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

1 (More) Relation Algebra 2
   1.1 Relation algebras satisfying the Tarski rule . . . . . . . . 6
   1.2 Relation algebras satisfying the point axiom . . . . . . . . 9

2 Relational Characterisation of Paths 13
   2.1 Consequences without the Tarski rule . . . . . . . . . . . . 14
   2.2 Consequences with the Tarski rule . . . . . . . . . . . . . 26

3 Relational Characterisation of Rooted Paths 34
   3.1 Consequences without the Tarski rule . . . . . . . . . . . . 35
   3.2 Consequences with the Tarski rule . . . . . . . . . . . . . 39
   3.3 Consequences with the Tarski rule and the point axiom . . . 46

4 Correctness of Path Algorithms 48
   4.1 Construction of a path . . . . . . . . . . . . . . . . . . . . 49
   4.2 Topological sorting . . . . . . . . . . . . . . . . . . . . . . 51
   4.3 Construction of a tree . . . . . . . . . . . . . . . . . . . . 53
   4.4 Construction of a non-empty cycle . . . . . . . . . . . . . 54
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-Functions

begin

no-notation trancl ((−) [1000] 999)

context relation-algebra begin

notation converse ((−T) [102] 101)

abbreviation bijective
  where bijective x ≡ is-inj x ∧ is-sur x

abbreviation reflexive
  where reflexive R ≡ 1' ≤ R

abbreviation symmetric
  where symmetric R ≡ R = R^T

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

General theorems

lemma $x$-leq-triple-$x$:
\[
x \leq x;x^T;x
\]
(\textit{proof})

lemma \textit{inj-triple}:
\begin{align*}
\text{assumes } & \text{\textit{is-inj } } x \\
\text{shows } & x = x;x^T;x
\end{align*}
(\textit{proof})

lemma \textit{p-fun-triple}:
\begin{align*}
\text{assumes } & \text{\textit{is-p-fun } } x \\
\text{shows } & x = x;x^T;x
\end{align*}
(\textit{proof})

lemma \textit{loop-backward-forward}:
\[
x^T \leq -(1') + x
\]
(\textit{proof})

lemma \textit{inj-sur-semi-swap}:
\begin{align*}
\text{assumes } & \text{\textit{is-sur } } z \\
\text{and } & \text{\textit{is-inj } } x \\
\text{shows } & z \leq y;x \implies x \leq y^T;z
\end{align*}
(\textit{proof})

lemma \textit{inj-sur-semi-swap-short}:
\begin{align*}
\text{assumes } & \text{\textit{is-sur } } z \\
\text{and } & \text{\textit{is-inj } } x \\
\text{shows } & z \leq y^T;x \implies x \leq y;z
\end{align*}
(\textit{proof})

lemma \textit{bij-swap}:
\begin{align*}
\text{assumes } & \text{\textit{bijective } } z \\
\text{and } & \text{\textit{bijective } } x \\
\text{shows } & z \leq y^T;x \leftrightarrow x \leq y;z
\end{align*}
(\textit{proof})

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

lemma \textit{ss422iv}:
\begin{align*}
\text{assumes } & \text{\textit{is-p-fun } } y \\
\text{and } & x \leq y \\
\text{and } & y;1 \leq x;1 \\
\text{shows } & x = y
\end{align*}
(\textit{proof})

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

lemma \textit{ss423conv}: 

3
assumes bijective \( x \)
shows \( x : y \leq z \iff y \leq x^T : z \)
\( \langle \text{proof} \rangle \)

**lemma ss423bij:**
assumes bijective \( x \)
shows \( y : x^T \leq z \iff y \leq z : x \)
\( \langle \text{proof} \rangle \)

**lemma inj-distr:**
assumes is-inj \( z \)
shows \( (x \cdot y) : z = (x : z) \cdot (y : z) \)
\( \langle \text{proof} \rangle \)

**lemma test-converse:**
\( x \cdot 1' = x^T \cdot 1' \)
\( \langle \text{proof} \rangle \)

**lemma injective-down-closed:**
assumes is-inj \( x \)
and \( y \leq x \)
shows is-inj \( y \)
\( \langle \text{proof} \rangle \)

**lemma injective-sup:**
assumes is-inj \( t \)
and \( e : t^T \leq 1' \)
and is-inj \( e \)
shows is-inj \( (t + e) \)
\( \langle \text{proof} \rangle \)

Some (more) results about vectors

**lemma vector-meet-comp:**
assumes is-vector \( v \)
and is-vector \( w \)
shows \( v \cdot w^T = v \cdot w^T \)
\( \langle \text{proof} \rangle \)

**lemma vector-meet-comp':**
assumes is-vector \( v \)
shows \( v \cdot v^T = v \cdot v^T \)
\( \langle \text{proof} \rangle \)

**lemma vector-meet-comp-x:**
\( x : 1 \cdot x^T = x : 1 \cdot 1 \cdot x^T \)
\( \langle \text{proof} \rangle \)

**lemma vector-meet-comp-x':**
\( x : 1 \cdot x = x : 1 \cdot 1 \cdot x \)
proof

lemma vector-prop1:
assumes is-vector v
shows \(-v^Tv = 0\)

(proof)

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)

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)

Some (more) results about points

definition point
where point \(x \equiv \text{is-vector } x \land \text{bijective } x\)

lemma point-swap:
assumes point \(p\)
and point \(q\)
shows \(p \leq x; q \leftrightarrow q \leq x^T; p\)

(proof)

Some (more) results about singletons

abbreviation singleton
where singleton \(x \equiv \text{bijective } (x;1) \land \text{bijective } (x^T;1)\)

lemma singleton-injective:
assumes singleton \(x\)
shows is-inj \(x\)

(proof)

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 \((t + e)\)

(proof)
lemma singleton-is-point:
  assumes singleton p
  shows point (p;1)
⟨proof⟩

lemma singleton-transp:
  assumes singleton p
  shows singleton (pT)
⟨proof⟩

lemma point-to-singleton:
  assumes singleton p
  shows singleton (1′·p;pT)
⟨proof⟩

lemma singleton-singletonT:
  assumes singleton p
  shows p;pT ≤ 1′
⟨proof⟩

Minimality
abbreviation minimum where minimum x v ≡ v ·− (xT; v)

Regressively finite
abbreviation regressively-finite where regressively-finite x ≡ ∀ v . is-vector v ∧ v ≤ xT; v −→ v = 0

lemma regressively-finite-minimum:
  regressively-finite R −→ is-vector v −→ v ≠ 0 −→ minimum R v ≠ 0
⟨proof⟩

lemma regressively-finite-irreflexive:
  assumes regressively-finite x
  shows x ≤ −1′
⟨proof⟩

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 $1;p^T=p^T$
\langle proof\rangle

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

**Lemma** point-singleton:
assumes is-point $p$
and is-vector $v$
and $v \neq 0$
and $v \leq p$
shows $v = p$
\langle proof\rangle

**Lemma** point-not-equal-aux:
assumes is-point $p$
and is-point $q$
shows $p \neq q \leftrightarrow p \cdot -q \neq 0$
\langle proof\rangle

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 $p \neq q \leftrightarrow p \leq -q$
and $p \leq -q \leftrightarrow p; q^T \leq -1'$
and $p; q^T \leq -1' \leftrightarrow p^T; q \leq 0$
\langle proof\rangle

**Lemma** point-is-point:
point $x \leftrightarrow$ is-point $x$
\langle proof\rangle

**Lemma** point-in-vector-or-complement:
assumes point $p$
and is-vector $v$
shows $p \leq v \lor p \leq -v$
\langle proof\rangle

**Lemma** point-in-vector-or-complement-iff:
assumes point $p$
and is-vector $v$
shows $p \leq v \leftrightarrow \neg(p \leq -v)$
\langle proof\rangle

**Lemma** different-points-consequences:
assumes point $p$
and point $q$

7
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⟩

Some (more) results about singletons

lemma singleton-pq:
assumes point \( p \)
and point \( q \)
shows singleton \( (p; q^T) \)
⟨proof⟩

lemma singleton-equal-aux:
assumes singleton \( p \)
and singleton \( q \)
and \( q \leq p \)
shows \( p \leq q; 1 \)
⟨proof⟩

lemma singleton-equal:
assumes singleton \( p \)
and singleton \( q \)
and \( q \leq p \)
shows \( q = p \)
⟨proof⟩

lemma singleton-nonsplit:
assumes singleton \( p \)
and \( x \leq p \)
shows \( x = 0 \lor x = p \)
⟨proof⟩

lemma singleton-nonzero:
assumes singleton \( p \)
shows \( p \neq 0 \)
⟨proof⟩

lemma singleton-sum:
assumes singleton \( p \)
shows \( p \leq x + y \iff (p \leq x \land p \leq y) \)
⟨proof⟩

lemma singleton-iff:
singleton \( x \iff x \neq 0 \land x^T; 1; x + x; 1; x^T \leq 1' \)
⟨proof⟩

lemma singleton-not-atom-in-relation-algebra-tarski:
assumes $p \neq 0$
and $\forall x . x \leq p \rightarrow x = 0 \lor x = p$
shows singleton $p$
nitpick [expect=genuine] ⟨proof⟩
end

1.2 Relation algebras satisfying the point axiom
class relation-algebra-point = relation-algebra +
assumes point-axiom: $x \neq 0 \rightarrow (\exists y z . \text{point } y \land \text{point } z \land y z^T \leq x)$
begin
Some (more) results about points

lemma point-exists:
$\exists x . \text{point } x$
⟨proof⟩

lemma point-below-vector:
assumes is-vector $v$
and $v \neq 0$
shows $\exists x . \text{point } x \land x \leq v$
⟨proof⟩
end

