x^T;1$
using *connected-root-iff2 backward-finite-path-root-def* **by** *blast*

lemma *non-empty-cycle-root-loop*:
assumes *non-empty-cycle-root r x*
shows $r \leq x^{T+};r$
using *assms connected-root-iff3 backward-finite-path-root-def* **by** *fastforce*

lemma *cycle-root-end-empty*:
assumes *terminating-path-root-end r x e*
and *many-strongly-connected x*
shows $x = 0$
by (*metis assms has-root-contra point-swap backward-finite-path-root-def backward-finite-path-root-move-root star-conv*)

lemma *cycle-root-end-empty-var*:
assumes *terminating-path-root-end r x e*
and $x \neq 0$
shows ¬ *many-strongly-connected x*
using *assms cycle-root-end-empty* **by** *blast*

Terminating path

lemma *terminating-path-root-end-connected*:
assumes *terminating-path-root-end r x e*
shows $x;1 \leq x^+;e$
proof –
have $x;1 \leq x;x^T;1$
by (*metis comp-assoc inf-top.left-neutral modular-var-2*)
also have ... $= x;x^{T+};r$
using *assms backward-finite-path-root-def connected-root-iff3 comp-assoc* **by** *fastforce*
also have ... $\leq x;x^{T+};x^\star;e$
by (*simp add*: *assms comp-assoc mult-isol*)
also have ... $= x;x^T;(x^\star + x^{T\star});e$
using *assms cancel-separate-p-fun-converse comp-assoc backward-finite-path-root-def* **by** *fastforce*
also have ... $= x;x^T;(x^+ + x^{T\star});e$
by (*simp add*: *star-star-plus*)
also have ... $= x;x^T;x^+;e + x;x^{T+};e$
by (*simp add*: *comp-assoc distrib-left*)
also have ... $= x;x^T;x^+;e$
by (*simp add*: *assms comp-assoc independence1*)
also have ... $\leq x^+;e$
by (*metis assms annil independence1 is-inj-def mult-isor mult-oner backward-finite-path-root-def*)
finally show *?thesis* **.**
qed

lemma *terminating-path-root-end-forward-finite*:
assumes *terminating-path-root-end r x e*
shows *backward-finite-path-root e* (x^T)
using *assms terminating-path-root-end-connected inj-p-fun connected-root-iff2 backward-finite-path-root-def* **by** *force*

end

3.2 Consequences with the Tarski rule

context *relation-algebra-rtc-tarski*
begin

Some (more) results about points

lemma *point-reachable-converse*:
assumes *is-vector v*
and $v \neq 0$
and *point r*
and $v \leq x^{T+};r$
shows $r \leq x^{+};v$
proof –
have $v^T;v \neq 0$
by (*metis assms*(*2*) *inf.idem inf-bot-right mult-1-right schroeder-1*)
hence $1;v^T;v = 1$
using *assms*(*1*) *is-vector-def mult-assoc tarski* **by** *force*
hence *1*: $r = r;v^T;v$
by (*metis assms*(*3*) *is-vector-def mult-assoc point-def*)
have $v;r^T \leq x^{T+}$
using *assms*(*3*,*4*) *point-def ss423bij* **by** *simp*
hence $r;v^T \leq x^{+}$
by (*metis conv-contrav conv-invol conv-iso star-conv star-slide-var*)
thus *?thesis*
using *1* **by** (*metis mult-isor*)
qed

Roots

lemma *root-in-start-points*:
assumes *connected-root r x*
and *is-vector r*
and $x \neq 0$
and $x;r = 0$
shows $r \leq$ *start-points x*
proof –
have $r = r;x;1$
by (*metis assms*(*2*,*3*) *comp-assoc is-vector-def tarski*)
also have ... $\leq x;1$
by (*metis assms*(*1*) *comp-assoc one-idem-mult phl-seq top-greatest*)
finally show *?thesis*
using *assms*(*4*) *comp-res compl-bot-eq compl-le-swap1 inf.boundedI* **by** *blast*

qed

lemma *root-equals-start-points*:
assumes *backward-terminating-path-root r x*
and $x \neq 0$
shows $r = $ *start-points x*
using *assms antisym point-def backward-finite-path-root-def start-points-in-root root-in-start-points*
by *fastforce*

lemma *root-equals-end-points*:
assumes *backward-terminating-path-root r* (x^T)
and $x \neq 0$
shows $r = $ *end-points x*
by (*metis assms conv-invol step-has-target ss-p18 root-equals-start-points*)

lemma *root-in-edge-sources*:
assumes *connected-root r x*
and $x \neq 0$
and *is-vector r*
shows $r \leq x;1$
proof −
have $r;1;x;1 \leq x^+;1$
using *assms*(*1*,*3*) *is-vector-def mult-isor* **by** *fastforce*
thus *?thesis*
by (*metis assms*(*2*) *comp-assoc conway.dagger-unfoldl-distr dual-order.trans maddux-20 sup.commute*
sup-absorb2 tarski top-greatest)
qed

Rooted Paths

lemma *non-empty-path-root-iff-aux*:
assumes *path-root r x*
and $x \neq 0$
shows $r \leq (x + x^T);1$
proof −
have $(r;x \cdot 1');1 = (x^T;r^T \cdot 1');1$
by (*metis conv-contrav conv-e conv-times inf.cobounded2 is-test-def test-eq-conv*)
also have ... $\leq x^T;r^T;1$
using *mult-subdistr* **by** *blast*
also have ... $\leq x^T;1$
by (*metis mult-assoc mult-double-iso one-idem-mult top-greatest*)
finally have *1*: $(r;x \cdot 1');1 \leq x^T;1$ **.**
have $r \leq r;1;x;1$
using *assms*(*2*) *comp-assoc maddux-20 tarski* **by** *fastforce*
also have ... $= r;x;1$
using *assms*(*1*) *path-root-def point-def is-vector-def* **by** *simp*
also have ... $= (r;x \cdot (x^\star + x^{T\star}));1$

using *assms*(*1*) *path-root-def* **by** (*simp add*: *inf.absorb-iff1*)
also have ... = $(r;x \cdot (x^+ + x^{T+} + 1'));1$
by (*metis star-star-plus star-unfoldl-eq sup-commute sup-left-commute*)
also have ... ≤ $(x^+ + x^{T+} + (r;x \cdot 1'));1$
by (*metis inf-le2 inf-sup-distrib1 mult-isor order-refl sup-mono*)
also have ... ≤ $x;1 + x^T;1 + (r;x \cdot 1');1$
by (*simp add*: *plus-top*)
also have ... = $x;1 + x^T;1$
using *1 sup.coboundedI2 sup.order-iff* **by** *fastforce*
finally show *?thesis*
by *simp*
qed

Backwards terminating and backwards finite

lemma *backward-terminating-path-root-2*:
assumes *backward-terminating-path-root r x*
shows *backward-terminating x*
using *assms backward-terminating-iff2 path-def backward-terminating-path-root-aux*
backward-finite-path-connected backward-finite-path-root-def **by** *blast*

lemma *backward-terminating-path-root*:
assumes *backward-terminating-path-root r x*
shows *backward-terminating-path x*
using *assms backward-finite-path-root-path backward-terminating-path-root-2* **by** *fastforce*

(Non-empty) Cycle

lemma *cycle-iff*:
assumes *point r*
shows $x;r \neq 0 \longleftrightarrow r \leq x^T;1$
by (*simp add*: *assms no-end-point-char-converse*)

lemma *non-empty-cycle-root-iff*:
assumes *connected-root r x*
and *point r*
shows $x;r \neq 0 \longleftrightarrow r \leq x^{T+};r$
using *assms connected-root-iff3 cycle-iff* **by** *simp*

lemma *backward-finite-path-root-terminating-or-cycle*:
backward-finite-path-root r x ⟷ *backward-terminating-path-root r x* ∨ *non-empty-cycle-root r x*
using *cycle-iff backward-finite-path-root-def* **by** *blast*

lemma *non-empty-cycle-root-msc*:
assumes *non-empty-cycle-root r x*
shows *many-strongly-connected x*
proof −
let *?p* = $x^T;r$

have *1*: *is-point ?p*
  unfolding *is-point-def*
  using *conjI assms is-vector-def mult-assoc point-def inj-compose p-fun-inj cycle-iff backward-finite-path-root-def root-cycle-converse* **by** *fastforce*
have $?p \leq x^{T+};?p$
  by (*metis assms comp-assoc mult-isol star-slide-var non-empty-cycle-root-loop*)
hence $?p \leq x^{+};?p$
  using *1 bot-least point-def point-is-point point-reachable-converse* **by** *blast*
also have ... $= x^{\star};(x;x^{T});r$
  by (*metis comp-assoc star-slide-var*)
also have ... $\leq x^{\star};1';r$
  using *assms is-inj-def mult-double-iso backward-finite-path-root-def* **by** *blast*
finally have *2*: $?p \leq x^{\star};r$
  by *simp*
have $x^{T};x^{\star};r = ?p + x^{T};x^{+};r$
  by (*metis conway.dagger-unfoldl-distr distrib-left mult-assoc*)
also have ... $\leq ?p + 1';x^{\star};r$
  by (*metis assms is-p-fun-def join-isol mult-assoc mult-isor backward-finite-path-root-def*)
also have ... $= x^{\star};r$
  using *2* **by** (*simp add*: *sup-absorb2*)
finally have *3*: $x^{T\star};r \leq x^{\star};r$
  by (*metis star-inductl comp-assoc conway.dagger-unfoldl-distr le-supI order-prop*)
have $x^{T} \leq x^{T+};r$
  by (*metis assms maddux-20 connected-root-iff3 backward-finite-path-root-def*)
also have ... $\leq x^{\star};r$
  using *3* **by** (*metis assms conway.dagger-unfoldl-distr sup-absorb2 non-empty-cycle-root-loop*)
finally have *4*: $x^{T} \leq x^{\star};r$ **.**
have $x^{T} \leq x^{T};x;x^{T}$
  by (*metis conv-invol x-leq-triple-x*)
also have ... $\leq 1;x;x^{T}$
  by (*simp add*: *mult-isor*)
also have ... $= r^{T};x^{+};x^{T}$
  using *assms connected-root-iff4 backward-finite-path-root-def* **by** *fastforce*
also have ... $\leq r^{T};x^{\star}$
  by (*metis assms is-inj-def mult-1-right mult-assoc mult-isol backward-finite-path-root-def star-slide-var*)
finally have $x^{T} \leq x^{\star};r \cdot r^{T};x^{\star}$
  using *4* **by** *simp*
also have ... $= x^{\star};r \cdot 1;r^{T};x^{\star}$
  by (*metis assms conv-contrav conv-one is-vector-def point-def backward-finite-path-root-def*)
also have ... $= (x^{\star};r \cdot 1);r^{T};x^{\star}$
  by (*metis* (*no-types*, *lifting*) *assms is-vector-def mult-assoc point-def backward-finite-path-root-def vector-1*)
also have ... $= x^{\star};r;r^{T};x^{\star}$

by *simp*
also have ... $\leq x^\star;x^\star$
by (*metis assms is-inj-def mult-1-right mult-assoc mult-isol mult-isor point-def backward-finite-path-root-def*)
also have ... $\leq x^\star$
by *simp*
finally show *?thesis*
by (*simp add*: *many-strongly-connected-iff-1*)
qed

