Abstract

We develop Stone-Kleene relation algebras, which expand Stone relation algebras with a Kleene star operation to describe reachability in weighted graphs. Many properties of the Kleene star arise as a special case of a more general theory of iteration based on Conway semirings extended by simulation axioms. This includes several theorems representing complex program transformations. We formally prove the correctness of Conway’s automata-based construction of the Kleene star of a matrix. We prove numerous results useful for reasoning about weighted graphs.
1 Synopsis and Motivation

This document describes the following five theory files:

* Iterings describes a general iteration operation that works for many different computation models. We first consider equational axioms based on variants of Conway semirings. We expand these structures by generalised simulation axioms, which hold in total and general correctness models, not just in partial correctness models like the induction axioms. Simulation axioms are still powerful enough to prove separation theorems and Back’s atomicity refinement theorem [4].

* Kleene Algebras form a particular instance of iterings in which the iteration is implemented as a least fixpoint. We implement them based on Kozen’s axioms [13], but most results are inherited from Conway semirings and iterings.

* Kleene Relation Algebras introduces Stone-Kleene relation algebras, which combine Stone relation algebras and Kleene algebras. This is similar to relation algebras with transitive closure [16] but allows us to talk about reachability in weighted graphs. Many results in this theory are useful for verifying the correctness of Prim’s minimum spanning tree algorithm.

* Subalgebras of Kleene Relation Algebras studies the regular elements of a Stone-Kleene relation algebra and shows that they form a Kleene relation subalgebra.

* Matrix Kleene Algebras lifts the Kleene star to finite square matrices using Conway’s automata-based construction. This involves an operation to restrict matrices to specific indices and a calculus for such restrictions. An implementation for the Kleene star of matrices was given in [3] without proof; this is the first formally verified correctness proof.

The development is based on a theory of Stone relation algebras [11, 12]. We apply Stone-Kleene relation algebras to verify Prim’s minimum spanning tree algorithm in Isabelle/HOL in [10].

Related libraries for Kleene algebras, regular algebras and relation algebras in the Archive of Formal Proofs are [1, 2, 8]. Kleene algebras are covered in the theory Kleene_Algebra/Kleene_Algebra.thy, but unlike the present development it is not based on general algebras using simulation axioms, which are useful to describe various computation models. The theory Regular_Algebras/Regular_Algebras.thy compares different axiomatisations of regular algebras. The theory Kleene_Algebra/Matrix.thy covers matrices over dioids, but does not implement the Kleene star of matrices. The theory Relation_Algebra/Relation_Algebra_RTC.thy combines
Kleene algebras and relation algebras, but is very limited in scope and not applicable as we need the weaker axioms of Stone relation algebras.

2 Iterings

This theory introduces algebraic structures with an operation that describes iteration in various relational computation models. An iteration describes the repeated sequential execution of a computation. This is typically modelled by fixpoints, but different computation models use different fixpoints in the refinement order. We therefore look at equational and simulation axioms rather than induction axioms. Our development is based on [9] and the proposed algebras generalise Kleene algebras.

We first consider a variant of Conway semirings [5] based on idempotent left semirings. Conway semirings expand semirings by an iteration operation satisfying Conway’s sumstar and productstar axioms [7]. Many properties of iteration follow already from these equational axioms.

Next we introduce iterings, which use generalised versions of simulation axioms in addition to sumstar and productstar. Unlike the induction axioms of the Kleene star, which hold only in partial-correctness models, the simulation axioms are also valid in total and general correctness models. They are still powerful enough to prove the correctness of complex results such as separation theorems of [6] and Back’s atomicity refinement theorem [4, 17].
lemma circ-mult-sub:
1 ⊔ x * (y * x)° * y ≤ (x * y)°
by (metis sup-right-isotone circ-left-slide circ-left-unfold mult-assoc mult-right-isotone)

lemma circ-right-unfold-sub:
1 ⊔ x° * x ≤ x°
by (metis circ-mult-sub mult-1-left mult-1-right)

lemma circ-zero:
bot° = 1
by (metis sup-monoid.add-0-right circ-left-unfold mult-left-zero)

lemma circ-increasing:
x ≤ x°
by (metis le-supI2 circ-left-unfold circ-right-unfold-sub mult-1-left mult-right-sub-dist-sup-left order-trans)

lemma circ-reflexive:
1 ≤ x°
by (metis sup-left-divisibility circ-left-unfold)

lemma circ-mult-increasing:
x ≤ x° * x°
by (metis circ-reflexive mult-right-isotone mult-1-right)

lemma circ-mult-increasing-2:
x ≤ x° * x°
by (metis circ-reflexive mult-left-isotone mult-1-left)

lemma circ-transitive-equal:
x° * x° = x°
by (metis sup-idem circ-sup-1 circ-left-unfold mult-assoc)

While iteration is not idempotent, a fixpoint is reached after applying this operation twice. Iteration is idempotent for the unit.

lemma circ-circ-circ:
x°°°° = x°°
by (metis sup-idem circ-sup-1 circ-increasing circ-transitive-equal le-iiff-sup)

lemma circ-one:
1° = 1°°
by (metis circ-circ-circ circ-zero)

lemma circ-sup-sub:
(x° * y°)° * x° ≤ (x ⊔ y)°
by (metis circ-sup-1 circ-left-slide)

lemma circ-plus-one:
\[ x^\circ = 1 \sqcup x^\circ \]

by (metis le-iff-sup circ-reflexive)

Iteration satisfies a characteristic property of reflexive transitive closures.

**Lemma** \texttt{circ-rtc-2}:
\[ 1 \sqcup x \sqcup x^\circ \circ x^\circ = x^\circ \]
by (metis sup-assoc circ-increasing circ-plus-one circ-transitive-equal le-iff-sup)

**Lemma** \texttt{mult-zero-circ}:
\[ (x * \text{bot})^\circ = 1 \sqcup x * \text{bot} \]
by (metis circ-left-unfold mult-assoc mult-left-zero)

**Lemma** \texttt{mult-zero-sup-circ}:
\[ (x \sqcup y * \text{bot})^\circ = x^\circ \circ (y * \text{bot})^\circ \]
by (metis circ-sup-1 mult-assoc mult-left-zero)

**Lemma** \texttt{circ-plus-sub}:
\[ x^\circ \circ x \leq x \circ x^\circ \]
by (metis circ-left-slide mult-1-left mult-1-right)

**Lemma** \texttt{circ-loop-fixpoint}:
\[ y \circ (y^\circ \circ z) \sqcup z = y^\circ \circ z \]
by (metis sup-commute circ-left-unfold mult-assoc mult-1-left mult-right-dist-sup)

**Lemma** \texttt{left-plus-below-circ}:
\[ x \circ x^\circ \leq x^\circ \]
by (metis sup.cobounded2 circ-left-unfold)

**Lemma** \texttt{right-plus-below-circ}:
\[ x^\circ \circ x \leq x^\circ \]
using circ-right-unfold-sub by auto

**Lemma** \texttt{circ-sup-upper-bound}:
\[ x \leq z^\circ \implies y \leq z^\circ \implies x \sqcup y \leq z^\circ \]
by simp

**Lemma** \texttt{circ-mult-upper-bound}:
\[ x \leq z^\circ \implies y \leq z^\circ \implies x \circ y \leq z^\circ \]
by (metis mult-isotone circ-transitive-equal)

**Lemma** \texttt{circ-sub-dist}:
\[ x^\circ \leq (x \sqcup y)^\circ \]
by (metis circ-sup-sub circ-plus-one mult-1-left mult-right-sub-dist-sup-left order-trans)

**Lemma** \texttt{circ-sub-dist-1}:
\[ x \leq (x \sqcup y)^\circ \]
using circ-increasing le-supE by blast
lemma circ-sub-dist-2:
\[ x \ast y \leq (x \sqcup y)^\circ \]
by (metis sup-commute circ-mult-upper-bound circ-sub-dist-1)

lemma circ-sub-dist-3:
\[ x^\circ \ast y^\circ \leq (x \sqcup y)^\circ \]
by (metis sup-commute circ-mult-upper-bound circ-sub-dist)

lemma circ-isotone:
\[ x \leq y \implies x^\circ \leq y^\circ \]
by (metis circ-sub-dist le-iff-sup)

lemma circ-sup-2:
\[ (x \sqcup y)^\circ \leq (x^\circ \ast y^\circ)^\circ \]
by (metis sup bounded-iff circ-increasing circ-isotone circ-reflexive mult-isotone mult-1-left mult-1-right)

lemma circ-sup-one-left-unfold:
\[ 1 \leq x \implies x \ast x^\circ = x^\circ \]
by (metis antisym le-iff-sup mult-1-left mult-right-sub-dist-sup-left left-plus-below-circ)

lemma circ-sup-one-right-unfold:
\[ 1 \leq x \implies x^\circ \ast x = x^\circ \]
by (metis antisym le-iff-sup mult-left-sub-dist-sup-left mult-1-right right-plus-below-circ)

lemma circ-decompose-4:
\[ (x^\circ \ast y^\circ)^\circ = x^\circ \ast (y^\circ \ast x^\circ)^\circ \]
by (metis sup-assoc sup-commute circ-sup-1 circ-loop-fixpoint circ-plus-one circ-rtc-2 circ-transitive-equal mult-assoc)

lemma circ-decompose-5:
\[ (x^\circ \ast y^\circ)^\circ = (y^\circ \ast x^\circ)^\circ \]
by (metis circ-decompose-4 circ-loop-fixpoint antisym mult-right-sub-dist-sup-right mult-assoc)

lemma circ-decompose-6:
\[ x^\circ \ast (y \ast x^\circ)^\circ = y^\circ \ast (x \ast y^\circ)^\circ \]
by (metis sup-commute circ-sup-1)

lemma circ-decompose-7:
\[ (x \sqcup y)^\circ = x^\circ \ast y^\circ \ast (x \sqcup y)^\circ \]
by (metis circ-sup-1 circ-decompose-6 circ-transitive-equal mult-assoc)

lemma circ-decompose-8:
\[ (x \sqcup y)^\circ = (x \sqcup y)^\circ \ast x^\circ \ast y^\circ \]
by (metis antisym eq-refl mult-assoc mult-isotone mult-1-right)
lemma circ-decompose-9:
\[(x^o \ast y^o)^o = x^o \ast y^o \ast (x^o \ast y^o)^o\]
by (metis circ-decompose-4 mult-assoc)

lemma circ-decompose-10:
\[(x^o \ast y^o)^o = (x^o \ast y^o)^o \ast x^o \ast y^o\]
by (metis sup-ge2 circ-loop-fixpoint circ-reflexive circ-sup-one-right-unfold mult-assoc order-trans)

lemma circ-back-loop-prefixpoint:
\[(z \ast y^o) \ast y \sqcup z \leq z \ast y^o\]
by (metis sup bounded-iff circ-left-unfold mult-assoc mult-left-sub-dist-sup-left mult-right-isotone mult-1-right right-plus-below-circ)

We obtain the fixpoint and prefixpoint properties of iteration, but not least or greatest fixpoint properties.

lemma circ-loop-is-fixpoint:
\[\text{is-fixpoint} (\lambda x . y \ast x \sqcup z) (y^o \ast z)\]
by (metis circ-loop-fixpoint is-fixpoint-def)

lemma circ-back-loop-is-prefixpoint:
\[\text{is-prefixpoint} (\lambda x . x \ast y \sqcup z) (z \ast y^o)\]
by (metis circ-back-loop-prefixpoint is-prefixpoint-def)

lemma circ-circ-sup:
\[(1 \sqcup x)^o = x^{oo}\]
by (metis sup-commute circ-sup-1 circ-decompose-4 circ-zero mult-1-right)

lemma circ-circ-mult-sub:
\[x^o \ast f^o \leq x^{oo}\]
by (metis circ-incresing circ-isotone circ-mult-upper-bound circ-reflexive)

lemma left-plus-circ:
\[(x \ast x^o)^o = x^o\]
by (metis circ-left-unfold circ-sup-1 mult-1-right mult-sub-right-one sup.absorb1 mult-assoc)

lemma right-plus-circ:
\[(x^o \ast x)^o = x^o\]
by (metis sup-commute circ-isotone circ-loop-fixpoint circ-plus-sub circ-sub-dist eq-iff left-plus-circ)

lemma circ-square:
\[(x \ast x)^o \leq x^o\]
by (metis circ-increasing circ-isotone left-plus-circ mult-right-isotone)

lemma circ-mult-sub-sup:
lemma circ-sup-mult-zero:
\(x^o \cdot y = (x \sqcup y \cdot \text{bot})^o \cdot y\)

proof

have \((x \sqcup y \cdot \text{bot})^o \cdot y = x^o \cdot (1 \sqcup y \cdot \text{bot}) \cdot y\)

by (metis mult-zero-sup-circ mult-zero-circ)

also have \(\ldots = x^o \cdot (y \sqcup y \cdot \text{bot})\)

by (metis mult-assoc mult-1-left mult-left-zero mult-right-dist-sup)

also have \(\ldots = x^o \cdot y\)

by (metis sup-commute le-iff-sup zero-right-mult-decreasing)

finally show \(?thesis\)

by simp

qed

lemma troeger-1:
\((x \sqcup y)^o = x^o \cdot (1 \sqcup y \cdot (x \sqcup y)^o)\)

by (metis circ-sup-1 circ-left-unfold mult-assoc)

lemma troeger-2:
\((x \sqcup y)^o \cdot z = x^o \cdot (y \cdot (x \sqcup y)^o \cdot z \sqcup z)\)

by (metis circ-sup-1 circ-loop-fixpoint mult-assoc)

lemma troeger-3:
\((x \sqcup y \cdot \text{bot})^o = x^o \cdot (1 \sqcup y \cdot \text{bot})\)

by (metis mult-zero-sup-circ mult-zero-circ)

lemma circ-sup-sub-sup-one-1:
\(x \sqcup y \leq x^o \cdot (1 \sqcup y)\)

by (metis circ-increasing circ-left-unfold mult-1-left mult-left-sub-dist-sup mult-right-sub-dist-sup-left order-trans sup-mono)

lemma circ-sup-sub-sup-one-2:
\(x \sqcup y \leq x^o \cdot (x \sqcup y)\)

by (metis circ-sup-sub-sup-one-1 circ-transitive-equal mult-assoc mult-right-isotone)

lemma circ-sup-sub-sup-one:
\(x \cdot x^o \cdot (x \sqcup y) \leq x \cdot x^o \cdot (1 \sqcup y)\)

by (metis circ-sup-sub-sup-one-2 mult-assoc mult-right-isotone)

lemma circ-square-2:
\((x \cdot x)^o \cdot (x \sqcup 1) \leq x^o\)

by (metis sup.bounded-iff circ-increasing circ-mult-upper-bound circ-reflexive circ-square)

lemma circ-extra-circ:
\((y \cdot x^o)^o = (y \cdot x^o \cdot x^o)^o\)
by (metis circ-decompose-6 circ-transitive-equal left-plus-circ mult-assoc)

**Lemma** circ-circ-sub-mult:
\[1^\circ \ast x^\circ \leq x^\circ\]
by (metis circ-increasing circ-isotone circ-mult-upper-bound circ-reflexive)

**Lemma** circ-decompose-11:
\[(x^\circ \ast y^\circ)^\circ = (x^\circ \ast y^\circ)^\circ \ast x^\circ\]
by (metis circ-decompose-10 circ-decompose-4 circ-decompose-5 circ-decompose-9 left-plus-circ)

**Lemma** circ-mult-below-circ-circ:
\[(x \ast y)^\circ \leq (x^\circ \ast y^\circ)^\circ \ast x^\circ\]
by (metis circ-increasing circ-isotone circ-reflexive dual-order.trans mult-left-isotone mult-right-isotone mult-1-right)

end

The next class considers the interaction of iteration with a greatest element.

**Class** bounded-left-conway-semiring = bounded-idempotent-left-semiring + left-conway-semiring
begin

**Lemma** circ-top:
\[\top^\circ = \top\]
by (simp add: antisym circ-increasing)

**Lemma** circ-right-top:
\[x^\circ \ast \top = \top\]
by (metis sup-right-top circ-loop-fixpoint)

**Lemma** circ-left-top:
\[\top \ast x^\circ = \top\]
by (metis circ-right-top circ-top circ-decompose-11)

**Lemma** mult-top-circ:
\[(x \ast \top)^\circ = 1 \sqcup x \ast \top\]
by (metis circ-left-top circ-left-unfold mult-assoc)

end

**Class** left-zero-conway-semiring = idempotent-left-zero-semiring + left-conway-semiring
begin

**Lemma** mult-zero-sup-circ-2:
(x ⊔ y * bot)° = x° ⊔ x° * y * bot
by (metis mult-assoc mult-left-dist-sup mult-1-right troeger-3)

lemma circ-unfold-sum:
(x ⊔ y)° = x° ⊔ x° * y * (x ⊔ y)°
by (metis mult-assoc mult-left-dist-sup mult-1-right troeger-1)
end

The next class assumes the full sliding equation.

class left-conway-semiring-1 = left-conway-semiring +
assumes circ-right-slide: x * (y * x)° ≤ (x * y)° * x
begin

lemma circ-slide-1:
x * (y * x)° = (x * y)° * x
by (metis antisym circ-left-slide circ-right-slide)

This implies the full unfold rules and Conway’s productstar.

lemma circ-right-unfold-1:
1 ⊔ x° * x = x°
by (metis circ-left-unfold circ-slide-1 mult-1-left mult-1-right)

lemma circ-mult-1:
(x * y)° = 1 ⊔ x * (y * x)° * y
by (metis circ-left-unfold circ-slide-1 mult-assoc)

lemma circ-sup-9:
(x ⊔ y)° = (x° * y)° * x°
by (metis circ-sup-1 circ-slide-1)

lemma circ-plus-same:
x° * x = x * x°
by (metis circ-slide-1 mult-1-left mult-1-right)

lemma circ-decompose-12:
x° * y° ≤ (x° * y)° * x°
by (metis circ-sup-9 circ-sub-dist-3)
end

class left-zero-conway-semiring-1 = left-zero-conway-semiring +
left-conway-semiring-1
begin

lemma circ-back-loop-fixpoint:
(z * y)° * y ⊔ z = z * y°
by (metis sup-commute circ-left-unfold circ-plus-same mult-associ mult-left-dist-sup mult-1-right)
lemma circ-back-loop-is-fixpoint:
  is-fixpoint (λx . x * y ⊔ z) (z * y°)
by (metis circ-back-loop-fixpoint is-fixpoint-def)

lemma circ-elimination:
  x * y = bot → x * y° ≤ x
by (metis sup-monoid.add-0-left circ-back-loop-fixpoint circ-plus-same
    mult-assoc mult-left-zero order-refl)

end

2.2 Iterings

This section adds simulation axioms to Conway semirings. We consider several classes with increasingly general simulation axioms.

class itering-1 = left-conway-semiring-1 +
  assumes circ-simulate: z * x ≤ y * z → z * x° ≤ y° * z
begin

lemma circ-circ-mult:
  1° * x° = x°
by (metis antisym circ-circ-sup circ-reflexive circ-simulate circ-sub-dist-3
    circ-sup-one-left-unfold circ-transitive-equal mult-1-left order-refl)

lemma sub-mult-one-circ:
  x * 1° ≤ 1° * x
by (metis circ-simulate mult-1-left mult-1-right order-refl)

  The left simulation axioms is enough to prove a basic import property
  of tests.

lemma circ-import:
  assumes p ≤ p * p
  and p ≤ 1
  and p * x ≤ x * p
  shows p * x° = p * (p * x)°
proof –
  have p * x ≤ p * (p * x * p) * p
    by (metis assms coreflexive-transitive eq-iff test-preserves-equation mult-assoc)
  hence p * x° ≤ p * (p * x)°
    by (metis (no-types) assms circ-simulate circ-slide-1 test-preserves-equation)
  thus ?thesis
    by (metis assms(2) circ-isotone mult-left-isotone mult-1-left mult-right-isotone antisym)
qed

end
Including generalisations of both simulation axioms allows us to prove separation rules.

class itering-2 = left-conway-semiring-1 +
assumes circ-simulate-right: \( z \ast x \leq y \ast z \sqcup w \implies z \ast x^o \leq y^o \ast (z \sqcup w \ast x^o) \)
assumes circ-simulate-left: \( x \ast z \leq z \ast y \sqcup w \implies x^o \ast z \leq (z \sqcup x^o \ast w) \ast y^o \)
begin
subclass itering-1
apply unfold-locales
by (metis sup-monoid.add-0-right circ-simulate-right mult-left-zero)

lemma circ-simulate-left-1:
assumes \( y \ast x \leq x \ast y \)
shows \( (x \sqcup y)^o = x^o \ast y^o \)
proof –
have \( y^o \ast x \leq x \ast y^o \sqcup y^o \ast bot \)
by (metis assms circ-simulate-left-1)
hence \( y^o \ast x \ast y^o \leq x \ast y^o \sqcup y^o \ast bot \ast y^o \)
by (metis mult-assoc mult-left-isotone mult-right-dist-sup)
also have \( ... = x \ast y^o \sqcup y^o \ast bot \)
by (metis circ-transitive-equal mult-assoc mult-left-zero)
finally have \( y^o \ast (x \ast y^o)^o \leq x^o \ast (y^o \sqcup y^o \ast bot) \)
using circ-simulate-right mult-assoc by fastforce
also have \( ... = x^o \ast y^o \)
by (simp add: sup-absorb1 zero-right-mult-decreasing)
finally have \( (x \sqcup y)^o \leq x^o \ast y^o \)
by (simp add: circ-decompose-6 circ-sup-1)
thus \( ?thesis \)
by (simp add: antisym circ-sub-dist-3)
qed

lemma circ-circ-mult-1:
\( x^o \ast 1^o = x^o \)
by (metis sup-commute circ-circ-sup circ-separate-1 mult-1-left mult-1-right order-refl)
end

With distributivity, we also get Back’s atomicity refinement theorem.

class itering-3 = itering-2 + left-zero-conway-semiring-1
begin
lemma circ-simulate-1:
assumes \( y \ast x \leq x \ast y \)

shows $y^o * x^o \leq x^o * y^o$

proof –
have $y * x^o \leq x^o * y$
  by (metis assms circ-simulate)

hence $y^o * x^o \leq x^o * y^o \sqcup y^o * \bot$
  by (metis circ-simulate-left-1)

thus \( \exists \)thesis
  by (metis sup-assoc sup-monoid.add-0-right circ-loop-fixpoint mult-assoc
      mult-left-zero mult-zero-sup-circ-2)
qed

lemma atomicity-refinement:
assumes $s = s * q$
  and $x = q * x$
  and $q * b = \bot$
  and $r * b \leq b * r$
  and $r * l \leq l * r$
  and $x * l \leq l * x$
  and $b * l \\leq l * b$
  and $q * l \leq l * q$
  and $r^o * q \leq q * r^o$
  and $q \leq l$

shows $s * (x \sqcup b \sqcup r \sqcup l)^o * q \leq s * (x * b^o * q \sqcup r \sqcup l)^o$

proof –
have $(x \sqcup b \sqcup r) * l \leq l * (x \sqcup b \sqcup r)$
  using assms(5–7) mult-left-dist-sup mult-right-dist-sup semiring.add-mono
by presburger
hence $s * (x \sqcup b \sqcup r \sqcup l)^o * q = s * l^o * (x \sqcup b \sqcup r)^o * q$
  by (metis sup-commute circ-separate-1 mult-assoc)
also have $\ldots = s * l^o * b^o * r^o * q * (x * b^o * r^o * q)^o$

proof –
have $(b \sqcup r)^o = b^o * r^o$
  by (simp add: assms(4) circ-separate-1)

hence $b^o * r^o * (q * (x * b^o * r^o))^o = (x \sqcup b \sqcup r)^o$
  by (metis (full-types) assms(2) circ-sup-1 sup-assoc sup-commute
      mult-assoc)
thus \( \exists \)thesis
  by (metis circ-slide-1 mult-assoc)

qed

also have $\ldots \leq s * l^o * b^o * r^o * q * (x * b^o * q * r^o)^o$
  by (metis assms(9) circ-isotone mult-assoc mult-right-isotone)

also have $\ldots \leq s * g * l^o * b^o * r^o * (x * b^o * q * r^o)^o$
  by (metis assms(1,10) mult-left-isotone mult-right-isotone mult-1-right)

also have $\ldots \leq s * l^o * q * b^o * r^o * (x * b^o * q * r^o)^o$
  by (metis assms(1,8) circ-simulate mult-assoc mult-left-isotone
      mult-right-isotone)

also have $\ldots \leq s * l^o * r^o * (x * b^o * q * r^o)^o$
  by (metis assms(3,10) sup-monoid.add-0-left circ-back-loop-fixpoint
      circ-plus-same mult-assoc mult-left-zero mult-left-isotone mult-right-isotone
      mult-right-isotone
also have \[ s \leq (x \ast b^\circ \ast q \sqcup r \sqcup l)^\circ \]
by (metis sup-commute circ-sup-1 circ-sub-dist-3 mult-assoc
mult-right-isotone)
finally show \(?thesis

qed
end

The following class contains the most general simulation axioms we consider. They allow us to prove further separation properties.