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

lemma atom-is-singleton:
assumes $p \neq 0$
and $\forall x . x \leq p \rightarrow x = 0 \lor x = p$
shows singleton $p$
⟨proof⟩

lemma singleton-iff-atom:
singleton $p \iff p \neq 0 \land (\forall x . x \leq p \rightarrow x = 0 \lor x = p)$
⟨proof⟩

lemma maddux-tarski:
assumes $x \neq 0$
shows $\exists y . y \neq 0 \land y \leq x \land \text{is-p-fun } y$
⟨proof⟩

Intermediate Point Theorem [10, Proposition 2.4.8]

lemma intermediate-point-theorem:
assumes point $p$
and point $r$
shows $p \leq x; y; r \iff (\exists q . \text{point } q \land p \leq x; q \land q \leq y; r)$
proof

end

custom relation-algebra
begin
lemma unfoldl-inductl-implies-unfoldr:
assumes \( \forall x. 1' + x; (rtc x) \leq rtc x \)
and \( \forall x y z. x + y; z \leq z \implies rtc(y); x \leq z \)
shows \( 1' + rtc(x); x \leq rtc x \)
⟨proof⟩

lemma star-transpose-swap:
assumes \( \forall x. 1' + x; (rtc x) \leq rtc x \)
and \( \forall x y z. x + y; z \leq z \implies rtc(y); x \leq z \)
shows \( rtc(x^T) = (rtc x)^T \)
⟨proof⟩

lemma unfoldl-inductl-implies-inductr:
assumes \( \forall x. 1' + x; (rtc x) \leq rtc x \)
and \( \forall x y z. x + y; z \leq z \implies rtc(y); x \leq z \)
shows \( x + z; y \leq z \implies x; rtc(y) \leq z \)
⟨proof⟩
end

custom relation-algebra-rtc
begin
abbreviation tc ((\cdot^+) [101] 100) where tc x \equiv x; x^*
abbreviation is-acyclic
where is-acyclic x \equiv x^+ \leq -1'
General theorems
lemma star-denest-10:
assumes x; y = 0
shows \( (x+y)^* = y; y^*; x^* + x^* \)
⟨proof⟩

lemma star-star-plus:
\( x^* + y^* = x^+ + y^* \)
⟨proof⟩

The following two lemmas are from [6].

lemma cancel-separate:
assumes $x : y \leq 1'$
shows $x^* : y^* \leq x^* + y^*$

⟨proof⟩

lemma cancel-separate-inj-converse:
assumes is-inj $x$
shows $x^* ; x^T* = x^* + x^T*$
⟨proof⟩

lemma cancel-separate-p-fun-converse:
assumes is-p-fun $x$
shows $x^T* ; x* = x* + x^T*$
⟨proof⟩

lemma cancel-separate-converse-idempotent:
assumes is-inj $x$
and is-p-fun $x$
shows $(x^* + x^T*) ; (x^* + x^T*) = x^* + x^T*$
⟨proof⟩

lemma triple-star:
assumes is-inj $x$
and is-p-fun $x$
shows $x^* ; x^T* ; x^* = x^* + x^T*$
⟨proof⟩

lemma inj-xts:
assumes is-inj $x$
shows $x ; x^T* \leq x^* + x^T*$
⟨proof⟩

lemma plus-top:
$x^+ ; 1 = x ; 1$
⟨proof⟩

lemma top-plus:
$1 ; x^+ = 1 ; x$
⟨proof⟩

lemma plus-conv:
$(x^+)^T = x^T+$
⟨proof⟩

lemma inj-implies-step-forwards-backwards:
assumes is-inj $x$
shows $x^* ; (x^+ ; 1^+) ; 1 \leq x^T ; 1$
⟨proof⟩

Acyclic relations
The following result is from [4].

lemma acyclic-inv:
assumes is-acyclic t
  and is-vector v
  and e ≤ v;−v^T
  and t ≤ v;v^T
shows is-acyclic (t + e)
⟨proof⟩

lemma acyclic-single-step:
assumes is-acyclic x
shows x ≤ −1'
⟨proof⟩

lemma acyclic-reachable-points:
assumes is-point p
  and is-point q
  and p ≤ x;q
  and is-acyclic x
shows p≠q
⟨proof⟩

lemma acyclic-trans:
assumes is-acyclic x
shows x ≤ −(x^T+)  
⟨proof⟩

lemma acyclic-trans'}:
assumes is-acyclic x
shows x* ≤ −(x^T+)
⟨proof⟩

Regressively finite

lemma regressively-finite-acyclic:
assumes regressively-finite x
shows is-acyclic x
⟨proof⟩

notation power (infixr ↑ 80)

lemma power-suc-below-plus:
  x ↑ Suc n ≤ x^+  
⟨proof⟩

end

class relation-algebra-rtc-tarski = relation-algebra-rtc + relation-algebra-tarski
begin
\textbf{lemma point-loop-not-acyclic}:  
\textbf{assumes} is-point \( p \)  
\textbf{and} \( p \leq x \uparrow \text{Succ } n \); \( p \)  
\textbf{shows} \( \neg \text{is-acyclic } x \)  
\langle proof \rangle  
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.
\textbf{lemma acyclic-power-vanishes}:  
\textbf{assumes} is-acyclic \( x \)  
\textbf{shows} \( \exists n . x \uparrow \text{Succ } n = 0 \)  
\langle proof \rangle  
Hence finite acyclic relations are regressively finite.
\textbf{lemma acyclic-regressively-finite}:  
\textbf{assumes} is-acyclic \( x \)  
\textbf{shows} regressively-finite \( x \)  
\langle proof \rangle  
\textbf{lemma acyclic-is-regressively-finite}:  
is-acyclic \( x \) \( \iff \) regressively-finite \( x \)  
\langle proof \rangle  
end
end

2 Relational Characterisation of Paths

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

imports More-Relation-Algebra
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⟩

2.1 Consequences without the Tarski rule

Definitions for path classifications

abbreviation connected
where connected x ≡ x;1;x ≤ x^* + x^{T*}

abbreviation many-strongly-connected
where many-strongly-connected x ≡ x^* = x^{T*}

abbreviation one-strongly-connected
where one-strongly-connected x ≡ x^T;1;x^T ≤ x^*

definition path
where path x ≡ connected x ∧ is-p-fun x ∧ is-inj x

abbreviation cycle
where cycle x ≡ path x ∧ many-strongly-connected x

abbreviation start-points
where start-points x ≡ x;1 · -(x^T;1)

abbreviation end-points
where end-points x ≡ x^T;1 · -(x;1)

abbreviation no-start-points
where no-start-points x ≡ x;1 ≤ x^T;1

abbreviation no-end-points
where no-end-points x ≡ x^T;1 ≤ x;1

abbreviation no-start-end-points
where no-start-end-points x ≡ x;1 = x^T;1
abbreviation has-start-points
  where has-start-points x ≡ 1 = -(1;x)x;1

abbreviation has-end-points
  where has-end-points x ≡ 1 = 1;x;-(x;1)

abbreviation has-start-end-points
  where has-start-end-points x ≡ 1 = -(1;x);1 · 1;x;-(x;1)

abbreviation backward-terminating
  where backward-terminating x ≡ x ≤ -(1;x);x;1

abbreviation forward-terminating
  where forward-terminating x ≡ x ≤ 1;x;-(x;1)

abbreviation terminating
  where terminating x ≡ x ≤ -(1;x);x;1 · 1;x;-(x;1)

abbreviation backward-finite
  where backward-finite x ≡ x ≤ x^T* + -(1;x);x;1

abbreviation forward-finite
  where forward-finite x ≡ x ≤ x^T* + 1;x;-(x;1)

abbreviation finite
  where finite x ≡ x ≤ x^T* + -(1;x);x;1 · 1;x;-(x;1))

abbreviation no-start-points-path
  where no-start-points-path x ≡ path x ∧ no-start-points x

abbreviation no-end-points-path
  where no-end-points-path x ≡ path x ∧ no-end-points x

abbreviation no-start-end-points-path
  where no-start-end-points-path x ≡ path x ∧ no-start-end-points x

abbreviation has-start-points-path
  where has-start-points-path x ≡ path x ∧ has-start-points x

abbreviation has-end-points-path
  where has-end-points-path x ≡ path x ∧ has-end-points x

abbreviation has-start-end-points-path
  where has-start-end-points-path x ≡ path x ∧ has-start-end-points x

abbreviation backward-terminating-path
  where backward-terminating-path x ≡ path x ∧ backward-terminating x
abbreviation forward-terminating-path
  where forward-terminating-path x ≡ path x ∧ forward-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 ≤ y, z
  and x · z = 0
  shows x ≤ y’; z
⟨proof⟩

lemma reachable-imp:
  assumes point p
  and point q
  and p’; q ≤ p’; p
  shows p ≤ p’; q
⟨proof⟩

Basic equivalences

lemma no-start-end-points-iff:
  no-start-end-points x ←→ no-start-points x ∧ no-end-points x
⟨proof⟩

lemma has-start-end-points-iff:
  has-start-end-points x ←→ has-start-points x ∧ has-end-points x
⟨proof⟩

lemma terminating-iff:
  terminating x ←→ backward-terminating x ∧ forward-terminating x
⟨proof⟩

lemma finite-iff:
  finite x ←→ backward-finite x ∧ forward-finite x
⟨proof⟩

lemma no-start-end-points-path-iff:
  no-start-end-points-path x ←→ no-start-points-path x ∧ no-end-points-path x
⟨proof⟩
lemma has-start-end-points-path-iff:
has-start-end-points-path x ⇐⇒ has-start-points-path x ∧ has-end-points-path x
⟨proof⟩

lemma terminating-path-iff:
terminating-path x ⇐⇒ backward-terminating-path x ∧ forward-terminating-path x
⟨proof⟩

lemma finite-path-iff:
finite-path x ⇐⇒ backward-finite-path x ∧ forward-finite-path x
⟨proof⟩