lemma *non-empty-cycle-root-msc-cycle*:
assumes *non-empty-cycle-root r x*
shows *cycle x*
using *assms backward-finite-path-root-path non-empty-cycle-root-msc* by *fastforce*

lemma *non-empty-cycle-root-non-empty*:
assumes *non-empty-cycle-root r x*
shows $x \neq 0$
using *assms cycle-iff annil backward-finite-path-root-def* by *blast*

lemma *non-empty-cycle-root-rtc-symmetric*:
assumes *non-empty-cycle-root r x*
shows $x^\star;r = x^{T\star};r$
using *assms non-empty-cycle-root-msc* by *fastforce*

lemma *non-empty-cycle-root-point-exchange*:
assumes *non-empty-cycle-root r x*
and *point p*
shows $r \leq x^\star;p \longleftrightarrow p \leq x^\star;r$
by (*metis assms*(*1*,*2*) *inj-sur-semi-swap point-def non-empty-cycle-root-msc backward-finite-path-root-def star-conv*)

lemma *non-empty-cycle-root-rtc-tc*:
assumes *non-empty-cycle-root r x*
shows $x^\star;r = x^+;r$
proof (*rule antisym*)
have $r \leq x^+;r$
using *assms many-strongly-connected-iff-7 non-empty-cycle-root-loop non-empty-cycle-root-msc*
by *simp*
thus $x^\star;r \leq x^+;r$
using *sup-absorb2* by *fastforce*
next
show $x^+;r \leq x^\star;r$
by (*simp add*: *mult-isor*)
qed

lemma *non-empty-cycle-root-no-start-end-points*:
assumes *non-empty-cycle-root r x*

shows $x;1 = x^T;1$
using *assms many-strongly-connected-implies-no-start-end-points non-empty-cycle-root-msc* **by** *blast*

lemma *non-empty-cycle-root-move-root*:
assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows *non-empty-cycle-root q x*
by (*metis assms cycle-iff dual-order.trans backward-finite-path-root-move-root start-points-in-root*
root-equals-start-points non-empty-cycle-root-non-empty)

lemma *non-empty-cycle-root-loop-converse*:
assumes *non-empty-cycle-root r x*
shows $r \leq x^+;r$
using *assms less-eq-def non-empty-cycle-root-rtc-tc* **by** *fastforce*

lemma *non-empty-cycle-root-move-root-same-reachable*:
assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows $x^\star;r = x^\star;q$
by (*metis assms many-strongly-connected-iff-7 connected-root-iff3 connected-root-move-root*
backward-finite-path-root-def non-empty-cycle-root-msc non-empty-cycle-root-rtc-tc)

lemma *non-empty-cycle-root-move-root-same-reachable-2*:
assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows $x^\star;r = x^{T\star};q$
using *assms non-empty-cycle-root-move-root-same-reachable non-empty-cycle-root-msc* **by** *simp*

lemma *non-empty-cycle-root-move-root-msc*:
assumes *non-empty-cycle-root r x*
shows $x^{T\star};q = x^\star;q$
using *assms non-empty-cycle-root-msc* **by** *simp*

lemma *non-empty-cycle-root-move-root-rtc-tc*:
assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows $x^\star;q = x^+;q$
using *assms non-empty-cycle-root-move-root non-empty-cycle-root-rtc-tc* **by** *blast*

lemma *non-empty-cycle-root-move-root-loop-converse*:

assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows $q \leq x^{T+};q$
using *assms non-empty-cycle-root-loop non-empty-cycle-root-move-root* **by** *blast*

lemma *non-empty-cycle-root-move-root-loop*:
assumes *non-empty-cycle-root r x*
and *point q*
and $q \leq x^\star;r$
shows $q \leq x^+;q$
using *assms non-empty-cycle-root-loop-converse non-empty-cycle-root-move-root*
by *blast*

lemma *non-empty-cycle-root-msc-plus*:
assumes *non-empty-cycle-root r x*
shows $x^+;r = x^{T+};r$
using *assms many-strongly-connected-iff-7 non-empty-cycle-root-msc* **by** *fastforce*

lemma *non-empty-cycle-root-tc-start-points*:
assumes *non-empty-cycle-root r x*
shows $x^+;r = x;1$
by (*metis assms connected-root-iff3 backward-finite-path-root-def non-empty-cycle-root-msc-plus*
non-empty-cycle-root-no-start-end-points)

lemma *non-empty-cycle-root-rtc-start-points*:
assumes *non-empty-cycle-root r x*
shows $x^\star;r = x;1$
by (*simp add*: *assms non-empty-cycle-root-rtc-tc non-empty-cycle-root-tc-start-points*)

lemma *non-empty-cycle-root-converse-start-end-points*:
assumes *non-empty-cycle-root r x*
shows $x^T \leq x;1;x$
by (*metis assms conv-contrav conv-invol conv-one inf.boundedI maddux-20 maddux-21 vector-meet-comp-x*
non-empty-cycle-root-no-start-end-points)

lemma *non-empty-cycle-root-start-end-points-plus*:
assumes *non-empty-cycle-root r x*
shows $x;1;x \leq x^+$
using *assms eq-iff one-strongly-connected-iff one-strongly-connected-implies-7-eq*
backward-finite-path-connected non-empty-cycle-root-msc **by** *blast*

lemma *non-empty-cycle-root-converse-plus*:
assumes *non-empty-cycle-root r x*
shows $x^T \leq x^+$
using *assms many-strongly-connected-iff-2 non-empty-cycle-root-msc* **by** *blast*

lemma *non-empty-cycle-root-plus-converse*:
assumes *non-empty-cycle-root r x*
shows $x^+ = x^{T+}$
using *assms many-strongly-connected-iff-7 non-empty-cycle-root-msc* **by** *fastforce*

lemma *non-empty-cycle-root-converse*:
assumes *non-empty-cycle-root r x*
shows *non-empty-cycle-root r* (x^T)
by (*metis assms conv-invol inj-p-fun connected-root-iff3 backward-finite-path-root-def non-empty-cycle-root-msc-plus non-empty-cycle-root-tc-start-points*)

lemma *non-empty-cycle-root-move-root-forward*:
assumes *non-empty-cycle-root r x*
and *point q*
and $r \leq x^\star;q$
shows *non-empty-cycle-root q x*
by (*metis assms backward-finite-path-root-move-root non-empty-cycle-root-no-start-end-points non-empty-cycle-root-point-exchange non-empty-cycle-root-rtc-start-points*)

lemma *non-empty-cycle-root-move-root-forward-cycle*:
assumes *non-empty-cycle-root r x*
and *point q*
and $r \leq x^\star;q$
shows $x;q \neq 0 \wedge x^T;q \neq 0$
by (*metis assms comp-assoc independence1 ss-p18 non-empty-cycle-root-move-root-forward non-empty-cycle-root-msc-plus non-empty-cycle-root-non-empty non-empty-cycle-root-tc-start-points*)

lemma *non-empty-cycle-root-equivalences*:
assumes *non-empty-cycle-root r x*
and *point q*
shows $(r \leq x^\star;q \longleftrightarrow q \leq x^\star;r)$
and $(r \leq x^\star;q \longleftrightarrow x;q \neq 0)$
and $(r \leq x^\star;q \longleftrightarrow x^T;q \neq 0)$
and $(r \leq x^\star;q \longleftrightarrow q \leq x;1)$
and $(r \leq x^\star;q \longleftrightarrow q \leq x^T;1)$
using *assms cycle-iff no-end-point-char non-empty-cycle-root-no-start-end-points non-empty-cycle-root-point-exchange non-empty-cycle-root-rtc-start-points*
by *metis+*

lemma *non-empty-cycle-root-chord*:
assumes *non-empty-cycle-root r x*
and *point p*
and *point q*
and $r \leq x^\star;p$

and $r \leq x^\star;q$
shows $p \leq x^\star;q$
using *assms non-empty-cycle-root-move-root-same-reachable non-empty-cycle-root-point-exchange*
by *fastforce*

lemma *non-empty-cycle-root-var-axioms-2*:
non-empty-cycle-root $r\ x \longleftrightarrow x;1 \leq x^+;r \wedge$ *is-inj* $x \wedge$ *is-p-fun* $x \wedge$ *point* $r \wedge r \leq x;1$
apply (*rule iffI*)
apply (*metis eq-iff backward-finite-path-root-def non-empty-cycle-root-no-start-end-points non-empty-cycle-root-tc-start-points*)
by (*metis conv-invol p-fun-inj connected-root-iff2 connected-root-iff3 non-empty-cycle-root-var-axioms-1 non-empty-cycle-root-msc-plus non-empty-cycle-root-rtc-start-points non-empty-cycle-root-rtc-tc*)

lemma *non-empty-cycle-root-var-axioms-3*:
non-empty-cycle-root $r\ x \longleftrightarrow x;1 \leq x^+;r \wedge$ *is-inj* $x \wedge$ *is-p-fun* $x \wedge$ *point* $r \wedge r \leq x^+;x;1$
apply (*rule iffI*)
apply (*metis comp-assoc eq-refl backward-finite-path-root-def star-inductl-var-eq2 non-empty-cycle-root-no-start-end-points non-empty-cycle-root-rtc-start-points non-empty-cycle-root-tc-start-points*)
by (*metis annir comp-assoc conv-contrav no-end-point-char non-empty-cycle-root-var-axioms-2*)

lemma *non-empty-cycle-root-subset-equals*:
assumes *non-empty-cycle-root* $r\ x$
and *non-empty-cycle-root* $r\ y$
and $x \leq y$
shows $x = y$
proof −
have $y;x^{T\star};r = y;x^{T+};r$
using *assms*(*1*) *comp-assoc non-empty-cycle-root-msc non-empty-cycle-root-msc-plus non-empty-cycle-root-rtc-tc* **by** *fastforce*
also have ... $\leq y;y^T;x^{T\star};r$
using *assms*(*3*) *comp-assoc conv-iso mult-double-iso* **by** *fastforce*
also have ... $\leq x^{T\star};r$
using *assms*(*2*) *backward-finite-path-root-def is-inj-def*
by (*meson dual-order.trans mult-isor order.refl prod-star-closure star-ref*)
finally have $r + y;x^{T\star};r \leq x^{T\star};r$
by (*metis conway.dagger-unfoldl-distr le-supI sup.cobounded1*)
hence $y^\star;r \leq x^{T\star};r$
by (*simp add*: *comp-assoc rtc-inductl*)
hence $y;1 \leq x;1$
using *assms*(*1*,*2*) *non-empty-cycle-root-msc*

non-empty-cycle-root-rtc-start-points **by** *fastforce*
thus *?thesis*
using *assms*(*2*,*3*) *backward-finite-path-root-def ss422iv* **by** *blast*
qed

lemma *non-empty-cycle-root-subset-equals-change-root*:
assumes *non-empty-cycle-root r x*
and *non-empty-cycle-root q y*
and $x \leq y$
shows $x = y$
proof –
have $r \leq y;1$
by (*metis assms*(*1*,*3*) *dual-order.trans mult-isor*
non-empty-cycle-root-no-start-end-points)
hence *non-empty-cycle-root r y*
by (*metis assms*(*1*,*2*) *connected-root-move-root backward-finite-path-root-def*
non-empty-cycle-root-no-start-end-points
non-empty-cycle-root-rtc-start-points)
thus *?thesis*
using *assms*(*1*,*3*) *non-empty-cycle-root-subset-equals* **by** *blast*
qed