class \textit{itering} = idempotent-left-zero-semiring + circ +
assumes \textit{circ-sup}: \((x \sqcup y)^\circ = (x^\circ \ast y)^\circ \ast x^\circ\)
assumes \textit{circ-mult}: \((x \ast y)^\circ = I \sqcup x \ast (y \ast x)^\circ \ast y\)
assumes \textit{circ-simulate-right-plus}: \(z \ast x \leq y \ast y^\circ \ast z \sqcup w \rightarrow z \ast x^\circ \leq y^\circ \ast (z \sqcup w) \ast y^\circ\)
assumes \textit{circ-simulate-left-plus}: \(x \ast z \leq z \ast y^\circ \sqcup w \rightarrow x^\circ \ast z \leq (z \sqcup x^\circ \ast w) \ast y^\circ\)
begin

lemma \textit{circ-right-unfold}:
\(I \sqcup x^\circ \ast x = x^\circ\)
by (metis circ-mult mult-1-left mult-1-right)

lemma \textit{circ-slide}:
\(x \ast (y \ast x)^\circ = (x \ast y)^\circ \ast x\)
proof –
have \(x \ast (y \ast x)^\circ = Rf x \ (y \ast I \sqcup y \ast (x \ast (y \ast x)^\circ \ast y)) \ast x\)
by (metis (no-types) circ-mult mult-1-left mult-1-right mult-left-zero mult-right-isotone mult-right-dist-sup mult-1-right mult-1-left-dist-sup mult-assoc)
thus \(?thesis
by (metis (no-types) circ-mult mult-1-left mult-left-dist-sup mult-assoc)
qed

subclass \textit{itering-3}
apply unfold-locale
apply (metis circ-mult mult-1-left mult-1-right)
apply (metis circ-slide order-refl)
apply (metis circ-sup circ-slide)
apply (metis circ-slide order-refl)
apply (metis sup-left-isotone circ-right-unfold mult-left-isotone
mult-left-sub-dist-sup-left mult-1-right order-trans circ-simulate-right-plus
mult-1-right order-trans circ-simulate-left-plus)

lemma \textit{circ-simulate-right-plus-1}:
\(z \ast x \leq y \ast y^\circ \ast z \Rightarrow z \ast x^\circ \leq y^\circ \ast z\)
by (metis sup-monoid.add-0-right circ-simulate-right-plus mult-left-zero)

14
lemma circ-simulate-left-plus-1:
\( x \cdot z \leq z \cdot y \Rightarrow x \cdot z \leq z \cdot y \sqcup x \cdot \bot \)
by (metis sup-monoid.add-0-right circ-simulate-left-plus mult-assoc
mult-left-zero mult-right-dist-sup)

lemma circ-simulate-2:
\( y \cdot x \sqsubseteq x \cdot y \iff y \cdot x \sqsubseteq x \cdot y \sqcup y \cdot \bot \)
apply (rule iffI)
apply (metis sup-assoc sup-monoid.add-0-right circ-loop-fixpoint
circ-simulate-left-plus-1 mult-assoc mult-left-zero mult-zero-sup-circ-2)
by (metis circ-increasing mult-left-isotone order-trans)

lemma circ-simulate-absorb:
\( y \cdot x \leq x \Rightarrow y \cdot x \leq x \sqcup y \cdot \bot \)
by (metis assms circ-simulate-left-plus-1 circ-zero mult-1-right)

lemma circ-separate-mult-1:
\( y \cdot x \leq x \cdot y \Rightarrow (x \cdot y) \cdot x \leq x \cdot y \cdot x \cdot \bot \)
by (metis circ-mult-sub-sup circ-separate-1)

lemma separation:
assumes \( y \cdot x \leq x \cdot y \)
shows \( (x \sqcup y) \cdot x \cdot y \)
proof –
have \( y \cdot x \cdot y \cdot x \leq x \cdot y \sqcup y \cdot y \cdot \bot \)
by (metis assms circ-simulate-left-plus-1 circ-transitive-equal mult-assoc
mult-left-isotone)
thus ?thesis
by (metis sup-ge2 circ-isotone circ-mult-upper-bound circ-sub-dist separation)
qed

lemma simulation:
\( y \cdot x \leq x \cdot y \Rightarrow y \cdot x \leq x \cdot y \)
by (metis sup-ge2 circ-isotone circ-mult-upper-bound circ-sub-dist separation)

lemma circ-simulate-4:
assumes \( y \cdot x \leq x \cdot x \cdot (1 \sqcup y) \)
shows \( y^o \ast x^o \leq x^o \ast y^o \)

**proof** –

have \( x \sqcup (x \ast x^o \ast x \sqcup x \ast x) = x \ast x^o \)

by (metis (no-types) circ-back-loop-fixpoint mult-right-dist-sup sup-commute)

**hence** \( x \leq x \ast x^o \ast I \sqcup x \ast x^o \ast y \)

by (metis mult-1-right sup-assoc sup-ge1)

**hence** \( (I \sqcup y) \ast x \leq x \ast x^o \ast (I \sqcup y) \)

using assms mult-left-dist-sup mult-right-dist-sup by force

**hence** \( y \ast x^o \leq x^o \ast y^o \)

by (metis circ-sup-sub-sup-one circ-increasing circ-reflexive
circ-simulate-right-plus-1 mult-right-isotone mult-right-sub-dist-sup-right
order-trans)

thus \(?thesis \)

by (metis circ-simulate-2)

qed

**lemma** circ-simulate-5:

\( y \ast x \leq x \ast x^o \ast (x \sqcup y) \Rightarrow y^o \ast x^o \leq x^o \ast y^o \)

by (metis circ-sup-sub-sup-one circ-simulate-4 order-trans)

**lemma** circ-simulate-6:

\( y \ast x \leq x \ast (x \sqcup y) \Rightarrow y^o \ast x^o \leq x^o \ast y^o \)

by (metis sup-commute circ-back-loop-fixpoint circ-simulate-5
mult-right-sub-dist-sup-left order-trans)

**lemma** circ-separate-4:

**assumes** \( y \ast x \leq x \ast x^o \ast (I \sqcup y) \)

**shows** \( (x \sqcup y)^o = x^o \ast y^o \)

**proof** –

have \( y \ast x \ast x^o \leq x \ast x^o \ast (I \sqcup y) \ast x^o \)

by (simp add: assms mult-left-isotone)

also have \( ... = x \ast x^o \sqcup x \ast x^o \ast y \ast x^o \)

by (simp add: circ-transitive-equal mult-left-dist-sup mult-right-dist-sup
mult-assoc)

also have \( ... \leq x \ast x^o \sqcup x \ast x^o \ast x^o \ast y^o \)

by (metis assms sup-right-isotone circ-simulate-2 circ-simulate-4 mult-assoc
mult-right-isotone)

finally have \( y \ast x \ast x^o \leq x \ast x^o \ast y^o \)

by (metis circ-reflexive circ-transitive-equal le-iff-sup mult-assoc
mult-right-isotone mult-1-right)

thus \(?thesis \)

by (metis circ-sup-1 left-plus-circ mult-assoc separation)

qed

**lemma** circ-separate-5:

\( y \ast x \leq x \ast x^o \ast (x \sqcup y) \Rightarrow (x \sqcup y)^o = x^o \ast y^o \)

by (metis circ-sup-sub-sup-one circ-separate-4 order-trans)

**lemma** circ-separate-6:
\[ y \cdot x \leq x \cdot (x \sqcup y) \implies (x \sqcup y)^\circ = x^\circ \cdot y^\circ \]

by (metis sup-commute circ-back-loop-fixpoint circ-separate-5
mult-right-sub-dist-sup-left order-trans)

end

class bounded-itering = bounded-idempotent-left-zero-semiring + itering
begin
subclass bounded-left-conway-semiring ..

end

We finally expand Conway semirings and iterings by an element that
 corresponds to the endless loop.

class \( L = \)
  fixes \( L :: \alpha \)

class left-conway-semiring-L = left-conway-semiring + L +
  assumes one-circ-mult-split: \( 1^\circ \cdot x = L \sqcup x \)
  assumes L-split-sup: \( x \cdot (y \sqcup L) \leq x \cdot y \sqcup L \)
begin

lemma L-def:
  \( L = 1^\circ \cdot \bot \)
  by (metis sup-monoid.add-0-right one-circ-mult-split)

lemma one-circ-split:
  \( 1^\circ = L \sqcup 1 \)
  by (metis mult-1-right one-circ-mult-split)

lemma one-circ-circ-split:
  \( 1^{\circ\circ} = L \sqcup 1 \)
  by (metis circ-one one-circ-split)

lemma sub-mult-one-circ:
  \( x \cdot 1^\circ \leq 1^\circ \cdot x \)
  by (metis L-split-sup sup-commute mult-1-right one-circ-mult-split)

lemma one-circ-mult-split-2:
  \( 1^\circ \cdot x = x \cdot 1^\circ \sqcup L \)
proof
  have 1: \( x \cdot 1^\circ \leq L \sqcup x \)
    using one-circ-mult-split sub-mult-one-circ by presburger
  have \( x \sqcup x \cdot 1^\circ = x \cdot 1^\circ \)
    by (meson circ-back-loop-prefixpoint le-iff-sup sup.boundedE)
  thus \( \bot \)
using 1 by (simp add: le_iff_sup one-circ-mult-split sup-assoc sup-commute) qed

lemma sub-mult-one-circ-split:
  \(x \cdot 1^\circ \leq x \sqcup L\)
  by (metis sup-commute one-circ-mult-split sub-mult-one-circ)

lemma sub-mult-one-circ-split-2:
  \(x \cdot 1^\circ \leq x \sqcup 1^\circ\)
  by (metis L-def sup-right-isotone order-trans sub-mult-one-circ-split zero-right-mult-decreasing)

lemma L-split:
  \(x \cdot L \leq x \cdot \bot \sqcup L\)
  by (metis L-split-sup sup-monoid.add-0-left)

lemma L-left-zero:
  \(L \cdot x = L\)
  by (metis L-def mult-assoc mult-left-zero)

lemma one-circ-L:
  \(1^\circ \cdot L = L\)
  by (metis L-def circ-transitive-equal mult-assoc)

lemma mult-L-circ:
  \((x \cdot L)^\circ = 1 \sqcup x \cdot L\)
  by (metis L-left-zero circ-left-unfold mult-assoc)

lemma mult-L-circ-mult:
  \((x \cdot L)^\circ \cdot y = y \sqcup x \cdot L\)
  by (metis L-left-zero mult-L-circ mult-assoc mult-1-left mult-right-dist-sup)

lemma circ-L:
  \(L^\circ = L \sqcup 1\)
  by (metis L-left-zero sup-commute circ-left-unfold)

lemma L-below-one-circ:
  \(L \leq 1^\circ\)
  by (metis L-def zero-right-mult-decreasing)

lemma circ-circ-mult-1:
  \(x^\circ \cdot 1^\circ = x^{2\circ}\)
  by (metis L-left-zero sup-commute circ-sup-1 circ-circ-sup mult-zero-circ one-circ-split)

lemma circ-circ-mult:
  \(1^\circ \cdot x^\circ = x^{2\circ}\)
  by (metis antisym circ-circ-mult-1 circ-circ-sub-mult sub-mult-one-circ)
lemma circ-circ-split:
\[ x^0 = L \sqcup x^0 \]
by (metis circ-circ-mult one-circ-mult-split)

lemma circ-sup-6:
\[ L \sqcup (x \sqcup y)^0 = (x^0 \cdot y^0)^0 \]
by (metis sup-assoc sup-commute circ-sup-1 circ-circ-sup circ-circ-split
circ-decompose-4)

end

class itering-L = itering + L +
  assumes L-def: \( L = 1^0 \cdot \bot \)
begin

lemma one-circ-split:
\[ 1^0 = L \sqcup 1 \]
by (metis L-def sup-commute antisym circ-sup-upper-bound circ-reflexive
circ-simulate-absorb mult-1-right order-refl zero-right-mult-decreasing)

lemma one-circ-mult-split:
\[ 1^0 \cdot x = L \sqcup x \]
by (metis L-def sup-commute circ-loop-fixpoint mult-assoc mult-left-zero
mult-zero-circ one-circ-split)

lemma sub-mult-one-circ-split:
\[ x \cdot 1^0 \leq x \sqcup L \]
by (metis sup-commute one-circ-mult-split sub-mult-one-circ)

lemma sub-mult-one-circ-split-2:
\[ x \cdot 1^0 \leq x \sqcup 1^0 \]
by (metis L-def sup-right-isotone order-trans sub-mult-one-circ-split
zero-right-mult-decreasing)

lemma L-split:
\[ x \cdot L \leq x \cdot \bot \sqcup L \]
by (metis L-def mult-assoc mult-left-isotone mult-right-dist-sup
sub-mult-one-circ-split-2)

subclass left-conway-semiring-L
  apply unfold-locales
  apply (metis L-def sup-commute circ-loop-fixpoint mult-assoc mult-left-zero
mult-zero-circ one-circ-split)
  by (metis sup-commute mult-assoc mult-left-isotone one-circ-mult-split
sub-mult-one-circ)

lemma circ-left-induct-mult-L:
\[ L \leq x \Rightarrow x \cdot y \leq x \Rightarrow x \cdot y^0 \leq x \]
by (metis circ-one circ-simulate le-iff-sup one-circ-mult-split)
lemma circ-left-induct-mult-iff-L:
\[ L \leq x \implies x \ast y \leq x \iff x \ast y^\circ \leq x \]
by (metis sup.bounded-iff circ-back-loop-fixpoint circ-left-induct-mult-L le-iff-sup)

lemma circ-left-induct-L:
\[ L \leq x \implies x \ast y \sqcup z \leq x \implies z \ast y^\circ \leq x \]
by (metis sup.bounded-iff circ-left-induct-mult-L le-iff-sup mult-right-dist-sup)
end

3 Kleene Algebras

Kleene algebras have been axiomatised by Kozen to describe the equational theory of regular languages [13]. Binary relations are another important model. This theory implements variants of Kleene algebras based on idempotent left semirings [15]. The weakening of some semiring axioms allows the treatment of further computation models. The presented algebras are special cases of iterings, so many results can be inherited.

theory Kleene-Algebras
imports Iterings
begin

We start with left Kleene algebras, which use the left unfold and left induction axioms of Kleene algebras.

class star =
fixes star :: 'a ⇒ 'a (^:: [100] 100)

class left-kleene-algebra = idempotent-left-semiring + star +
assumes star-left-unfold : I \sqcup y \ast y^* \leq y^*
assumes star-left-induct : z \sqcup y \ast x \leq x \implies y^* \ast z \leq x
begin

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

abbreviation tc (^:: [100] 100) where tc x ≡ x \ast x^*

lemma star-left-unfold-equal:
\[ I \sqcup x \ast x^* = x^* \]
by (metis sup-right-isotone antisym mult-right-isotone mult-1-right star-left-induct star-left-unfold)
This means that for some properties of Kleene algebras, only one inequality can be derived, as exemplified by the following sliding rule.

**Lemma star-left-slide:**

\[(x \ast y)^* \ast x \leq x \ast (y \ast x)^*\]

by (metis mult-assoc mult-left-sub-dist-sup mult-1-right star-left-induct star-left-unfold-equal)

**Lemma star-isotone:**

\[x \leq y \Rightarrow x^* \leq y^*\]

by (metis sup-right-isotone mult-left-isotone order-trans star-left-unfold mult-1-right star-left-induct)

**Lemma star-sup-1:**

\[(x \sqcup y)^* = x^* \ast (y \ast x^*)^*\]

**Proof** (rule antisym)

have \(y \ast x^* \ast (y \ast x^*)^* \leq (y \ast x^*)^*\)

using sup-right-divisibility star-left-unfold-equal by auto

also have \(\ldots \leq x^* \ast (y \ast x^*)^*\)

using mult-left-isotone sup-left-divisibility star-left-unfold-equal by fastforce

finally have \((x \sqcup y) \ast (x^* \ast (y \ast x^*)^*) \leq x^* \ast (y \ast x^*)^*\)

by (metis le-supI mult-right-dist-sup mult-right-sub-dist-sup-right mult-assoc star-left-unfold-equal)

hence \(1 \sqcup (x \sqcup y) \ast (x^* \ast (y \ast x^*)^*) \leq x^* \ast (y \ast x^*)^*\)

using reflexive-mult-closed star-left-unfold by auto

thus \((x \sqcup y)^* \leq x^* \ast (y \ast x^*)^*\)

using star-left-induct by force

next have \(x^* \ast (y \ast x^*)^* \leq x^* \ast (y \ast (x \sqcup y)^*)^*\)

by (simp add: mult-right-isotone star-isotone)

also have \(\ldots \leq x^* \ast ((x \sqcup y) \ast (x \sqcup y)^*)^*\)

by (simp add: mult-right-isotone mult-right-sub-dist-sup-right star-isotone)

also have \(\ldots \leq x^* \ast (x \sqcup y)^{**}\)

using mult-right-isotone star-left-unfold star-isotone by auto

also have \(\ldots \leq (x \sqcup y)^* \ast (x \sqcup y)^*\)

by (simp add: mult-left-isotone star-isotone)

also have \(\ldots \leq (x \sqcup y)^*\)

by (metis sup.bounded-iff mult-1-right star-left-induct star-left-unfold)

finally show \(x^* \ast (y \ast x^*)^* \leq (x \sqcup y)^*\)

by simp

qed

end

We now show that left Kleene algebras form iterings. A sublocale is used instead of a subclass, because iterings use a different iteration operation.

**Sublocale** left-kleene-algebra < star: left-conway-semiring where circ = star

apply unfold-locale
apply (rule star-left-unfold-equal)
apply (rule star-left-slide)
A number of lemmas in this class are taken from Georg Struth's Kleene algebra theory [2].

**Lemma star-sub-one:**
\[ x \leq 1 \implies x^* = 1 \]
by (metis sup-right-isotone eq-iff le-iff-sup mult-1-right star-left-induct)

**Lemma star-one:**
\[ 1^* = 1 \]
by (simp add: star-sub-one)

**Lemma star-left-induct-mult:**
\[ x \cdot y \leq y \implies x^* \cdot y \leq y \]
by (simp add: star-left-induct)

**Lemma star-left-induct-mult-iff:**
\[ x \cdot y \leq y \iff x^* \cdot y \leq y \]
using mult-left-isotone order-trans star.circ-increasing star-left-induct-mult by blast

**Lemma star-involutive:**
\[ x^* = x^{**} \]
using star.circ-circ-sap star-sup-1 star-one by auto

**Lemma star-sup-one:**
\[ (1 \sqcup x)^* = x^* \]
using star.circ-circ-sap star-involutive by auto

**Lemma star-left-induct-equal:**
\[ z \sqcup x \cdot y = y \implies x^* \cdot z \leq y \]
by (simp add: star-left-induct)

**Lemma star-left-induct-mult-equal:**
\[ x \cdot y = y \implies x^* \cdot y \leq y \]
by (simp add: star-left-induct-mult)

**Lemma star-star-upper-bound:**
\[ x^* \leq z^* \implies x^{**} \leq z^* \]
using star-involutive by auto

**Lemma star-simulation-left:**
assumes \[ x \cdot z \leq z \cdot y \]
shows \[ x^* \cdot z \leq z \cdot y^* \]
proof –
have \( x \cdot z \cdot y^* \leq z \cdot y \cdot y^* \)
by (simp add: assms mult-left-isotone)
also have \( \ldots \leq z \cdot y^* \)
by (simp add: mult-right-isotone star.left-plus-below-circ mult-assoc)
finally have \( z \uplus x \cdot z \cdot y^* \leq z \cdot y^* \)
using star.circ-back-loop-prefixpoint by auto
thus \( ?thesis \)
by (simp add: star-left-induct mult-assoc)

qed

\[ \text{lemma quasicomm-1:} \]
\[ y \cdot x \leq x \cdot (x \uplus y)^* \iff y^* \cdot x \leq x \cdot (x \uplus y)^* \]
by (metis mult-isotone order-refl order-trans star.circ-increasing star-involutive star-simulation-left)

\[ \text{lemma star-rtc-3:} \]
\[ 1 \uplus x \uplus y \cdot y = y \Rightarrow x^* \leq y \]
by (metis sup.bounded-iff le-iff-sup mult-left-sub-dist-sup-left mult-1-right star-left-induct-mult-iff star-circ-sub-dist)

\[ \text{lemma star-decompose-1:} \]
\[ (x \uplus y)^* = (x^* \cdot y^*)^* \]
apply (rule antisym)
apply (simp add: star.circ-sup-2)
using star.circ-sub-dist-3 star-isotone star-involutive by fastforce

\[ \text{lemma star-sum:} \]
\[ (x \uplus y)^* = (x^* \uplus y^*)^* \]
using star-decompose-1 star-involutive by auto

\[ \text{lemma star-decompose-3:} \]
\[ (x^* \cdot y^*)^* = x^* \cdot (y^* \cdot x^*)^* \]
using star-sup-1 star-decompose-1 by auto

In contrast to iterings, we now obtain that the iteration operation results in least fixpoints.

\[ \text{lemma star-loop-least-fixpoint:} \]
\[ y \cdot x \uplus z = x \Rightarrow y^* \cdot z \leq x \]
by (simp add: sup-commute star-left-induct-equal)

\[ \text{lemma star-loop-is-least-fixpoint:} \]
\[ \text{is-least-fixpoint } (\lambda x . y \cdot x \uplus z) (y^* \cdot z) \]
by (simp add: is-least-fixpoint-def star.circ-loop-fixpoint star-loop-least-fixpoint)

\[ \text{lemma star-loop-mu:} \]
\[ \mu (\lambda x . y \cdot x \uplus z) = y^* \cdot z \]
by (metis least-fixpoint-same star-loop-is-least-fixpoint)

\[ \text{lemma affine-has-least-fixpoint:} \]

23
\begin{verbatim}
has-least-fixpoint \((\lambda x . y \ast x \sqcup z)\)
by (metis has-least-fixpoint-def star-loop-least-fixpoint)

lemma star-outer-increasing:
\(x \leq y^* \ast x \ast y^*\)
by (metis star.circ-back-loop-prefixpoint star.circ-loop-fixpoint sup.boundedE)

end

We next add the right induction rule, which allows us to strengthen many
inequalities of left Kleene algebras to equalities.