Closure under converse

lemma connected-conv:
connected x ⇐⇒ connected (xT)
⟨proof⟩

lemma conv-many-strongly-connected:
many-strongly-connected x ⇐⇒ many-strongly-connected (xT)
⟨proof⟩

lemma conv-one-strongly-connected:
one-strongly-connected x ⇐⇒ one-strongly-connected (xT)
⟨proof⟩

lemma conv-path:
path x ⇐⇒ path (xT)
⟨proof⟩

lemma conv-cycle:
cycle x ⇐⇒ cycle (xT)
⟨proof⟩

lemma conv-no-start-points:
no-start-points x ⇐⇒ no-end-points (xT)
⟨proof⟩

lemma conv-no-start-end-points:
no-start-end-points x ⇐⇒ no-start-end-points (xT)
⟨proof⟩

lemma conv-has-start-points:
has-start-points x ⇐⇒ has-end-points (xT)
⟨proof⟩

lemma conv-has-start-end-points:
has-start-end-points x ⇐⇒ has-start-end-points (xT)
lemma conv-backward-terminating:
  backward-terminating \( x \) \iff\ forward-terminating \( x^T \)
(proof)

lemma conv-terminating:
  terminating \( x \) \iff\ terminating \( x^T \)
(proof)

lemma conv-backward-finite:
  backward-finite \( x \) \iff\ forward-finite \( x^T \)
(proof)

lemma conv-finite:
  finite \( x \) \iff\ finite \( x^T \)
(proof)

lemma conv-no-start-points-path:
  no-start-points-path \( x \) \iff\ no-end-points-path \( x^T \)
(proof)

lemma conv-no-start-end-points-path:
  no-start-end-points-path \( x \) \iff\ no-start-end-points-path \( x^T \)
(proof)

lemma conv-has-start-points-path:
  has-start-points-path \( x \) \iff\ has-end-points \( x^T \)
(proof)

lemma conv-has-start-end-points-path:
  has-start-end-points-path \( x \) \iff\ has-start-end-points-path \( x^T \)
(proof)

lemma conv-backward-terminating-path:
  backward-terminating-path \( x \) \iff\ forward-terminating-path \( x^T \)
(proof)

lemma conv-terminating-path:
  terminating-path \( x \) \iff\ terminating-path \( x^T \)
(proof)

lemma conv-backward-finite-path:
  backward-finite-path \( x \) \iff\ forward-finite-path \( x^T \)
(proof)

lemma conv-finite-path:
  finite-path \( x \) \iff\ finite-path \( x^T \)
(proof)
Equivalences for connected

**lemma connected-iff2:**

*assumes* is-inj $x$

*and* is-p-fun $x$

*shows* connected $x \iff x; 1; x^T \leq x^* + x^{T*}$

(proof)

**lemma connected-iff3:**

*assumes* is-inj $x$

*and* is-p-fun $x$

*shows* connected $x \iff x^T; 1; x \leq x^* + x^{T*}$

(proof)

**lemma connected-iff4:**

connected $x \iff x^T; 1; x \leq x^* + x^{T*}$

(proof)

**lemma connected-iff5:**

connected $x \iff x^T; 1; x \leq x^* + x^{T*}$

(proof)

**lemma connected-iff6:**

*assumes* is-inj $x$

*and* is-p-fun $x$

*shows* connected $x \iff (x^+); 1; (x^+)^T \leq x^* + x^{T*}$

(proof)

**lemma connected-iff7:**

*assumes* is-inj $x$

*and* is-p-fun $x$

*shows* connected $x \iff (x^+)^T; 1; x^+ \leq x^* + x^{T*}$

(proof)

**lemma connected-iff8:**

connected $x \iff (x^+)^T; 1; (x^+)^T \leq x^* + x^{T*}$

(proof)

Equivalences and implications for many-strongly-connected

**lemma many-strongly-connected-iff-1:**

many-strongly-connected $x \iff x^T \leq x^*$

(proof)

**lemma many-strongly-connected-iff-2:**

many-strongly-connected $x \iff x^T \leq x^+$

(proof)

**lemma many-strongly-connected-iff-3:**

many-strongly-connected $x \iff x \leq x^{T*}$

(proof)
lemma many-strongly-connected-iff-4:
many-strongly-connected x $\iff$ $x \leq x^T$
⟨proof⟩

lemma many-strongly-connected-iff-5:
many-strongly-connected x $\iff$ $x^*; x^T \leq x^+$
⟨proof⟩

lemma many-strongly-connected-iff-6:
many-strongly-connected x $\iff$ $x^T; x^* \leq x^+$
⟨proof⟩

lemma many-strongly-connected-iff-7:
many-strongly-connected x $\iff$ $x^T = x^+$
⟨proof⟩

lemma many-strongly-connected-iff-5-eq:
many-strongly-connected x $\iff$ $x^*; x^T = x^+$
⟨proof⟩

lemma many-strongly-connected-iff-6-eq:
many-strongly-connected x $\iff$ $x^T; x^* = x^+$
⟨proof⟩

lemma many-strongly-connected-implies-no-start-end-points:
assumes many-strongly-connected x
shows no-start-end-points x
⟨proof⟩

lemma many-strongly-connected-implies-8:
assumes many-strongly-connected x
shows $x; x^T \leq x^+$
⟨proof⟩

lemma many-strongly-connected-implies-9:
assumes many-strongly-connected x
shows $x^T; x \leq x^+$
⟨proof⟩

lemma many-strongly-connected-implies-10:
assumes many-strongly-connected x
shows $x; x^T; x^* \leq x^+$
⟨proof⟩

lemma many-strongly-connected-implies-10-eq:
assumes many-strongly-connected x
shows $x; x^T; x^* = x^+$
⟨proof⟩
lemma many-strongly-connected-implies-11:
  assumes many-strongly-connected x
  shows $x^*;x^T;x \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-implies-11-eq:
  assumes many-strongly-connected x
  shows $x^*;x^T;x = x^+$
  ⟨proof⟩

lemma many-strongly-connected-implies-12:
  assumes many-strongly-connected x
  shows $x^*;x;x^T \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-implies-12-eq:
  assumes many-strongly-connected x
  shows $x^*;x;x^T = x^+$
  ⟨proof⟩

lemma many-strongly-connected-implies-13:
  assumes many-strongly-connected x
  shows $x^T;x;x^* \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-implies-13-eq:
  assumes many-strongly-connected x
  shows $x^T;x;x^* = x^+$
  ⟨proof⟩

lemma many-strongly-connected-iff-8:
  assumes is-p-fun x
  shows many-strongly-connected x $\iff x;x^T \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-iff-9:
  assumes is-inj x
  shows many-strongly-connected x $\iff x^T;x \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-iff-10:
  assumes is-p-fun x
  shows many-strongly-connected x $\iff x;x^T;x^* \leq x^+$
  ⟨proof⟩

lemma many-strongly-connected-iff-10-eq:
  assumes is-p-fun x
  shows many-strongly-connected x $\iff x;x^T;x^* = x^+$
  ⟨proof⟩
lemma many-strongly-connected-iff-11:
  assumes is-inj x
  shows many-strongly-connected $x \iff x^\ast; x^T; x \leq x^+$
  ⟨proof ⟩

lemma many-strongly-connected-iff-11-eq:
  assumes is-inj x
  shows many-strongly-connected $x \iff x^\ast; x^T; x = x^+$
  ⟨proof ⟩

lemma many-strongly-connected-iff-12:
  assumes is-p-fun x
  shows many-strongly-connected $x \iff x^\ast; x^T \leq x^+$
  ⟨proof ⟩

lemma many-strongly-connected-iff-12-eq:
  assumes is-p-fun x
  shows many-strongly-connected $x \iff x^\ast; x^T = x^+$
  ⟨proof ⟩

lemma many-strongly-connected-iff-13:
  assumes is-inj x
  shows many-strongly-connected $x \iff x^T; x^\ast \leq x^+$
  ⟨proof ⟩

lemma many-strongly-connected-iff-13-eq:
  assumes is-inj x
  shows many-strongly-connected $x \iff x^T; x^\ast = x^+$
  ⟨proof ⟩