lemma *non-empty-cycle-root-equivalences-2*:
assumes *non-empty-cycle-root r x*
shows ($v \leq x^\star;r \longleftrightarrow v \leq x^T;1$)
and ($v \leq x^\star;r \longleftrightarrow v \leq x;1$)
using *assms non-empty-cycle-root-no-start-end-points*
non-empty-cycle-root-rtc-start-points
by *metis+*

lemma *cycle-root-non-empty*:
assumes $x \neq 0$
shows *cycle-root r x* $\longleftrightarrow$ *non-empty-cycle-root r x*
proof
assume *1*: *cycle-root r x*
have $r \leq r;1;x;1$
using *assms comp-assoc maddux-20 tarski* **by** *fastforce*
also have ... $\leq (x^+ \cdot x^T;1);1$
using *1* **by** (*simp add*: *is-vector-def mult-isor point-def*)
also have ... $\leq x^T;1$
by (*simp add*: *ra-1*)
finally show *non-empty-cycle-root r x*
using *1 backward-finite-path-root-def inf.boundedE* **by** *blast*
next
assume *non-empty-cycle-root r x*
thus *cycle-root r x*
by (*metis backward-finite-path-root-def inf.orderE maddux-20*
non-empty-cycle-root-plus-converse
ra-1)

qed

Start points and end points

lemma *start-points-path-aux*:
assumes *backward-finite-path-root r x*
and *start-points x* $\neq$ *0*
shows *x*;*r* = *0*
by (*metis assms compl-inf-bot inf.commute*
non-empty-cycle-root-no-start-end-points
backward-finite-path-root-terminating-or-cycle)

lemma *start-points-path*:
assumes *backward-finite-path-root r x*
and *start-points x* $\neq$ *0*
shows *backward-terminating-path-root r x*
by (*simp add*: *assms start-points-path-aux*)

lemma *root-in-start-points-2*:
assumes *backward-finite-path-root r x*
and *start-points x* $\neq$ *0*
shows *r* $\leq$ *start-points x*
by (*metis assms conv-zero eq-refl galois-aux2 root-equals-start-points*
start-points-path-aux)

lemma *root-equals-start-points-2*:
assumes *backward-finite-path-root r x*
and *start-points x* $\neq$ *0*
shows *r* = *start-points x*
by (*metis assms inf-bot-left ss-p18 root-equals-start-points start-points-path*)

lemma *start-points-injective*:
assumes *backward-finite-path-root r x*
shows *is-inj* (*start-points x*)
by (*metis assms compl-bot-eq inj-def-var1 point-def backward-finite-path-root-def*
top-greatest
root-equals-start-points-2)

lemma *backward-terminating-path-root-aux-2*:
assumes *backward-finite-path-root r x*
and *start-points x* $\neq$ *0* $\vee$ *x* = *0*
shows $x \leq x^{T\star};-(x^T;1)$
using *assms bot-least backward-terminating-path-root-aux start-points-path* **by**
blast

lemma *start-points-not-zero-iff*:
assumes *backward-finite-path-root r x*
shows *x*;*r* = *0* $\wedge$ *x* $\neq$ *0* $\longleftrightarrow$ *start-points x* $\neq$ *0*
by (*metis assms conv-zero inf-compl-bot backward-finite-path-root-def*
start-points-not-zero-contra

start-points-path-aux)

Backwards terminating and backwards finite: Part II

lemma *backward-finite-path-root-acyclic-terminating-aux*:
assumes *backward-finite-path-root r x*
and *is-acyclic x*
shows $x;r = 0$
proof (*cases* $x = 0$)
assume $x = 0$
thus *?thesis*
by *simp*
next
assume $x \neq 0$
hence *1*: $r \leq x;1$
using *assms*(*1*) *has-root-contra no-end-point-char backward-finite-path-root-def* **by** *blast*
have $r\cdot(x^T;1) = r\cdot(x^{T+};r)$
using *assms*(*1*) *connected-root-iff3 backward-finite-path-root-def* **by** *fastforce*
also have ... $\leq r\cdot(-1';r)$
by (*metis assms*(*2*) *conv-compl conv-contrav conv-e conv-iso meet-isor mult-isor star-conv star-slide-var*)
also have ... $= 0$
by (*metis* (*no-types*) *assms*(*1*) *inj-distr annil inf-compl-bot mult-1-left point-def backward-finite-path-root-def*)
finally have $r \leq$ *start-points x*
using *1 galois-aux inf.boundedI le-bot* **by** *blast*
thus *?thesis*
using *assms*(*1*) *annir le-bot start-points-path* **by** *blast*
qed

lemma *backward-finite-path-root-acyclic-terminating-iff*:
assumes *backward-finite-path-root r x*
shows *is-acyclic x* $\longleftrightarrow x;r = 0$
apply (*rule iffI*)
apply (*simp add*: *assms backward-finite-path-root-acyclic-terminating-aux*)
using *assms backward-finite-path-root-path-root path-root-acyclic* **by** *blast*

lemma *backward-finite-path-root-acyclic-terminating*:
assumes *backward-finite-path-root r x*
and *is-acyclic x*
shows *backward-terminating-path-root r x*
by (*simp add*: *assms backward-finite-path-root-acyclic-terminating-aux*)

lemma *non-empty-cycle-root-one-strongly-connected*:
assumes *non-empty-cycle-root r x*
shows *one-strongly-connected x*
by (*metis assms one-strongly-connected-iff order-trans star-1l star-star-plus*

sup.absorb2
non-empty-cycle-root-msc non-empty-cycle-root-start-end-points-plus)

lemma *backward-finite-path-root-nodes-reachable*:
assumes *backward-finite-path-root r x*
and $v \leq x;1 + x^T;1$
and *is-sur v*
shows $r \leq x^\star;v$
proof −
have $v \leq x;1 + x^{T+};r$
using *assms connected-root-iff3 backward-finite-path-root-def* **by** *fastforce*
also have ... $\leq x^{T\star};r + x^{T+};r$
using *assms*(*1*) *join-iso start-points-in-root-aux* **by** *blast*
also have ... $= x^{T\star};r$
using *mult-isor sup.absorb1* **by** *fastforce*
finally show *?thesis*
using *assms*(*1*,*3*)
by (*simp add*: *inj-sur-semi-swap point-def backward-finite-path-root-def*
star-conv
inj-sur-semi-swap-short)
qed

lemma *terminating-path-root-end-backward-terminating*:
assumes *terminating-path-root-end r x e*
shows *backward-terminating-path-root r x*
using *assms non-empty-cycle-root-move-root-forward-cycle*
backward-finite-path-root-terminating-or-cycle **by** *blast*

lemma *terminating-path-root-end-converse*:
assumes *terminating-path-root-end r x e*
shows *terminating-path-root-end e* (x^T) *r*
by (*metis assms terminating-path-root-end-backward-terminating*
backward-finite-path-root-def
conv-invol terminating-path-root-end-forward-finite point-swap star-conv)

lemma *terminating-path-root-end-forward-terminating*:
assumes *terminating-path-root-end r x e*
shows *backward-terminating-path-root e* (x^T)
using *assms terminating-path-root-end-converse* **by** *blast*

end

3.3 Consequences with the Tarski rule and the point axiom

context *relation-algebra-rtc-tarski-point*
begin

Rooted paths

lemma *path-root-iff*:
$(\exists r\ .\ path\text{-}root\ r\ x) \longleftrightarrow path\ x$

proof
  assume $\exists r$. *path-root r x*
  thus *path x*
    using *path-def path-iff-backward point-def path-root-def* **by** *blast*
next
  assume *1*: *path x*
  show $\exists r$. *path-root r x*
  proof (*cases* $x = 0$)
    assume $x = 0$
    thus *?thesis*
      by (*simp add*: *is-inj-def is-p-fun-def point-exists path-root-def*)
  next
    assume $\neg(x = 0)$
    hence $x;1 \neq 0$
      by (*simp add*: *ss-p18*)
    from *this* **obtain** *r* **where** *2*: *point r* $\wedge$ $r \leq x;1$
      using *comp-assoc is-vector-def one-idem-mult point-below-vector* **by** *fastforce*
    hence $r;x \leq x;1;x$
      by (*simp add*: *mult-isor*)
    also have ... $\leq x^\star + x^{T\star}$
      using *1 path-def* **by** *blast*
    finally show *?thesis*
      using *1 2 path-def path-root-def* **by** *blast*
  qed
qed

lemma *non-empty-path-root-iff*:
  ($\exists r$. *path-root r x* $\wedge$ $r \leq (x + x^T);1$) $\longleftrightarrow$ *path x* $\wedge$ $x \neq 0$
apply (*rule iffI*)
 using *non-empty-cycle-root-non-empty path-root-def*
zero-backward-terminating-path-root path-root-iff
 apply *fastforce*
using *path-root-iff non-empty-path-root-iff-aux* **by** *blast*

(Non-empty) Cycle

lemma *non-empty-cycle-root-iff*:
  ($\exists r$. *non-empty-cycle-root r x*) $\longleftrightarrow$ *cycle x* $\wedge$ $x \neq 0$
proof
  assume $\exists r$. *non-empty-cycle-root r x*
  thus *cycle x* $\wedge$ $x \neq 0$
    using *non-empty-cycle-root-msc-cycle non-empty-cycle-root-non-empty* **by**
fastforce
next
  assume *1*: *cycle x* $\wedge$ $x \neq 0$
  hence $x^T;1 \neq 0$
    using *many-strongly-connected-implies-no-start-end-points ss-p18* **by** *blast*
  from *this* **obtain** *r* **where** *2*: *point r* $\wedge$ $r \leq x^T;1$
    using *comp-assoc is-vector-def one-idem-mult point-below-vector* **by** *fastforce*
  have *3*: $x^T;1;x^T \leq x^\star$

    using *1 one-strongly-connected-iff path-def* **by** *blast*
  have $r;x \leq x^T;1;x$
    using *2* **by** (*simp add*: *is-vector-def mult-isor point-def*)
  also have ... $\leq x^T;1;x;x^T;x$
    using *comp-assoc mult-isol x-leq-triple-x* **by** *fastforce*
  also have ... $\leq x^T;1;x^T;x$
    by (*metis mult-assoc mult-double-iso top-greatest*)
  also have ... $\leq x^\star;x$
    using *3 mult-isor* **by** *blast*
  finally have *connected-root r x*
    by (*simp add*: *star-slide-var*)
  hence *non-empty-cycle-root r x*
    using *1 2 path-def backward-finite-path-root-def* **by** *fastforce*
  thus $\exists r$. *non-empty-cycle-root r x* **..**
qed

lemma *non-empty-cycle-subset-equals*:
  assumes *cycle x*
    and *cycle y*
    and $x \leq y$
    and $x \neq 0$
  shows $x = y$
by (*metis assms le-bot non-empty-cycle-root-subset-equals-change-root non-empty-cycle-root-iff*)

lemma *cycle-root-iff*:
  ($\exists r$. *cycle-root r x*) $\longleftrightarrow$ *cycle x*
proof (*cases* $x = 0$)
  assume $x = 0$
  thus *?thesis*
    using *path-def point-exists* **by** *fastforce*
next
  assume $x \neq 0$
  thus *?thesis*
    using *cycle-root-non-empty non-empty-cycle-root-iff* **by** *simp*
qed