class strong-left-kleene-algebra = left-kleene-algebra +
  assumes star-right-induct: \(z \sqcup x \ast y \leq x \longrightarrow z \ast y^* \leq x\)
begin

lemma star-plus:
\(y^* \ast y = y \ast y^*\)
proof (rule antisym)
  show \(y^* \ast y \leq y \ast y^*\)
    by (simp add: star.circ-plus-sub)
next
  have \(y^* \ast y \ast y \leq y^* \ast y\)
    by (simp add: mult-left-isotone star-right-below-circ)
  hence \(y \sqcup y^* \ast y \ast y \leq y^* \ast y\)
    by (simp add: star.circ-mult-increasing-2)
  thus \(y \ast y^* \leq y^* \ast y\)
    using star-right-induct by blast
qed

lemma star-slide:
\(((x \ast y)^* \ast x = x \ast (y \ast x)^*)\)
proof (rule antisym)
  show \(((x \ast y)^* \ast x \leq x \ast (y \ast x)^*)\)
    by (rule star-left-slide)
next
  have \(x \sqcup (x \ast y)^* \ast x \ast y \ast x \leq (x \ast y)^* \ast x\)
    by (metis (full-types) sup.commute eq-refl star.circ-loop-fixpoint mult.assoc star-plus)
  thus \(x \ast (y \ast x)^* \leq (x \ast y)^* \ast x\)
    by (simp add: mult.assoc star-right-induct)
qed

lemma star-simulation-right:
  assumes \(z \ast x \leq y \ast z\)
  shows \(z \ast x^* \leq y^* \ast z\)
proof
  have \(y^* \ast z \ast x \leq y^* \ast z\)
end
\end{verbatim}
by (metis assms dual-order.trans mult-isotone mult-left-sub-dist-sup-right
star.circ-loop-fixpoint star.circ-transitive-equal sup.cobounded1 mult-assoc)
thus ?thesis
by (metis le-supI star.circ-loop-fixpoint star-right-induct sup.cobounded2)
qed

Again we inherit results from the itering hierarchy.

sublocale strong-left-kleene-algebra <
itering-1 where
circ = star
apply unfold-locales
apply (simp add: star-slide)
by (simp add: star-simulation-right)

context strong-left-kleene-algebra
begin

lemma star-right-induct-mult:
y * x ≤ y =⇒ y * x* ≤ y
by (simp add: star-right-induct)

lemma star-right-induct-mult-iff:
y * x ≤ y =⇒ y * x* ≤ y
using mult-right-isotone order-trans star.circ-increasing star-right-induct-mult
by blast

lemma star-simulation-right-equal:
z * x = y * z =⇒ z * x* = y* * z
by (metis eq-iff star-simulation-left star-simulation-right)

lemma star-simulation-star:
x * y ≤ y * x =⇒ x* * y* ≤ y* * x*
by (simp add: star-simulation-left star-simulation-right)

lemma star-right-induct-equal:
z ⊔ y * x = y =⇒ z * x* ≤ y
by (simp add: star-right-induct)

lemma star-right-induct-mult-equal:
y * x = y =⇒ y * x* ≤ y
by (simp add: star-right-induct-mult)

lemma star-back-loop-least-fixpoint:
x * y ⊔ z = x =⇒ z * y* ≤ x
by (simp add: sup-commute star-right-induct-equal)

lemma star-back-loop-is-least-fixpoint:
is-least-fixpoint (λx . x * y ⊔ z) (z * y*)
proof (unfold is-least-fixpoint-def, rule conjI)
have \((z * y) * y \sqcup z) * y \leq z * y * y \sqcup z\)
  using le-supI1 mult-left-isotone star.circ-back-loop-prefixpoint by auto
hence \(z * y * y \leq z * y * y \sqcup z\)
  by (simp add: star-right-induct)
thus \(z * y * y \sqcup z = z * y\)
  using antisym star.circ-back-loop-prefixpoint by auto

next
show \(\forall x. x * y \sqcup z = x \rightarrow z * y \leq x\)
  by (simp add: star-back-loop-least-fixpoint)

qed

lemma star-back-loop-mu:
  \(\mu (\lambda x . x * y \sqcup z) = z * y\)
  by (metis least-fixpoint-same star-back-loop-is-least-fixpoint)

lemma star-square:
  \(x * = (I \sqcup x) * (x * x)\)
proof
  let \(f = \lambda y . y * x \sqcup I\)
  have 1: isotone \(f\)
    by (metis sup-left-isotone isotone-def mult-left-isotone)
  have \(f \circ f = (\lambda y . y * (x * x) \sqcup (I \sqcup x))\)
    by (simp add: sup-assoc sup-commute mult-assoc mult-right-dist-sup o-def)
  thus \(?thesis\)
    using 1 by (metis no-types antisym mult-left-sub-dist-sup star.
circ-square-2 has-least-fixpoint-def star-back-loop-is-least-fixpoint)

qed

lemma star-square-2:
  \(x * = (x * x) * (x \sqcup I)\)
proof
  have \((I \sqcup x) * (x * x)* = (x * x)* * I \sqcup x * (x * x)\)
    using mult-right-dist-sup by force
  thus \(?thesis\)
    by (metis (no-types) antisym mult-left-sub-dist-sup star.circ-square-2 star.
circ-slide sup-commute star-square)

qed

lemma star-circ-simulate-right-plus:
assumes \(z * x \leq y * y * z \sqcup w\)
  shows \(z * x* \leq y * (z \sqcup w * x)\)
proof
  have \((z \sqcup w * x*) * x \leq z * x \sqcup w * x\)
    using mult-right-dist-sup star.circ-back-loop-prefixpoint sup-right-isotone by auto
  also have \(z \sqcup w * x* \leq y * y * z \sqcup w * x\)
    using assms sup-left-isotone by blast
  also have \(z \sqcup w * x* \leq y * y * z \sqcup w * x\)
    using le-supI1 star.circ-back-loop-prefixpoint sup-commute by auto
also have ... ≤ y* (z ∪ w * x*)
  by (metis sup.bounded-iff mult-isotone mult-left-isotone mult-left-one
  mult-left-sub-dist-sup-left star.circ-reflexive star.left-plus-below-circ)
finally have y* (z ∪ w * x*) * x ≤ y* (z ∪ w * x*)
  by (metis mult-assoc mult-right-isotone star.circ-transitive-equal)
thus ?thesis
  by (metis sup.bounded-iff star-right-induct mult-left-sub-dist-sup-left
  star.circ-loop-fixpoint)
qd

lemma transitive-star:
x * x ≤ x ⇒ x* = 1 ∪ x
  by (metis order.antisym star.circ-mult-increasing-2 star.circ-plus-same
  star-left-induct-mult star-left-unfold-equal)

end

The following class contains a generalisation of Kleene algebras, which
lacks the right zero axiom.

class left-zero-kleene-algebra = idempotent-left-zero-semiring +
  strong-left-kleene-algebra
begin

lemma star-star-absorb:
y* * (y* * x)* * y* = (y* * x)* * y*
  by (metis star.circ-transitive-equal star-slide mult-assoc)

lemma star-circ-simulate-left-plus:
assumes x * z ≤ z * y* ∪ w
  shows x* * z ≤ (z ∪ x* * w) * y*
proof –
  have x * (x* * (w * y*)) ≤ x* * (w * y*)
    by (metis (no-types) mult-right-sub-dist-sup-left star.circ-loop-fixpoint
      mult-assoc)
hence x * ((z ∪ x * w) * y*) ≤ x * z * y* ∪ x* * w * y*
  using mult-left-dist-sup mult-right-dist-sup-sup-right-isotone mult-assoc
    by presburger
also have ... ≤ (z * y* ∪ w) * y* ∪ x* * w * y*
  using assms mult-isotone semiring.add-right-mono by blast
also have ... = z * y* ∪ w * y* ∪ x* * w * y*
  by (simp add: mult-right-dist-sup star.circ-transitive-equal mult-assoc)
also have ... = (z ∪ w ∪ x* * w) * y*
  by (simp add: mult-right-dist-sup)
also have ... = (z ∪ x* * w) * y*
  by (metis sup-assoc sup-ge2 le-iff-sup star.circ-loop-fixpoint)
finally show ?thesis
  by (metis sup.bounded-iff mult-left-sub-dist-sup-left mult-1-right
  mult-right-sub-dist-sup-left star.circ-loop-fixpoint)
\[
\text{lemma} \ star-one-sup-below:
\begin{align*}
x \ast y^* \ast (1 \sqcup z) \le x \ast (y \sqcup z)^*
\end{align*}
\]
\textbf{proof} –
\begin{align*}
\text{have } & y^* \ast z \le (y \sqcup z)^* \\
\text{using } & \text{sup-ge2 order-trans star.circ-increasing star.circ-mult-upper-bound} \star \text{circ-sub-dist by blast} \\
\text{hence } & y^* \sqcup y^* \ast z \le (y \sqcup z)^* \\
\text{by } & (\text{simp add: star.circ-sup-upper-bound star.circ-sub-dist}) \\
\text{hence } & y^* \ast (1 \sqcup z) \le (y \sqcup z)^* \\
\text{by } & (\text{simp add: mult-left-dist-sup}) \\
\text{thus } & \text{?thesis by (metis mult-right-isotone mult-assoc)}
\end{align*}
\textbf{qed}

The following theorem is similar to the puzzle where persons insert themselves always in the middle between two groups of people in a line. Here, however, items in the middle annihilate each other, leaving just one group of items behind.

\textbf{lemma} cancel-separate:
\begin{align*}
\text{assumes } x \ast y \le 1 \\
\text{shows } x^* \ast y^* \le x^* \sqcup y^*
\end{align*}
\textbf{proof} –
\begin{align*}
\text{have } & x \ast y^* = x \sqcup x \ast y \ast y^* \\
\text{by } & (\text{metis mult-assoc mult-left-dist-sup mult-1-right star-left-unfold-equal}) \\
\text{also have } & \ldots \le x \sqcup y^* \\
\text{by } & (\text{meson assms dual-order.trans order.refl star.circ-mult-upper-bound star.circ-reflexive sup-right-isotone}) \\
\text{also have } & \ldots \le x^* \sqcup y^* \\
\text{using } & \text{star.circ-increasing sup-left-isotone by auto} \\
\text{finally have } & 1: x \ast y^* \le x^* \sqcup y^* \\
\text{have } & x \ast (x^* \sqcup y^*) = x \ast x^* \sqcup x \ast y^* \\
\text{by } & (\text{simp add: mult-left-dist-sup}) \\
\text{also have } & \ldots \le x^* \sqcup y^* \\
\text{using } & 1 \text{ by (metis sup.bounded-iff sup-ge1 order-trans star.left-plus-below-circ)} \\
\text{finally have } & 2: x \ast (x^* \sqcup y^*) \le x^* \sqcup y^* \\
\text{have } & y^* \le x^* \sqcup y^* \\
\text{by } & \text{simp} \\
\text{hence } & y^* \sqcup x \ast (x^* \sqcup y^*) \le x^* \sqcup y^* \\
\text{using } & 2 \text{ sup.bounded-iff by blast} \\
\text{thus } & \text{?thesis by (metis star-left-induct)}
\end{align*}
\textbf{qed}

\textbf{lemma} star-separate:
\begin{verbatim}
assumes \( x \ast y = \text{bot} \)
and \( y \ast y = \text{bot} \)
shows \((x \sqcup y)^* = x^* \sqcup y \ast x^* \)

proof -
have 1: \( y^* = 1 \sqcup y \)
  using assms(2) by (simp add: transitive-star)
have \((x \sqcup y)^* = y^* \ast (x \ast y)^* \)
  using \( \ast \) by (simp add: star.circ-decompose-6 star-sup-1)
also have \(... = (1 \sqcup y) \ast x^* \)
  using 1 by (simp add: assms mult-left-dist-sup)
also have \(... = x^* \sqcup y \ast x^* \)
  by (simp add: mult-right-dist-sup)
finally show \( ?\text{thesis} \).
qed
end

We can now inherit from the strongest variant of iterings.

sublocale left-zero-kleene-algebra < star: itering where \( \text{circ} = \text{star} \)
apply unfold-locales
apply (metis star.circ-sup-9)
apply (metis star.circ-mult-1)
apply (simp add: star-circ-simulate-right-plus)
by (simp add: star-circ-simulate-left-plus)

category left-zero-kleene-algebra
begin

lemma star-absorb:
\( x \ast y = \text{bot} \Rightarrow x \ast y^* = x \)
by (metis sup.bounded-iff antisym_conv star.circ-back-loop-prefixpoint star.circ-elimination)

lemma star-separate-2:
assumes \( x \ast z^+ \ast y = \text{bot} \)
and \( y \ast z^+ \ast y = \text{bot} \)
and \( z \ast x = \text{bot} \)
shows \((x^* \sqcup y \ast x^*) \ast (z \ast (1 \sqcup y \ast x^*))^* = z^* \ast (x^* \sqcup y \ast x^*) \ast z^* \)

proof -
have 1: \( x^* \ast z^+ \ast y = z^+ \ast y \)
  by (metis assms mult-assoc mult-1-left mult-left-zero star.circ-zero star-simulation-right-equal)
have 2: \( z^* \ast (x^* \sqcup y \ast x^*) \ast z^+ \leq z^* \ast (x^* \sqcup y \ast x^*) \ast z^* \)
  by (simp add: mult-right-isotone star.left-plus-below-circ)
have \( z^* \ast z^+ \ast y \ast x^* \leq z^* \ast y \ast x^* \)
  by (metis mult-left-isotone star.left-plus-below-circ star.right-plus-circ)
\end{verbatim}
star-plus)

also have ... ≤ z∗ · (x∗ ⊔ y ∗ x∗)
  by (simp add: mult-assoc mult-left-sub-dist-sup-right)
also have ... ≤ z∗ · (x∗ ⊔ y ∗ x∗) * z∗
  using sup-right-divisibility star.circ-back-loop-firpoint by blast
finally have 3: z∗ · z∗ ⊔ y ∗ x∗ ≤ z∗ · (x∗ ⊔ y ∗ x∗) * z∗

have z∗ · (x∗ ⊔ y ∗ x∗) * z∗ · (z ∗ (1 ⊔ y ∗ x∗)) = z∗ · (x∗ ⊔ y ∗ x∗) * z∗ ⊔
 z∗ · (x∗ ⊔ y ∗ x∗) * z∗ 
  by (metis multi-1-right semiring.distrib-left star.circ-plus-same mult-assoc)
also have ... = z∗ · (x∗ ⊔ y ∗ x∗) * z∗ ⊔ z∗ · (1 ⊔ y) * x∗ * z∗ ⊔ y ∗ x∗
  by (simp add: semiring.distrib-right mult-assoc)
also have ... = z∗ · (x∗ ⊔ y ∗ x∗) * z∗ ⊔ z∗ · (1 ⊔ y) * z∗ ⊔ y ∗ x∗
  using 1 by (simp add: mult-assoc)
also have ... = z∗ · (x∗ ⊔ y ∗ x∗) * z∗ ⊔ z∗ · z∗ · y * z∗ ⊔ y * z∗ · y
  using mult-left-dist-sup mult-right-dist-sup sup-associative
also have ... = z∗ · (x∗ ⊔ y ∗ x∗) * z∗ 
  by (metis assms(2) mult-left-dist-sup mult-left-zero sup-commute
   sup-monoid.add-0-left mult-assoc)
also have ... ≤ z∗ · (x∗ ⊔ y ∗ x∗) * z∗
  using 2 3 by simp
finally have (x∗ ⊔ y ∗ x∗) ⊔ z∗ · (x∗ ⊔ y ∗ x∗) * z∗ · (z ∗ (1 ⊔ y ∗ x∗)) ≤
 z∗ · (x∗ ⊔ y ∗ x∗) * z∗ 
  by (simp add: star-right-increasing)

hence 4: (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ⊔ y ∗ x∗)) ≤ z∗ · (x∗ ⊔ y ∗ x∗) * z∗
  by (simp add: star-right-induct)
have 5: (x∗ ⊔ y ∗ x∗) * z∗ ≤ (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ⊔ y ∗ x∗))
  by (metis sup-ge1 mult-right-associative mult-right-1-right star-isotone)
have z · (x∗ ⊔ y ∗ x∗) = z ∗ x∗ ⊔ z ∗ y ∗ x∗
  by (simp add: mult-assoc mult-left-dist-sup)
also have ... = z ⊔ z ∗ y ∗ x∗
  by (simp add: assms star-absorb)
also have ... = z ∗ (1 ∪ y ∗ x∗)
  by (simp add: mult-assoc mult-left-dist-sup)
also have ... ≤ (z ∗ (1 ∪ y ∗ x∗))
  by (simp add: star.circ-increasing)
also have ... ≤ (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ∪ y ∗ x∗))
  by (metis le-supE mult-right-dist-sup-left star.circ-loop-firpoint
finally have z · (x∗ ⊔ y ∗ x∗) ≤ (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ∪ y ∗ x∗))
  by (metis mult-assoc mult-left-associative star.circ-transitive-true)

hence z · (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ⊔ y ∗ x∗)) ≤ (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ∪ y ∗ x∗))
  by (metis mult-assoc mult-left-associative star.circ-transitive-true)

hence z∗ · (x∗ ⊔ y ∗ x∗) · z∗ ≤ (x∗ ⊔ y ∗ x∗) * (z ∗ (1 ∪ y ∗ x∗))
  using 5 by (metis star-right-induct sup.bounded-if mult-assoc

thus ?thesis
  using 4 by (simp add: antisym)

qed
A Kleene algebra is obtained by requiring an idempotent semiring.

class kleene-algebra = left-zero-kleene-algebra + idempotent-semiring

The following classes are variants of Kleene algebras expanded by an additional iteration operation. This is useful to study the Kleene star in computation models that do not use least fixpoints in the refinement order as the semantics of recursion.

class left-kleene-conway-semiring = left-kleene-algebra + left-conway-semiring

begin

lemma star-below-circ:
  \( x^* \leq x^0 \)
  by (metis circ-left-unfold mult-1-right order-refl star-left-induct)

lemma star-zero-below-circ-mult:
  \( x^* \otimes \bot \leq x^0 \otimes y \)
  by (simp add: mult-isotone star-below-circ)

lemma star-mult-circ:
  \( x^* \otimes x^0 = x^0 \)
  by (metis sup-right-divisibility antisym circ-left-unfold star-left-induct-mult star.circ-loop-fixpoint)

lemma circ-mult-star:
  \( x^* \otimes x^0 = x^0 \)
  by (metis sup-assoc sup.bounded-iff circ-left-unfold circ rtc-2 eq iff left-plus-circ star.circ-sup-star star.circ-back-loop-prefixpoint star.circ-increasing star-below-circ star-mult-circ star-sup-one)

lemma circ-star:
  \( x^{*0} = x^0 \)
  by (metis antisym circ-reflexive circ-transitive-equal star.circ-increasing star-sup-one-right-unfold star-left-induct-mult-equal)

lemma star-circ:
  \( x^{00} \leq x^0 \)
  by (metis antisym circ-circ-sup circ-sub-dist le-iff-sup star.circ rtc-2 star-below-circ)

lemma circ-sup-3:
  \( (x^0 \otimes y^0)^* \leq (x \uplus y)^0 \)
  using circ-star circ-sub-dist-3 star-isotone by fastforce

end

class left-zero-kleene-conway-semiring = left-zero-kleene-algebra + itering
begin

subclass left-kleene-conway-semiring ..

lemma circ-isolate:
\[ x^\circ = x^\circ \star \bot \sqcup x^\star \]
\hspace{1em}by (metis sup-commute antisym circ-sup-upper-bound circ-mult-star circ-simulate-absorb star.left-plus-below-circ star-below-circ zero-right-mult-decreasing)

lemma circ-isolate-mult:
\[ x^\circ \star y = x^\circ \star \bot \sqcup x^\star \star y \]
\hspace{1em}by (metis circ-isolate mult-assoc mult-left-zero mult-right-dist-sup)

lemma circ-isolate-mult-sub:
\[ x^\circ \star y \leq x^\circ \sqcup x^\star \star y \]
\hspace{1em}by (metis sup-left-isotone circ-isolate-mult zero-right-mult-decreasing)

lemma circ-sub-decompose:
\[ (x^\circ \star y)^\circ \leq (x^\star \star y)^\circ \star x^\circ \]
proof –
\hspace{1em}have \( x^\star \star y \sqcup x^\circ \star \bot = x^\circ \star y \)
\hspace{1em}by (metis sup.commute circ-isolate-mult)
\hspace{1em}hence \( (x^\star \star y)^\circ \star x^\circ = ((x^\circ \star y)^\circ \sqcup x^\circ)^\circ \star x^\circ \)
\hspace{1em}by (metis circ-star circ-sup-9 circ-sup-mult-zero star-decompose-1)
\hspace{1em}thus \hspace{1em}thesis
\hspace{1em}by (metis circ-star le-iff-sup star.circ-decompose-7 star.circ-unfold-sum)
qed

lemma circ-sup-4:
\[ (x \sqcup y)^\circ = (x^\star \star y)^\circ \star x^\circ \]
apply (rule antisym)
apply (metis circ-sup circ-sub-decompose circ-transitive-equal mult-assoc mult-left-isotone)
by (metis circ-sup circ-isotone mult-left-isotone star-below-circ)

lemma circ-sup-5:
\[ (x^\circ \star y)^\circ \star x^\circ = (x^\star \star y)^\circ \star x^\circ \]
using circ-sup-4 circ-sup-9 by auto

lemma plus-circ:
\[ (x^\star \star x)^\circ = x^\circ \]
\hspace{1em}by (metis sup-idem circ-sup-4 circ-decompose-7 circ-star star.circ-decompose-5 star.right-plus-circ)

end

The following classes add a greatest element.

32
class bounded-left-kleene-algebra = bounded-idempotent-left-semiring + left-kleene-algebra

sublocale bounded-left-kleene-algebra < star: bounded-left-conway-semiring
where circ = star ..

class bounded-left-zero-kleene-algebra = bounded-idempotent-left-semiring + left-zero-kleene-algebra

sublocale bounded-left-zero-kleene-algebra < star: bounded-itering where circ = star ..

class bounded-kleene-algebra = bounded-idempotent-semiring + kleene-algebra

sublocale bounded-kleene-algebra < star: bounded-itering where circ = star ..

We conclude with an alternative axiomatisation of Kleene algebras.

class kleene-algebra-var = idempotent-semiring + star +
assumes star-left-unfold-var : 1 ∪ y * y* ≤ y*
assumes star-left-induct-var : y * x ≤ x → y* * x ≤ x
assumes star-right-induct-var : x * y ≤ x → x * y* ≤ x
begin
subclass kleene-algebra
  apply unfold-locales
  apply (rule star-left-unfold-var)
  apply (meson sup.bounded-iff mult-right-isotone order-trans star-left-induct-var)
  by (meson sup.bounded-iff mult-left-isotone order-trans star-right-induct-var)
end
end

4 Kleene Relation Algebras

This theory combines Kleene algebras with Stone relation algebras. Relation algebras with transitive closure have been studied by [16]. The weakening to Stone relation algebras allows us to talk about reachability in weighted graphs, for example.