Equivalences and implications for one-strongly-connected

lemma one-strongly-connected-iff:
  one-strongly-connected $x \iff$ connected $x \land$ many-strongly-connected $x$
  ⟨proof ⟩

lemma one-strongly-connected-iff-1:
  one-strongly-connected $x \iff x^T; 1; x^T \leq x^+$
  ⟨proof ⟩

lemma one-strongly-connected-iff-1-eq:
  one-strongly-connected $x \iff x^T; 1; x^T = x^+$
  ⟨proof ⟩

lemma one-strongly-connected-iff-2:
  one-strongly-connected $x \iff x; 1; x \leq x^{T\ast}$
  ⟨proof ⟩
lemma one-strongly-connected-iff-3:
  one-strongly-connected x ⟷ x ⋆ x ≤ x^T_+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-3-eq:
  one-strongly-connected x ⟷ x ⋆ x = x^T_+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-4-eq:
  one-strongly-connected x ⟷ x^T ⋆ 1 ⋆ x = x^+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-5-eq:
  one-strongly-connected x ⟷ x ⋆ 1 ⋆ x = x^+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-6-aux:
  x ⋆ x^+ ≤ x ⋆ 1 ⋆ x 
  ⟨proof⟩

lemma one-strongly-connected-implies-6-eq:
  assumes one-strongly-connected x
  shows x ⋆ 1 ⋆ x = x ⋆ x^+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-7-aux:
  x^+ ≤ x ⋆ 1 ⋆ x 
  ⟨proof⟩

lemma one-strongly-connected-implies-7-eq:
  assumes one-strongly-connected x
  shows x ⋆ 1 ⋆ x = x^+ 
  ⟨proof⟩

lemma one-strongly-connected-implies-8:
  assumes one-strongly-connected x
  shows x ⋆ 1 ⋆ x ≤ x^* 
  ⟨proof⟩

lemma one-strongly-connected-iff-4:
  assumes is-inj x
  shows one-strongly-connected x ⟷ x^T ⋆ 1 ⋆ x ≤ x^+ 
  ⟨proof⟩

lemma one-strongly-connected-iff-5:
  assumes is-p-fun x
  shows one-strongly-connected x ⟷ x ⋆ 1 ⋆ x^T ≤ x^+ 
  ⟨proof⟩
lemma one-strongly-connected-iff-6:
 assumes is-p-fun x
 and is-inj x
 shows one-strongly-connected x ←→ x;1;x ≤ x;x^+
 ⟨proof⟩

lemma one-strongly-connected-iff-6-eq:
 assumes is-p-fun x
 and is-inj x
 shows one-strongly-connected x ←→ x;1;x = x;x^+
 ⟨proof⟩

Start points and end points

lemma start-end-implies-terminating:
 assumes has-start-points x
 and has-end-points x
 shows terminating x
 ⟨proof⟩

lemma start-points-end-points-conv:
 start-points x = end-points (x^T)
 ⟨proof⟩

lemma start-point-at-most-one:
 assumes path x
 shows is-inj (start-points x)
 ⟨proof⟩

lemma start-point-zero-point:
 assumes path x
 shows start-points x = 0 ∨ is-point (start-points x)
 ⟨proof⟩

lemma start-point-iff1:
 assumes path x
 shows is-point (start-points x) ←→ ¬(no-start-points x)
 ⟨proof⟩

lemma end-point-at-most-one:
 assumes path x
 shows is-inj (end-points x)
 ⟨proof⟩

lemma end-point-zero-point:
 assumes path x
 shows end-points x = 0 ∨ is-point (end-points x)
 ⟨proof⟩

lemma end-point-iff1:
assumes path $x$
shows is-point (end-points $x$) $\iff \neg$(no-end-points $x$)

\langle proof \rangle

lemma predecessor-point':
assumes path $x$
and point $s$
and point $e$
and $e; s^T \leq x$
shows $x; s = e$
\langle proof \rangle

lemma predecessor-point:
assumes path $x$
and point $s$
and point $e$
and $e; s^T \leq x$
shows point($x; s$)
\langle proof \rangle

lemma points-of-path-iff:
shows ($x + x^T; 1 = x^T; 1 + \text{start-points}(x)$
and ($x + x^T; 1 = x; 1 + \text{end-points}(x)$
\langle proof \rangle

Path concatenation preliminaries

lemma path-concat-aux-1:
assumes $x; 1 \cdot y; 1 \cdot y^T; 1 = 0$
and end-points $x = \text{start-points} y$
shows $x; 1 \cdot y; 1 = 0$
\langle proof \rangle

lemma path-concat-aux-2:
assumes $x; 1 \cdot x^T; 1 \cdot y^T; 1 = 0$
and end-points $x = \text{start-points} y$
shows $x^T; 1 \cdot y^T; 1 = 0$
\langle proof \rangle

lemma path-concat-aux3-1:
assumes path $x$
shows $x; 1; x^T \leq x^* + x^{T*}$
\langle proof \rangle

lemma path-concat-aux3-2:
assumes path $x$
shows $x^T; 1; x \leq x^* + x^{T*}$
\langle proof \rangle

lemma path-concat-aux3-3:
\textbf{lemma path-concat-aux-3}: \\
\textbf{assumes path } x \\
\textbf{and } y \leq x^* + x^T \\
\textbf{and } z \leq x^* + x^T \\
\textbf{shows } y;1;z \leq x^* + x^T \\
\langle \text{proof} \rangle

\textbf{lemma path-concat-aux-4}: \\
x^* + x^T \leq x^* + x^T;1 \\
\langle \text{proof} \rangle

\textbf{lemma path-concat-aux-5}: \\
\textbf{assumes path } x \\
\textbf{and } y \leq \text{start-points } x \\
\textbf{and } z \leq x + x^T \\
\textbf{shows } y;1;z \leq x^* \\
\langle \text{proof} \rangle

\textbf{lemma path-conditions-disjoint-points-iff}: \\
x;1 \cdot (x^T;1 + y;1) \cdot y^T;1 = 0 \land \text{start-points } x \cdot \text{end-points } y = 0 \iff x;1 \cdot y^T;1 = 0 \\
\langle \text{proof} \rangle

\textbf{2.2 Consequences with the Tarski rule}

\textbf{context relation-algebra-rtc-tarski}

\textbf{begin}

\begin{itemize}
\item \textbf{General theorems}
\item \textbf{lemma reachable-implies-predecessor}: \\
\textbf{assumes } p \neq q \\
\textbf{and } point p \\
\textbf{and } point q \\
\textbf{and } x^*;q \leq x^T;p \\
\textbf{shows } x;q \neq 0 \\
\langle \text{proof} \rangle
\item \textbf{lemma acyclic-imp-one-step-different-points}: \\
\textbf{assumes } is-acyclic x \\
\textbf{and } point p \\
\textbf{and } point q \\
\textbf{and } p \leq x;q \\
\textbf{shows } p \leq -q \land p \neq q \\
\langle \text{proof} \rangle
\end{itemize}

\textbf{end}
Start points and end points

**Lemma** start-point-iff2:
assumes path x
shows is-point (start-points x) ↔ has-start-points x
(proof)

**Lemma** end-point-iff2:
assumes path x
shows is-point (end-points x) ↔ has-end-points x
(proof)

**Lemma** edge-is-path:
assumes is-point p
and is-point q
shows path (p;q^T)
(proof)

**Lemma** edge-start:
assumes is-point p
and is-point q
and p ≠ q
shows start-points (p;q^T) = p
(proof)

**Lemma** edge-end:
assumes is-point p
and is-point q
and p ≠ q
shows end-points (p;q^T) = q
(proof)

**Lemma** loop-no-start:
assumes is-point p
shows start-points (p;p^T) = 0
(proof)

**Lemma** loop-no-end:
assumes is-point p
shows end-points (p;p^T) = 0
(proof)

**Lemma** start-point-no-predecessor:
x;start-points(x) = 0
(proof)

**Lemma** end-point-no-successor:
x^T;end-points(x) = 0
(proof)
lemma start-to-end:
  assumes path x
  shows start-points(x);end-points(x)^T ≤ x^*
⟨proof⟩

lemma path-acyclic:
  assumes has-start-points-path x
  shows is-acyclic x
⟨proof⟩

Equivalences for terminating

lemma backward-terminating-iff1:
  assumes path x
  shows backward-terminating x ←→ has-start-points x ∨ x = 0
⟨proof⟩

lemma backward-terminating-iff2-aux:
  assumes path x
  shows x;1 · 1;x^T · -(1;x) ≤ x^T*
⟨proof⟩

lemma backward-terminating-iff2:
  assumes path x
  shows backward-terminating x ←→ x ≤ x^T*;-(x^T;1)
⟨proof⟩

lemma backward-terminating-iff3-aux:
  assumes path x
  shows x^T;1 · 1;x^T · -(1;x) ≤ x^T*
⟨proof⟩

lemma backward-terminating-iff3:
  assumes path x
  shows backward-terminating x ←→ x^T ≤ x^T*;-(x^T;1)
⟨proof⟩

lemma backward-terminating-iff4:
  assumes path x
  shows backward-terminating x ←→ x ≤ -(1;x);x^*
⟨proof⟩

lemma forward-terminating-iff1:
  assumes path x
  shows forward-terminating x ←→ has-end-points x ∨ x = 0
⟨proof⟩

lemma forward-terminating-iff2:
  assumes path x
  shows forward-terminating x ←→ x^T ≤ x^*;-(x;1)
lemma forward-terminating-iff3:
assumes path x
shows forward-terminating x \iff x \leq x^*;-(x;1)
(proof)

lemma forward-terminating-iff4:
assumes path x
shows forward-terminating x \iff x \leq -(1;x^T);x^T*
(proof)

lemma terminating-iff1:
assumes path x
shows terminating x \iff has-start-end-points x \lor x = 0
(proof)

lemma terminating-iff2:
assumes path x
shows terminating x \iff x \leq x^T*;-(x^T;1) \cdot -(1;x^T);x^T*
(proof)

lemma terminating-iff3:
assumes path x
shows terminating x \iff x \leq x^*;-(x;1) \cdot -(1;x);x^*
(proof)

lemma backward-terminating-path-irreflexive:
assumes backward-terminating-path x
shows x \leq -1'
(proof)

lemma forward-terminating-path-end-points-1:
assumes forward-terminating-path x
shows x \leq x^+;end-points x
(proof)

lemma forward-terminating-path-end-points-2:
assumes forward-terminating-path x
shows x^T \leq x^*;end-points x
(proof)

lemma forward-terminating-path-end-points-3:
assumes forward-terminating-path x
shows start-points x \leq x^+;end-points x
(proof)

lemma backward-terminating-path-start-points-1:
assumes backward-terminating-path x
shows \( x^T \leq x^{T^+}; \text{start-points } x \)
(proof)