Backwards terminating and backwards finite

lemma *backward-terminating-path-root-iff*:
  ($\exists r$. *backward-terminating-path-root r x*) $\longleftrightarrow$ *backward-terminating-path x*
proof
  assume $\exists r$. *backward-terminating-path-root r x*
  thus *backward-terminating-path x*
    using *backward-terminating-path-root* **by** *fastforce*
next
  assume *1*: *backward-terminating-path x*
  show $\exists r$. *backward-terminating-path-root r x*
  proof (*cases* $x = 0$)
    assume $x = 0$

thus *?thesis*
using *point-exists zero-backward-terminating-path-root* **by** *blast*
next
let *?r* = *start-points x*
assume $x \neq 0$
hence *2*: *is-point ?r*
using *1 start-point-iff2 backward-terminating-iff1* **by** *fastforce*
have *3*: $x;?r = 0$
by (*metis inf-top.right-neutral modular-1-aux'*)
have $x;1;x \leq x;1;x;x^T;x$
using *comp-assoc mult-isol x-leq-triple-x* **by** *fastforce*
also have ... $\leq (x^\star + x^{T\star});x^T;x$
using *1 mult-isor path-def* **by** *blast*
also have ... $= (1' + x^+ + x^{T+});x^T;x$
by (*metis star-star-plus star-unfoldl-eq sup.commute*)
also have ... $= x^T;x + x^+;x^T;x + x^{T+};x^T;x$
by (*metis distrib-right' mult-onel*)
also have ... $= x^T;(x + x^{T\star};x^T;x) + x^+;x^T;x$
using *comp-assoc distrib-left sup.commute sup.assoc* **by** *simp*
also have ... $\leq x^T;1 + x^+;x^T;x$
using *join-iso mult-isol* **by** *fastforce*
also have ... $\leq x^T;1 + x^+;1'$
using *1* **by** (*metis comp-assoc join-isol mult-isol path-def is-p-fun-def*)
finally have $-(x^T;1) \cdot x;1;x \leq x^+$
by (*simp add*: *galois-1 inf.commute*)
hence $?r;x \leq x^+$
by (*metis inf-commute one-compl ra-1*)
hence *backward-terminating-path-root ?r x*
using *1 2 3* **by** (*simp add*: *point-is-point backward-finite-path-root-def path-def*)
thus *?thesis* **..**
qed
qed

lemma *non-empty-backward-terminating-path-root-iff*:
backward-terminating-path-root (*start-points x*) $x \longleftrightarrow$ *backward-terminating-path* $x \wedge x \neq 0$
apply (*rule iffI*)
apply (*metis backward-finite-path-root-path backward-terminating-path-root-2 conv-zero*
inf.cobounded1 non-empty-cycle-root-non-empty)
using *backward-terminating-path-root-iff root-equals-start-points* **by** *blast*

lemma *non-empty-backward-terminating-path-root-iff'*:
backward-finite-path-root (*start-points x*) $x \longleftrightarrow$ *backward-terminating-path* $x \wedge x \neq 0$
using *start-point-no-predecessor non-empty-backward-terminating-path-root-iff* **by** *simp*

lemma *backward-finite-path-root-iff*:
  ($\exists$ *r* . *backward-finite-path-root r x*) $\longleftrightarrow$ *backward-finite-path x*
proof
  assume $\exists$ *r* . *backward-finite-path-root r x*
  thus *backward-finite-path x*
    by (*meson backward-finite-iff-msc non-empty-cycle-root-msc*
backward-finite-path-root-path
              backward-finite-path-root-terminating-or-cycle
backward-terminating-path-root)
next
  assume *backward-finite-path x*
  thus $\exists$ *r* . *backward-finite-path-root r x*
    by (*metis backward-finite-iff-msc point-exists non-empty-cycle-root-iff*
             zero-backward-terminating-path-root backward-terminating-path-root-iff)
qed

lemma *non-empty-backward-finite-path-root-iff*:
  ($\exists$ *r* . *backward-finite-path-root r x* $\wedge$ $r \leq x;1$) $\longleftrightarrow$ *backward-finite-path x* $\wedge$ *x*
$\neq 0$
apply (*rule iffI*)
 apply (*metis backward-finite-path-root-iff annir backward-finite-path-root-def*
le-bot
              no-end-point-char ss-p18)
using *backward-finite-path-root-iff backward-finite-path-root-def point-def*
root-in-edge-sources **by** *blast*

Terminating

lemma *terminating-path-root-end-aux*:
  assumes *terminating-path x*
    shows $\exists$ *r e* . *terminating-path-root-end r x e*
proof (*cases* $x = 0$)
  assume $x = 0$
  thus *?thesis*
    using *point-exists zero-backward-terminating-path-root* **by** *fastforce*
next
  assume *1*: $\neg(x = 0)$
  have *2*: *backward-terminating-path x*
    using *assms* **by** *simp*
  from *this* **obtain** *r* **where** *3*: *backward-terminating-path-root r x*
    using *backward-terminating-path-root-iff* **by** *blast*
  have *backward-terminating-path* (x^T)
    using *2* **by** (*metis assms backward-terminating-iff1*
conv-backward-terminating-path conv-invol
                    conv-zero inf-top.left-neutral)
  from *this* **obtain** *e* **where** *4*: *backward-terminating-path-root e* (x^T)
    using *backward-terminating-path-root-iff* **by** *blast*
  have $r \leq x;1$
    using *1 3 root-in-edge-sources backward-finite-path-root-def point-def* **by**
fastforce

also have ... = $x^+;e$
using *4 connected-root-iff3 backward-finite-path-root-def* **by** *fastforce*
also have ... $\leq$ $x^\star;e$
by (*simp add*: *mult-isor*)
finally show *?thesis*
using *3 4 backward-finite-path-root-def* **by** *blast*
qed

lemma *terminating-path-root-end-iff*:
($\exists$ *r e* . *terminating-path-root-end r x e*) $\longleftrightarrow$ *terminating-path x*
proof
assume *1*: $\exists$ *r e* . *terminating-path-root-end r x e*
show *terminating-path x*
proof (*cases x = 0*)
assume *x = 0*
thus *?thesis*
by (*simp add*: *is-inj-def is-p-fun-def path-def*)
next
assume $\neg$(*x = 0*)
hence *2*: $\neg$ *many-strongly-connected x*
using *1 cycle-root-end-empty* **by** *blast*
hence *3*: *backward-terminating-path x*
using *1 backward-terminating-path-root*
terminating-path-root-end-backward-terminating **by** *blast*
have $\exists$ *e* . *backward-finite-path-root e* (x^T)
using *1 terminating-path-root-end-converse* **by** *blast*
hence *backward-terminating-path* (x^T)
using *1 backward-terminating-path-root terminating-path-root-end-converse*
by *blast*
hence *forward-terminating-path x*
by (*simp add*: *conv-backward-terminating-path*)
thus *?thesis*
using *3* **by** (*simp add*: *inf.boundedI*)
qed
next
assume *terminating-path x*
thus $\exists$ *r e* . *terminating-path-root-end r x e*
using *terminating-path-root-end-aux* **by** *blast*
qed

lemma *non-empty-terminating-path-root-end-iff*:
terminating-path-root-end (*start-points x*) *x* (*end-points x*) $\longleftrightarrow$
terminating-path x $\wedge$ *x* $\neq$ *0*
apply (*rule iffI*)
apply (*metis conv-zero non-empty-backward-terminating-path-root-iff*
terminating-path-root-end-iff)
using *terminating-path-root-end-iff terminating-path-root-end-forward-terminating*
root-equals-end-points terminating-path-root-end-backward-terminating
root-equals-start-points

by *blast*

lemma *non-empty-finite-path-root-end-iff*:
finite-path-root-end (*start-points* x) x (*end-points* x) $\longleftrightarrow$ *terminating-path* $x \wedge x \neq 0$
using *non-empty-terminating-path-root-end-iff end-point-no-successor* **by** *simp*

end

end

4 Correctness of Path Algorithms

To show that our theory of paths integrates with verification tasks, we verify the correctness of three basic path algorithms. Algorithms at the presented level are executable and can serve prototyping purposes. Data refinement can be carried out to move from such algorithms to more efficient programs. The total-correctness proofs use a library developed in [7].

theory *Path-Algorithms*

imports *Aggregation-Algebras.Hoare-Logic Rooted-Paths*

begin

no-notation
trancl ((-$^+$) [*1000*] *999*)

class *choose-singleton-point-signature* =
fixes *choose-singleton* :: $'a \Rightarrow 'a$
fixes *choose-point* :: $'a \Rightarrow 'a$

class *relation-algebra-rtc-tarski-choose-point* =
relation-algebra-rtc-tarski + *choose-singleton-point-signature* +
assumes *choose-singleton-singleton*: $x \neq 0 \Longrightarrow$ *singleton* (*choose-singleton* x)
assumes *choose-singleton-decreasing*: *choose-singleton* $x \leq x$
assumes *choose-point-point*: *is-vector* $x \Longrightarrow x \neq 0 \Longrightarrow$ *point* (*choose-point* x)
assumes *choose-point-decreasing*: *choose-point* $x \leq x$
begin

no-notation
composition (**infixl** ; *75*) **and**
times (**infixl** $*$ *70*)

notation
composition (**infixl** $*$ *75*)

4.1 Construction of a path

Our first example is a basic greedy algorithm that constructs a path from a vertex x to a different vertex y of a directed acyclic graph D.

abbreviation *construct-path-inv q x y D W* ≡
  is-acyclic D ∧ *point x* ∧ *point y* ∧ *point q* ∧
  $D^\star * q \leq D^{T\star} * x$ ∧ $W \leq D$ ∧ *terminating-path W* ∧
  ($W = 0 \longleftrightarrow q=y$) ∧ ($W \neq 0 \longleftrightarrow q =$ *start-points W* ∧ $y =$ *end-points W*)

abbreviation *construct-path-inv-simp q x y D W* ≡
  is-acyclic D ∧ *point x* ∧ *point y* ∧ *point q* ∧
  $D^\star * q \leq D^{T\star} * x$ ∧ $W \leq D$ ∧ *terminating-path W* ∧
  $q =$ *start-points W* ∧ $y =$ *end-points W*

lemma *construct-path-pre*:
 assumes *is-acyclic D*
   and *point y*
   and *point x*
   and $D^\star * y \leq D^{T\star} * x$
  shows *construct-path-inv y x y D 0*
 apply (*intro conjI*, *simp-all add*: *assms is-inj-def is-p-fun-def path-def*)
 using *assms*(*2*) *cycle-iff* **by** *fastforce*

The following three lemmas are auxiliary lemmas for *construct-path-inv*. They are pulled out of the main proof to have more structure.