Many results in this theory are used in the correctness proof of Prim’s minimum spanning tree algorithm. In particular, they are concerned with the exchange property, preservation of parts of the invariant and with establishing parts of the postcondition.

theory Kleene-Relation-Algebras

imports Stone-Relation-Algebras Relation-Algebras Kleene-Algebras

33
We first note that bounded distributive lattices can be expanded to Kleene algebras by reusing some of the operations.

\[
\text{sublocale bounded-distrib-lattice < comp-inf: bounded-kleene-algebra where star = } \lambda x . \text{top and one = top and times = inf}
\]

apply unfold-locales
apply (simp add: inf.assoc)
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
apply simp
by (simp add: inf-assoc)

Kleene star and the relational operations are reasonably independent. The only additional axiom we need in the generalisation to Stone-Kleene relation algebras is that star distributes over double complement.

\[
\text{class stone-kleene-relation-algebra = stone-relation-algebra + kleene-algebra + assumes pp-dist-star: } \\
\quad \neg \neg (x^*) = (\neg \neg x)^*
\]

begin

subclass bounded-kleene-algebra ..

lemma regular-closed-star:
regular x \Rightarrow regular (x^*)
by (simp add: pp-dist-star)

lemma conv-star-conv:
\( x^* \leq x^{T+T} \)
proof
have \( x^{T+T} * x^T \leq x^{T+T} \)
  by (simp add: star_right-plus-below-circ)
  hence \( I : x * x^{T+T} \leq x^{T+T} \)
  using conv-dist-comp cone_isotone by fastforce
  have \( 1 \leq x^{T+T} \)
    by (simp add: reflexive-conv-closed star_circ_reflexive)
  hence \( I \sqcup x * x^{T+T} \leq x^{T+T} \)
  using \( I \) by simp
  thus \( \neg \neg \)thesis
    using star_left_induct by fastforce

34
It follows that star and converse commute.

**Lemma** \(\text{conv-star-commute:} \)
\[
x^{*T} = x^{T*}
\]

**Proof** (rule \textit{antisym})

\[
ex^{*T} \leq x^{T*}
\]
using \textit{conv-star-conv conv-isotone by fastforce}

**Next**

\[
x^{T*} \leq x^{*T}
\]
by (metis \textit{conv-star-conv conv-involutive})

\textit{qed}

**Abbreviation** \textit{acyclic} :: \(\forall a \Rightarrow \text{bool}\)
where \textit{acyclic} \(x \equiv x^+ \leq -1\)

**Abbreviation** \textit{forest} :: \(\forall a \Rightarrow \text{bool}\)
where \textit{forest} \(x \equiv \text{injective} x \land \text{acyclic} x\)

**Lemma** \textit{forest-bot}:

\textit{forest bot}
by simp

**Lemma** \textit{acyclic-star-below-complement}:

\textit{acyclic} \(w \iff w^{TT} \leq -w\)
by (simp add: \textit{conv-star-commute schroeder-4-p})

**Lemma** \textit{acyclic-asymmetric}:

\textit{acyclic} \(w \Rightarrow w^T \sqcap w = \text{bot}\)
using \textit{acyclic-star-below-complement inf.order-leseq-imp pseudo-complement star.circ-increasing by blast}

**Lemma** \textit{vector-star-1}:

\textit{assumes} \textit{vector} \(x\)
\textit{shows} \(x^T \ast (x \ast x^{T})^* \leq x^T\)

**Proof**

\textit{have} \(x^T \ast (x \ast x^{T})^* = (x^T \ast x)^* \ast x^T\)
by (simp add: \textit{star-slide})
\textit{also have} \(\ldots \leq \text{top} \ast x^T\)
by (simp add: \textit{mult-left-isotone})
\textit{also have} \(\ldots = x^T\)
using \textit{assms vector-conv-covector by auto}
finally show \(?thesis\)

\textit{qed}

**Lemma** \textit{vector-star-2}:

\textit{vector} \(x \Rightarrow x^T \ast (x \ast x^{T})^* \leq x^T \ast \text{bot}^*\)
by (simp add: \textit{star-absorb vector-star-1})

**Lemma** \textit{vector-vector-star}:
vector \( v \mapsto (v \ast v^T)^* = I \sqcup v \ast v^T \)

by (simp add: transitive-star vv-transitive)

**lemma** forest-separate:

assumes forest \( x \)

shows \( x^* \ast x^T \ast \sqcap x^T \ast x \leq 1 \)

**proof**

have \( x^* \ast 1 \leq -x^T \)

using assms schroeder-5-p by force

hence \( 1 : x^* \sqcap x^T = \bot \)

by (simp add: pseudo-complement)

have \( x^* \sqcap x^T \ast x = (I \sqcup x^* \ast x) \sqcap x^T \ast x \)

using star.circ-right-unfold-1 by simp

also have \( \ldots = (I \sqcup x^T \ast x) \sqcup (x^* \ast x \sqcap x^T \ast x) \)

by (simp add: inf-sup-distrib2)

also have \( \ldots \leq 1 \sqcup (x^* \sqcap x^T) \ast x \)

by (simp add: assms injective-comp-right-dist-inf)

also have \( \ldots = 1 \)

using 1 by simp

finally have \( 2 : x^* \sqcap x^T \ast x \leq 1 \)

hence \( 3 : x^T \ast x^T \ast x \leq 1 \)

by (metis (mono-tags, lifting) cone-star-commute conv-dist-comp conv-dist-inf cone-involutive coreflexive-symmetric)

also have \( \ldots \leq 1 \)

by (metis assms (1) conv-involutive dedekind-1 inf-top.left-neutral)

also have \( \ldots \leq 1 \)

using 2 3 by (simp add: inf-sup-distrib2)

finally show \( \text{thesis} \)

.. .

**qed**

**lemma** cut-reachable:

assumes \( v^T = r^T \ast t^* \)

and \( t \leq g \)

shows \( v \ast -v^T \sqcap g \leq (r^T \ast g^*)^T \ast (r^T \ast g^*) \)

**proof**

have \( v \ast -v^T \sqcap g \leq v \ast \top \sqcap g \)

using inf.sup-left-isotone mult-right-isotone top-greatest by blast

also have \( \ldots = (r^T \ast t^*)^T \ast \top \sqcap g \)

by (metis assms (1) conv-involutive)

also have \( \ldots \leq (r^T \ast g^*)^T \ast \top \sqcap g \)

using assms (2) conv-isotone inf.sup-left-isotone mult-left-isotone mult-right-isotone star-isotone by auto

also have \( \ldots \leq (r^T \ast g^*)^T \ast ((r^T \ast g^*) \ast g) \)

by (metis conv-involutive dedekind-1 inf-top.left-neutral)

also have \( \ldots \leq (r^T \ast g^*)^T \ast (r^T \ast g^*) \)

36
The following lemma shows that the nodes reachable in the graph can be reached by only using edges between reachable nodes.

**Lemma reachable-restrict:**

- **Assumes** vector $r$
- **Shows** $r^T * g^T = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^*$

**Proof**:
- Have 1: $r^T \leq r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^*$
  - Using mult-right-isotone mult-1-right star.circ-reflexive by fastforce
- Have 2: covector $(r^T * g^*)$
  - Using assms covector-mult-closed vector-conv-covector by auto
- Finally have $r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * g \leq r^T * g^* * g$
  - By (simp add: mult-left-isotone mult-right-isotone star-isotone)
- Also have $... \leq r^T * g^*$
  - By (simp add: mult-assoc mult-right-isotone star.left-plus below-circ star-plus)
- Finally have $r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * g = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * g \cap r^T * g^*$
  - By (simp add: le-iff-inf)
- Also have $... = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * (g \cap r^T * g^*)$
  - Using assms covector-comp-inf covector-mult-closed vector-conv-covector by auto
- Also have $... = (r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* \cap r^T * g^*) * (g \cap r^T * g^*)$
  - By (simp add: inf.absorb2 inf-commute mult-right-isotone star-isotone)
- Also have $... = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * (g \cap r^T * g^* \cap (r^T * g^*)^T)$
  - Using 2 by (metis comp-inf-vector-1)
- Also have $... = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * ((r^T * g^*)^T \cap r^T * g^* \cap g)$
  - Using inf-commute inf-associ ac by simp
- Also have $... = r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^* * ((r^T * g^*)^T * (r^T * g^*) \cap g)$
  - Using 2 by (metis covector-conv-covector inf-top.right-neutral vector-inf-comp)
- Also have $... \leq r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^*$
  - By (simp add: mult-assoc mult-right-isotone star.left-plus below-circ star-plus)
- Finally have $r^T * g^* \leq r^T * ((r^T * g^*)^T * (r^T * g^*) \cap g)^*$
  - Using 1 star-right-induct by auto
- Thus $?thesis$
- By (simp add: inf.eq-iff mult-right-isotone star-isotone)

**QED**

The following lemma shows that the predecessors of visited nodes in the minimum spanning tree extending the current tree have all been visited.

**Lemma predecessors-reachable:**

- **Assumes** vector $r$
- **And** injective $r$
and $v^T = r^T * t^*$
and forest $w$
and $t \leq w$
and $w \leq (r^T * g^*)^T * (r^T * g^*) \cap g$
and $r^T * g^* \leq r^T * w^*$
shows $w * v \leq v$

proof –

have $w * r \leq (r^T * g^*)^T * (r^T * g^*) * r$
  using assms(6) mult-left-isotone by auto
also have ... $\leq (r^T * g^*)^T * \top$
  by (simp add: mult-assoc mult-right-isotone)
also have ... $= (r^T * g^*)^T$
  by (simp add: assms(1) comp-associative conv-dist-comp)
also have ... $\leq (r^T * w^*)^T$
  by (simp add: assms(7) conv-isotone)
also have ... $= w^T * r$
  by (simp add: conv-dist-comp conv-star-commute)
also have ... $\leq -w * r$
  using assms(4) by (simp add: mult-left-isotone
acyclic-star-below-complement)
also have ... $\leq -(w * r)$
  by (simp add: assms(2) comp-injective-below-complement)
finally have 1: $w * r = \bot$
  by (simp add: le-iff-inf)

have $v = t^T * r$
  by (metis assms(3) conv-dist-comp conv-involutive conv-star-commute)
also have ... $= t^T * v \sqcup r$
  by (simp add: calculation star.circ-loop-fixpoint)
also have ... $\leq w^T * v \sqcup r$
  using assms(5) comp-isotone conv-isotone semiring.add-right-mono by auto
finally have $w * v \leq w * w^T * v \sqcup w * r$
  by (simp add: comp-left-dist-sup mult-assoc mult-right-isotone)
also have ... $= w * w^T * v$
  using 1 by simp
also have ... $\leq v$
  using assms(4) by (simp add: star-left-induct-mult-iff star-sub-one)
finally show ?thesis

qed

4.1 Preservation of Invariant

The following results are used for proving the correctness of Prim’s minimum spanning tree algorithm. We first treat the preservation of the invariant. The following lemma shows that the while-loop preserves that $v$ represents the nodes of the constructed tree. The remaining lemmas in this section show that $t$ is a spanning tree. The exchange property is treated in the following two sections.
lemma \texttt{reachable-inv}:
assumes \texttt{vector v}
and \texttt{e \leq v * -v^T}
and \texttt{e * t = bot}
and \texttt{v^T = r^T * t^*}
shows \texttt{(v \cup e^T * top)^T = r^T * (t \cup e)^*}

proof –
have 1: \texttt{v^T \leq r^T * (t \cup e)^*}
  by (simp add: \texttt{assms(4) mult-right-isotone star.circ-sub-dist})
have 2: \texttt{(e^T * top)^T = top * e}
  by (simp add: \texttt{conv-dist-comp})
also have ... = \texttt{top * (v * -v^T \cap e)}
  by (simp add: \texttt{assms(2) inf-absorb2})
also have ... \leq \texttt{top * (v * top \cap e)}
  using \texttt{inf.sup-left-isotone mult-right-isotone top-greatest by blast}
also have ... = \texttt{top * v^T * e}
  by (simp add: \texttt{comp-inf-vector inf.sup-monoid.add-commute})
also have ... = \texttt{v^T * e}
  using \texttt{assms(1) vector-cone-covector by auto}
also have ... \leq \texttt{r^T * (t \cup e)^* * e}
  using 1 by (simp add: \texttt{mult-left-isotone})
also have ... \leq \texttt{r^T * (t \cup e)^* * (t \cup e)}
  by (simp add: \texttt{mult-right-isotone})
also have ... \leq \texttt{r^T * (t \cup e)^*}
  by (simp add: \texttt{comp-associative mult-right-isotone star.right-plus-below-circ})
finally have 3: \texttt{(v \cup e^T * top)^T \leq r^T * (t \cup e)^*}
  using 1 by (simp add: \texttt{conv-dist-sup})
have \texttt{r^T \leq r^T * t^*}
  using \texttt{sup.bounded-iff star.circ-back-loop-prefix-fixpoint by blast}
also have ... \leq \texttt{(v \cup e^T * top)^T}
  by (metis \texttt{assms(4) conv-isotone sup-ge1})
finally have 4: \texttt{r^T * r^T \leq (v \cup e^T * top)^T}
  .
  have \texttt{(v \cup e^T * top)^T * (t \cup e) = (v \cup e^T * top)^T * t \cup (v \cup e^T * top)^T * e}
    by (simp add: \texttt{mult-left-dist-sup})
  also have ... \leq \texttt{(v \cup e^T * top)^T * t \cup top * e}
    using \texttt{comp-isotone semiring.add-left-mono by auto}
  also have ... = \texttt{v^T * t \cup top * e * t \cup top * e}
    using 2 by (simp add: \texttt{conv-dist-sup mult-right-dist-sup})
  also have ... = \texttt{v^T * t \cup top * e}
    by (simp add: \texttt{assms(3) comp-associative})
  also have ... \leq \texttt{r^T * t^* \cup top * e}
    by (metis \texttt{assms(4) star.circ-back-loop-fixpoint sup-ge1 sup-left-isotone})
  also have ... = \texttt{v^T \cup top * e}
    by (simp add: \texttt{assms(4)})
finally have 5: \texttt{(v \cup e^T * top)^T * (t \cup e) \leq (v \cup e^T * top)^T}
  using 2 by (simp add: \texttt{conv-dist-sup})
have \texttt{r^T * (t \cup e)^* \leq (v \cup e^T * top)^T * (t \cup e)^*}
  using 4 by (simp add: \texttt{mult-left-isotone})
also have ... ≤ (v ∪ e^T * top)^T
  using 5 by (simp add: star-right-induct-mult)
finally show ?thesis
  using 3 by (simp add: inf_eq_iff)
qed

The next result is used to show that the while-loop preserves acyclicity of the constructed tree.

**lemma acyclic-inv:**
assumes acyclic t
  and vector v
  and e ≤ v * -v^T
  and t ≤ v * v^T
shows acyclic (t ⊔ e)

proof –
  have t^+ * e ≤ t^+ * v * -v^T
    by (simp add: assms(3) comp-associative mult-right-isotone)
also have ... ≤ v * v^T * t^* * v * -v^T
    by (simp add: assms(4) mult-left-isotone)
also have ... ≤ v * top * -v^T
    by (metis mult-assoc mult-left-isotone mult-right-isotone top-greatest)
also have ... = v * -v^T
    by (simp add: assms(2))
also have ... ≤ -1
    by (simp add: pp-increasing Schroeder-3-p)
finally have 1: t^+ * e ≤ -1

  have 2: e * t^* = e
    using assms(2-4) et(1) star-absorb by blast
have e^* = 1 ⊔ e ⊔ e * e * e^*
    by (metis star.circ-loop-fixpoint star-square-2 sup-commute)
also have ... = 1 ⊔ e
    using assms(2,3) ee comp-left-zero bot-least sup-absorb1 by simp
finally have 3: e^* = 1 ⊔ e

  have e ≤ v * -v^T
    by (simp add: assms(3))
also have ... ≤ -1
    by (simp add: pp-increasing Schroeder-3-p)
finally have 4: t^+ * e ⊔ e ≤ -1
    using 1 by simp
have (t ⊔ e)^+ = (t ⊔ e) * t^* * (e * t^*)^*
    using star-sup-I mult-assoc by simp
also have ... = (t ⊔ e) * t^* * (1 ⊔ e)
    using 2 3 by simp
also have ... = t^+ * (1 ⊔ e) ⊔ e * t^* * (1 ⊔ e)
    by (simp add: comp-right-dist-sup)
also have ... = t^+ * (1 ⊔ e) ⊔ e * (1 ⊔ e)
    using 2 by simp

40
also have ... = \( t^+ \cup (I \cup e) \cup e \)
using 3 by (metis star-absorb assms(2,3) ee)
also have ... = \( t^+ \cup t^+ \cup e \cup e \)
by (simp add: mult-left-dist-sup)
also have ... \( \leq -1 \)
using 4 by (metis assms(1) sup.absorb1 sup.orderI sup-assoc)
finally show \(?thesis\).

qed

The following lemma shows that the extended tree is in the component reachable from the root.

**Lemma** mst-subgraph-inv-2:

**Assumes** regular \((v \ast v^T)\)
and \(t \leq v \ast v^T \sqcap -g\)
and \(v^T = r^T \ast t^*\)
and \(e \leq v \ast -v^T \sqcap -g\)
and vector \(v\)
and regular \((v \cup e \ast v^T \ast top)\)
shows \(t \cup e \leq (r^T \ast (-((v \cup e \ast v^T \ast top) \ast (v \cup e \ast v^T \ast top) \sqcap g) \ast )) \ast (r^T \ast (-((v \cup e \ast v^T \ast top) \ast (v \cup e \ast v^T \ast top) \sqcap g) \ast ))\)

**Proof**

let \(?v = v \cup e \ast v^T \ast top\)
let \(?G = ?v \ast ?v^T \sqcap g\)
let \(?c = r^T \ast (-(-?G) \ast )\)

have \(v^T \leq r^T \ast (-(-v \ast v^T \sqcap g) \ast )\)
using assms(1-3) inf-pp-commute mult-right-isotone star-isotone by auto
also have ... \(\leq ?c\)
using comp-inf.mult-right-isotone comp-isotone conv-isotone inf.commute mult-right-isotone pp-isotone star-isotone sup.cobounded1 by presburger

finally have 2; \(v^T \leq ?c \land v \leq ?c^T\)
by (metis conv-isotone conv-involutive)

have \(t \leq v \ast v^T\)
using assms(2) by auto

hence 3; \(t \leq ?v^T \ast ?c\)
using 2 order-trans mult-isotone by blast

have \(e \leq v \ast top \sqcap -g\)
by (metis assms(4,5) inf.bounded-iff inf.sup-left-divisibility mult-right-isotone top.extremum)

hence \(e \leq v \ast top \sqcap top \ast e \sqcap -g\)
by (simp add: top-left-mult-increasing inf.boundedI)

hence \(e \leq v \ast top \ast e \sqcap -g\)
by (metis comp-inf-covector inf.absorb2 mult-assoc top.extremum)

hence \(t \cup e \leq (v \ast v^T \sqcap -g) \sqcup (v \ast top \ast e \sqcap -g)\)
using assms(2) sup-mono by blast
also have ... \(= v \ast ?v^T \sqcap -g\)
by (simp add: inf-sup-distrib2 mult-assoc mult-left-dist-sup conv-dist-comp conv-dist-sup)
also have ... \(\leq -?-?G\)
using assms(6) comp-left-increasing-sup inf.sup-left-isotone pp-dist-inf by auto

finally have 4: \( t \sqcup e \leq \_\_\_ G \)

\begin{itemize}
  \item have \( e \leq e \star e^T \star e \)
    \begin{itemize}
      \item by (simp add: ex231c)
    \end{itemize}
  \item also have \( \ldots \leq v \star -v^T \star -v \star v^T \star e \)
    \begin{itemize}
      \item by (metis assms(4) multi-left-isotone conv-isotone conv-dist-comp mult-assoc mult-iso conv-involutive conv-complement inf.boundedE)
    \end{itemize}
\end{itemize}

\begin{itemize}
  \item also have \( \ldots \leq v \star \top \star v^T \star e \)
    \begin{itemize}
      \item by (metis mult-assoc multi-left-isotone mult-right-isotone top.extremum)
    \end{itemize}
  \item also have \( \ldots = v \star r^T \star t^* \star e \)
    \begin{itemize}
      \item using assms(3,5) by (simp add: mult-assoc)
    \end{itemize}
  \item also have \( \ldots \leq v \star r^T \star (t \sqcup e)^* \)
    \begin{itemize}
      \item by (simp add: comp-associative multi-right-isotone star.circ-mul-upper-bound star.circ-sub-dist-1 star-isotone sup-commute)
    \end{itemize}
\end{itemize}

\begin{itemize}
  \item also have \( \ldots \leq v \star ?c \)
    \begin{itemize}
      \item using 4 by (simp add: multi-assoc multi-right-isotone star-isotone)
    \end{itemize}
  \item also have \( \ldots \leq ?c^T \star ?c \)
    \begin{itemize}
      \item using 2 by (simp add: multi-left-isotone)
    \end{itemize}
\end{itemize}

finally show \( \_\_\_ \theta \_\_\_ \)

using 3 by simp

qed

lemma span-inv:
assumes \( e \leq v \star -v^T \)

and vector \( v \)

and atom \( e \)

and \( t \leq (v \star v^T) \sqcap g \)

and \( g^T = g \)

and \( v^T = r^T \star t^* \)

and injective \( r \)

and \( r^T \leq v^T \)

and \( r^T \star ((v \star v^T) \sqcap g)^* \leq r^T \star t^* \)

shows \( r^T \star ((v \star e^T \star \top) \star (v \star e^T \star \top)^T) \sqcap g)^* \leq r^T \star (t \sqcup e)^* \)

proof

let \( ?d = (v \star v^T) \sqcap g \)

have 1: \( (v \sqcup e^T \star \top) \star (v \sqcup e^T \star \top)^T = v \star v^T \sqcup v \star v^T \star e \sqcup e^T \star v \star v^T \sqcup e^T \star e \)

using assms(1-3) ve-dist by simp

have \( t^T \leq ?d^T \)

using assms(4) conv-isotone by simp

also have \( \ldots = (v \star v^T) \sqcap g^T \)

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

also have \( \ldots = ?d \)

by (simp add: assms(5))

finally have 2: \( t^T \leq ?d \)

have \( v \star v^T = (r^T \star t^*)^T \star (r^T \star t^*) \)

42
by (metis assms(6) conv-associative)
also have ... = \( t^{T^*} \cdot (r \cdot r^{T^*}) \cdot t^* \)
  by (simp add: comp-associative conv-dist-comp conv-star-commute)
also have ... \( \leq \) \( t^{T^*} \cdot 1 \cdot t^* \)
  by (simp add: assms(7) mult-left-isotone star-right-induct-mult-iff
star-sub-one)
also have ... = \( t^{T^*} \cdot t^* \)
  by simp
also have ... \( \leq \) \( \bar{d}^* \cdot t^* \)
  using 2 by (simp add: comp-left-isotone star.circ-isotone)
also have ... \( \leq \) \( \bar{d}^* \cdot \bar{d}^* \)
  using assms(4) mult-right-isotone star-isotone by simp
also have 3: ... = \( \bar{d}^* \)
  by (simp add: star.circ-transitive-equal)
finally have 4: \( v \cdot v^{T} \leq \bar{d}^* \)
  by (simp add: conv-associative mult-right-isotone star.circ-plus-same
star.left-plus-below-circ)
have \( r^{T^*} \cdot \bar{d}^* \cdot (v \cdot v^{T} \cap g) \leq r^{T^*} \cdot \bar{d}^* \)
  by (simp add: comp-associative mult-right-isotone star.circ-plus-same
mult-left-dist-sup mult-right-dist-sup)
also have ... \( \leq \) \( r^{T^*} \cdot \bar{d}^* \cdot e \)
  using 3 4 by (metis comp-associative comp-isotone eq-refl)
finally have 6: \( r^{T^*} \cdot \bar{d}^* \cdot (v \cdot v^{T} \cap e) \leq r^{T^*} \cdot \bar{d}^* \cdot e \)
  by (simp add: assms(1,2) mult-assoc mult-right-dist-sup mult-right-zero
sup-bot-right \( vTeT \))
also have ... \( \leq \) \( v^{T} \cdot e^{T} \cdot e \)
  by (simp add: assms(8) comp-isotone)
also have ... = \( bot \)
  using \( vTeT \) assms(1,2) by simp
finally show \( r^{T^*} \cdot (1 \cup v \cdot v^{T}) \cdot e^{T} \cdot x = bot \)
  by (simp add: le-bot)
qed
have \( r^{T^*} \cdot \bar{d}^* \cdot (e^{T} \cdot v \cdot v^{T} \cap g) \leq r^{T^*} \cdot \bar{d}^* \cdot v \cdot v^{T} \)
  by (simp add: comp-associative comp-right-isotone)
also have ... \( \leq \) \( r^{T^*} \cdot (1 \cup v \cdot v^{T}) \cdot e^{T} \cdot v \cdot v^{T} \)
  by (metis assms(2) star.circ-isotone vector-vector-star inf-le1
comp-associative comp-right-isotone comp-left-isotone)
also have ... = \( bot \)
  using 7 by simp
finally have 8: \( r^T * ?d^* * (e^T * v * v^T \sqcap g) = \bot \)
  by (simp add: le-bot)
have \( r^T * ?d^* * (e^T * e \sqcap g) \leq r^T * ?d^* * e \)
  by (simp add: comp-associative comp-right-isotone)
also have \( ... \leq r^T * (1 \sqcup v * v^T) * e \)
  by (metis assms(2) star.circ-isotone vector-vector-star inf-le1
                           comp-associative comp-right-isotone comp-left-isotone)
also have \( ... = \bot \)
  using 7 by simp
finally have 9: \( r^T * ?d^* * (e^T * e \sqcap g) = \bot \)
  by (simp add: le-bot)
have \( r^T * ?d^* * ((v \sqcup e^T * \text{top}) * (v \sqcup e^T * \text{top}) \sqcap g) = r^T * ?d^* * ((v * v^T \sqcup e * e \sqcup e^T * v * v^T \sqcup e^T * e) \sqcap g) \)
  using 1 by simp
also have \( ... = r^T * ?d^* * ((v * v^T \sqcap g) \sqcup (v * v^T * e \sqcap g) \sqcup (e^T * v * v^T \sqcap g) \sqcup (e^T * e \sqcap g)) \)
  by (simp add: inf-sup-distrib2)
also have \( ... = r^T * ?d^* * (v \sqcup v^T \sqcap g) \sqcup r^T * ?d^* * (v \sqcup v^T * e \sqcap g) \sqcup r^T * ?d^* * (e^T * e \sqcap g) \)
  by (simp add: comp-left-dist-sup)
also have \( ... = r^T * ?d^* * (v \sqcup v^T \sqcap g) \sqcup r^T * ?d^* * (v \sqcup v^T * e \sqcap g) \)
  using 8 9 by simp
also have \( ... \leq r^T * ?d^* \sqcup r^T * ?d^* * e \)
  using 5 6 sup.mono by simp
also have \( ... = r^T * ?d^* * (1 \sqcup e) \)
  by (simp add: mult-left-dist-sup)
finally have 10: \( r^T * ?d^* * ((v \sqcup e^T * \text{top}) * (v \sqcup e^T * \text{top}) \sqcap g) \leq r^T * ?d^* * (1 \sqcup e) \)
  by simp
have \( r^T * ?d^* * c * (v * v^T \sqcap g) \leq r^T * ?d^* * c * v * v^T \)
  by (simp add: comp-associative comp-right-isotone)
also have \( ... = \bot \)
  by (metis assms(1,2) comp-associative comp-right-zero ev comp-left-zero)
finally have 11: \( r^T * ?d^* * e * (v * v^T \sqcap g) = \bot \)
  by (simp add: le-bot)
have \( r^T * ?d^* * e * (v * v^T * e \sqcap g) \leq r^T * ?d^* * e * v * v^T * e \)
  by (simp add: comp-associative comp-right-isotone)
also have \( ... = \bot \)
  by (metis assms(1,2) comp-associative comp-right-zero ev comp-left-zero)
finally have 12: \( r^T * ?d^* * e * (v * v^T * e \sqcap g) = \bot \)
  by (simp add: le-bot)
have \( r^T * ?d^* * e * (e^T * v * v^T \sqcap g) \leq r^T * ?d^* * e * e^T * v * v^T \)
  by (simp add: comp-associative comp-right-isotone)
also have \( ... \leq r^T * ?d^* * 1 * v * v^T \)
  by (metis assms(3) atom-injective comp-associative comp-left-isotone
                           comp-right-isotone)
also have \( ... = r^T * ?d^* * v * v^T \)
  by simp
also have \( ... \leq r^T * ?d^* * ?d^* \)
  by simp