**lemma** backward-terminating-path-start-points-2:
**assumes** backward-terminating-path \( x \)
**shows** \( x \leq x^{T^+}; \text{start-points } x \)
(proof)

**lemma** backward-terminating-path-start-points-3:
**assumes** backward-terminating-path \( x \)
**shows** end-points \( x \leq x^{T^+}; \text{start-points } x \)
(proof)

**lemma** path-aux1a:
**assumes** forward-terminating-path \( x \)
**shows** \( x \neq 0 \iff \text{end-points } x \neq 0 \)
(proof)

**lemma** path-aux1b:
**assumes** backward-terminating-path \( y \)
**shows** \( y \neq 0 \iff \text{start-points } y \neq 0 \)
(proof)

**lemma** path-aux1:
**assumes** forward-terminating-path \( x \)
and backward-terminating-path \( y \)
**shows** \( x \neq 0 \lor y \neq 0 \iff \text{end-points } x \neq 0 \lor \text{start-points } y \neq 0 \)
(proof)

**Equivalences for finite**

**lemma** backward-finite-iff-msc:
backward-finite \( x \iff \text{many-strongly-connected } x \lor \text{backward-terminating } x \)
(proof)

**lemma** forward-finite-iff-msc:
forward-finite \( x \iff \text{many-strongly-connected } x \lor \text{forward-terminating } x \)
(proof)

**lemma** finite-iff-msc:
finite \( x \iff \text{many-strongly-connected } x \lor \text{terminating } x \)
(proof)

**Path concatenation**

**lemma** path-concatenation:
**assumes** forward-terminating-path \( x \)
and backward-terminating-path \( y \)
and end-points $x = \text{start-points } y$
and $x_1 \cdot (x^T; 1 + y; 1) \cdot y^T; 1 = 0$
shows path $(x+y)$

lemma path-concatenation-with-edge:
assumes $x \neq 0$
and forward-terminating-path $x$
and is-point $q$
and $q \leq -(1; x)$
shows path $(x+(\text{end-points } x); q^T)$

lemma path-concatenation-cycle-free:
assumes forward-terminating-path $x$
and backward-terminating-path $y$
and end-points $x = \text{start-points } y$
and $x_1 \cdot y^T; 1 = 0$
shows path $(x+y)$

lemma path-concatenation-start-points-approx:
assumes end-points $x = \text{start-points } y$
shows start-points $(x+y) \leq \text{start-points } x$

lemma path-concatenation-end-points-approx:
assumes end-points $x = \text{start-points } y$
shows end-points $(x+y) \leq \text{end-points } y$

lemma path-concatenation-start-points:
assumes end-points $x = \text{start-points } y$
and $x_1 \cdot y^T; 1 = 0$
shows start-points $(x+y) = \text{start-points } x$

lemma path-concatenation-end-points:
assumes end-points $x = \text{start-points } y$
and $x_1 \cdot y^T; 1 = 0$
shows end-points $(x+y) = \text{end-points } y$

lemma path-concatenation-cycle-free-complete:
assumes forward-terminating-path $x$
and backward-terminating-path $y$
and end-points $x = \text{start-points } y$
and $x_1 \cdot y^T; 1 = 0$
shows path $(x+y) \land \text{start-points } (x+y) = \text{start-points } x \land \text{end-points } (x+y)$
Path restriction (path from a given point)

**lemma** reachable-points-iff:
*assumes* point \( p \)
*shows* \( (x^T \cdot p \cdot x) = (x^T \cdot p \cdot l) \cdot x \)

**lemma** path-from-given-point:
*assumes* path \( x \)
*and* point \( p \)
*shows* \( \text{path}(x^T \cdot p \cdot x) \)
*and* start-points\( (x^T \cdot p \cdot x) \leq p \)
*and* end-points\( (x^T \cdot p \cdot x) \leq \text{end-points}(x) \)

**lemma** path-from-given-point':
*assumes* has-start-points-path \( x \)
*and* point \( p \)
*and* \( p \leq x; l \)
*shows* \( \text{path}(x^T \cdot p \cdot x) \)
*and* start-points\( (x^T \cdot p \cdot x) = p \)
*and* end-points\( (x^T \cdot p \cdot x) = \text{end-points}(x) \)

**Cycles**

**lemma** selfloop-is-cycle:
*assumes* is-point \( x \)
*shows* cycle \( (x; x^T) \)

**lemma** start-point-no-cycle:
*assumes* has-start-points-path \( x \)
*shows* \( \neg \text{cycle} \ x \)

**lemma** end-point-no-cycle:
*assumes* has-end-points-path \( x \)
*shows* \( \neg \text{cycle} \ x \)

**lemma** cycle-no-points:
*assumes* cycle \( x \)
*shows* start-points \( x = 0 \)
*and* end-points \( x = 0 \)

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⟩

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 · (x^T;1 + y;1) · y^T;1 = 0
  shows cycle(x + y)
⟨proof⟩

lemma path-edge-equals-cycle:
  assumes has-start-end-points-path x
  shows cycle(x + end-points(x);(start-points x)^T)
⟨proof⟩

  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⟩

lemma cycle-remove-edge’:
  assumes cycle x
  and point s
  and point e
  and s ≠ e
  and e; s^T ≤ x
  shows path(x · −(e; s^T))
  and s = start-points (x · −(e; s^T))
  and e = end-points (x · −(e; s^T))
⟨proof⟩

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 ≠ 0
  shows xᵀ; 1 ≠ 0
⟨proof⟩

lemma end-point-char:
  xᵀ; p = 0 ⟷ p ≤ -(x; 1)
⟨proof⟩

end

context relation-algebra-tarski
begin

  General theorems concerning points

lemma successor-point:
  assumes is-inj x
  and point r
  and x; r ≠ 0
  shows point (x; r)
⟨proof⟩

lemma no-end-point-char:
  assumes point p
  shows xᵀ; p ≠ 0 ⟷ p ≤ x; 1
⟨proof⟩

lemma no-end-point-char-converse:
  assumes point p
  shows x; p ≠ 0 ⟷ p ≤ xᵀ; 1
⟨proof⟩

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 ≡ r;x ≤ x* + xT* ∧ is-inj x ∧ is-p-fun x ∧ point r

abbreviation connected-root
where connected-root r x ≡ r;x ≤ x+

definition backward-finite-path-root
where backward-finite-path-root r x ≡ connected-root r x ∧ is-inj x ∧ is-p-fun x ∧ point r

abbreviation backward-terminating-path-root
where backward-terminating-path-root r x ≡ backward-finite-path-root r x ∧ x;r
= 0

abbreviation cycle-root
where cycle-root r x ≡ r;x ≤ x+ · xT;1 ∧ is-inj x ∧ is-p-fun x ∧ point r

abbreviation non-empty-cycle-root
where non-empty-cycle-root r x ≡ backward-finite-path-root r x ∧ r ≤ xT;1

abbreviation finite-path-root-end
where finite-path-root-end r x e ≡ backward-finite-path-root r x ∧ point e ∧ r ≤ x*;e

abbreviation terminating-path-root-end
where terminating-path-root-end r x e ≡ finite-path-root-end r x e ∧ xT;e = 0

Equivalent formulations of connected-root

lemma connected-root-iff1:
assumes point r
shows connected-root r x ⊣→ 1;x ≤ rT;x+
⟨proof⟩

lemma connected-root-iff2:
assumes point r
shows connected-root r x ⊣→ xT;1 ≤ xT+r;r
⟨proof⟩

lemma connected-root-aux:
xT+r;r ≤ xT;1
⟨proof⟩

lemma connected-root-iff3:
assumes point r
shows connected-root \( r \ x \leftrightarrow x^T;1 = x^{T^*};r \)

\( \langle \text{proof} \rangle \)

**lemma** connected-root-iff4:
assumes point \( r \)
shows connected-root \( r \ x \leftrightarrow 1;x = r^T;x^+ \)

\( \langle \text{proof} \rangle \)

**Consequences of connected-root**

**lemma** has-root-contra:
assumes connected-root \( r \ x \)
and point \( r \)
and \( x^T;r = 0 \)
shows \( x = 0 \)

\( \langle \text{proof} \rangle \)

**lemma** has-root:
assumes connected-root \( r \ x \)
and point \( r \)
and \( x \neq 0 \)
shows \( x^T;r \neq 0 \)

\( \langle \text{proof} \rangle \)

**lemma** connected-root-move-root:
assumes connected-root \( r \ x \)
and \( q \leq x^*;r \)
shows connected-root \( q \ x \)

\( \langle \text{proof} \rangle \)

**lemma** root-cycle-converse:
assumes connected-root \( r \ x \)
and point \( r \)
and \( x;r \neq 0 \)
shows \( x^T;r \neq 0 \)

\( \langle \text{proof} \rangle \)

**Rooted paths**

**lemma** path-iff-aux-1:
assumes bijective \( r \)
shows \( r;x \leq x^* + x^{T^*} \leftrightarrow x \leq r^T;(x^* + x^{T^*}) \)

\( \langle \text{proof} \rangle \)

**lemma** path-iff-aux-2:
assumes bijective 2-
shows \( r;x \leq x^* + x^{T^*} \leftrightarrow x^T \leq (x^* + x^{T^*});r \)

\( \langle \text{proof} \rangle \)