lemma *path-inv-points*:
 assumes *construct-path-inv q x y D W* ∧ $q \neq x$
 shows *point q*
  and *point* (*choose-point* ($D*q$))
 using *assms* **apply** *blast*
by (*metis assms choose-point-point comp-assoc is-vector-def point-def reachable-implies-predecessor*)

lemma *path-inv-choose-point-decrease*:
 assumes *construct-path-inv q x y D W* ∧ $q \neq x$
  shows $W{\neq}0 \Longrightarrow$ *choose-point* $(D*q) \leq -((W +$ *choose-point* $(D*q) * q^T)^T*1)$
proof −
 let *?q* = *choose-point* ($D*q$)
 let $?W = W + ?q * q^T$
 assume *as*: $W{\neq}0$
 hence $q*W \leq W^+$
  by (*metis assms conv-contrav conv-invol conv-iso conv-terminating-path forward-terminating-path-end-points-1 plus-conv point-def ss423bij terminating-path-iff*)
 hence $?q \cdot W^T*1 \leq D*q \cdot W^{T+}*q$
  using *choose-point-decreasing meet-iso meet-isor inf-mono assms connected-root-iff2* **by** *simp*
 also have ... $\leq (D \cdot D^{T+})*q$
  by (*metis assms inj-distr point-def conv-contrav conv-invol conv-iso meet-isor mult-isol-var mult-isor star-conv star-slide-var star-subdist*

sup.commute sup.orderE)
also have ... $\leq$ *0*
by (*metis acyclic-trans assms conv-zero step-has-target eq-iff galois-aux ss-p18*)
finally have *a*: *?q* $\leq -(W^T * 1)$
using *galois-aux le-bot* **by** *blast*

have *point ?q*
using *assms* **by**(*rule path-inv-points*(*2*))
hence *?q* $\leq -(q * ?q^T * 1)$
by (*metis assms acyclic-imp-one-step-different-points*(*2*) *point-is-point*
choose-point-decreasing edge-end end-point-char end-point-no-successor)
with *a* **show** *?thesis*
by (*simp add*: *inf.boundedI*)
qed

lemma *end-points*:
assumes *construct-path-inv q x y D W* $\wedge$ *q* $\neq$ *x*
shows *choose-point* (*D∗q*) = *start-points* (*W* + *choose-point* (*D∗q*) ∗ q^T)
and *y* = *end-points* (*W* + *choose-point* (*D∗q*) ∗ q^T)
proof −
let *?q* = *choose-point* (*D∗q*)
let *?W* = *W* + *?q* ∗ q^T
show *1*: *?q* = *start-points ?W*
proof (*rule antisym*)
show *start-points ?W* $\leq$ *?q*
by (*metis assms*(*1*) *path-inv-points*(*2*)
acyclic-imp-one-step-different-points(*2*)
choose-point-decreasing edge-end edge-start sup.commute
path-concatenation-start-points-approx point-is-point eq-iff sup-bot-left)
show *?q* $\leq$ *start-points ?W*
proof −
have *a*: *?q* = $?q * q^T * 1$
by (*metis assms*(*1*) *comp-assoc point-equations*(*1*) *point-is-point aux4*
conv-zero
choose-point-decreasing choose-point-point conv-contrav conv-one
point-def
inf.orderE inf-compl-bot inf-compl-bot-right is-vector-def maddux-142
sup-bot-left sur-def-var1)
hence *?q* =($q \cdot -q$) + ($?q \cdot -q \cdot -(?W^T * 1)$)
by (*metis assms path-inv-points*(*2*) *path-inv-choose-point-decrease*
acyclic-imp-one-step-different-points(*1*) *choose-point-decreasing*
inf.orderE
inf-compl-bot sup-inf-absorb edge-start point-is-point sup-bot-left)
also have ... $\leq (W * 1 \cdot -(?W^T * 1) \cdot -q) + (?q \cdot -q \cdot -(?W^T * 1))$
by *simp*
also have ... = $(W * 1 + ?q) \cdot -(q + ?W^T * 1)$
by (*metis compl-sup inf-sup-distrib2 meet-assoc sup.commute*)
also have ... $\leq ?W * 1 \cdot -(?W^T * 1)$
using *a* **by** (*metis inf.left-commute distrib-right′ compl-sup*

inf.cobounded2)
    finally show *?q* ≤ *start-points ?W* **.**
  qed
qed
show *y* = *end-points ?W*
proof −
  have *point-nq*: *point ?q*
    using *assms* **by**(*rule path-inv-points*(*2*))
  hence *yp*: *y* ≤ −*?q*
    using *1 assms*
    by (*metis acyclic-imp-one-step-different-points*(*2*) *choose-point-decreasing cycle-no-points*(*1*)
        finite-iff finite-iff-msc forward-finite-iff-msc path-aux1a path-edge-equals-cycle
        point-is-point point-not-equal(*1*) *terminating-iff1*)
  have $y = y + (W*1 \cdot -(W^T*1) \cdot -(W*1))$
    by (*simp add*: *inf.commute*)
  also have ... $= y + (q \cdot -(W*1))$
    using *assms* **by** *fastforce*
  also have ... $= y + (q \cdot -(W*1) \cdot -?q)$
    by (*metis calculation sup-assoc sup-inf-absorb*)
  also have ... $= (y \cdot -?q) + (q \cdot -(W*1) \cdot -?q)$
    using *yp* **by** (*simp add*: *inf.absorb1*)
  also have ... $= (W^T*1 \cdot -(W*1) \cdot -?q) + (q \cdot -(W*1) \cdot -?q)$
    using *assms* **by** *fastforce*
  also have ... $= (W^T*1 + q) \cdot -(W*1) \cdot -?q$
    by (*simp add*: *inf-sup-distrib2*)
  also have ... $= (W^T*1 + q) \cdot -(W*1 + ?q)$
    by (*simp add*: *inf.assoc*)
  also have ... $= (W^T*1 + q*?q^T*1) \cdot -(W*1 + ?q*q^T*1)$
    using *point-nq*
    by(*metis assms*(*1*) *comp-assoc conv-contrav conv-one is-vector-def point-def sur-def-var1*)
  also have ... $= (?W^T)*1 \cdot -(?W*1)$
    by *simp*
  finally show *?thesis* **.**
qed
qed

lemma *construct-path-inv*:
assumes *construct-path-inv q x y D W* ∧ *q* ≠ *x*
shows *construct-path-inv* (*choose-point* (*D*∗*q*)) *x y D* (W + *choose-point* (*D*∗*q*)∗q^T)
proof (*intro conjI*)
let *?q* = *choose-point* (*D*∗*q*)
let $?W = W + ?q * q^T$
show *is-acyclic D*
  using *assms* **by** *blast*
show *point-y*: *point y*

using *assms* **by** *blast*
show *point x*
using *assms* **by** *blast*
show $?W \leq D$
using *assms choose-point-decreasing le-sup-iff point-def ss423bij inf.boundedE* **by** *blast*
show $D^\star * ?q \leq D^{T\star} * x$
proof −
have $D^+ * q \leq D^{T\star} * x$
using *assms conv-galois-2 order-trans star-1l* **by** *blast*
thus *?thesis*
by (*metis choose-point-decreasing comp-assoc dual-order.trans mult-isol star-slide-var*)
qed
show *point-nq*: *point ?q*
using *assms* **by**(*rule path-inv-points*(*2*))
show *pathW*: *path ?W*
proof(*cases* $W=0$)
assume $W=0$
thus *?thesis*
using *assms edge-is-path point-is-point point-nq* **by** *simp*
next
assume *a*: $W \neq 0$
have *b*: $?q*q^T \leq 1*?q*q^T*-(?q*q^T*1)$
proof −
have $?q*q^T \leq 1$ **by** *simp*
thus *?thesis*
using *assms point-nq*
by(*metis different-points-consequences*(*1*) *point-def sur-def-var1 acyclic-imp-one-step-different-points*(*2*) *choose-point-decreasing comp-assoc is-vector-def point-def point-equations*(*3*,*4*) *point-is-point*)
qed
have *c*: $W \leq -(1*W)*W*1$
using *assms terminating-path-iff* **by** *blast*
have *d*: $(?q*q^T)^T*1 \cdot -((?q*q^T)*1) = W*1 \cdot -(W^T*1)$
using *a*
by (*metis assms path-inv-points*(*2*) *acyclic-reachable-points choose-point-decreasing edge-end point-is-point comp-assoc point-def sur-total total-one*)
have *e*: $?q*q^T*1 \cdot W^T*1 = 0$
proof −
have $?q*q^T*1 \cdot W^T*1 = ?q \cdot W^T*1$
using *assms point-nq*
by (*metis comp-assoc conv-contrav conv-one is-vector-def point-def sur-def-var1*)
also have ... $\leq -(?W^T*1) \cdot ?W^T*1$
using *assms path-inv-choose-point-decrease*
by (*smt a conv-contrav conv-iso conv-one inf-mono less-eq-def subdistl-eq*)

also have ... $\leq$ *0*
using *compl-inf-bot eq-refl* **by** *blast*
finally show *?thesis*
using *bot-unique* **by** *blast*
qed
show *?thesis*
using *b c d e* **by** (*metis assms comp-assoc edge-is-path path-concatenation-cycle-free point-is-point sup.commute point-nq*)
qed
show $?W = 0 \longleftrightarrow ?q = y$
apply (*rule iffI*)
apply (*metis assms conv-zero dist-alt edge-start inf-compl-bot-right modular-1-aux' modular-2-aux' point-is-point sup.left-idem sup-bot-left point-nq*)
by (*smt assms end-points*(*1*) *conv-contrav conv-invol cycle-no-points*(*1*) *end-point-iff2 has-start-end-points-iff path-aux1b path-edge-equals-cycle point-is-point start-point-iff2 sup-bot-left top-greatest pathW*)
show $?W \neq 0 \longleftrightarrow ?q = start\text{-}points\ ?W \wedge y = end\text{-}points\ ?W$
apply (*rule iffI*)
using *assms end-points* **apply** *blast*
using *assms* **by** *force*
show *terminating ?W*
by (*smt assms end-points end-point-iff2 has-start-end-points-iff point-is-point start-point-iff2 terminating-iff1 pathW point-nq*)
qed

theorem *construct-path-partial*: *VARS p q W*
$\{ \text{is-acyclic } D \wedge \text{point } y \wedge \text{point } x \wedge D^\star * y \leq D^{T\star} * x \}$
$W := 0$;
$q := y$;
WHILE $q \neq x$
INV { *construct-path-inv q x y D W* }
DO $p :=$ *choose-point* $(D*q)$;
$W := W + p*q^T$;
$q := p$
OD
$\{ W \leq D \wedge \text{terminating-path } W \wedge (W=0 \longleftrightarrow x=y) \wedge (W\neq 0 \longleftrightarrow x = \text{start-points } W \wedge y = \text{end-points } W) \}$
apply *vcg*
using *construct-path-pre* **apply** *blast*
using *construct-path-inv* **apply** *blast*
by *fastforce*

end

For termination, we additionally need finiteness.

context *finite*

begin

lemma *decrease-set*:
  assumes $\forall x::'a . Q\ x \longrightarrow P\ x$
    and $P\ w$
    and $\neg\ Q\ w$
   shows $card\ \{\ x\ .\ Q\ x\ \} < card\ \{\ x\ .\ P\ x\ \}$
by (*metis Collect-mono assms card-seteq finite mem-Collect-eq not-le*)

end

class *relation-algebra-rtc-tarski-choose-point-finite* =
    relation-algebra-rtc-tarski-choose-point +
relation-algebra-rtc-tarski-point-finite
begin

lemma *decrease-variant*:
  assumes $y \leq z$
    and $w \leq z$
    and $\neg\ w \leq y$
   shows $card\ \{\ x\ .\ x \leq y\ \} < card\ \{\ x\ .\ x \leq z\ \}$
by (*metis Collect-mono assms card-seteq linorder-not-le dual-order.trans finite-code mem-Collect-eq*)

lemma *construct-path-inv-termination*:
  assumes *construct-path-inv* $q\ x\ y\ D\ W \wedge q \neq x$
   shows $card\ \{\ z\ .\ z \leq -(W + choose\text{-}point\ (D{*}q){*}q^T)\ \} < card\ \{\ z\ .\ z \leq -W\ \}$
proof −
  let $?q = choose\text{-}point\ (D{*}q)$
  let $?W = W + ?q * q^T$
  show *?thesis*
  proof (*rule decrease-variant*)
   show $-?W \leq -W$
    by *simp*
   show $?q * q^T \leq -W$
    by (*metis assms galois-aux inf-compl-bot-right maddux-142 mult-isor order-trans top-greatest*)
   show $\neg\ (?q * q^T \leq -?W)$
    using *assms end-points*(*1*)
    by (*smt acyclic-imp-one-step-different-points*(*2*) *choose-point-decreasing compl-sup inf.absorb1*
      inf-compl-bot-right sup.commute sup-bot.left-neutral conv-zero end-points(*2*))
  qed
qed

theorem *construct-path-total*: *VARS p q W*
  [*is-acyclic* $D \wedge$ *point* $y \wedge$ *point* $x \wedge D^\star{*}y \leq D^{T\star}{*}x$]