44
using 4 by (simp add: mult-right-isotone mult-assoc)
also have ... = r^T * ?d^*
  by (simp add: star.circ-transitive-equivalent comp-associative)
finally have 13: r^T * ?d^* * e * (e^T * v * v^T \sqcap g) ≤ r^T * ?d^*
  have r^T * ?d^* * e * (e^T * e \sqcap g) ≤ r^T * ?d^* * e * e^T * e
  by (simp add: comp-associative comp-right-isotone)
also have ... ≤ r^T * ?d^* * I * e
  by (metis assms(3) atom-injective comp-associative comp-left-isotone)
also have ... = r^T * ?d^* * e
  by simp
finally have 14: r^T * ?d^* * e * (e^T * e \sqcap g) ≤ r^T * ?d^* * e
  have r^T * ?d^* * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) = r^T * ?d^* * e *
  ((v * v^T \sqcup v * v^T \sqcap e * e^T * v * v^T \sqcup e^T * e) \sqcap g)
    using 1 by simp
also have ... = r^T * ?d^* * e * ((v * v^T \sqcap g) \sqcup (v * v^T * e \sqcap g) \sqcup (e^T * v * v^T \sqcap g) \sqcup (v^T * e \sqcap g))
    by (simp add: inf-sup-distrib2)
also have ... = r^T * ?d^* * e * (v * v^T \sqcap g) \sqcup r^T * ?d^* * e * (v * v^T * e \sqcap g) \sqcup r^T * ?d^* * e * (e^T * e \sqcap g)
    by (simp add: comp-left-dist-sup)
also have ... = r^T * ?d^* * e * (e^T * v * v^T \sqcap g) \sqcup r^T * ?d^* * e * (e^T * e \sqcap g)
    using 11 12 by simp
also have ... ≤ r^T * ?d^* \sqcup r^T * ?d^* * e
    using 13 14 sup-mono by simp
also have ... = r^T * ?d^* * (1 \sqcup e)
    by (simp add: mult-left-dist-sup)
finally have 15: r^T * ?d^* * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) ≤ r^T * ?d^* * (1 \sqcup e)
    by simp
have r^T ≤ r^T * ?d^*
  using mult-right-isotone star.circ-reflexive by fastforce
also have ... ≤ r^T * ?d^* * (1 \sqcup e)
  by (simp add: semiring.distrib-left)
finally have 16: r^T ≤ r^T * ?d^* * (1 \sqcup e)
  have r^T * ?d^* * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) = r^T * ?d^* *
  ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \sqcup r^T * ?d^* * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g)
    by (simp add: semiring.distrib-left semiring.distrib-right)
also have ... ≤ r^T * ?d^* * (1 \sqcup e)
    using 10 15 le-supI by simp
finally have r^T * ?d^* * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) ≤ r^T * ?d^* * (1 \sqcup e)
  hence r^T \sqcup r^T * ?d^* * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) ≤ r^T * ?d^* * (1 \sqcup e)
using 16 sup-least by simp
hence \(r^T \ast ((v \sqcup e^T \ast \text{top}) \ast (v \sqcup e^T \ast \text{top})^T \cap g)^* \leq r^T \ast \text{top} \ast e^T \ast (1 \sqcup e)\)
by (simp add: star-right-induct)
also have \(... \leq r^T \ast t^* \ast (t \sqcup e)^*\)
by (simp add: assms(9) mult-left-isotone)
also have \(...) \leq r^T \ast (t \sqcup e)^*\)
by (simp add: star-one-sup-below)
finally show ?thesis.
qed

4.2 Exchange gives Spanning Trees

The following abbreviations are used in the spanning tree application to construct the new tree for the exchange property. It is obtained by replacing an edge with one that has minimal weight and reversing the path connecting these edges. Here, \(w\) represents a weighted graph, \(v\) represents a set of nodes and \(e\) represents an edge.

abbreviation \(E \::\) 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a where \(E w v e \equiv w \cap \ldots \cap v^* \cap \ldots \cap v^T \cap \ldots \cap \text{top} \ast e \ast w^T\ast\)
abbreviation \(P \::\) 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a where \(P w v e \equiv w \cap \ldots \cap v^* \cap \ldots \cap v^T \cap \ldots \cap \text{top} \ast e \ast w^T\ast\)
abbreviation \(EP \::\) 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a where \(EP w v e \equiv w \cap \ldots \cap (EP w v e) \sqcup (P w v e)^T \sqcup e\)

abbreviation \(W \::\) 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a where \(W w v e \equiv (w \cap -(EP w v e)) \sqcup (P w v e)^T \sqcup e\)

The lemmas in this section are used to show that the relation after exchange represents a spanning tree. The results in the next section are used to show that it is a minimum spanning tree.

lemma exchange-injective-3:
assumes \(c \leq v \ast -v^T\)
and vector \(v\)
shows \((w \cap -(EP w v e)) \ast e^T = \text{bot}\)
proof –
have 1: \(\text{top} \ast e \leq -v^T\)
by (simp add: assms schroeder-4-p vTeT)

have \(\text{top} \ast e \leq \text{top} \ast e \ast w^T\ast\)
using sup-right-divisibility star.circ-back-loop-fixpoint by blast
hence \(\text{top} \ast e \leq -v^T \cap \text{top} \ast e \ast w^T\ast\)

using 1 by simp
hence \(\text{top} \ast e \leq -(w \cap -(EP w v e))\)
by (metis inf.assoc inf-import-p le-infl2 p-antitone p-antitone-iff)

hence \((w \cap -(EP w v e)) \ast e^T \leq \text{bot}\)
using p-top schroeder-4-p by blast

thus ?thesis
using le-bot by simp
qed
lemma exchange-injective-6:
assumes atom e
  and forest w
shows \((P \ w \ v \ e)^T * e^T = \bot\)
proof
  have \(e^T * top * e \leq - -1\)
    using assms(1) point-injective by auto
  hence 1: \(e * -1 * e^T \leq \bot\)
    by (metis conv-involutive p-top triple-schroeder-p)
  have \((P \ w \ v \ e)^T * e^T \leq (w \cap top * e * w^T)^T * e^T\)
    using comp-inf.mult-left-isotone conv-dist-inf mult-left-isotone by simp
  also have ... = \((w^T \cap w^T * e^T * top) * e^T\)
    by (simp add: conv-star-commute inf-vector-comp)
  also have ... \leq \((w^T \cap w^T * e^T * top \cap e) * (e^T \cap w^+ * e^T * top)\)
    by (metis dedekind mult-assoc conv-involutive inf-commute)
  also have ... \leq \((w^T * e^T * top \cap e) * (w^+ * e^T * top)\)
    by (simp add: mult-isotone)
  also have ... \leq (top * e) \((w^T * e^T * top)\)
    by (simp add: mult-left-isotone)
  also have ... = top * e * w^+ * e^T * top
    using mult-assoc by simp
  also have ... \leq top * e * -1 * e^T * top
    using assms(2) mult-left-isotone mult-right-isotone by simp
  also have ... \leq bot
    using 1 by (metis le-bot semiring.mult-not-zero mult-assoc)
finally show ?thesis
  using le-bot by simp
qed

The graph after exchanging is injective.

lemma exchange-injective:
assumes atom e
  and \(e \leq v * -v^T\)
  and forest w
  and vector v
shows injective \((W \ w \ v \ e)\)
proof
  have 1: \((w \cap -(EP \ w \ v \ e)) * (w \cap -(EP \ w \ v \ e))^T \leq 1\)
  proof
    have \((w \cap -(EP \ w \ v \ e)) * (w \cap -(EP \ w \ v \ e))^T \leq w * w^T\)
      by (simp add: comp-isotone conv-isotone)
    also have ... \leq 1
      by (simp add: assms(3))
  finally show ?thesis
  qed
have 2: \((w \cap -(EP w v e)) \star (P w v e)^{TT} \leq 1\)

proof -
  have \(\text{top} \star (P w v e)^{T} = \text{top} \star (w^{T} \cap -v \star -v^{T} \cap w^{T \star} \star e^{T} \star \text{top})\)
  by (simp add: conv-associative conv-complement conv-dist-comp)
  also have \(... = \text{top} \star e \star w^{T \star} \star (w^{T} \cap -v \star -v^{T})\)
  by (metis comp-inf-vector conv-dist-comp conv-involutive inf-top-left)
  also have \(... \leq \text{top} \star e \star w^{T \star} \star (w^{T} \cap \text{top} \star -v^{T})\)
  using comp-inf.
star.circ-plus-same mult-right-isotone mult-left-isotone mult-right-isotone
by presburger
also have \(... \leq e \star w^{T \star} \star \text{top} \star e \star w^{T^{*}}\)
by (simp add: comp-associative comp-isotone infboseI1)
also have \(... \leq \text{top} \star e \star 1 \star e \star w^{T^{*}}\)
by (metis comp-left-isotone comp-right-isotone conv-top mult-assoc)
finally have \((P w v e)^{T} \star (P w v e)^{TT} \leq \text{top} \star e \star w^{T \star} \star w^{T^{*}} \cap w^{T} \star w\)
by (simp add: conv-isotone inf.left-commute inf.sup-monoid.add-commute)
also have \(... \leq 1\)
by (simp add: assms(3) forest-separate)
finally show \?thesis.

qed

have 3: \((w \cap -(EP w v e)) \star e^{T} \leq 1\)
by (metis assms(2,4) exchange-injective-3 bot-least)

have 4: \((P w v e)^{T} \star (w \cap -(EP w v e))^{T} \leq 1\)
using 2 conv-dist-comp coreflexive-symmetric by fastforce

have 5: \((P w v e)^{T} \star (P w v e)^{TT} \leq 1\)
proof -
  have \((P w v e)^{T} \star (P w v e)^{TT} \leq (\text{top} \star e \star w^{T^{*}})^{T} \star (\text{top} \star e \star w^{T^{*}})\)
  by (simp add: conv-dist-inf mult-isotone)
  also have \(... = w^{*} \star e^{T} \star \text{top} \star e \star w^{T^{*}}\)
  using conv-star-commute conv-dist-comp conv-involutive conv-top mult-assoc
by presburger
  also have \(... \leq w^{*} \star 1 \star e \star w^{T^{*}}\)
  by (metis comp-left-isotone comp-right-isotone mult-assoc assms(1))
point-injective)
finally have \((P w v e)^{T} \star (P w v e)^{TT} \leq w^{*} \star w^{T^{*}} \cap w^{T} \star w\)
by (simp add: conv-isotone inf.left-commute inf.sup-monoid.add-commute)
mult-isotone)
also have \(... \leq 1\)
by (simp add: assms(3) forest-separate)
finally show \?thesis.

qed

have 6: \((P w v e)^{T} \star e^{T} \leq 1\)
using assms exchange-injective-6 bot-least by simp
have 7: \( e \ast (w \cap -(EP w v e))^T \leq 1 \)
  using 3 by (metis conv-dist-comp conv-involutive coreflexive-symmetric)
have 8: \( e \ast (P w v e)^{TT} \leq 1 \)
  using 6 conv-dist-comp coreflexive-symmetric by fastforce
have 9: \( e \ast e^T \leq 1 \)
  by (simp add: assms(1) atom-injective)
have \((W w v e) \ast (W w v e)^T = (w \cap -(EP w v e))^T \sqcup (w \cap -(EP w v e))^T \sqcup (w \cap -(EP w v e))^T \sqcup (P w v e)^T \sqcup (w \cap -(EP w v e))^T \sqcup (P w v e)^T \ast (w \cap -(EP w v e))^T \sqcup (P w v e)^T \ast e^T \sqcup e \ast (w \cap -(EP w v e))^T \sqcup (P w v e)^T \ast e^T \sqcup e \ast e^T \)
  using comp-left-dist-sup comp-right-dist-sup conv-dist-sup sup assoc by simp
also have ... \leq 1
  using 1 2 3 4 5 6 7 8 9 by simp
finally show \(?thesis \)
  qed

lemma pv:
  assumes vector v
  shows \((P w v e)^T \ast v = bot \)
proof –
  have \((P w v e)^T \ast v \leq (-v \ast -v^T)^T \ast v \)
    by (meson conv-isotone inf-le1 inf-le2 mult-left-isotone order-trans)
also have ... \equal{-v \ast -v^T \ast v \)
    by (simp add: conv-complement conv-dist-comp)
also have ... \equal{bot \)
    by (simp add: assms vector-vector-comp mult-assoc)
finally show \(?thesis \)
    by (simp add: antisym)
  qed

lemma vector-pred-inv:
  assumes atom e
  and \( e \leq v \ast -v^T \)
  and forest w
  and vector v
  and \( w \ast v \leq v \)
  shows \((W w v e) \ast (v \sqcup e^T \ast top) \leq v \sqcup e^T \ast top \)
proof –
  have \((W w v e) \ast e^T \ast top \leq (w \cap -(EP w v e))^T \ast e^T \ast top \sqcup (P w v e)^T \ast e^T \ast top \)
    by (simp add: mult-right-dist-sup)
also have ... \equal{e \ast e^T \ast top \)
    using assms exchange-injective-3 exchange-injective-6 comp-left-zero by simp
also have ... \leq v \ast -v^T \ast e^T \ast top \)
    by (simp add: assms(2) comp-isotone)
also have ... \leq v \ast top \)
    by (simp add: comp-associative mult-right-isotone)
also have ... \equal{v \)

49
by \((\text{simp add: assms(4)})\)

finally have \(1\): \((W \, w \, v \, e) \ast e^T \ast \text{top} \leq v\).

have \((W \, w \, v \, e) \ast v = (w \cap -(EP \, w \, v \, e)) \ast v \cup (P \, w \, v \, e)^T \ast v \cup e \ast v\)
by \((\text{simp add: mult-right-dist-sup})\)
also have \(... = (w \cap -(EP \, w \, v \, e)) \ast v\)
by \((\text{metis assms(2,4) pv ev sup-bot-right})\)
also have \(... \leq w \ast v\)
by \((\text{simp add: mult-left-isotone})\)
finally have \(2\): \((W \, w \, v \, e) \ast v \leq v\)
using \(\text{assms(5) order-trans by blast}\)

have \((W \, w \, v \, e) \ast (v \cup e^T \ast \text{top}) = (W \, w \, v \, e) \ast v \cup (W \, w \, v \, e) \ast e^T \ast \text{top}\)
by \((\text{simp add: semiring.distrib-left mult-assoc})\)
also have \(... \leq v\)
using \(1, 2\) by simp
also have \(... \leq v \cup e^T \ast \text{top}\)
by simp
finally show \(?thesis\)

qed

The graph after exchanging is acyclic.

**lemma** exchange-acyclic:

assumes vector \(v\)
and \(e \leq v \ast -v^T\)
and \(w \ast v \leq v\)
and acyclic \(w\)

shows acyclic \((W \, w \, v \, e)\)

**proof** –

have \(1\): \((P \, w \, v \, e)^T \ast e = \text{bot}\)

proof –

have \((P \, w \, v \, e)^T \ast e \leq (-v \ast -v^T)^T \ast e\)
by \((\text{meson conv-order dual-order.trans inf.cobounded1 inf.cobounded2 mult-left-isotone})\)
also have \(... = -v \ast -v^T \ast e\)
by \((\text{simp add: conv-complement conv-dist-comp})\)
also have \(... \leq -v \ast -v^T \ast v \ast -v^T\)
by \((\text{simp add: assms(2) comp-associative mult-right-isotone})\)
also have \(... = \text{bot}\)
by \((\text{simp add: assms(1) covector-vector-comp mult-assoc})\)
finally show \(?thesis\)
by \((\text{simp add: bot-unique})\)

qed

have \(2\): \(e \ast e = \text{bot}\)
using \(\text{assms(1,2) ee by auto}\)

have \(3\): \((w \cap -(EP \, w \, v \, e)) \ast (P \, w \, v \, e)^T = \text{bot}\)

proof –

have \(\text{top} \ast (P \, w \, v \, e) \leq \text{top} \ast (-v \ast -v^T \cap \text{top} \ast e \ast w^T)^*\)
using \(\text{comp-inf.mult-semi-associative mult-right-isotone by auto}\)

50
also have ... \leq top \ast -v \ast -v^T \sqcap top \ast top \ast e \ast w^{T*}
   by (simp add: comp-inf-covector mult-assoc)
also have ... \leq top \ast -v^T \sqcap top \ast e \ast w^{T*}
   by (meson comp-inf.comp-isotope mult-left-isotone top.extremum)
also have ... = -v^T \sqcap top \ast e \ast w^{T*}
   by (simp add: asms(1) vector-conv-compl)
finally have top \ast (P \ w \ v \ e) \leq -\big(\w \sqcap -EP \ w \ v \ e\big)
   by (metis inf.assoc inf-import-p le-infI2 p-antitone p-antitone-iff)
hence \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big) \ast (P \ w \ v \ e)^T \leq bot
   by (simp add: comp-associative p-top schroeder-4-p)
   using bot-unique by blast
thus \?thesis
have 4: \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e \leq v \ast top
proof –
  have e \leq v \ast top
    using asms(2) dual-order.trans mult-right-isotone top-greatest by blast
hence 5: \w \sqcup w \ast v \ast top \leq v \ast top
   by (simp add: asms(1,3) comp-associative)
have \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e \leq w^+ \ast e
   by (simp add: comp-isotope star-isotone)
also have ... \leq w^+ \ast e
   by (simp add: mult-left-isotope star.left-plus-below-circ)
also have ... \leq v \ast top
   using 5 by (simp add: comp-associative star-left-induct)
finally show \?thesis
qed
have 9: (P \ w \ v \ e)^T \ast \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e = bot
proof –
  have (P \ w \ v \ e)^T \ast \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e \leq (P \ w \ v \ e)^T \ast v \ast top
    using 7 by (simp add: mult-assoc mult-right-isotone)
also have ... = bot
   by (simp add: asms(1) pv)
finally show \?thesis
   using bot-unique by blast
qed
have 10: e \ast \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e = bot
proof –
  have e \ast \big(\w \sqcap -\big(EP \ w \ v \ e\big)\big)^+ \ast e \leq e \ast v \ast top
    using 7 by (simp add: mult-assoc mult-right-isotone)
also have ... \leq v \ast -v^T \ast v \ast top
by (simp add: assms(2) mult-left-isotone)
also have ... = bot
by (simp add: assms(1) covector-vector-comp mult-assoc)
finally show \(?thesis
using bot-unique by blast
qed

have 11: \(e \cdot (P w v e)^T \cdot (w \sqcap - (EP w v e))^* \leq v \cdot -v^T\)
proof
have 12: \(-v^T \cdot w \leq -v^T\)
  by (metis assms(3) conv-complement order-lesseq-imp pp-increasing
schroder-6-p)
  have \(v \cdot -v^T \cdot (w \sqcap - (EP w v e)) \leq v \cdot -v^T \cdot w\)
  by (simp add: comp-isotone star-isotone)
also have ... \(\leq v \cdot -v^T\)
  using 12 by (simp add: comp-isotone comp-associative)
finally have 13: \(v \cdot -v^T \cdot (w \sqcap - (EP w v e)) \leq v \cdot -v^T\)

have 14: \((P w v e)^T \leq -v \cdot -v^T\)
  by (metis conv-complement conv-dist-comp conv-involutive conv-order inf-le1
inf-le2 order-trans)
  have \(e \cdot (P w v e)^T \cdot (w \sqcap - (EP w v e))^* \leq v \cdot -v^T \cdot (P w v e)^T\)
  by (simp add: assms(2) mult-left-isotone)
also have ... \(\leq v \cdot -v^T \sqcup v \cdot -v^T \cdot (P w v e)^T^+\)
  by (metis mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute)
also have ... \(\leq v \cdot -v^T \sqcup v \cdot -v^T \cdot (P w v e)^T^* \cdot (P w v e)^T\)
  by (simp add: mult-assoc star-plus)
also have ... \(\leq v \cdot -v^T \sqcup v \cdot top \cdot -v^T\)
  using 14 mult-assoc mult-right-isotone sup-right-isotone by simp
also have ... \(\leq v \cdot -v^T \sqcup v \cdot -v^T \cdot (P w v e)^T^* \cdot -v \cdot -v^T\)
  by (metis top-greatest mult-right-isotone mult-left-isotone mult-assoc
sup-right-isotone)
also have ... \(\leq v \cdot -v^T\)
  by (simp add: assms(1))
finally have \(e \cdot (P w v e)^T^* \cdot (w \sqcap - (EP w v e))^* \leq v \cdot -v^T \cdot (w \sqcap - (EP w v e))^*\)
  by (simp add: mult-left-isotone)
also have ... \(\leq v \cdot -v^T\)
  using 13 by (simp add: star-right-induct-mult)
finally show \(?thesis

qed

have 15: \((w \sqcap - (EP w v e))^+ \cdot (P w v e)^T^* \cdot (w \sqcap - (EP w v e))^* \leq -1\)
proof
  have \((w \sqcap - (EP w v e))^+ \cdot (P w v e)^T^* \cdot (w \sqcap - (EP w v e))^* \cdot (w \sqcap - (EP w v e))^*\)
  using 5 by simp
  also have ... \(\leq (w \sqcap - (EP w v e))^+\)
  by (simp add: mult-assoc star.circ-transitive-equal)
  also have ... \(\leq w^+\)
by (simp add: comp-isotone star-isotone)
finally show \( \text{thesis} \)
  using assms(4) by simp
qed

have \( 16: (P \ w \ v \ e)^T * (w \cap -(EP \ w \ v \ e))^* * (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \leq -1 \)
  proof -
   have \( (w \cap -(EP \ w \ v \ e))^* * (P \ w \ v \ e)^{T^*} \leq (w \cap -(EP \ w \ v \ e))^* \)
     by (simp add: mult-right-isotone star.left-plus-below-circ)
   also have \( \ldots = (w \cap -(EP \ w \ v \ e))^* \)
     using 5 by simp
   also have \( \ldots \leq w^+ \)
     by (simp add: comp-isotone star-isotone)
   finally have \( (w \cap -(EP \ w \ v \ e))^* * (P \ w \ v \ e)^{T^*} \leq -1 \)
     using assms(4) by simp
   hence \( 17: (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \leq -1 \)
     by (simp add: comp-commute-below-diversity)
   have \( (P \ w \ v \ e)^{T^*} \leq w^{T^*} \)
     by (simp add: comp-isotone conv-dist-inf inf.left-commute inf.sup-monoid.add-commute star-isotone)
   also have \( \ldots \leq -1 \)
     by (simp add: conv-dist-comp conv-star-commute star-plus)
   also have \( \ldots < -1 \)
     using assms(4) conv-complement cone-isotone by force
   finally have \( 18: (P \ w \ v \ e)^{T^*} \leq -1 \)
   have \( (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* * (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     \( = (P \ w \ v \ e)^{T^*} * ((w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     using 6 by (simp add: comp-associative)
   also have \( \ldots = (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     \( \lor (w \cap -(EP \ w \ v \ e))^* \)
     by (simp add: mult-left-dist-sup mult-right-dist-sup)
   also have \( \ldots = (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     \( \lor (w \cap -(EP \ w \ v \ e))^* \)
     by (simp add: mult-associative star.circ-transitive-equal)
   also have \( \ldots = (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     \( \lor (w \cap -(EP \ w \ v \ e))^* \)
     using star-left-unfold-equal by simp
   also have \( \ldots = (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \)
     \( \lor (w \cap -(EP \ w \ v \ e))^* \)
     by (simp add: mult-left-dist-sup sup.left-commute sup-commute)
   also have \( \ldots = ((P \ w \ v \ e)^{T} \lor (P \ w \ v \ e)^{T^*}) * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} \)
     \( \lor (w \cap -(EP \ w \ v \ e))^* \)
     by (simp add: mult-right-dist-sup)
   also have \( \ldots = (P \ w \ v \ e)^{T^*} * (w \cap -(EP \ w \ v \ e))^* \lor (P \ w \ v \ e)^{T^*} \)
     using star.circ-mult-increasing by (simp add: le-iff-sup)
   also have \( \ldots \leq -1 \)
     using 17 18 by simp