**lemma** path-iff-backward:
assumes is-inj \( x \)

36
and is-p-fun x
and point r
and r;x ≤ x* + xT*
shows connected x
⟨proof⟩

lemma empty-path-root-end:
assumes terminating-path-root-end r x e
shows e = r ↔ x = 0
⟨proof⟩

lemma path-root-acyclic:
assumes path-root r x
and x;r = 0
shows is-acyclic x
⟨proof⟩

Start points and end points

lemma start-points-in-root-aux:
assumes backward-finite-path-root r x
shows x;1 ≤ xT*;r
⟨proof⟩

lemma start-points-in-root:
assumes backward-finite-path-root r x
shows start-points x ≤ r
⟨proof⟩

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⟩

lemma start-points-not-zero:
assumes connected-root r x
and point r
and x ≠ 0
and x;r = 0
shows start-points x ≠ 0
⟨proof⟩

Backwards terminating and backwards finite

lemma backward-terminating-path-root-aux:
assumes backward-terminating-path-root r x
shows x ≤ xT*;−(xT*;1)
⟨proof⟩
lemma backward-finite-path-connected-aux:  
assumes backward-finite-path-root r x  
shows \( x^T \bowtie r \leq x^* + x^{T*} \)  
⟨proof⟩

lemma backward-finite-path-connected:  
assumes backward-finite-path-root r x  
shows connected x  
⟨proof⟩

lemma backward-finite-path-root-path:  
assumes backward-finite-path-root r x  
shows path x  
⟨proof⟩

lemma backward-finite-path-root-path-root:  
assumes backward-finite-path-root-root r x  
shows path-root r x  
⟨proof⟩

lemma zero-backward-terminating-path-root:  
assumes point r  
shows backward-terminating-path-root r 0  
⟨proof⟩

lemma backward-finite-path-root-move-root:  
assumes backward-finite-path-root-move-root r x  
and point q  
and \( q \leq x^* \bowtie r \)  
shows backward-finite-path-root q x  
⟨proof⟩

Cycle

lemma non-empty-cycle-root-var-axioms-1:  
non-empty-cycle-root r x \( \iff \) \( x^T \bowtie 1 \leq x^{T+} \bowtie r \land \text{is-inj } x \land \text{is-p-fun } x \land \text{point } r \land r \leq x^{T+} \bowtie 1 \)  
⟨proof⟩

lemma non-empty-cycle-root-loop:  
assumes non-empty-cycle-root-loop r x  
shows \( r \leq x^{T+} \bowtie r \)  
⟨proof⟩

lemma cycle-root-end-empty:  
assumes terminating-path-root-end r x e  
and many-strongly-connected x  
shows \( x = 0 \)  
⟨proof⟩
lemma cycle-root-end-empty-var:
  assumes terminating-path-root-end r x e
  and x ≠ 0
  shows ¬ many-strongly-connected x
  ⟨proof⟩

Terminating path

lemma terminating-path-root-end-connected:
  assumes terminating-path-root-end r x e
  shows x;1 ≤ x⁺;e
  ⟨proof⟩

lemma terminating-path-root-end-forward-finite:
  assumes terminating-path-root-end r x e
  shows backward-finite-path-root e (xᵀ)
  ⟨proof⟩

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 ≠ 0
  and point r
  and v ≤ xᵀ⁺;r
  shows r ≤ x⁺;v
  ⟨proof⟩

Roots

lemma root-in-start-points:
  assumes connected-root r x
  and is-vector r
  and x ≠ 0
  and x;r = 0
  shows r ≤ start-points x
  ⟨proof⟩

lemma root-equals-start-points:
  assumes backward-terminating-path-root r x
  and x ≠ 0
  shows r = start-points x
  ⟨proof⟩

end
lemma root-equals-end-points:
  assumes backward-terminating-path-root r (x^T)
  and x ≠ 0
  shows r = end-points x
⟨proof⟩

lemma root-in-edge-sources:
  assumes connected-root r x
  and x ≠ 0
  and is-vector r
  shows r ≤ x;1
⟨proof⟩

Rooted Paths
lemma non-empty-path-root-iff-aux:
  assumes path-root r x
  and x ≠ 0
  shows r ≤ (x + x^T);1
⟨proof⟩

Backwards terminating and backwards finite
lemma backward-terminating-path-root-2:
  assumes backward-terminating-path-root r x
  shows backward-terminating-path-root r x
⟨proof⟩

lemma backward-terminating-path-root:
  assumes backward-terminating-path-root r x
  shows backward-terminating-path x
⟨proof⟩

(Non-empty) Cycle
lemma cycle-iff:
  assumes point r
  shows x;r ≠ 0 ←→ r ≤ x^T;1
⟨proof⟩

lemma non-empty-cycle-root-iff:
  assumes connected-root r x
  and point r
  shows x;r ≠ 0 ←→ r ≤ x^T+r
⟨proof⟩

lemma backward-finite-path-root-terminating-or-cycle:
  backward-finite-path-root r x ←→ backward-terminating-path-root r x \lor
  non-empty-cycle-root r x
⟨proof⟩

lemma non-empty-cycle-root-msc:
assumes non-empty-cycle-root r x
shows many-strongly-connected x
⟨proof⟩

lemma non-empty-cycle-root-mse-cycle:
assumes non-empty-cycle-root r x
shows cycle x
⟨proof⟩

lemma non-empty-cycle-root-non-empty:
assumes non-empty-cycle-root r x
shows x ≠ 0
⟨proof⟩

lemma non-empty-cycle-root-rtc-symmetric:
assumes non-empty-cycle-root r x
shows x*;r = r x*;r
⟨proof⟩

lemma non-empty-cycle-root-point-exchange:
assumes non-empty-cycle-root r x
and point p
shows r ≤ x*;p ↔ p ≤ x*;r
⟨proof⟩

lemma non-empty-cycle-root-rtc-tc:
assumes non-empty-cycle-root r x
shows x*;r = x r*;r
⟨proof⟩

lemma non-empty-cycle-root-no-start-end-points:
assumes non-empty-cycle-root r x
shows x;1 = x r;1
⟨proof⟩

lemma non-empty-cycle-root-move-root:
assumes non-empty-cycle-root r x
and point q
and q ≤ x*;r
shows non-empty-cycle-root q x
⟨proof⟩

lemma non-empty-cycle-root-loop-converse:
assumes non-empty-cycle-root r x
shows r ≤ x r;1
⟨proof⟩

lemma non-empty-cycle-root-move-root-same-reachable:
assumes non-empty-cycle-root r x
and point q
and \( q \leq x^*;r \)
shows \( x^*:r = x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-move-root-same-reachable-2:
assumes non-empty-cycle-root \( r \) \( x \)
and point q
and \( q \leq x^*:r \)
shows \( x^*:r = x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-move-root-msc:
assumes non-empty-cycle-root \( r \) \( x \)
shows \( x^*:q = x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-move-root-rtc-tc:
assumes non-empty-cycle-root \( r \) \( x \)
and point q
and \( q \leq x^*:r \)
shows \( x^*:q = x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-move-root-loop-converse:
assumes non-empty-cycle-root \( r \) \( x \)
and point q
and \( q \leq x^*:r \)
shows \( q \leq x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-move-root-loop:
assumes non-empty-cycle-root \( r \) \( x \)
and point q
and \( q \leq x^*:r \)
shows \( q \leq x^*:q \)
⟨proof⟩

lemma non-empty-cycle-root-msc-plus:
assumes non-empty-cycle-root \( r \) \( x \)
shows \( x^+:r = x^{T+}:r \)
⟨proof⟩

lemma non-empty-cycle-root-rtc-start-points:
assumes non-empty-cycle-root \( r \) \( x \)
shows \( x^*:r = x:1 \)
⟨proof⟩

lemma non-empty-cycle-root-rtc-start-points:
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} x^*;r = x;1 \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-converse-start-end-points: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} x^T \leq x;1;x \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-start-end-points-plus: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} x;1;x \leq x^+ \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-converse-plus: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} x^+ \leq x^T \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-plus-converse: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} x^+ = x^{T+} \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-converse: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{shows} non-empty-cycle-root q \ (x^T) \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-move-root-forward: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{and} point q \\
\textbf{and} r \leq x^*;q \\
\textbf{shows} non-empty-cycle-root q \ x \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-move-root-forward-cycle: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{and} point q \\
\textbf{and} r \leq x^*;q \\
\textbf{shows} x;q \neq 0 \land x^T;q \neq 0 \\
\langle proof \rangle

\textbf{lemma} non-empty-cycle-root-equivalences: \\
\textbf{assumes} non-empty-cycle-root r \ x \\
\textbf{and} point q \\
\textbf{shows} (r \leq x^*;q \iff q \leq x^*;r) \\
\textbf{and} (r \leq x^*;q \iff x;q \neq 0) \\
\textbf{and} (r \leq x^*;q \iff x^T;q \neq 0)
and \((r \leq x^*; q \leftrightarrow q \leq x;1)\)
and \((r \leq x^*; q \leftrightarrow q \leq x^T;1)\)

\(\langle\text{proof}\rangle\)

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

\(\langle\text{proof}\rangle\)

**lemma** non-empty-cycle-root-var-axioms-2:
non-empty-cycle-root \(r x \iff x;1 \leq x^+;r \land \text{is-inj } x \land \text{is-p-fun } x \land \text{point } r \land r \leq x;1\)

\(\langle\text{proof}\rangle\)

**lemma** non-empty-cycle-root-var-axioms-3:
non-empty-cycle-root \(r x \iff x;1 \leq x^+;r \land \text{is-inj } x \land \text{is-p-fun } x \land \text{point } r \land r \leq x^+;x;1\)

\(\langle\text{proof}\rangle\)

**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 \(x = y\)

\(\langle\text{proof}\rangle\)