$W := 0$;
$q := y$;
WHILE $q \neq x$
INV { *construct-path-inv* $q\ x\ y\ D\ W$ }
VAR { *card* { z . $z \leq -W$ } }
DO $p :=$ *choose-point* $(D*q)$;
$W := W + p*q^T$;
$q := p$
OD
[$W \leq D$ ∧ *terminating-path* W ∧ $(W{=}0 \longleftrightarrow x{=}y)$ ∧ $(W{\neq}0 \longleftrightarrow x =$ *start-points* $W \wedge y =$ *end-points* $W)$]
apply *vcg-tc*
using *construct-path-pre* **apply** *blast*
apply (*rule CollectI*, *rule conjI*)
using *construct-path-inv* **apply** *blast*
using *construct-path-inv-termination* **apply** *clarsimp*
by *fastforce*

end

4.2 Topological sorting

In our second example we look at topological sorting. Given a directed acyclic graph, the problem is to construct a linear order of its vertices that contains x before y for each edge (x, y) of the graph. If the input graph models dependencies between tasks, the output is a linear schedule of the tasks that respects all dependencies.

context *relation-algebra-rtc-tarski-choose-point*
begin

abbreviation *topological-sort-inv*
where *topological-sort-inv* $q\ v\ R\ W \equiv$
regressively-finite $R \wedge R \cdot v*v^T \leq W^+ \wedge$ *terminating-path* $W \wedge W*1 = v \cdot -q \wedge$
$(W = 0 \vee q =$ *end-points* $W) \wedge$ *point* $q \wedge R*v \leq v \wedge q \leq v \wedge$ *is-vector* v

lemma *topological-sort-pre*:
assumes *regressively-finite* R
shows *topological-sort-inv* (*choose-point* (*minimum* R *1*)) (*choose-point* (*minimum* R *1*)) R *0*
proof (*intro conjI*,*simp-all add*:*assms*)
let *?q* = *choose-point* $(-(R^T * 1))$
show *point-q*: *point ?q*
using *assms* **by** (*metis* (*full-types*) *annir choose-point-point galois-aux2 is-inj-def is-sur-def*
is-vector-def one-idem-mult point-def ss-p18 inf-top-left one-compl)

show $R \cdot ?q * ?q^T \leq 0$
by (*metis choose-point-decreasing conv-invol end-point-char eq-iff inf-bot-left schroeder-2*)
show *path 0*
by (*simp add*: *is-inj-def is-p-fun-def path-def*)
show $R{*}?q \leq ?q$
by (*metis choose-point-decreasing compl-bot-eq conv-galois-1 inf-compl-bot-left2 le-inf-iff*)
show *is-vector ?q*
using *point-q point-def* **by** *blast*
qed

lemma *topological-sort-inv*:
assumes $v \neq 1$
and *topological-sort-inv q v R W*
shows *topological-sort-inv* (*choose-point* (*minimum R* (− *v*))) (*v* + *choose-point* (*minimum R* (− *v*))) *R* ($W + q * $ *choose-point* (*minimum R* (− *v*))^T)
proof (*intro conjI*)
let *?p* = *choose-point* (*minimum R* (−*v*))
let $?W = W + q{*}?p^T$
let $?v = v + ?p$
show *point-p*: *point ?p*
using *assms*
by (*metis choose-point-point compl-bot-eq double-compl galois-aux2 comp-assoc is-vector-def*
vector-compl vector-mult)
hence *ep-np*: *end-points* $(q{*}?p^T) = ?p$
using *assms*(*2*)
by (*metis aux4 choose-point-decreasing edge-end le-supI1 point-in-vector-or-complement-iff*
point-is-point)
hence *sp-q*: *start-points* $(q{*}?p^T) = q$
using *assms*(*2*) *point-p*
by (*metis* (*no-types*, *lifting*) *conv-contrav conv-invol edge-start point-is-point*)
hence *ep-sp*: $W \neq 0 \Longrightarrow$ *end-points* $W =$ *start-points* $(q{*}?p^T)$
using *assms*(*2*) **by** *force*
have $W{*}1 \cdot (q{*}?p^T)^T{*}1 = v{\cdot}{-}q{\cdot}?p$
using *assms*(*2*) *point-p is-vector-def mult-assoc point-def point-equations*(*3*) *point-is-point*
by *auto*
hence *1*: $W{*}1 \cdot (q{*}?p^T)^T{*}1 = 0$
by (*metis choose-point-decreasing dual-order.trans galois-aux inf.cobounded2 inf.commute*)

show *regressively-finite R*
using *assms*(*2*) **by** *blast*
show $R \cdot ?v{*}?v^T \leq ?W^+$
proof −

have *a*: $R \cdot v{*}v^T \leq ?W^+$
using *assms*(*2*) **by** (*meson mult-isol-var order.trans order-prop star-subdist*)
have *b*: $R \cdot v{*}?p^T \leq ?W^+$
proof –
have $R \cdot v{*}?p^T \leq W{*}1{*}?p^T + q{*}?p^T$
by (*metis inf-le2 assms*(*2*) *aux4 double-compl inf-absorb2 distrib-right*)
also have ... $= W{*}?p^T + q{*}?p^T$
using *point-p* **by** (*metis conv-contrav conv-one is-vector-def mult-assoc point-def*)
also have ... $\leq W^+{*}end\text{-}points\ W{*}?p^T + q{*}?p^T$
using *assms*(*2*)
by (*meson forward-terminating-path-end-points-1 join-iso mult-isor terminating-path-iff*)
also have ... $\leq W^+{*}q{*}?p^T + q{*}?p^T$
using *assms*(*2*) **by** (*metis annil eq-refl*)
also have ... $= W^\star{*}q{*}?p^T$
using *conway.dagger-unfoldl-distr mult-assoc sup-commute* **by** *fastforce*
also have ... $\leq ?W^+$
by (*metis mult-assoc mult-isol-var star-slide-var star-subdist sup-ge2*)
finally show *?thesis* **.**
qed
have *c*: $R \cdot ?p{*}v^T \leq ?W^+$
proof –
have $v \leq -?p$
using *choose-point-decreasing compl-le-swap1 inf-le1 order-trans* **by** *blast*
hence $R{*}v \leq -?p$
using *assms*(*2*) *order.trans* **by** *blast*
thus *?thesis*
by (*metis galois-aux inf-le2 schroeder-2*)
qed
have *d*: $R \cdot ?p{*}?p^T \leq ?W^+$
proof –
have $R \cdot ?p{*}?p^T \leq R \cdot 1'$
using *point-p is-inj-def meet-isor point-def* **by** *blast*
also have ... $= 0$
using *assms*(*2*) *regressively-finite-irreflexive galois-aux* **by** *blast*
finally show *?thesis*
using *bot-least inf.absorb-iff2* **by** *simp*
qed
have $R \cdot ?v{*}?v^T = (R \cdot v{*}v^T) + (R \cdot v{*}?p^T) + (R \cdot ?p{*}v^T) + (R \cdot ?p{*}?p^T)$
by (*metis conv-add distrib-left distrib-right inf-sup-distrib1 sup.commute sup.left-commute*)
also have ... $\leq ?W^+$
using *a b c d* **by** (*simp add*: *le-sup-iff*)
finally show *?thesis* **.**
qed
show *pathW*: *path ?W*
proof (*cases* $W = 0$)
assume $W = 0$

thus *?thesis*
using *assms(2) point-p edge-is-path point-is-point sup-bot-left* **by** *auto*
next
assume *a1*: $W \neq 0$
have *fw-path*: *forward-terminating-path W*
using *assms(2) terminating-iff* **by** *blast*
have *bw-path*: *backward-terminating-path* $(q*?p^T)$
using *assms point-p sp-q*
by (*metis conv-backward-terminating conv-has-start-points conv-path edge-is-path*
forward-terminating-iff1 point-is-point start-point-iff2)
show *?thesis*
using *fw-path bw-path ep-sp 1 a1 path-concatenation-cycle-free* **by** *blast*
qed
show *terminating ?W*
proof (*rule start-end-implies-terminating*)
show *has-start-points ?W*
apply (*cases* $W = 0$)
using *assms(2) sp-q pathW*
apply (*metis (no-types, lifting) point-is-point start-point-iff2 sup-bot.left-neutral*)
using *assms(2) ep-sp 1 pathW*
by (*metis has-start-end-points-iff path-concatenation-start-points start-point-iff2*
terminating-iff1)
show *has-end-points ?W*
apply (*cases* $W = 0$)
using *point-p ep-np ep-sp pathW end-point-iff2 point-is-point* **apply** *force*
using *point-p ep-np ep-sp 1 pathW*
by (*metis end-point-iff2 path-concatenation-end-points point-is-point*)
qed
show $?W*1 = ?v \cdot -?p$
proof −
have $?W*1 = v$
by (*metis assms(2) point-p is-vector-def mult-assoc point-def point-equations(3)*
point-is-point aux4 distrib-right' inf-absorb2 sup.commute)
also have ... $= v \cdot -?p$
by (*metis choose-point-decreasing compl-le-swap1 inf.coboundedI inf.orderE order-trans*)
finally show *?thesis*
by (*simp add: inf-sup-distrib2*)
qed
show $?W = 0 \vee ?p = $ *end-points ?W*
using *ep-np ep-sp 1* **by** (*metis path-concatenation-end-points sup-bot-left*)
show $R*?v \leq ?v$
using *assms(2)*
by (*meson choose-point-decreasing conv-galois-1 inf.coboundedI2 order.trans sup.coboundedI1*

sup-least)
show *?p* ≤ *?v*
by *simp*
show *is-vector ?v*
using *assms*(*2*) *point-p point-def vector-add* **by** *blast*
qed

lemma *topological-sort-post*:
assumes ¬ *v* ≠ *1*
and *topological-sort-inv q v R W*
shows $R \le W^+ \wedge$ *terminating-path W* $\wedge (W + W^T) * 1 = -1' * 1$
proof (*intro conjI*,*simp-all add*:*assms*)
show $R \le W^+$
using *assms* **by** *force*
show *backward-terminating W* $\wedge W \le 1 * W * (- v + q)$
using *assms* **by** *force*
show $v \cdot - q + W^T * 1 = - 1' * 1$
proof (*cases W* = *0*)
assume *W* = *0*
thus *?thesis*
using *assms*
by (*metis compl-bot-eq conv-one conv-zero double-compl inf-top.left-neutral is-inj-def*
le-bot mult-1-right one-idem-mult point-def ss-p18 star-zero sup.absorb2 top-le)
next
assume *a1*: *W* ≠ *0*
hence $-1' \ne 0$
using *assms backward-terminating-path-irreflexive le-bot* **by** *fastforce*
hence $1 = 1 * -1' * 1$
by (*simp add*: *tarski*)
also have ... $= -1' * 1$
by (*metis comp-assoc distrib-left mult-1-left sup-top-left distrib-right sup-compl-top*)
finally have *a*: $1 = -1' * 1$ **.**
have $W * 1 + W^T * 1 = 1$
using *assms a1* **by** (*metis double-compl galois-aux4 inf.absorb-iff2 inf-top.left-neutral*)
thus *?thesis*
using *a* **by** (*simp add*: *assms*(*2*))
qed
qed