53
finally show \( \exists \text{thesis} \)

\[\text{qed}\]
\[\text{have 19: } e * (w \sqcap -(EP w v e))^T * (P w v e)^T * (w \sqcap -(EP w v e))^T * (w \sqcap -(EP w v e))^* \leq -1\]
\[\text{proof} -\]
\[\text{have } e * (w \sqcap -(EP w v e))^T * (P w v e)^T * (w \sqcap -(EP w v e))^* = e * ((w \sqcap -(EP w v e))^T \sqcup (P w v e)^T) * (w \sqcap -(EP w v e))^*\]
\[\text{using 6 by simp add: mult-assoc}\]
\[\text{also have } ... = e * (w \sqcap -(EP w v e))^T * (w \sqcap -(EP w v e))^* \sqcup e * (P w v e)^T * (w \sqcap -(EP w v e))^*\]
\[\text{by simp add: mult-left-dist-sup mult-right-dist-sup}\]
\[\text{also have } ... = e * (w \sqcap -(EP w v e))^T \sqcup e * (P w v e)^T * (w \sqcap -(EP w v e))^*\]
\[\text{by simp add: mult-assoc star.circ-transitive-equal}\]
\[\text{also have } ... \leq e * (P w v e)^T * (w \sqcap -(EP w v e))^T \sqcup e * (P w v e)^T * (w \sqcap -(EP w v e))^*\]
\[\text{using mult-right-isotone star.left-plus-below-circ by auto}\]
\[\text{also have } ... \leq v^T\]
\[\text{using 11 by simp}\]
\[\text{also have } ... \leq -1\]
\[\text{by simp add: pp-increasing schroeder-3-p}\]
\[\text{finally show } \exists \text{thesis}\]

\[\text{qed}\]
\[\text{have 20: } (W w v e) * (w \sqcap -(EP w v e))^T * (P w v e)^T * (w \sqcap -(EP w v e))^T * (w \sqcap -(EP w v e))^* \leq -1\]
\[\text{using 15 16 19 by simp add: comp-right-dist-sup}\]
\[\text{have 21: } (w \sqcap -(EP w v e))^T * e * (P w v e)^T * (w \sqcap -(EP w v e))^* \leq -1\]
\[\text{proof -}\]
\[\text{have } (w \sqcap -(EP w v e))^T * v * -v^T \leq w * v * -v^T\]
\[\text{by simp add: comp-isotone star-isotone}\]
\[\text{also have } ... \leq v^T\]
\[\text{by simp add: assms(3) mult-left-isotone}\]
\[\text{finally have 22: } (w \sqcap -(EP w v e))^T * v * -v^T \leq v * -v^T\]
\[\text{have } (w \sqcap -(EP w v e))^T * e * (P w v e)^T * (w \sqcap -(EP w v e))^* \leq (w \sqcap -(EP w v e))^T * v * -v^T\]
\[\text{using 11 by simp add: mult-right-isotone mult-assoc}\]
\[\text{also have } ... \leq (w \sqcap -(EP w v e))^T * v * -v^T\]
\[\text{using mult-left-isotone star.left-plus-below-circ by blast}\]
\[\text{also have } ... \leq -1\]
\[\text{by simp add: star-left-induct-mult mult-assoc}\]
\[\text{also have } ... \leq -1\]
\[\text{by simp add: pp-increasing schroeder-3-p}\]
\[\text{finally show } \exists \text{thesis}\]
qed

have 23: \((P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast e \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \leq -1

proof =  
  have \((P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast e = (P \, w \, v \, e)^T \ast e \sqcup (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast e
  using comp-left-dist-sup mult-assoc star.circ-loop-fixpoint sup-commute by auto
  
  also have ... = bot
  using 1 9 by simp
  
finally show \(?thesis
  by simp

qed

have 24: \(e \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast e \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \leq -1\)

proof =  
  have \(e \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast e = e \ast e \sqcup e \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast e
  using comp-left-dist-sup mult-assoc star.circ-loop-fixpoint sup-commute by auto
  
  also have ... = bot
  using 2 10 by simp
  finally show \(?thesis
  by simp

qed

have 25: \((W \, w \, v \, e) \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast e \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast \leq -1

  using 21 23 24 by (simp add: comp-right-dist-sup)

have \((W \, w \, v \, e) \ast = ((P \, w \, v \, e)^T \sqcup e \ast (P \, w \, v \, e)^T) \ast ((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast ((P \, w \, v \, e)^T \sqcup e \ast (P \, w \, v \, e)^T)\) \ast

  by (metis star-sup-1 sup.left-commute sup-commute)

also have ... = \(((P \, w \, v \, e)^T \sqcup e \ast (P \, w \, v \, e)^T) \ast ((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast ((P \, w \, v \, e)^T \sqcup e \ast (P \, w \, v \, e)^T)\) \ast

  using 1 2 star-separate by auto

also have ... = \(((P \, w \, v \, e)^T \sqcup e \ast (P \, w \, v \, e)^T) \ast ((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (1 \sqcup e \ast (P \, w \, v \, e)^T)\) \ast

  using 4 mult-left-dist-sup by auto

also have ... = \((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast

  using 3 9 10 star-separate-2 by blast

also have ... = \((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast

  by (simp add: semiring.distrib-left semiring.distrib-right mult-assoc)

finally have \((W \, w \, v \, e) \ast = (W \, w \, v \, e) \ast ((w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast

  by simp

also have ... = \((W \, w \, v \, e) \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (P \, w \, v \, e)^T \ast (w \, \sqcap \, -\,(EP \, w \, v \, e)) \ast (w \, \sqcap \, -\,(EP \, w \, v \, e))\) \ast
by (simp add: comp-left-dist-sup comp-associative)
also have ... ≤ −1
using 20 25 by simp
finally show thesis
.
qed

The following lemma shows that an edge across the cut between visited nodes and unvisited nodes does not leave the component of visited nodes.

lemma mst-subgraph-inv:
  assumes e ≤ v ∗ −vT ∩ g
  and t ≤ g
  and vT = rT ∗ t∗
  shows e ≤ (rT ∗ g∗)T + (rT ∗ g∗) ∩ g
proof –
  have e ≤ v ∗ −vT ∩ g
    by (rule assms(1))
  also have ... ≤ v ∗ (−vT ∩ vT ∗ g) ∩ g
    by (simp add: dedekind-1)
  also have ... ≤ v ∗ vT ∩ g ∩ g
    by (simp add: assms(3))
  also have ... = v ∗ (rT ∗ t∗) ∗ g ∩ g
    by (simp add: assms(3))
  also have ... = (rT ∗ t∗)T ∗ (rT ∗ t∗) ∗ g ∩ g
    by (metis assms(3) conv-involutive)
  also have ... ≤ (rT ∗ t∗)T ∗ (rT ∗ g∗) ∗ g ∩ g
    using assms(2) comp-inf mult-left-isotone comp-isotone star-isotone by auto
  also have ... ≤ (rT ∗ t∗)T ∗ (rT ∗ g∗) ∩ g
    using inf.sup-right-isotone inf-commute mult-assoc mult-right-isotone
    star.left-plus-below-circ star-plus by presburger
  also have ... ≤ (rT ∗ g∗)T ∗ (rT ∗ g∗) ∩ g
    using assms(2) comp-inf mult-left-isotone conv-dist-comp conv-isotone
    mult-left-isotone star-isotone by auto
  finally show thesis
.
qed

The following lemmas show that the tree after exchanging contains the currently constructed and tree and its extension by the chosen edge.

lemma mst-extends-old-tree:
  assumes t ≤ w
  and t ≤ v ∗ vT
  and vector v
  shows t ≤ W w v e
proof –
  have t ∩ EP w v e ≤ t ∩ −vT
    by (simp add: inf.coboundedI2 inf.sup-monoid.add-assoc)
  also have ... ≤ v ∗ vT ∩ −vT
    by (simp add: assms(2) inf.coboundedI1)
also have \( \leq \) \( \text{bot} \)
by (simp add: assms(\ref{eq:refl}) covector-vector-comp eq-refl schroeder-2)

finally have \( t \leq -(EP \ w \ v \ e) \)
using le-\( \text{bot} \) pseudo-complement by blast

hence \( t \leq w \wedge -(EP \ w \ v \ e) \)
using assms(\ref{eq:refl}) by simp

thus \( ?\text{thesis} \)
by (simp add: le-supI2 sup-commute)

qed

lemma \( \text{mst-extends-new-tree}: \)
\( t \leq w \implies t \leq v \ast v \)
\( T = \implies \text{vector } v \)
\( \leq \wedge -(EP \ w \ v \ e) \)

using mst-extends-old-tree by auto

The following lemma shows that the nodes reachable in the tree after exchange contain the nodes reachable in the tree before exchange.

lemma \( \text{mst-reachable-inv}: \)
assumes regular \( (EP \ w \ v \ e) \)
and vector \( r \)
and \( e \leq v \ast -v \)
and vector \( v \)
and \( v \)
\( T = \ast t \)
and \( t \leq w \)
and \( t \leq v \ast v \)
and \( w \ast v \leq v \)

shows \( r \ast e \leq r \ast (W \ w \ v \ e) \)

proof –

have 1: \( r \ast \leq r \ast (W \ w \ v \ e) \)
using sup.bounded-iff star.circ-back-loop-prefixpoint by blast

have top \ast e \ast (w \wedge -v) \ast \wedge -v \ast (w \wedge -v) \ast \ast (w \wedge -v)
by (simp add: assms(4) covector-comp-inf vector-conv-compl)

also have \( \leq \text{top} \ast e \ast (w \wedge -v) \)
by (simp add: comp-isotone mult-assoc star.circ-plus-same star.left-plus-below-circ)

finally have 2: \( \text{top} \ast e \ast (w \wedge -v) \ast \wedge -v \leq \text{top} \ast e \ast (w \wedge -v) \ast \wedge -v \)

by (simp add: shunting-var-p)

have 3: \( -v \ast w \leq \text{top} \ast e \ast (w \wedge -v) \ast \wedge -v \)
by (metis assms(8) conv-dist-comp conv-order mult-assoc order pp-comp-semi-commute pp-isotone sup.coboundedI1 sup-commute)

have 4: \( \text{top} \ast e \leq \text{top} \ast e \ast (w \wedge -v) \ast \wedge -v \)
using sup-right-divisibility star.circ-back-loop-fixpoint le-supI1 by blast

have \( (\text{top} \ast e \ast (w \wedge -v) \ast \wedge -v) \ast w \ast (w \wedge -v) \ast w \ast (w \wedge -v) \ast w \ast \)
\( \wedge -v \ast w \)

by (simp add: comp-right-dist-sup)

also have \( \leq \text{top} \ast e \ast (w \wedge -v) \ast \wedge -v \)
using 2 3 by simp

finally have \( \text{top} \ast e \ast (\text{top} \ast e \ast (w \wedge -v) \ast \wedge -v) \ast w \leq \text{top} \ast e \ast \)
\[(w^T \cap -v^T)^* \sqcup -v^T\]

using 4 by simp

hence 5: \(top * e * w^T * \leq top * e * (w^T \cap -v^T)^* \sqcup -v^T\)

by (simp add: star-right-induct)

have 6: \(top * e \leq top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*\)

using sup-right-divisibility star.circ-back-loop-fixpoint by blast

have \((top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* )^T \leq (top * e * w^T^*)^T\)

by (simp add: star-isotone mult-right-isotone conv-isotone inf-assoc)

also have \(\ldots = w^* * e^T * top\)

by (simp add: conv-dist-comp conv-star-commute mult-assoc)

finally have 7: \((top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*)^T \leq w^* * e^T * top\)

have \((top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*)^T \leq (top * e * (-v * -v^T)^*)^T\)

by (simp add: conv-isotone inf-commute mult-right-isotone star-isotone le-infI2)

also have \(\ldots \leq (top * v * -v^T * (-v * -v^T)^*)^T\)

by (metis assms(3) conv-isotone mult-left-isotone mult-right-isotone mult-assoc)

also have \(\ldots = (top * v * (-v^T * -v)^* * -v^T)^T\)

by (simp add: mult-assoc star-slide)

also have \(\ldots \leq (top * -v^T)^T\)

using conv-order mult-left-isotone by auto

also have \(\ldots = -v\)

by (simp add: assms(4) conv-complement vector-conv-compl)

finally have 8: \((top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*)^T \leq w^* * e^T * top \cap -v\)

using 7 by simp

have covector \((top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*)^T\)

by (simp add: covector-mult-closed)

hence \(top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T \cap -v^T) = top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T \cap -v^T \sqcap (top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*^T))\)

by (metis comp-inf-vector-1 inf.idem)

also have \(\ldots \leq (top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T \cap -v^T \sqcap w^* * e^T * top \cap -v)\)

using 8 mult-right-isotone inf.sup-right-isotone inf-assoc by simp

also have \(\ldots = top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T \cap (-v \cap -v^T) \sqcap w^* * e^T * top)\)

using inf-assoc by (simp add: inf-assoc)

also have \(\ldots = top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)\)

using assms(4) conv-complement vector-complement-closed vector-covector by fastforce

also have \(\ldots \leq (top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^*^T\)

by (simp add: comp-associative comp-isotone star.circ-plus-same star.left-plus-below-circ)

finally have 9: \(top * e \sqcup top * e * (w^T \cap -v * -v^T \sqcap w^* * e^T * top)^* * (w^T\)
\( \square -v^T \leq \text{top} * e * (w^T \cap -v * -v^T \cap w^* * e^T * \text{top})^* \)

using 6 by simp

have \( EP w v e \leq -v^T \cap \text{top} * e * w^T \)

using \( \text{inf.sup-left-isotone by auto} \)

also have \( \ldots \leq \text{top} * e * (w^T \cap -v * -v^T \cap w^* * e^T * \text{top})^* \)

using 5 by \( \text{metis inf-commute shunting-ear-p} \)

also have \( \ldots \leq \text{top} * e * (w^T \cap -v * -v^T \cap w^* * e^T * \text{top})^* \)

using 9 by \( \text{simp add: star-right-induct} \)

finally have \( 10: EP w v e \leq \text{top} * e * (P w v e)^T \)

by \( \text{simp add: conv-complement conv-dist-comp conv-dist-inf} \)

conv-star-commute mult-assoc

have \( \text{top} * e = \text{top} * (v * -v^T \cap e) \)

by \( \text{simp add: assms(3) inf.absorb2} \)

also have \( \ldots \leq \text{top} * (v * \text{top} \cap e) \)

using \( \text{inf.sup-right-isotone inf-commute mult-right-isotone top-greatest by presburger} \)

also have \( \ldots = (\text{top} \cap (v * \text{top})^T) * e \)

using \( \text{assms(4) covector-inf-comp-3 by presburger} \)

also have \( \ldots = \text{top} * v^T * e \)

by \( \text{simp add: conv-dist-comp} \)

also have \( \ldots = \text{top} * r^T * t^* * e \)

by \( \text{simp add: assms(5) comp-associative} \)

also have \( \ldots \leq \text{top} * r^T * (W w v e)^* * e \)

by \( \text{metis assms(4,6,7) mst-extends-old-tree star-isotone mult-left-isotone} \)

mult-right-isotone

finally have \( 11: \text{top} * e \leq \text{top} * r^T * (W w v e)^* * e \)

have \( r^T * (W w v e)^* * (EP w v e) \leq r^T * (W w v e)^* * (\text{top} * e * (P w v e)^T) \)

using 10 mult-right-isotone by blast

also have \( \ldots = r^T * (W w v e)^* * \text{top} * e * (P w v e)^T \)

by \( \text{simp add: mult-assoc} \)

also have \( \ldots \leq \text{top} * e * (P w v e)^T \)

by \( \text{metis comp-associative comp-inf-covector inf.idem} \)

inf.sup-right-divisibility

also have \( \ldots \leq \text{top} * r^T * (W w v e)^* * e * (P w v e)^T \)

using 11 by \( \text{simp add: mult-left-isotone} \)

also have \( \ldots = r^T * (W w v e)^* * e * (P w v e)^T \)

using \( \text{assms(2) vector-covector by auto} \)

also have \( \ldots \leq r^T * (W w v e)^* * (W w v e)^* * (P w v e)^T \)

by \( \text{simp add: mult-left-isotone mult-right-isotone} \)

also have \( \ldots \leq r^T * (W w v e)^* * (W w v e)^* \)

by \( \text{meson dual-order.trans mult-right-isotone star-isotone sup-ge1 sup-ge2} \)

also have \( \ldots \leq r^T * (W w v e)^* \)

by \( \text{metis mult-assoc mult-right-isotone star.circ-transitive-equal} \)

star.left-plus-below-circ

finally have \( 12: r^T * (W w v e)^* * (EP w v e) \leq r^T * (W w v e)^* \)

have \( r^T * (W w v e)^* * w \leq r^T * (W w v e)^* * (w \square EP w v e) \)

59
4.3 Exchange gives Minimum Spanning Trees

The lemmas in this section are used to show that the after exchange we obtain a minimum spanning tree. The following lemmas show various interactions between the three constituents of the tree after exchange.

**lemma epm-1:**

vector v $\Rightarrow$ E w v e $\sqcup$ P w v e = EP w v e

by (metis inf-commute inf-sup-distrib1 mult-assoc mult-right-dist-sup regular-closed-p regular-complement-top vector-conv-compl)

**lemma epm-2:**

assumes regular (EP w v e)

and vector v

shows (w $\sqcap$ -(EP w v e)) $\sqcup$ P w v e $\sqcup$ E w v e = w

proof –

have (w $\sqcap$ -(EP w v e)) $\sqcup$ P w v e $\sqcup$ E w v e = (w $\sqcap$ -(EP w v e)) $\sqcup$ EP w v e

using epm-1 sup-assoc sup-commute assms(2) by (simp add: inf-sup-distrib1)

also have ... = w $\sqcup$ EP w v e

by (metis assms(1) inf-top-right-neutral regular-complement-top sup-inf-distrib2)

also have ... = w

by (simp add: sup-inf-distrib1)

finally show ?thesis

qed

**lemma epm-4:**

by (simp add: inf-assoc)

also have ... = $r^T$ * (W w v e)* * ((w $\sqcup$ EP w v e) $\sqcap$ -(EP w v e) $\sqcup$ EP w v e)

by (metis assms(1) inf-top-right stone)

also have ... = $r^T$ * (W w v e)* * (w $\sqcap$ -(EP w v e)) $\sqcup$ $r^T$ * (W w v e)* * (EP w v e)

by (simp add: sup-inf-distrib2)

also have ... = $r^T$ * (W w v e)* * (w $\sqcap$ -(EP w v e)) $\sqcup$ $r^T$ * (W w v e)* * (EP w v e)

by (simp add: comp-left-dist-sup)

also have ... $\leq$ $r^T$ * (W w v e)* * (W w v e) $\sqcup$ $r^T$ * (W w v e)* * (EP w v e)

using mult-right-isotone sup-left-isotone by auto

also have ... $\leq$ $r^T$ * (W w v e)* $\sqcup$ $r^T$ * (W w v e)* * (EP w v e)

using mult-assoc mult-right-isotone star.circ-plus-same

star.left-plus-below-circ sup-left-isotone by auto

also have ... = $r^T$ * (W w v e)*

using 12 sup.absorb1 by blast

finally have $r^T$ $\sqcup$ $r^T$ * (W w v e)* * w $\leq$ $r^T$ * (W w v e)*

using 1 by simp

thus ?thesis

by (simp add: star-right-induct)

qed
assumes $e \leq w$
and injective $w$
and $w * v \leq v$
and $e \leq v * -v^T$
shows $\top * e * w^T \leq \top * v^T$

proof –

have $w^* \leq v$
  by (simp add: assms(3) star-left-induct-mult)

hence $1: v^T * w^T \leq v^T$
  using conv-star-commute conv-dist-comp conv-isotone by fastforce

have $e * w^T \leq w * w^T \cap e * w^T$
  by (simp add: assms(1) mult-left-isotone)

also have $... \leq 1 \cap e * w^T$
  using assms(2) inf.sup-left-isotone by auto

also have $... = 1 \cap w * e^T$
  using calculation conv-dist-comp conv-involutive coreflexive-symmetric by fastforce

also have $... \leq w * e^T$
  by simp

also have $... \leq w * -v * v^T$
  by (metis assms(4) conv-complement conv-dist-comp conv-involutive conv-order mult-assoc mult-right-isotone)

also have $... \leq \top * v^T$
  by (simp add: mult-left-isotone)

finally have $\top * e * w^T \leq \top * v^T * w^T$
  by (metis antisym comp-associative comp-isotone dense-top-closed
   mult-left-isotone transitive-top-closed)

also have $... \leq \top * v^T$
  using $1$ by (simp add: mult-assoc mult-right-isotone)

finally show $?thesis$
.