**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 \(x = y\)

\(\langle\text{proof}\rangle\)

**lemma** non-empty-cycle-root-equivalences-2:
assumes non-empty-cycle-root \(r x\)
shows \((v \leq x^*;r \leftrightarrow v \leq x^T;1)\)
and \((v \leq x^*;r \leftrightarrow v \leq x;1)\)

\(\langle\text{proof}\rangle\)

**lemma** cycle-root-non-empty:
assumes \(x \neq 0\)
shows cycle-root \(r x \iff \text{non-empty-cycle-root } r x\)

\(\langle\text{proof}\rangle\)

Start points and end points
lemma start-points-path-aux:
assumes backward-finite-path-root r x 
and start-points x ≠ 0 
s shows \( x; r = 0 \)
⟨proof⟩

lemma start-points-path:
assumes backward-finite-path-root r x 
and start-points x ≠ 0 
s shows backward-terminating-path-root r x
⟨proof⟩

lemma root-in-start-points-2:
assumes backward-finite-path-root r x 
and start-points x ≠ 0 
s shows \( r \leq \text{start-points } x \)
⟨proof⟩

lemma root-equals-start-points-2:
assumes backward-finite-path-root r x 
and start-points x ≠ 0 
s shows \( r = \text{start-points } x \)
⟨proof⟩

lemma start-points-injective:
assumes backward-finite-path-root r x 
s shows is-inj (start-points x)
⟨proof⟩

lemma backward-terminating-path-root-aux-2:
assumes backward-finite-path-root r x 
and start-points x ≠ 0 \lor x = 0 
s shows \( x \leq x^{T^*}; (-x^{T}; 1) \)
⟨proof⟩

lemma start-points-not-zero-iff:
assumes backward-finite-path-root r x 
s shows \( x; r = 0 \land x \neq 0 \iff \text{start-points } x \neq 0 \)
⟨proof⟩

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 
s shows \( x; r = 0 \)
⟨proof⟩

lemma backward-finite-path-root-acyclic-terminating-iff:
assumes backward-finite-path-root r x
shows is-acyclic $x \iff x; r = 0$
(proof)

lemma backward-finite-path-root-acyclic-terminating:
assumes backward-finite-path-root $r x$
and is-acyclic $x$
s shows backward-terminating-path-root $r x$
(proof)

lemma non-empty-cycle-root-one-strongly-connected:
assumes non-empty-cycle-root $r x$
s shows one-strongly-connected $x$
(proof)

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 $v$
s shows $r \leq x^*; v$
(proof)

lemma terminating-path-root-end-backward-terminating:
assumes terminating-path-root-end $r x e$
s shows backward-terminating-path-root $r x$
(proof)

lemma terminating-path-root-end-converse:
assumes terminating-path-root-end $r x e$
s shows terminating-path-root-end $e (x^T) r$
(proof)

lemma terminating-path-root-end-forward-terminating:
assumes terminating-path-root-end $r x e$
s shows backward-terminating-path-root $e (x^T)$
(proof)

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

lemma non-empty-path-root-iff:
$(\exists r. \text{path-root } r x \land r \leq (x + x^T); 1) \iff \text{path } x \land x \neq 0$
(Non-empty) Cycle

**Lemma** non-empty-cycle-root-iff:
\[(\exists r \cdot \text{non-empty-cycle-root } r \ x) \iff \text{cycle } x \land x \neq 0\]

**Lemma** non-empty-cycle-subset-equals:
\[
\text{assumes cycle } x \\
\text{and cycle } y \\
\text{and } x \leq y \\
\text{and } x \neq 0 \\
\text{shows } x = y
\]

**Lemma** cycle-root-iff:
\[(\exists r \cdot \text{cycle-root } r \ x) \iff \text{cycle } x\]

Backwards terminating and backwards finite

**Lemma** backward-terminating-path-root-iff:
\[(\exists r \cdot \text{backward-terminating-path-root } r \ x) \iff \text{backward-terminating-path } x\]

**Lemma** non-empty-backward-terminating-path-root-iff:
\[
\text{backward-terminating-path-root (start-points } x) \ x \iff \text{backward-terminating-path } x \land x \neq 0
\]

**Lemma** non-empty-backward-terminating-path-root-iff’:
\[
\text{backward-finite-path-root (start-points } x) \ x \iff \text{backward-terminating-path } x \land x \neq 0
\]

**Lemma** backward-finite-path-root-iff:
\[(\exists r \cdot \text{backward-finite-path-root } r \ x) \iff \text{backward-finite-path } x\]

**Lemma** non-empty-backward-finite-path-root-iff:
\[
(\exists r \cdot \text{backward-finite-path-root } r \ x \land r \leq x;1) \iff \text{backward-finite-path } x \land x \neq 0
\]

Terminating

**Lemma** terminating-path-root-end-aux:
\[
\text{assumes terminating-path } x \\
\text{shows } \exists r \ e \ . \ \text{terminating-path-root-end } r \ x \ e
\]
**Lemma** terminating-path-root-end-iff:

\((\exists r e. \text{terminating-path-root-end } r e) \iff \text{terminating-path } x\)  
(proof)

**Lemma** non-empty-terminating-path-root-end-iff:

\(\text{terminating-path-root-end } (\text{start-points } x) \land (\text{end-points } x) \iff \text{terminating-path } x \land x \neq 0\)  
(proof)

**Lemma** non-empty-finite-path-root-end-iff:

\(\text{finite-path-root-end } (\text{start-points } x) \land (\text{end-points } x) \iff \text{terminating-path } x \land x \neq 0\)  
(proof)

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**

\(\text{trunctl } ((. \to) \{1000\} 999)\)

**Class** choose-singleton-point-signature =

- fixes choose-singleton :: '\(a \Rightarrow \text{'}a\)'
- fixes choose-point :: '\(a \Rightarrow \text{'}a\)'

**Class** relation-algebra-rtc-tarski-choose-point =

relation-algebra-rtc-tarski + choose-singleton-point-signature +
- assumes choose-singleton-singleton: \(x \neq 0 \Rightarrow \text{singleton } (\text{choose-singleton } x)\)
- assumes choose-singleton-decreasing: \(\text{choose-singleton } x \leq x\)
- assumes choose-point-point: is-vector \(x \Rightarrow x \neq 0 \Rightarrow \text{point } (\text{choose-point } x)\)
- assumes choose-point-decreasing: \(\text{choose-point } x \leq x\)

**Begin**

**No-notation**

\(\text{composition } (\text{infixl } 75) \text{ and }\)
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^* \ast q \leq D^T \ast x \wedge W \leq D \wedge \text{terminating-path} W \wedge 
(W = 0 \iff q = y) \wedge (W \neq 0 \iff q = \text{start-points} W \wedge y = \text{end-points} W)

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

lemma construct-path-pre:
assumes is-acyclic D
and point y
and point x
and \( D^* \ast y \leq D^T \ast x \)
shows construct-path-inv y x y D 0
(proof)

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 \( \neq x \)
shows point q
and point (choose-point \((D \ast q)\))
(proof)

lemma path-inv-choose-point-decrease:
assumes construct-path-inv q x y D W \( \neq x \)
shows \( W \neq 0 \implies \text{choose-point} \((D \ast q)\) \leq -(W + \text{choose-point} \((D \ast q) \ast q^T\))^T \ast 1)
(proof)

lemma end-points:
assumes construct-path-inv q x y D W \( \neq x \)
shows \( \text{choose-point} \((D \ast q)\) = \text{start-points} \((W + \text{choose-point} \((D \ast q) \ast q^T\)) \)
and \( y = \text{end-points} \((W + \text{choose-point} \((D \ast q) \ast q^T\)) \)
(proof)

lemma construct-path-inv:
assumes construct-path-inv q x y D W \( \neq x \)
shows construct-path-inv (choose-point (D*q)) x y D (W + choose-point (D*q)*q^T)

(proof)

theorem construct-path-partial: VARS p q W
{ is-acyclic D ∧ point y ∧ point x ∧ D*y ≤ D*x }
W := 0;
q := y;
WHILE q ≠ 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 ≤ D ∧ terminating-path W ∧ (W=0 ↔ x=y) ∧ (W≠0 ↔ x = start-points W ∧ y = end-points W) }
(proof)

end

For termination, we additionally need finiteness.

context finite
begin

lemma decrease-set:
assumes ∀ x::'a . Q x → P x
and P w
and ¬ Q w
shows card { x . Q x } < card { x . P x }
(proof)

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 ≤ z
and w ≤ z
and ¬ w ≤ y
shows card { x . x ≤ y } < card { x . x ≤ z }
(proof)

lemma construct-path-inv-termination:
assumes construct-path-inv q x y D W ∧ q ≠ x
shows card { z . z ≤ -(W + choose-point (D*q)*q^T) } < card { z . z ≤ -W }

\begin{proof}

**Theorem** \textit{construct-path-total}: \texttt{VARS p q W}

\[\text{is-acyclic } D \land \text{point } y \land \text{point } x \land D^*y \leq D^*x \]

\[
\begin{array}{l}
\text{W} := 0; \\
q := y; \\
\text{WHILE } q \neq x \\
\text{INV} \{ \text{construct-path-inv } q \ x \ y \ D \ W \} \\
\text{VAR} \{ \text{card } z . z \leq -W \} \\
\text{DO } p := \text{choose-point } (D*q); \\
\text{\hspace{1cm} W} := W + p*q^T; \\
\text{\hspace{1cm} q} := p \\
\text{OD}
\end{array}
\]

\[\text{W} \leq D \land \text{terminating-path } W \land (W=0 \leftrightarrow x=y) \land (W\neq0 \leftrightarrow x = \text{start-points } W \land y = \text{end-points } W) \]

\end{proof}

\section{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.