theorem *topological-sort-partial*: *VARS p q v W*
{ *regressively-finite R* }
W := *0*;
q := *choose-point* (*minimum R 1*);
v := *q*;
WHILE v ≠ *1*

```
    INV { topological-sort-inv q v R W }
     DO p := choose-point (minimum R (−v));
        W := W + q∗p^T;
        q := p;
        v := v + p
     OD
  { R ≤ W^+ ∧ terminating-path W ∧ (W + W^T)∗1 = −1′∗1 }
  apply vcg
  using topological-sort-pre apply blast
  using topological-sort-inv apply blast
  using topological-sort-post by blast

end

context relation-algebra-rtc-tarski-choose-point-finite
begin

lemma topological-sort-inv-termination:
  assumes v ≠ 1
      and topological-sort-inv q v R W
    shows card {z . z ≤ −(v + choose-point (minimum R (−v)))} < card { z . z
≤ −v }
proof (rule decrease-variant)
  let ?p = choose-point (minimum R (−v))
  let ?v = v + ?p
  show −?v ≤ −v
    by simp
  show ?p ≤ −v
    using choose-point-decreasing inf.boundedE by blast
  have point ?p
    using assms
    by (metis choose-point-point compl-bot-eq double-compl galois-aux2 comp-assoc
is-vector-def
              vector-compl vector-mult)
  thus ¬ (?p ≤ −?v)
    by (metis annir compl-sup inf.absorb1 inf-compl-bot-right maddux-20
no-end-point-char)
qed
```

Use precondition *is-acyclic* instead of *regressively-finite*. They are equivalent for finite graphs.

```
theorem topological-sort-total: VARS p q v W
  [ is-acyclic R ]
  W := 0;
  q := choose-point (minimum R 1);
  v := q;
  WHILE v ≠ 1
    INV { topological-sort-inv q v R W }
    VAR { card { z . z ≤ −v } }
```

$DO\ p := choose\text{-}point\ (minimum\ R\ (-v));$
$W := W + q*p^T;$
$q := p;$
$v := v + p$
OD
$[\ R \leq W^+ \wedge terminating\text{-}path\ W \wedge (W + W^T)*1 = -1'*1\ ]$
apply *vcg-tc*
apply (*drule acyclic-regressively-finite*)
using *topological-sort-pre* **apply** *blast*
apply (*rule CollectI*, *rule conjI*)
using *topological-sort-inv* **apply** *blast*
using *topological-sort-inv-termination* **apply** *auto*[*1*]
using *topological-sort-post* **by** *blast*

end

4.3 Construction of a tree

Our last application is a correctness proof of an algorithm that constructs a non-empty cycle for a given directed graph. This works in two steps. The first step is to construct a directed tree from a given root along the edges of the graph.

context *relation-algebra-rtc-tarski-choose-point*
begin

abbreviation *construct-tree-pre*
where $construct\text{-}tree\text{-}pre\ x\ y\ R \equiv y \leq R^{T\star}*x \wedge point\ x$
abbreviation *construct-tree-inv*
where $construct\text{-}tree\text{-}inv\ v\ x\ y\ D\ R \equiv construct\text{-}tree\text{-}pre\ x\ y\ R \wedge is\text{-}acyclic\ D \wedge is\text{-}inj\ D \wedge$
$D \leq R \wedge D*x = 0 \wedge v = x + D^T*1 \wedge x*v^T \leq D^\star \wedge D \leq v*v^T \wedge$
$is\text{-}vector\ v$
abbreviation *construct-tree-post*
where $construct\text{-}tree\text{-}post\ x\ y\ D\ R \equiv is\text{-}acyclic\ D \wedge is\text{-}inj\ D \wedge D \leq R \wedge D*x = 0 \wedge D^T*1 \leq D^{T\star}*x \wedge$
$D^\star*y \leq D^{T\star}*x$

lemma *construct-tree-pre*:
assumes *construct-tree-pre x y R*
shows *construct-tree-inv x x y 0 R*
using *assms* **by** (*simp add*: *is-inj-def point-def*)

lemma *construct-tree-inv-aux*:
assumes $\neg\ y \leq v$
and *construct-tree-inv v x y D R*
shows $singleton\ (choose\text{-}singleton\ (v*-v^T \cdot R))$
proof (*rule choose-singleton-singleton*, *rule notI*)
assume $v*-v^T \cdot R = 0$

hence $R^{T\star}*v \le v$
by (*metis galois-aux conv-compl conv-galois-1 conv-galois-2 conv-invol double-compl star-inductl-var*)
hence $y = 0$
using *assms* **by** (*meson mult-isol order-trans sup.cobounded1*)
thus *False*
using *assms point-is-point* **by** *auto*
qed

lemma *construct-tree-inv*:
assumes $\neg\ y \le v$
and *construct-tree-inv v x y D R*
shows *construct-tree-inv* (v + *choose-singleton* ($v*-v^T \cdot R)^T*1$) x y (D + *choose-singleton* ($v*-v^T \cdot R$)) R
proof (*intro conjI*)
let $?e$ = *choose-singleton* ($v*-v^T \cdot R$)
let $?D = D + ?e$
let $?v = v + ?e^T*1$
have *1*: $?e \le v*-v^T$
using *choose-singleton-decreasing inf.boundedE* **by** *blast*
show *point x*
by (*simp add*: *assms*)
show $y \le R^{T\star}*x$
by (*simp add*: *assms*)
show *is-acyclic ?D*
using *1 assms acyclic-inv* **by** *fastforce*
show *is-inj ?D*
using *1 construct-tree-inv-aux assms injective-inv* **by** *blast*
show $?D \le R$
apply (*rule sup.boundedI*)
using *assms* **apply** *blast*
using *choose-singleton-decreasing inf.boundedE* **by** *blast*
show $?D*x = 0$
proof −
have $?D*x = ?e*x$
by (*simp add*: *assms*)
also have ... $\le ?e*v$
by (*simp add*: *assms mult-isol*)
also have ... $\le v*-v^T*v$
using *1 mult-isor* **by** *blast*
also have ... $= 0$
by (*metis assms*(*2*) *annir comp-assoc vector-prop1*)
finally show *?thesis*
using *le-bot* **by** *blast*
qed
show $?v = x + ?D^T*1$
by (*simp add*: *assms sup-assoc*)
show $x*?v^T \le ?D^\star$

```
  proof -
    have x*?v^T = x*v^T + x*1*?e
      by (simp add: distrib-left mult-assoc)
    also have ... ≤ D^⋆ + x*1*(?e · v*-v^T)
      using 1 by (metis assms(2) inf.absorb1 join-iso)
    also have ... = D^⋆ + x*1*(?e · v · -v^T)
      by (metis assms(2) comp-assoc conv-compl inf.assoc vector-compl
vector-meet-comp)
    also have ... ≤ D^⋆ + x*1*(?e · v)
      using join-isol mult-subdistl by fastforce
    also have ... = D^⋆ + x*(1 · v^T)*?e
      by (metis assms(2) inf.commute mult-assoc vector-2)
    also have ... = D^⋆ + x*v^T*?e
      by simp
    also have ... ≤ D^⋆ + D^⋆*?e
      using assms join-isol mult-isor by blast
    also have ... ≤ ?D^⋆
      by (meson le-sup-iff prod-star-closure star-ext star-subdist)
    finally show ?thesis .
  qed
  show ?D ≤ ?v*?v^T
  proof (rule sup.boundedI)
    show D ≤ ?v*?v^T
      using assms
      by (meson conv-add distrib-left le-supI1 conv-iso dual-order.trans
mult-isol-var order-prop)
    have ?e ≤ v*(-v^T · v^T*?e)
      using 1 inf.absorb-iff2 modular-1' by fastforce
    also have ... ≤ v*1*?e
      by (simp add: comp-assoc le-infI2 mult-isol-var)
    also have ... ≤ ?v*?v^T
      by (metis conv-contrav conv-invol conv-iso conv-one mult-assoc mult-isol-var
sup.cobounded1
              sup-ge2)
    finally show ?e ≤ ?v*?v^T
      by simp
  qed
  show is-vector ?v
    using assms comp-assoc is-vector-def by fastforce
qed

lemma construct-tree-post:
  assumes y ≤ v
      and construct-tree-inv v x y D R
    shows construct-tree-post x y D R
proof -
  have v*x^T ≤ D^T⋆
    by (metis (no-types, lifting) assms(2) conv-contrav conv-invol conv-iso
star-conv)
```

hence *1*: $v \leq D^{T\star}*x$
using *assms point-def ss423bij* **by** *blast*
hence *2*: $D^T*1 \leq D^{T\star}*x$
using *assms le-supE* **by** *blast*
have $D^\star*y \leq D^{T\star}*x$
proof (*rule star-inductl*, *rule sup.boundedI*)
show $y \leq D^{T\star}*x$
using *1 assms order.trans* **by** *blast*
next
have $D*(D^{T\star}*x) = D*x + D*D^{T+}*x$
by (*metis conway.dagger-unfoldl-distr distrib-left mult-assoc*)
also have ... $= D*D^{T+}*x$
using *assms* **by** *simp*
also have ... $\leq 1'*D^{T\star}*x$
by (*metis assms*(*2*) *is-inj-def mult-assoc mult-isor*)
finally show $D*(D^{T\star}*x) \leq D^{T\star}*x$
by *simp*
qed
thus *construct-tree-post x y D R*
using *2 assms* **by** *simp*
qed

theorem *construct-tree-partial*: *VARS e v D*
{ *construct-tree-pre x y R* }
$D := 0$;
$v := x$;
WHILE $\neg\ y \leq v$
INV { *construct-tree-inv v x y D R* }
DO $e :=$ *choose-singleton* $(v*-v^T \cdot R)$;
$D := D + e$;
$v := v + e^T*1$
OD
{ *construct-tree-post x y D R* }
apply *vcg*
using *construct-tree-pre* **apply** *blast*
using *construct-tree-inv* **apply** *blast*
using *construct-tree-post* **by** *blast*

end

context *relation-algebra-rtc-tarski-choose-point-finite*
begin

lemma *construct-tree-inv-termination*:
assumes $\neg\ y \leq v$
and *construct-tree-inv v x y D R*
shows *card* { z . $z \leq -(v +$ *choose-singleton* $(v*-v^T \cdot R)^T*1)$ } < *card* { z . $z \leq -v$ }
proof (*rule decrease-variant*)

```
  let ?e = choose-singleton (v∗−v^T · R)
  let ?v = v + ?e^T∗1
  have 1: ?e ≤ v∗−v^T
    using choose-singleton-decreasing inf.boundedE by blast
  have 2: singleton ?e
    using construct-tree-inv-aux assms by simp
  show −?v ≤ −v
    by simp
  have ?e^T ≤ −v∗v^T
    using 1 conv-compl conv-iso by force
  also have ... ≤ −v∗1
    by (simp add: mult-isol)
  finally show ?e^T∗1 ≤ −v
    using assms by (metis is-vector-def mult-isor one-compl)
  thus ¬ (?e^T∗1 ≤ −?v)
    using 2 by (metis annir compl-sup inf.absorb1 inf-compl-bot-right surj-one
tarski)
qed

theorem construct-tree-total: VARS e v D
 [ construct-tree-pre x y R ]
 D := 0;
 v := x;
 WHILE ¬ y ≤ v
   INV { construct-tree-inv v x y D R }
   VAR { card { z . z ≤ −v } }
    DO e := choose-singleton (v∗−v^T · R);
       D := D + e;
       v := v + e^T∗1
    OD
 [ construct-tree-post x y D R ]
  apply vcg-tc
  using construct-tree-pre apply blast
  apply (rule CollectI, rule conjI)
  using construct-tree-inv apply blast
  using construct-tree-inv-termination apply force
  using construct-tree-post by blast

end
```