qed

lemma epm-5:
assumes $e \leq w$
and injective $w$
and $w * v \leq v$
and $e \leq v * -v^T$
and vector $v$
shows $P \ w \ v \ e = \text{bot}$

proof –

have $1: e = w \cap \top * e$
  by (simp add: assms(1,2) epm-3)

have $2: \top * e * w^T \leq \top * v^T$
  by (simp add: assms(1-4) epm-4)

have $3: -v * -v^T \cap \top * v^T = \text{bot}$
  by (simp add: assms(5) comp-associative covector-vector-comp
  inf.sup-monoid.add-commute schroeder-2)

have $P \ w \ v \ e = (w \cap -v * -v^T \cap \top * e) \sqcup (w \cap -v * -v^T \cap \top * e * w^T)$


by (metis inf-sup-distrib1 mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute)
also have ... \leq (e \cap -v \ast -v^T) \cup (w \cap -v \ast -v^T \cap top \ast e \ast w^{T'})
using 1 by (metis comp-inf.mult-semi-associative inf.sup-monoid.add-commute semiring.add-right-mono)
also have ... \leq (e \cap -v \ast -v^T) \cup (w \cap -v \ast -v^T \cap top \ast v^T)
using 2 by (metis sup-right-isotone inf.sup-right-isotone)
also have ... \leq (e \cap -v \ast -v^T) \cup (-v \ast -v^T \cap top \ast v^T)
using inf.sup-monoid mult-right-isotone inf-sup-distrib1 mult-assoc top-greatest by blast
also have ... = bot
using assms(5) inf-compl-bot vector-complement-closed by auto
finally show \( ? \text{thesis} \)
by (simp add: le-iff-inf)
qed

lemma epm-6:
assumes e \leq w
and injective w
and w \ast v \leq v
and e \leq v \ast -v^T
and vector v
shows \( E w v e = e \)
proof –
  have 1: e \leq - - v \ast -v^T
    using assms(4) mult-isotone order-lesseq-imp pp-increasing by blast
  have 2: top \ast e \ast w^{T'} \leq top \ast v^T
    by (simp add: assms(1-4) epm-4)
  have 3: e = w \cap top \ast e
    by (simp add: assms(1,2) epm-3)
  hence e \leq top \ast e \ast w^{T'}
    by (metis le-infI2 star.circ-back-loop-fixpoint sup.commute sup-commute sup-ge1)
  hence 4: e \leq E w v e
    using 1 by (simp add: assms(1))
  have 5: - - v \ast -v^T \cap top \ast v^T = bot
    by (simp add: assms(5) comp-associative covector-vector-comp inf.sup-monoid.add-commute Schroeder-2)
  have \( E w v e = (w \cap - - v \ast -v^T \cap top \ast e) \cup (w \cap - - v \ast -v^T \cap top \ast e \ast w^{T'}) \)
    by (metis inf-sup-distrib1 mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute)
  also have ... \leq (e \cap - - v \ast -v^T) \cup (w \cap - - v \ast -v^T \cap top \ast e \ast w^{T'})
    using 3 by (metis comp-inf.mult-semi-associative inf.sup-monoid.add-commute semiring.add-right-mono)
  also have ... \leq (e \cap - - v \ast -v^T) \cup (w \cap - - v \ast -v^T \cap top \ast v^T)
    using 2 by (metis sup-right-isotone inf.sup-right-isotone)
also have \( \leq (e \cap -v * -v^T) \sqcup (-v * -v^T \cap \text{top} * v^T) \)
using `inf.assoc le-infI2` by `auto`
also have \( \leq e \)
by `(simp add: 5)`
finally show `?thesis`
using `4` by `(simp add: antisym)`

**lemma epm-7:**

regular \((EP w v e) \Rightarrow e \leq w \Rightarrow \text{injective w} \Rightarrow w * v \leq v \Rightarrow e \leq v * -v^T \Rightarrow \text{vector v} \Rightarrow W w v e = w \)`
by `(metis conv-bot epm-2 epm-5 epm-6)`

**lemma epm-8:**

assumes `acyclic w` shows \((w \sqcap -(EP w v e)) \sqcap (P w v e)^T = \text{bot} \)`
proof –
have \((w \sqcap -(EP w v e)) \sqcap (P w v e)^T \leq w \sqcap w^T \)`
by `(meson conv-isotone inf-le1 inf-mono order-trans)`
thus `?thesis`
by `(metis assms acyclic-asymmetric inf.commute le-bot)`

**lemma epm-9:**

assumes `e \leq v * -v^T` and `vector v` shows \((w \sqcap -(EP w v e)) \sqcap e = \text{bot} \)`
proof –
have `1: e \leq -v^T` by `(metis assms complement-conv-sub vector-conv-covector ev p-antitone-iff p-bot)`

have \((w \sqcap -(EP w v e)) \sqcap e = (w \sqcap -v^T \sqcap e) \sqcup (w \sqcap -(\text{top} * e * w^T^*) \sqcap e) \)`
by `(simp add: inf-commute inf-sup-distrib1)`
also have \(\leq (\neg v^T \sqcap e) \sqcup -(\text{top} * e * w^T^*) \sqcap e \)`
using `comp-inf.mutl-left-isotone inf.cobounded2 semiring.add-mono` by `blast`
also have `1` by `(metis inf.sup-relative-same-increasing inf-commute inf-sup-distr1 madluz-3-13 regular-closed-p)`
also have \(\leq \text{bot} \)`
by `(metis inf.sup-relative-same-increasing inf-bot-right inf-commute inf-p mul-left-isotone star-outer-increasing top-greatest)`
finally show `?thesis`
by `(simp add: le-iff-inf)`

**lemma epm-10:**

assumes `e \leq v * -v^T` and `vector v`
shows \((P \ w \ v \ e)^T \sqcap e = \bot\)

proof –

have \((P \ w \ v \ e)^T \leq -v * -v^T\)
  by (simp add: conv-complement conv-dist-comp conv-dist-inf inf.absorb-iff1 inf.left-commute inf-commute)

hence \((P \ w \ v \ e)^T \sqcap e = \bot\)
  using assms(1) inf-mono by blast

also have \(...) \leq -v * top \sqcap v * top\)
  using inf.sup-mono mult-right-isotone top-greatest by blast

also have \(...) = \bot\)
  using assms(2) inf-compl-bot vector-complement-closed by auto

finally show \?thesis
  by (simp add: le-iff-inf)

qed

lemma epm-11:
  assumes vector \(v\)
  shows \((w \sqcap -(EP \ w \ v \ e)) \sqcap P \ w \ v \ e = \bot\)

proof –

have \(P \ w \ v \ e \leq EP \ w \ v \ e\)
  by (metis assms comp-isotone inf.sup-left-isotone inf.sup-right-isotone order.refl top-greatest vector-conv-compl)

thus \?thesis
  using inf-le2 order-trans p-antitone pseudo-complement by blast

qed

lemma epm-12:
  assumes vector \(v\)
  shows \((w \sqcap -(EP \ w \ v \ e)) \sqcap E \ w \ v \ e = \bot\)

proof –

have \(E \ w \ v \ e \leq EP \ w \ v \ e\)
  by (metis assms comp-isotone inf.sup-left-isotone inf.sup-right-isotone order.refl top-greatest vector-conv-compl)

thus \?thesis
  using inf-le2 order-trans p-antitone pseudo-complement by blast

qed

lemma epm-13:
  assumes vector \(v\)
  shows \(P \ w \ v \ e \sqcap E \ w \ v \ e = \bot\)

proof –

have \(P \ w \ v \ e \sqcap E \ w \ v \ e \leq -v * -v^T \sqcap -v * -v^T\)
  by (meson dual-order.trans inf.cobounded1 inf.sup-mono inf.le2)

also have \(...) \leq -v * top \sqcap -v * top\)
  using inf.sup-mono mult-right-isotone top-greatest by blast

also have \(...) = \bot\)
  using assms inf-compl-bot vector-complement-closed by auto

finally show \?thesis
  by (simp add: le-iff-inf)
The following lemmas show that the relation characterising the edge across the cut is an atom.

**Lemma atom-edge-1:**

assumes \( e \leq v \ast -v^T \cap g \)
and vector \( v \)
and \( v^T = r^T \ast t^* \)
and \( t \leq g \)
and \( r^T \ast g^* \leq r^T \ast w^* \)

shows \( \top \ast e \leq v^T \ast w^* \)

**Proof:**

- have \( \top \ast e \leq \top \ast (v \ast -v^T \cap g) \)
  using \( \text{assms}(1) \) \text{mult-right-isotone} by auto
- also have \( \ldots \leq \top \ast (v \ast \top \cap g) \)
  using \( \text{inf.sup-right-isotone inf-commute mult-right-isotone top-greatest} \) by presburger
- also have \( \ldots = v^T \ast g \)
  by \( \text{(metis assms(2) covector-inf-comp-3 inf-top.left-neutral)} \)
- also have \( \ldots = r^T \ast t^* \ast g \)
  by \( \text{(simp add: assms(3))} \)
- also have \( \ldots \leq r^T \ast g^* \ast g \)
  by \( \text{(simp add: assms(4) mult-left-isotone mult-right-isotone star-isotone)} \)
- also have \( \ldots \leq r^T \ast g^* \)
  by \( \text{(simp add: mult-assoc mult-right-isotone star.right-plus-below-circ)} \)
- also have \( \ldots \leq r^T \ast w^* \)
  by \( \text{(simp add: assms(5))} \)
- also have \( \ldots \leq v^T \ast w^* \)
  by \( \text{(metis assms(3) mult-left-isotone mult-right-isotone mult-1-right star.circ-reflexive)} \)
- finally show \( ?\text{thesis} \)

**QED**

**Lemma atom-edge-2:**

assumes \( e \leq v \ast -v^T \cap g \)
and vector \( v \)
and \( v^T = r^T \ast t^* \)
and \( t \leq g \)
and \( r^T \ast g^* \leq r^T \ast w^* \)
and \( w \ast v \leq v \)
and injective \( w \)

shows \( \top \ast e \ast w^T \ast x \leq v^T \ast w^* \)

**Proof:**

- have \( 1: \top \ast e \leq v^T \ast w^* \)
  using \( \text{assms}(1-5) \) \text{atom-edge-1} by blast
- have \( v^T \ast w^* \ast w^T = v^T \ast w^T \cup v^T \ast w^+ \ast w^T \)
  by \( \text{(metis mult-assoc mult-left-dist-sap star.circ-loop-firpoint sup-commute)} \)
- also have \( \ldots \leq v^T \cup v^T \ast w^+ \ast w^T \)

**QED**
by (metis assms(6) conv-dist-comp conv-isotone sup-left-isotone)
also have ... = v^T \sqcap v^T \ast w^* \ast (w \ast w^T)
  by (metis mult-assoc star-plus)
also have ... \leq v^T \sqcap v^T \ast w^*
  by (metis assms(7) mult-right-isotone mult-1-right sup-right-isotone)
also have ... = v^T \ast w^*
  by (metis star.circ-back-loop-fixpoint sup-absorb2 sup-ge2)
finally show ?thesis
  using 1 star-right-induct by auto
qed

lemma atom-edge-3:
  assumes e \leq v \ast -v^T \sqcap g
  and vector v
  and v^T = r^T \ast t^*
  and t \leq g
  and v^T \ast g^* \leq r^T \ast w^*
  and w \ast v \leq v
  and injective w
  and E w v e = bot
  shows e = bot
proof -
have bot = E w v e
  by (simp add: assms(8))
also have ... = w \sqcap -v \ast top \sqcap top \ast -v^T \sqcap top \ast e \ast w^T*
    by (metis assms(2) comp-inf-covector inf.assoc inf-top.left-neutral vector-conv-compl)
also have ... = w \sqcap top \ast e \ast w^T* \sqcap -v^T \sqcap -v
  using assms(2) inf.assoc inf.commute vector-conv-compl vector-complement-closed by (simp add: inf-assoc)
finally have 1: w \sqcap top \ast e \ast w^T* \sqcap -v^T \leq -v
  using shunting-1-pp by force
have w^* \ast e^T \ast top = (top \ast e \ast w^T*)^T
  by (simp add: conv-star-commute comp-associative conv-dist-comp)
also have ... \leq (w^T \ast w^*)^T
  using assms(1-7) atom-edge-2 by (simp add: conv-isotone)
also have ... = w^T* \ast v
  by (simp add: conv-star-commute conv-dist-comp)
finally have 2: w^* \ast e^T \ast top \leq w^T* \ast v
  using assms(2) conv-complement covector-inf-comp-3 inf-top.right-neutral vector-complement-closed
also have ... \leq -v \ast top
  using 1 by (simp add: comp-isotone)
also have ... = -v
  using assms(2) vector-complement-closed by auto

66
finally have \((w^T \cap w^* \ast e^T \ast \top) \ast \neg\neg v \leq \neg\neg v\)
using p-antitone-iff schroeder-3-p by auto
hence \(w^* \ast e^T \ast \top \cap w^T \ast \neg\neg v \leq \neg\neg v\)
by (simp add: inf-vector-comp)
hence \(3: w^T \ast \neg\neg v \leq \neg\neg v \sqcup -(w^* \ast e^T \ast \top)\)
by (simp add: inf.commute shunting-p)
also have \(\ldots \leq -v \sqcup -(w^* \ast e^T \ast \top)\)
by simp
also have \(w^T \ast ((w^* \ast e^T \ast \top)) \leq -v \sqcup -(w^* \ast e^T \ast \top)\)
using 3 by (simp add: mult-left-dist-sup)
hence \(w^T \ast (-v \sqcup -(w^* \ast e^T \ast \top)) \leq -v \sqcup -(w^* \ast e^T \ast \top)\)
using star-left-induct-mult-iff by blast
hence \(w^* \ast e^T \ast \top \cap w^T \ast \neg\neg v \leq \neg\neg v\)
by (simp add: inf-commute shunting-p)
hence \(w^* \ast e^T \ast \top \leq \neg\neg v\)
using 2 by (metis inf.absorb1 p-antitone-iff p-comp-pp vector-export-comp)
hence \(4: e^T \ast \top \leq \neg\neg v\)
by (metis mult-assoc star.circ-loop-fixpoint sup.bounded-iff)
also have \(\ldots \leq -v \ast \top\)
by (simp add: conv-complement conv-dist-comp mult-assoc mult-right-isotone)
also have \(\ldots = -v\)
using assms(2) vector-complement-closed by auto
finally have \(e^T \ast \top \leq \bot\)
using 4 shunting-1-pp by auto
hence \(e^T = \bot\)
using antisym bot-least top-right-mult-increasing by blast
thus \(?\thesis\)
using conv-bot by fastforce
qed
by simp add: inf.orderE mult-right-dist-sup
have \( \leq -v^T \cap v^T \cap w^* \)
  using assms(1) inf.sup-left-isotone mult-assoc right-isotone by auto
also have \( \leq \top \cap -v^T \cap v^T \cap w^* \)
  using inf.sup-left-isotone mult-left-isotone top-greatest by blast
also have \( = -v^T \cap v^T \cap w^* \)
  by (simp add: assms(2) vector-conv-compl)
also have \( \leq -v^T \cap (w \cap -v^T) \cap w^* \)
  using 3 by simp
also have \( = (\top \cap (\top \cap (\top \cap -v^T)) \cap (w \cap -v^T)) \cap w^* \)
  by (simp add: conv-complement)
also have \( = \top \cap (w \cap -v \cap -v^T) \cap w^* \)
  using assms(2) vector-inf-comp-3 inf-assoc inf-left-commute
vector-complement-closed by presburger
also have \( = \top \cap (w \cap -v \cap -v^T) \cap w^* \)
  by (metis assms(2) vector-complement-closed conv-complement inf-assoc vector-conv-covector)
finally have \( \leq \top \cap (\top \cap (\top \cap -v^T) \cap (w \cap -v^T)) \cap w^* \)
  by (metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top)
hence \( \leq \top \cap (w \cap -v \cap -v^T) \cap w^* \cap (e^T \cap \top) \)
  using assms(6) shunt-bijection by blast
also have \( = \top \cap (w \cap -v \cap -v^T) \cap (\top \cap e \cap w^*T) \cap \top \)
  by (simp add: conv-dist-comp top-mult-assoc)
also have \( = \top \cap (w \cap -v \cap -v^T \cap \top \cap e \cap w^*T \cap \top) \cap \top \)
  by (simp add: conv-inf-vector-1 mult-assoc)
finally show thesis
  by (simp add: conv-star-commute top-le)
qed
lemma atom-edge-5:
assumes vector v
and w ∗ v ≤ v
and injective w
and atom e
shows (E w v e)T * top * E w v e ≤ 1

proof –
have 1: eT * top * e ≤ 1
  by (simp add: assms(4) point-injective)
have E w v e ≤ v * top
  by (simp add: inf-commute le-infI2 mult-right-isotone)
hence 2: E w v e ≤ −−v
  by (simp add: assms(1) vector-complement-closed)
have 3: w * −−v ≤ −−v
  by (simp add: assms(2) p-antitone p-antitone-iff)
have w ∩ top * E w v e ≤ w * (E w v e)T * E w v e
  by (metis dedekind-2 inf.commute inf-top.ltop-neutral)
also have ... ≤ w * wT * E w v e
  by (simp add: conv-isotone le-infI1 mult-right-isotone)
also have ... ≤ E w v e
  by (metis assms(3) mult-left-isotone mult-left-one)
finally have 4: w ∩ top * E w v e ≤ E w v e
  by (simp add: conv-isotone le-infI1 mult-right-isotone)
also have ... ≤ w * wT * E w v e
  by (metis assms(1) vector-complement-closed)
also have ... ≤ −−v
  by (simp add: assms(2) p-antitone p-antitone-iff)
also have ... ≤ −−v
  by (simp add: conv-isotone le-infI1 mult-right-isotone)
also have ... ≤ E w v e
  by (simp add: mult-right-isotone)
finally have 5: wT ∩ top * E w v e ∩ −v = bot
  using shunting-1-pp by blast
hence 6: wT ∩ (E w v e)T * top ∩ −vT = bot
  using conv-complement conv-dist-comp conv-dist-inf conv-top conv-bot by force
have (E w v e)T * top * E w v e ≤ (top * e * wT*)T * top * (top * e * wT*)
  by (simp add: conv-isotone mult-isotone)
also have ... = w * eT * top * e * wT*
  by (metis assms(3) cancel-separate inf.eq-iff star.circ-sup-sub-sup-one star-circ-plus-one star-involutive)
also have ... = wT * wT * w
  by (metis assms(3) cancel-separate inf.eq-iff star.circ-sup-sub-sup-one star-circ-plus-one star-involutive)
also have ... = wT * wT * 1
  by (metis star.circ-plus-one star-left-unfold-equal sup.assoc sup.commute)
finally have 7: (E w v e)T * top * E w v e ≤ wT * wT * 1
  by (simp add: conv-isotone le-infI1 mult-right-isotone)

69
have $E w v e \leq -v * -v^T$
  by (simp add: le-infI1)
also have ... $\leq \top * -v^T$
  by (simp add: mult-left-isotone)
also have ... $= -v^T$
  by (simp add: assms(1) vector-conv-compl)
finally have $8: E w v e \leq -v^T$.
nhence $9: (E w v e)^T \leq -v$
  by (metis conv-complement conv-involutive conv-isotone)
have $(E w v e)^T * \top * E w v e = (w^+ \sqcup w^{T+} \sqcup \top) \cap (E w v e)^T * \top * E w v e$
  using $7$ by (simp add: inf.absorb-iff2)
also have ... $\leq (w^+ \sqcap (E w v e)^T * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * E w v e)\sqcap \\top$ * $E w v e$
  using comp-inf.mult-right-dist-sup sup-assoc sup-commute by auto
also have ... $\leq 1 \sqcup (w^+ \sqcap (E w v e)^T * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * -v^T)$
  using inf-le1 sup-left-isotone by blast
also have ... $\leq 1 \sqcup (w^+ \sqcap -v * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * -v^T)$
  using $8$ inf.sup-right-isolone mult-right-isotone sup-right-isotone by blast
also have ... $\leq 1 \sqcup (w^+ \sqcap -v * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * -v^T)$
  using $9$ by (metis inf.sup-right-isotone mult-left-isotone sup.commute sup-right-isotone)
also have ... $= 1 \sqcup (w^+ \sqcap -v * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * -v^T)$
  by (metis (no-types) vector-export-comp inf-top-right inf-assoc)
also have ... $= 1 \sqcup (w^+ \sqcap -v * \top * E w v e) \sqcup (w^{T+} \sqcap (E w v e)^T * \top * -v^T)$
  using assms(1) vector-complement-closed vector-conv-compl by auto
also have ... $= 1$
  using $5$ $6$ by (simp add: conv-star-commute conv-dist-comp inf.commute inf-assoc star.circ-plus-same)
finally show $\text{thesis}$.

qed

lemma atom-edge-6:

assumes vector $v$
  and $w * v \leq v$
  and injective $w$
  and atom $e$
shows $E w v e * \top * (E w v e)^T \leq 1$

proof -

have $E w v e * 1 * (E w v e)^T \leq w * w^T$
  using comp-isotone conv-order inf.coboundedI1 mult-one-associative by auto
also have ... $\leq 1$

70
by (simp add: assms(3))
finally have 1: \( E \cdot w \cdot v \cdot e \leq 1 \)
  have \((E \cdot w \cdot v \cdot e)T \cdot top \leq 1 \)
    by (simp add: assms atom-edge-5)
also have ... \leq -\overline{1} - \overline{1}
    by (simp add: pp-increasing)
finally have 2: \( E \cdot w \cdot v \cdot e \cdot -1 \leq \overline{bot} \)
    by (metis conv-involutive regular-closed-bot regular-dense-top
    triple-schroeder-p)
have \( E \cdot w \cdot v \cdot e \cdot top \cdot (E \cdot w \cdot v \cdot e)T = E \cdot w \cdot v \cdot e \cdot 1 \cdot (E \cdot w \cdot v \cdot e)T \cup E \cdot w \cdot v \cdot e \cdot -1 \cdot (E \cdot w \cdot v \cdot e)T \)
    by (metis mult-left-dist-sup mult-right-dist-sup regular-complement-top
    regular-one-closed)
also have ... \leq 1
    using 1 2 by (simp add: bot-unique)
finally show \(?thesis

qed

lemma atom-edge:
assumes e \leq v \cdot -v^T \cap g
  and vector v
  and v^T = \overline{r^T} \cdot \overline{t^*}
  and \( t \leq g \)
  and \( \overline{r^T} \cdot \overline{g^*} \leq \overline{r^T} \cdot \overline{w^*} \)
  and \( w \cdot v \leq v \)
  and injective w
  and atom e
shows atom \((E \cdot w \cdot v \cdot e)\)
proof (intro conjI)
  have \((E \cdot w \cdot v \cdot e)T \cdot top \cdot (E \cdot w \cdot v \cdot e)T \leq 1 \)
    using assms(2,6-8) atom-edge-6 by simp
  thus injective \((E \cdot w \cdot v \cdot e)T \cdot top\)
    by (metis conv-dist-comp conv-top mult-assoc top-mult-top)
next
  show surjective \((E \cdot w \cdot v \cdot e)T \cdot top\)
    using assms(1-5,8) atom-edge-4 mult-assoc by simp
next
  have \((E \cdot w \cdot v \cdot e)T \cdot top \leq 1 \)
    using assms(2,6-8) atom-edge-5 by simp
  thus injective \((E \cdot w \cdot v \cdot e)T \cdot top\)
    by (metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top)
next
  have top \cdot (E \cdot w \cdot v \cdot e)T = top
    using assms(1-5,8) atom-edge-4 by simp
  thus surjective \((E \cdot w \cdot v \cdot e)T \cdot top\)
    by (metis mult-assoc conv-dist-comp conv-top)
qed
4.4 Invariant implies Postcondition

The lemmas in this section are used to show that the invariant implies the postcondition at the end of the algorithm. The following lemma shows that the nodes reachable in the graph are the same as those reachable in the constructed tree.