\textbf{Context} \textit{relation-algebra-rtc-tarski-choose-point}

\begin{proof}

**Abbreviation** \textit{topological-sort-inv}

\texttt{where} \textit{topological-sort-inv } q \ v \ R \ W \equiv
\begin{align*}
\text{regressively-finite } R \land R \cdot v*v^T \leq W^+ \land \text{terminating-path } W \land W*1 = v \cdot q \land \\
(W = 0 \lor q = \text{end-points } W) \land \text{point } q \land R*v \leq v \land q \leq v \land \text{is-vector } v
\end{align*}

**Lemma** \textit{topological-sort-pre}:

\texttt{assumes} \text{regressively-finite } R

\texttt{shows} \textit{topological-sort-inv } (\text{choose-point } (\text{minimum } R \ 1)) (\text{choose-point } (\text{minimum } R \ 1)) \ R \ 0

\end{proof}

\begin{proof}

**Lemma** \textit{topological-sort-inv}:

\texttt{assumes} \(v \neq 1\) \and \textit{topological-sort-inv } q \ v \ R \ W

\texttt{shows} \textit{topological-sort-inv } (\text{choose-point } (\text{minimum } R \ (-v))) (v + \\
\text{choose-point } (\text{minimum } R \ (-v))) \ R \ (W + q \ast \text{choose-point } (\text{minimum } R \ (-v))^T)

\end{proof}
lemma topological-sort-post:
assumes \( \neg v \neq 1 \)
and topological-sort-inv \( q \ v \ R \ W \)
shows \( R \leq W^+ \land \text{terminating-path} \ W \land (W + W^T)^*1 = -1^*1 \)

(proof)

theorem topological-sort-partial: VARS \( p \ q \ v \ W \)
{ regressively-finite \( R \)}
\( W := 0; \)
\( q := \text{choose-point} \ (\text{minimum} \ R \ 1); \)
\( v := q; \)
WHILE \( v \neq 1 \)
INV { topological-sort-inv \( q \ v \ R \ W \) }
DO \( p := \text{choose-point} \ (\text{minimum} \ R \ (-v)); \)
\( W := W + q^*p^T; \)
\( q := p; \)
\( v := v + p \)
OD
{ \( R \leq W^+ \land \text{terminating-path} \ W \land (W + W^T)^*1 = -1^*1 \) }
(proof)

end

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

lemma topological-sort-inv-termination:
assumes \( v \neq 1 \)
and topological-sort-inv \( q \ v \ R \ W \)
shows \( \text{card} \ \{ z \ . \ z \leq -(v + \text{choose-point} \ (\text{minimum} \ R \ (-v))) \} < \text{card} \ \{ z \ . \ z \leq -v \} \)
(proof)

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 \( R \) }
\( W := 0; \)
\( q := \text{choose-point} \ (\text{minimum} \ R \ 1); \)
\( v := q; \)
WHILE \( v \neq 1 \)
INV { topological-sort-inv \( q \ v \ R \ W \) }
VAR { \text{card} \ \{ z \ . \ z \leq -v \} } 
DO \( p := \text{choose-point} \ (\text{minimum} \ R \ (-v)); \)
\( W := W + q^*p^T; \)
\( q := p; \)
\( v := v + p \)

52
\[ OD \]
\[
[ R \leq W^+ \land \text{terminating-path } W \land (W + W^T) \ast I = -1' \ast I ]
\]

\( \langle \text{proof} \rangle \)

## 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-tree-pre x y R \equiv y \leq R^T \ast x \land \text{point } x

abbreviation construct-tree-inv
  where construct-tree-inv v x y D R \equiv construct-tree-pre x y R \land \text{is-acyclic } D \land
  \text{is-inj } D \land
  D \leq R \land D \ast x = 0 \land v = x + D^T \ast 1 \land x \ast v^T \leq
  \text{is-vector } v

abbreviation construct-tree-post
  where construct-tree-post x y D R \equiv \text{is-acyclic } D \land \text{is-inj } D \land D \leq R \land D \ast x
  = 0 \land D^T \ast 1 \leq D^T \ast x \land
  D^* \ast y \leq D^T \ast x

lemma construct-tree-pre:
  assumes construct-tree-pre x y R
  shows construct-tree-inv x x y 0 R
  \( \langle \text{proof} \rangle \)

lemma construct-tree-inv-aux:
  assumes \neg y \leq v
  and construct-tree-inv v x y D R
  shows singleton \( \text{choose-singleton} (v \ast v^T \cdot R) \)
  \( \langle \text{proof} \rangle \)

lemma construct-tree-inv:
  assumes \neg y \leq v
  and construct-tree-inv v x y D R
  shows construct-tree-inv \( v + \text{choose-singleton} (v \ast v^T \cdot R)^T \ast 1 \) x y \( D + \text{choose-singleton} (v \ast v^T \cdot R) \) R
  \( \langle \text{proof} \rangle \)

lemma construct-tree-post:
  assumes y \leq v
  and construct-tree-inv v x y D R

53
shows construct-tree-post x y D R
(proof)

theorem construct-tree-partial: 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 }
  DO e := choose-singleton (v+−vT · R);
      D := D + e;
      v := v + eT*1
  OD
{ construct-tree-post x y D R }
(proof)
end

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

lemma construct-tree-inv-termination:
assumes ¬ y ≤ v
and construct-tree-inv v x y D R
shows card { z . z ≤ −(v + choose-singleton (v+−vT · R)T*1) } < card { z . z ≤ −v }
(proof)

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+−vT · R);
      D := D + e;
      v := v + eT*1
  OD
[ construct-tree-post x y D R ]
(proof)
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 ∧ point y ∧ y*xT ≤ R

lemma construct-cycle-pre:
assumes ¬ is-acyclic R
and y = choose-point ((R⁺ · 1')*1)
and x = choose-point (R*sy · RT*ty)
shows construct-cycle-inv x x y 0 R
⟨proof⟩

lemma construct-cycle-pre2:
assumes y ≤ v
and construct-cycle-inv v x y D R
shows construct-path-inv y x y D 0 ∧ D ≤ R ∧ D * x = 0 ∧ y * xT ≤ R
⟨proof⟩

lemma construct-cycle-post:
assumes ¬ q ≠ x
and (construct-path-inv q x y D W ∧ D ≤ R ∧ D * x = 0 ∧ y * xT ≤ R)
sows W + y * xT ≠ 0 ∧ W + y * xT ≤ R ∧ cycle (W + y * xT)
⟨proof⟩

theorem construct-cycle-partial: VARS e p q v x y C D W
{¬ is-acyclic R }
y := choose-point ((R⁺ · 1')*1);
x := choose-point (R*sy · RT*ty);
D := 0;
v := x;
WHILE ¬ y ≤ v
INV { construct-cycle-inv v x y D R }
DO e := choose-singleton (v*−vT · R);
D := D + e;
v := v + eT*1
OD;
comment { is-acyclic D ∧ point y ∧ point x ∧ D*sy ≤ D*T*x }; W := 0;
q := y;
WHILE q ≠ x
INV { construct-path-inv q x y D W ∧ D ≤ R ∧ D*x = 0 ∧ y*xT ≤ R }
DO p := choose-point (D*q);
W := W + p*qT;
q := p
OD;
comment { W ≤ D ∧ terminating-path W ∧ (W = 0 † q=y) ∧ (W ≠ 0

55
\[ q = \text{start-points } W \land y = \text{end-points } W \}\);
\[
C := W + y*x^T
\]  
\{ C \neq 0 \land C \leq R \land \text{cycle } C \}
\langle \text{proof} \rangle
end

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

\textbf{theorem} \textit{construct-cycle-total}: \textbf{VARS} e p q v x y C D W  
\[
\neg \text{is-acyclic } R
\]
\[
y := \text{choose-point } ((R^+ \cdot I)^*1);
\]
\[
x := \text{choose-point } (R^*y \cdot R^T*y);
\]
D := 0;
\[
v := x;
\]
\textbf{WHILE} \neg y \leq v
\[
\textbf{INV} \{ \text{construct-cycle-inv } v x y D R \}
\]
\[
\textbf{VAR} \{ \text{card } \{ z \mid z \leq -v \} \}
\]
\[
\textbf{DO} e := \text{choose-singleton } (v \cdot v^T \cdot R);
\]
\[
D := D + e;
\]
\[
v := v + e^T*1
\]
\textbf{OD};
\[
\textbf{comment} \{ \text{is-acyclic } D \land \text{point } y \land \text{point } x \land D^*y \leq D^T*x \};
\]
\[
W := 0;
\]
q := y;
\textbf{WHILE} q \neq x
\[
\textbf{INV} \{ \text{construct-path-inv } q x y D W \land D \leq R \land D*x = 0 \land y*x^T \leq R \}
\]
\[
\textbf{VAR} \{ \text{card } \{ z \mid z \leq -W \} \}
\]
\[
\textbf{DO} p := \text{choose-point } (D*q);
\]
\[
W := W + p*q^T;
\]
\[
q := p
\]
\textbf{OD};
\[
\textbf{comment} \{ W \leq D \land \text{terminating-path } W \land (W = 0 \iff q=y) \land (W \neq 0 \iff q = \text{start-points } W \land y = \text{end-points } W)}\};
\]
\[
C := W + y*x^T
\]  
\{ C \neq 0 \land C \leq R \land \text{cycle } C \}
\langle \text{proof} \rangle
end
\textbf{end}

\textbf{References}

\textit{Archive of Formal Proofs}, 2014.