4.4 Construction of a non-empty cycle

The second step is to construct a path from the root to a given vertex in the tree. Adding an edge back to the root gives the cycle.

```
context relation-algebra-rtc-tarski-choose-point
begin

abbreviation comment
  where comment - ≡ SKIP
```

abbreviation *construct-cycle-inv*
  where *construct-cycle-inv v x y D R* ≡ *construct-tree-inv v x y D R* ∧ *point y* ∧ $y{*}x^T \leq R$

lemma *construct-cycle-pre*:
 assumes ¬ *is-acyclic R*
   and $y = choose\text{-}point\ ((R^+ \cdot 1')*1)$
   and $x = choose\text{-}point\ (R^\star*y \cdot R^T*y)$
  shows *construct-cycle-inv x x y 0 R*
proof(*rule conjI*, *rule-tac* [*2*] *conjI*)
  show *point-y*: *point y*
   using *assms* **by** (*simp add*: *choose-point-point is-vector-def mult-assoc galois-aux ss-p18*)
  have $R^\star*y \cdot R^T*y \neq 0$
  proof
   have $R^+ \cdot 1' = (R^+)^T \cdot 1'$
    by (*metis* (*mono-tags*, *hide-lams*) *conv-e conv-times inf.cobounded1 inf.commute*
      many-strongly-connected-iff-6-eq mult-oner star-subid)
   also have ... $= R^{T+} \cdot 1'$
    using *plus-conv* **by** *fastforce*
   also have ... $\leq (R^{T\star} \cdot R)*R^T$
    by (*metis conv-contrav conv-e conv-invol modular-2-var mult-oner star-slide-var*)
   also have ... $\leq (R^{T\star} \cdot R)*1$
    by (*simp add*: *mult-isol*)
   finally have *a*: $(R^+ \cdot 1')*1 \leq (R^{T\star} \cdot R)*1$
    by (*metis mult-assoc mult-isor one-idem-mult*)
   assume $R^\star*y \cdot R^T*y = 0$
   hence $(R^\star \cdot R^T)*y = 0$
    using *point-y inj-distr point-def* **by** *blast*
   hence $(R^\star \cdot R^T)^T*1 \leq -y$
    by (*simp add*: *conv-galois-1*)
   hence $y \leq -((R^\star \cdot R^T)^T*1)$
    using *compl-le-swap1* **by** *blast*
   also have ... $= -((R^{T\star} \cdot R)*1)$
    by (*simp add*: *star-conv*)
   also have ... $\leq -((R^+ \cdot 1')*1)$
    using *a comp-anti* **by** *blast*
   also have ... $\leq -y$
    by (*simp add*: *assms galois-aux ss-p18 choose-point-decreasing*)
   finally have $y = 0$
    using *inf.absorb2* **by** *fastforce*
   thus *False*
    using *point-y annir point-equations*(*2*) *point-is-point tarski* **by** *force*
  qed
  hence *point-x*: *point x*
   by (*metis point-y assms*(*3*) *inj-distr is-vector-def mult-assoc point-def choose-point-point*)

hence $y \leq R^{T\star} * x$
by (*metis assms*(*3*) *point-y choose-point-decreasing inf-le1 order.trans point-swap star-conv*)
thus *tree-inv*: *construct-tree-inv x x y 0 R*
using *point-x construct-tree-pre* **by** *blast*
show $y * x^T \leq R$
proof −
have $x \leq R^\star{*}y \cdot R^T{*}y$
using *assms*(*3*) *choose-point-decreasing* **by** *blast*
also have ... $= (R^\star \cdot R^T){*}y$
using *point-y inj-distr point-def* **by** *fastforce*
finally have $x{*}y^T \leq R^\star \cdot R^T$
using *point-y point-def ss423bij* **by** *blast*
also have ... $\leq R^T$
by *simp*
finally show *?thesis*
using *conv-iso* **by** *force*
qed
qed

lemma *construct-cycle-pre2*:
assumes $y \leq v$
and *construct-cycle-inv v x y D R*
shows *construct-path-inv y x y D 0* $\wedge D \leq R \wedge D * x = 0 \wedge y * x^T \leq R$
proof(*intro conjI*, *simp-all add*: *assms*)
show $D^\star * y \leq D^{T\star} * x$
using *assms construct-tree-post* **by** *blast*
show *path 0*
by (*simp add*: *is-inj-def is-p-fun-def path-def*)
show $y \neq 0$
using *assms*(*2*) *is-point-def point-is-point* **by** *blast*
qed

lemma *construct-cycle-post*:
assumes $\neg\ q \neq x$
and (*construct-path-inv q x y D W* $\wedge D \leq R \wedge D * x = 0 \wedge y * x^T \leq R$)
shows $W + y * x^T \neq 0 \wedge W + y * x^T \leq R \wedge$ *cycle* $(W + y * x^T)$
proof(*intro conjI*)
let $?C = W + y{*}x^T$
show $?C \neq 0$
by (*metis assms acyclic-imp-one-step-different-points*(*2*) *no-trivial-inverse point-def ss423bij*
sup-bot.monoid-axioms monoid.left-neutral)
show $?C \leq R$
using *assms*(*2*) *order-trans sup.boundedI* **by** *blast*
show *path* $(W + y * x^T)$
by (*metis assms construct-tree-pre edge-is-path less-eq-def path-edge-equals-cycle*
point-is-point terminating-iff1)

show *many-strongly-connected* $(W + y * x^T)$
by (*metis assms construct-tree-pre bot-least conv-zero less-eq-def path-edge-equals-cycle star-conv star-subid terminating-iff1*)
qed

theorem *construct-cycle-partial*: *VARS e p q v x y C D W*
$\{ \neg$ *is-acyclic* $R \}$
$y :=$ *choose-point* $((R^+ \cdot 1') * 1)$;
$x :=$ *choose-point* $(R^\star * y \cdot R^T * y)$;
$D := 0$;
$v := x$;
WHILE $\neg\, y \leq v$
INV $\{$ *construct-cycle-inv* $v\ x\ y\ D\ R \}$
DO $e :=$ *choose-singleton* $(v * -v^T \cdot R)$;
$D := D + e$;
$v := v + e^T * 1$
OD;
comment $\{$ *is-acyclic* $D \wedge$ *point* $y \wedge$ *point* $x \wedge D^\star * y \leq D^{T\star} * x \}$;
$W := 0$;
$q := y$;
WHILE $q \neq x$
INV $\{$ *construct-path-inv* $q\ x\ y\ D\ W \wedge D \leq R \wedge D * x = 0 \wedge y * x^T \leq R \}$
DO $p :=$ *choose-point* $(D * q)$;
$W := W + p * q^T$;
$q := p$
OD;
comment $\{ W \leq D \wedge$ *terminating-path* $W \wedge (W = 0 \longleftrightarrow q = y) \wedge (W \neq 0$
$\longleftrightarrow q =$ *start-points* $W \wedge y =$ *end-points* $W) \}$;
$C := W + y * x^T$
$\{ C \neq 0 \wedge C \leq R \wedge$ *cycle* $C \}$
apply *vcg*
using *construct-cycle-pre* **apply** *blast*
using *construct-tree-inv* **apply** *blast*
using *construct-cycle-pre2* **apply** *blast*
using *construct-path-inv* **apply** *blast*
using *construct-cycle-post* **by** *blast*

end

context *relation-algebra-rtc-tarski-choose-point-finite*
begin

theorem *construct-cycle-total*: *VARS e p q v x y C D W*
$[\neg$ *is-acyclic* $R]$
$y :=$ *choose-point* $((R^+ \cdot 1') * 1)$;
$x :=$ *choose-point* $(R^\star * y \cdot R^T * y)$;
$D := 0$;
$v := x$;
WHILE $\neg\, y \leq v$

INV { *construct-cycle-inv v x y D R* }
VAR { *card* { *z* . $z \leq -v$ } }
DO e := *choose-singleton* $(v*-v^T \cdot R)$;
$D := D + e$;
$v := v + e^T*1$
OD;
comment { *is-acyclic* $D \wedge$ *point* $y \wedge$ *point* $x \wedge D^\star*y \leq D^{T\star}*x$ };
$W := 0$;
$q := y$;
WHILE $q \neq x$
INV { *construct-path-inv q x y D W* $\wedge D \leq R \wedge D*x = 0 \wedge y*x^T \leq R$ }
VAR { *card* { *z* . $z \leq -W$ } }
DO p := *choose-point* $(D*q)$;
$W := W + p*q^T$;
$q := p$
OD;
comment { $W \leq D \wedge$ *terminating-path* $W \wedge (W = 0 \longleftrightarrow q=y) \wedge (W \neq 0 \longleftrightarrow q =$ *start-points* $W \wedge y =$ *end-points* $W)$};
$C := W + y*x^T$
[$C \neq 0 \wedge C \leq R \wedge$ *cycle* C]
apply *vcg-tc*
using *construct-cycle-pre* **apply** *blast*
apply (*rule CollectI*, *rule conjI*)
using *construct-tree-inv* **apply** *blast*
using *construct-tree-inv-termination* **apply** *force*
using *construct-cycle-pre2* **apply** *blast*
apply (*rule CollectI*, *rule conjI*)
using *construct-path-inv* **apply** *blast*
using *construct-path-inv-termination* **apply** *clarsimp*
using *construct-cycle-post* **by** *blast*

end

end

References

[1] A. Armstrong, S. Foster, G. Struth, and T. Weber. Relation algebra. *Archive of Formal Proofs*, 2014.

[2] R. Berghammer, H. Furusawa, W. Guttmann, and P. Höfner. Relational characterisations of paths. *Journal of Logical and Algebraic Methods in Programming*, 2020. To appear.

[3] R. Diestel. *Graph Theory*. Springer, third edition, 2005.

[4] W. Guttmann. Stone-Kleene relation algebras. *Archive of Formal Proofs*, 2017.

[5] W. Guttmann. Stone relation algebras. *Archive of Formal Proofs*, 2017.

[6] W. Guttmann. An algebraic framework for minimum spanning tree problems. *Theoretical Comput. Sci.*, 744:37–55, 2018.

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

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

[9] K. C. Ng. *Relation Algebras with Transitive Closure*. PhD thesis, University of California, Berkeley, 1984.

[10] G. Schmidt and T. Ströhlein. *Relations and Graphs*. Springer, 1993.

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

[12] G. Tinhofer. *Methoden der angewandten Graphentheorie*. Springer, 1976.