**lemma** `span-post`:
- **assumes** `regular v` and `vector v` and `v^T = r^T * t^*` and `v * -v^T \cap g = bot` and `t \leq v * v^T \cap g` and `r^T * (v * v^T \cap g)^* \leq r^T * t^*`
- **shows** `v^T = r^T * g^*`

**proof** –
- let `?vv = v * v^T \cap g`
- have 1: `r^T \leq v^T`
  - using `assms(3)` `mult-right-isotone` `mult-1-right` `star.circ-reflexive` by `fastforce`
- have `v * top \cap g = (v * v^T \sqcup v * -v^T) \cap g`
  - by `(metis assms(1) conv-complement mult-left-dist-sup regular-complement-top)`
- also have `... = ?vv \sqcup (v * -v^T \cap g)`
  - by `(simp add: inf-sup-distrib2)`
- also have `... = ?vv`
  - by `(simp add: assms(4))`
- finally have 2: `v * top \cap g = ?vv`
  - by `simp`
- have `r^T * ?vv^* \leq v^T * ?vv^*`
  - using 1 by `(simp add: comp-left-isotone)`
- also have `... \leq v^T * (v * v^T)^*`
  - by `(simp add: comp-right-isotone star.circ-isotone)`
- also have `... \leq v^T`
  - by `(simp add: assms(2) vector-star-1)`
- finally have `r^T * ?vv^* \leq v^T`
  - by `simp`
- hence `r^T * ?vv^* * g = (r^T * ?vv^* \cap v^T) * g`
  - by `(simp add: inf.absorb1)`
- also have `... = r^T * ?vv^* * (v * top \cap g)`
  - by `(simp add: assms(2) covector-inf-comp-3)`
- also have `... = r^T * ?vv^* * ?vv`
  - using 2 by `simp`
- also have `... \leq r^T * ?vv^*`
  - by `(simp add: comp-associative comp-right-isotone star.left-plus-below-circ star-plus)`
- finally have `r^T \sqcup r^T * ?vv^* * g \leq r^T * ?vv^*`
  - using `star.circ-back-loop-prefixpoint` by `auto`
- hence `r^T * g^* \leq r^T * ?vv^*`
  - using `star-right-induct` by `blast`
- hence `r^T * g^* = r^T * ?vv^*`

72
by \((\text{simp add: antisym mult-right-isotone star-isotone})\)
also have \(... = r^T * t^*\)
using \(\text{assms(5,6) antisym mult-right-isotone star-isotone by auto}\)
also have \(... = v^T\)
by \((\text{simp add: assms(3)})\)
finally show \(?\text{thesis}\)
by simp
qed

The following lemma shows that the minimum spanning tree extending
a tree is the same as the tree at the end of the algorithm.

**Lemma mst-post:**
assumes \(\text{vector } r\)
and \(\text{injective } r\)
and \(v^T = r^T * t^*\)
and \(\text{forest } w\)
and \(t \leq w\)
and \(w \leq v * v^T\)
shows \(w = t\)

**Proof**

have 1: \(\text{vector } v\)
using \(\text{assms(1,3) covector-mult-closed vector-conv-covector by auto}\)
have \(w * v \leq v * v^T * v\)
by \((\text{simp add: assms(6) mult-left-isotone})\)
also have \(... \leq v\)
using \(1\) by \(\text{metis mult-assoc mult-right-isotone top-greatest}\)
finally have 2: \(w * v \leq v\).

have 3: \(r \leq v\)
by \((\text{metis assms(3) conv-order mult-right-isotone mult-l-right}\)
\text{star,circ-reflexive})
have 4: \(v \sqcap -r = t^{\ast} * r \sqcap -r\)
by \((\text{metis assms(3) conv-dist-comp conv-involutive conv-star-commute})\)
also have \(... = (r \cup t^{\ast} * r) \sqcap -r\)
using \(\text{mult-assoc star,circ-loop-fixpoint sup-commute by auto}\)
also have \(... \leq t^{\ast} * r\)
by \((\text{simp add: shunting})\)
also have \(... \leq t^T * top\)
by \((\text{simp add: comp-isotone mult-assoc})\)
finally have \(1 \sqcap (v \sqcap -r) * (v \sqcap -r)^T \leq 1 \sqcap t^T * top * (t^T * top)^T\)
using \(\text{conv-order inf.sup-right-isotone mult-isotone by auto}\)
also have \(... = 1 \sqcap t^T * top * t\)
by \((\text{metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top})\)
also have \(... \leq t^T * (top * t \sqcap t * I)\)
by \((\text{metis conv-involutive dedekind-1 inf.commute mult-assoc})\)
also have \(... \leq t^T * t\)
by \((\text{simp add: mult-right-isotone})\)
finally have 5: \(1 \sqcap (v \sqcap -r) * (v \sqcap -r)^T \leq t^T * t\).

73
have \( w \ast w^+ \leq -1 \)

by (metis assms(4) mult-right-isotone order-trans star.circ-increasing
star.left-plus-circ)

hence 6: \( w^{T^+} \leq -w \)

by (metis conv-star-commute mult-assoc mult-1-left triple-schroeder-p)

have \( w \ast r \cap w^{T^+} \ast r = (w \cap w^{T^+}) \ast r \)

using assms(2) by (simp add: injective-comp-right-dist-inf)

also have ... = bot

using 6 p-antitone pseudo-complement-pp semiring.mult-not-zero by blast

finally have 7: \( w \ast r \cap w^{T^+} \ast r = bot \)

have \(-1 \ast r \leq -r \)

using assms(2) schroeder-4-p by force

hence \(-1 \ast r \ast top \leq -r \)

by (simp add: assms(1) comp-associative)

hence 8: \( r^T \ast -1 \ast r \leq bot \)

by (simp add: mult-assoc schroeder-6-p)

have \( r^T \ast w \ast r \leq r^T \ast w^+ \ast r \)

by (simp add: mult-left-isotone mult-right-isotone star.circ-mult-increasing)

also have ... \( \leq r^T \ast -1 \ast r \)

by (simp add: assms(4) comp-isotone)

finally have \( r^T \ast w \ast r \leq bot \)

using 8 by simp

hence \( w \ast r \ast top \leq -r \)

by (simp add: mult-assoc schroeder-6-p)

hence \( w \ast r \leq -r \)

by (simp add: assms(1) comp-associative)

hence \( w \ast r \leq -r \cap w \ast v \)

using 3 by (simp add: mult-right-isotone)

also have ... \( \leq -r \cap v \)

using 2 by (simp add: le-iff-inf2)

also have ... \( = -r \cap t^T \ast r \)

using 4 by (simp add: inf-commute)

also have ... \( \leq -r \cap w^{T^+} \ast r \)

using assms(5) comp-inf.mult-right-isotone conv-isotone mult-left-isotone
star-isotone by auto

also have ... \( = -r \cap (r \cup w^{T^+} \ast r) \)

using mult-assoc star.circ-loop-fixpoint sup-commute by auto

also have ... \( \leq w^{T^+} \ast r \)

using inf.commute maddux-3-13 by auto

finally have \( w \ast r = bot \)

using 7 by (simp add: le-iff-inf)

hence \( w = w \cap top \ast -r^T \)

by (metis complement-conv-sub conv-dist-comp conv-involutive conv-bot
inf.assoc inf.orderE regular-closed-bot regular-dense-top top-left-mult-increasing)

also have ... \( = w \cap v \ast v^T \cap top \ast -r^T \)

by (simp add: assms(6) inf-absorb1)

also have ... \( \leq w \cap top \ast v^T \cap top \ast -r^T \)

using comp-inf.mult-left-isotone comp-inf.mult-right-isotone mult-left-isotone
by auto
also have ... = w \cap top \ast (v^T \cap -r^T)
using 1 assms(1) covector-inf-closed inf-assoc vector-conv-compl
vector-conv-covector by auto
also have ... = w \ast (1 \cap (v \cap -r) \ast top)
by (simp add: comp-inf-vector conv-complement conv-dist-inf)
also have ... = w \ast (1 \cap (v \cap -r) \ast (v \cap -r)^T)
by (metis conv-top dedekind-eq inf-commute inf-top-left mult-1-left
mult-1-right)
also have ... \leq w \ast t^T \ast t
using 5 by (simp add: comp-isotone mult-assoc)
also have ... \leq w \ast w^T \ast t
by (simp add: assms(5) comp-isotone conv-isotone)
also have ... \leq t
using assms(4) mult-left-isotone mult-1-left by fastforce
finally show ?thesis
by (simp add: assms(5) antisym)
qed

end

4.5 Related Structures

Stone algebras can be expanded to Stone-Kleene relation algebras by reusing
some operations.

sublocale stone-algebra < comp-inf: stone-kleene-relation-algebra where star =
\lambda x. top and one = top and times = inf and conv = id
apply unfold-locales
by simp

Every bounded linear order can be expanded to a Stone algebra, which
can be expanded to a Stone relation algebra, which can be expanded to a
Stone-Kleene relation algebra.

class linorder-stone-kleene-relation-algebra-expansion =
linorder-stone-relation-algebra-expansion + star +
assumes star-def [simp]: x^* = top
begin

subclass kleene-algebra
apply unfoldlocales
apply simp
apply (simp add: min.coboundedII min.commute)
by (simp add: min.coboundedII)

subclass stone-kleene-relation-algebra
apply unfold-locales
by simp

end
A Kleene relation algebra is based on a relation algebra.

\[
\text{class } \text{kleene-relation-algebra} = \text{relation-algebra} + \text{stone-kleene-relation-algebra}
\]

end

5 Subalgebras of Kleene Relation Algebras

In this theory we show that the regular elements of a Stone-Kleene relation algebra form a Kleene relation subalgebra.

theory Kleene-Relation-Subalgebras


begin

instantiation regular :: (stone-kleene-relation-algebra) kleene-relation-algebra

begin

lift-definition star-regular :: \('a\) regular \(\Rightarrow\) \('a\) regular is star

using regular-closed-p regular-closed-star

by blast

instance

apply intro-classes

apply (metis (mono-tags, lifting) star-regular.rep-eq less-eq-regular.rep-eq
left-kleene-algebra-class.star-left-unfold one-regular.rep-eq simp-regular
sup-regular.rep-eq times-regular.rep-eq)

apply (metis (mono-tags, lifting) less-eq-regular.rep-eq
left-kleene-algebra-class.star-left-induct simp-regular star-regular.rep-eq
sup-regular.rep-eq times-regular.rep-eq)

apply (metis (mono-tags, lifting) less-eq-regular.rep-eq
strong-left-kleene-algebra-class.star-right-induct simp-regular star-regular.rep-eq
sup-regular.rep-eq times-regular.rep-eq)

by simp

end

end

6 Matrix Kleene Algebras

This theory gives a matrix model of Stone-Kleene relation algebras. The main result is that matrices over Kleene algebras form Kleene algebras. The automata-based construction is due to Conway [7]. An implementation of the construction in Isabelle/HOL that extends [2] was given in [3] without a correctness proof.

For specifying the size of matrices, Isabelle/HOL’s type system requires
the use of types, not sets. This creates two issues when trying to implement Conway’s recursive construction directly. First, the matrix size changes for recursive calls, which requires dependent types. Second, some submatrices used in the construction are not square, which requires typed Kleene algebras [14], that is, categories of Kleene algebras.

Because these instruments are not available in Isabelle/HOL, we use square matrices with a constant size given by the argument of the Kleene star operation. Smaller, possibly rectangular submatrices are identified by two lists of indices: one for the rows to include and one for the columns to include. Lists are used to make recursive calls deterministic; otherwise sets would be sufficient.

theory Matrix-Kleene-Algebras


begin

6.1 Matrix Restrictions

In this section we develop a calculus of matrix restrictions. The restriction of a matrix to specific row and column indices is implemented by the following function, which keeps the size of the matrix and sets all unused entries to bot.

definition restrict-matrix :: \('a list \Rightarrow ('a,'b::bot) square \Rightarrow ('a,'b) square\) where restrict-matrix as f bs = (λ(i,j). if List.member as i ∧ List.member bs j then f(i,j) else bot)

The following function captures Conway’s automata-based construction of the Kleene star of a matrix. An index k is chosen and s contains all other indices. The matrix is split into four submatrices a, b, c, d including/not including row/column k. Four matrices are computed containing the entries given by Conway’s construction. These four matrices are added to obtain the result. All matrices involved in the function have the same size, but matrix restriction is used to set irrelevant entries to bot.

primrec star-matrix' :: \('a list \Rightarrow ('a,'b::\{star,times,bounded-semilattice-sup-bot\}) square \Rightarrow ('a,'b) square\) where
star-matrix' Nil g = mbot | star-matrix' (k#s) g = (let r = [k] in
  let a = r g r in
  let b = r g s in
  let c = s g r in
  let d = s g s in
  let as = r (star o a) r in
let ds = star-matrix' s d in
let e = a ⊕ b ⊙ ds ⊙ c in
let es = r (star o e)r in
let f = d ⊕ c ⊙ as ⊙ b in
let fs = star-matrix' s f in
es ⊕ as ⊙ b ⊙ fs ⊕ ds ⊙ c ⊙ es ⊕ fs

The Kleene star of the whole matrix is obtained by taking as indices
all elements of the underlying type 'a. This is conveniently supplied by the
enum class.

fun star-matrix :: ('a::enum,'b::{star,times,bounded-semilattice-sup-bot}) square ⇒ ('a,'b) square (⊂ [100] 100) where star-matrix f = star-matrix'
(enum-class.enum::'a list) f

The following lemmas deconstruct matrices with non-empty restrictions.

lemma restrict-empty-left:
[[]][f]ls = mbot
by (unfold restrict-matrix-def List.member-def bot-matrix-def) auto

lemma restrict-empty-right:
ks[[]][f] = mbot
by (unfold restrict-matrix-def List.member-def bot-matrix-def) auto

lemma restrict-nonempty-left:
fixes f :: ('a,'b::bounded-semilattice-sup-bot) square
shows (k#ks)[[]][f]ls = [k][[]][f]ls ⊕ ks[f]ls
by (unfold restrict-matrix-def List.member-def sup-matrix-def) auto

lemma restrict-nonempty-right:
fixes f :: ('a,'b::bounded-semilattice-sup-bot) square
shows ks[[]][f](l#ls) = ks[f][[]][l] ⊕ ks[f]ls
by (unfold restrict-matrix-def List.member-def sup-matrix-def) auto

lemma restrict-nonempty:
fixes f :: ('a,'b::bounded-semilattice-sup-bot) square
shows (k#ks)[[]][f](l#ls) = [k][[]][f][[]][l] ⊕ [k][[]][f]ls ⊕ ks[f][[]][l] ⊕ ks[f]ls
by (unfold restrict-matrix-def List.member-def sup-matrix-def) auto

The following predicate captures that two index sets are disjoint. This
has consequences for composition and the unit matrix.

abbreviation disjoint ks ls ≡ ¬(∃ x . List.member ks x ∧ List.member ls x)

lemma times-disjoint:
fixes f g :: ('a,'b::idempotent-semiring) square
assumes disjoint ls ms
shows ks[[]][f]ls ⊙ ms[[]][g]ms = mbot
proof (rule ext, rule prod-cases)
fix i j
have \((\langle f \rangle_ks \odot ms \langle g \rangle_ns)(i,j) = \bigsqcup_k (\langle f \rangle_ks)(i,k) * (ms \langle g \rangle_ns)(k,j)\)
by (simp add: times-matrix-def)
also have \(\ldots = \bigsqcup_k (\text{if List.member } ks \ i \land \text{List.member } ls \ k \ \text{then } f \ (i,k) \ \text{else} \ \text{bot}) * (\text{if List.member } ms \ k \land \text{List.member } ns \ j \ \text{then } g \ (k,j) \ \text{else} \ \text{bot})\)
by (simp add: restrict-matrix-def)
also have \(\ldots = \bigsqcup_k (\text{if List.member } ms \ k \land \text{List.member } ns \ j \ \text{then } \text{bot} * g \ (k,j) \ \text{else} \ (\text{if List.member } ks \ i \land \text{List.member } ls \ k \ \text{then } f \ (i,k) \ \text{else} \ \text{bot}) * \text{bot})\)
using \(\text{assms by (auto intro: sup-monoid.sum.cong)}\)
also have \(\ldots = \text{bot}\)
by (simp add: sup-monoid.sum.neutral)
also have \(\ldots = \text{mbot} \ (i,j)\)
by (simp add: bot-matrix-def)
finally show \((\langle f \rangle_ks \odot ms \langle g \rangle_ns)(i,j) = \text{mbot} \ (i,j)\)
qed

lemma one-disjoint:
assumes disjoint ks ls
shows \(\langle \langle mone \ :: (\prime a, \prime b :: idempotent-semiring) \ square \rangle \rangle_ks \odot \langle f \rangle_ls = \text{mbot}\)
proof (rule ext, rule prod-cases)
let \(?o = mone :: (\prime a, \prime b) \ square\)
fix \(i \ j\)
have \(\langle f \rangle_\langle ?o \rangle_ks \odot \langle f \rangle_ls \ (i,j) = (\text{if List.member } ks \ i \land \text{List.member } ls \ j \ \text{then } f \ (i,j) \ \text{else} \ \text{bot}) \ \text{if } i = j \ \text{then } 1 \ \text{else} \ \text{bot})\)
by (simp add: \langle f \rangle_ks \odot \langle f \rangle_ls \ associative)
also have \(\ldots = \text{bot}\)
using \(\text{assms by auto}\)
also have \(\ldots = \text{mbot} \ (i,j)\)
by (simp add: bot-matrix-def)
finally show \(\langle f \rangle_\langle ?o \rangle_ks \odot \langle f \rangle_ls \ (i,j) = \text{mbot} \ (i,j)\)
qed

The following predicate captures that an index set is a subset of another index set. This has consequences for repeated restrictions.

abbreviation is-sublist ks ls \(\equiv \forall x \ . \ \text{List.member } ks \ x \longrightarrow \text{List.member } ls \ x\)

lemma restrict-sublist:
assumes is-sublist ks ls
and is-sublist ms ns
shows \(ls \langle f \rangle_ks \odot ms \langle f \rangle_ns = ls \langle f \rangle_ms\)
proof (rule ext, rule prod-cases)
fix \(i \ j\)
show \(ls \langle f \rangle_\langle ks \rangle_ks \odot ms \langle f \rangle_ms \ (i,j) = (ls \langle f \rangle_ms) \ (i,j)\)
proof (cases List.member ls \(i \land \text{List.member } ms \ j\)
\(\quad \text{case True thus } ?\text{thesis}\)
by (simp add: \langle f \rangle_ks \odot \langle f \rangle_ms \ associative)
next
  case False thus \textit{?thesis}
    by (unfold restrict-matrix-def) auto
qed

lemma restrict-superlist:
assumes is-sublist \( ls \) \( ks \)
  and is-sublist \( ms \) \( ns \)
shows \( ks \langle ls \langle f \rangle ms \rangle ns = ls\langle f \rangle ms \)
proof (rule ext, rule prod-cases)
  fix \( i \) \( j \)
  show \( (ks \langle ls \langle f \rangle ms \rangle ns)) (i, j) = (ls\langle f \rangle ms) (i, j) \)
    by (simp add: assms restrict-matrix-def)
next
  case False thus \textit{?thesis}
    by (unfold restrict-matrix-def) auto
qed

The following lemmas give the sizes of the results of some matrix operations.

lemma restrict-sup:
fixes \( f \) \( g \) :: \((\_\_\_::bounded-semilattice-sup-bot) square\)
shows \( ks\langle f \oplus g \rangle ls = ks\langle f \rangle ls \oplus ks\langle g \rangle ls \)
by (unfold restrict-matrix-def sup-matrix-def) auto

lemma restrict-times:
fixes \( f \) \( g \) :: \((\_\_\_::idempotent-semiring) square\)
shows \( ks\langle ks\langle f \rangle ls \odot ls\langle g \rangle ms \rangle ms = ks\langle f \rangle ls \odot ls\langle g \rangle ms \)
proof (rule ext, rule prod-cases)
  fix \( i \) \( j \)
  have \( (ks\langle (ks\langle f \rangle ls \odot ls\langle g \rangle ms) \rangle ms)) (i, j) = (if List.member \( ls \) \( i \) \& List.member \( ms \) \( j \) then \( \bigsqcup \_ \_ \_ k \ (ks\langle f \rangle ls) (i, k) \ast (ls\langle g \rangle ms) (k, j) \) else bot) \)
    by (simp add: times-matrix-def restrict-matrix-def)
  also have ... = (if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( \bigsqcup \_ \_ \_ k \ (if List.member \( ls \) \( k \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) \ast (if List.member \( ls \) \( k \) \& List.member \( ms \) \( j \) then \( g \ (k, j) \) else bot)) else bot) \)
    by (auto intro: sup-monoid.sum.cong)
  also have ... = (\( \bigsqcup \_ \_ \_ k \) if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) else bot) \)
    by (auto intro: sup-monoid.sum.cong)
  also have ... = (\( \bigsqcup \_ \_ \_ k \) if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) else bot) \)
    by auto
  also have ... = (\( \bigsqcup \_ \_ \_ k \) if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) \ast (if List.member \( ls \) \( k \) \& List.member \( ms \) \( j \) then \( g \ (k, j) \) else bot)) \)
  also have ... = (\( \bigsqcup \_ \_ \_ k \) if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) \ast (if List.member \( ls \) \( k \) \& List.member \( ms \) \( j \) then \( g \ (k, j) \) else bot)) \)
  also have ... = (\( \bigsqcup \_ \_ \_ k \) if List.member \( ks \) \( i \) \& List.member \( ms \) \( j \) then \( f \ (i, k) \) else bot) \ast (if List.member \( ls \) \( k \) \& List.member \( ms \) \( j \) then \( g \ (k, j) \) else bot)) \)

80
by (auto intro: sup-monoid.sum.cong)
also have ... = (\bigl\{ k \cdot (k \cdot f) \cdot (i, k) \cdot (l \cdot g) \cdot (j, j) \bigr\)
by (simp add: restrict-matrix-def)
also have ... = (k \cdot f) \cdot (l \cdot g) \cdot (i, j)
by (simp add: times-matrix-def)
finally show (k \cdot (k \cdot f) \cdot (l \cdot g) \cdot (m)) \cdot (i, j) = (k \cdot f) \cdot (l \cdot g) \cdot (i, j)
qed

lemma restrict-star:
fixes g :: 'a::{kleene-algebra} square
shows (\star\cdot (\star\cdot t) \cdot g) \cdot t = \star\cdot (\star\cdot t) \cdot g
proof (induct arbitrary: g rule: list.induct)
case Nil show ?case
by (simp add: restrict-empty-left)
next
case (Cons k s)
let ?t = k \cdot s
assume \bigl\{ g\cdot\cdot (\cdot a, b\cdot) \cdot square . s\cdot (\star\cdot s) \cdot g \cdot s = \star\cdot (\star\cdot s) \cdot g \bigr\}
hence \bigl\{ g\cdot\cdot (\cdot a, b\cdot) \cdot square . ?t\cdot (\star\cdot s) \cdot g \cdot t = \star\cdot (\star\cdot s) \cdot g \bigr\}
by (metis member-rec(1) restrict-superlist)
show ?t\cdot (\star\cdot s) \cdot g \cdot t = \star\cdot (\star\cdot s) \cdot g
proof =
  let ?r = [k]
  let ?a = ?r\cdot g \cdot ?r
  let ?b = ?r\cdot g \cdot s
  let ?c = s\cdot g \cdot ?r
  let ?d = s\cdot g \cdot s
  let ?as = ?r\cdot (\star\cdot a \cdot ?a) \cdot ?r
  let ?ds = \star\cdot s \cdot ?d
  let ?e = ?a \oplus ?b \oplus ?ds \oplus ?c
  let ?es = ?r\cdot (\star\cdot a \cdot ?e) \cdot ?r
  let ?f = ?d \oplus ?c \oplus ?as \oplus ?b
  let ?fs = \star\cdot (\star\cdot s) \cdot ?f
  have 2: ?t\cdot (\star\cdot ?a) \cdot ?t = ?as \land ?t\cdot (\star\cdot ?b) \cdot ?t = ?b \land ?t\cdot (\star\cdot ?c) \cdot ?t = ?c \land ?t\cdot (\star\cdot ?es) \cdot ?t = ?es
    by (simp add: restrict-superlist member-def)
  have 3: ?t\cdot (\star\cdot ?ds) \cdot ?t = ?ds \land \star\cdot ?fs \cdot ?t = \star\cdot ?fs
    using 1 by simp
  have 4: ?t\cdot (\star\cdot ?as) \cdot ?t \circ ?t\cdot (\star\cdot ?b) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t \circ ?t\cdot (\star\cdot ?es) \cdot ?t = ?t\cdot (\star\cdot ?as) \cdot ?t \circ ?t\cdot (\star\cdot ?b) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t
    by (metis (no-types) restrict-times)
  have 5: ?t\cdot (\star\cdot ?ds) \cdot ?t \circ ?t\cdot (\star\cdot ?c) \cdot ?t \circ ?t\cdot (\star\cdot ?es) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t \circ ?t\cdot (\star\cdot ?es) \cdot ?t = ?t\cdot (\star\cdot ?ds) \cdot ?t \circ ?t\cdot (\star\cdot ?c) \cdot ?t \circ ?t\cdot (\star\cdot ?es) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t
    by (metis (no-types) restrict-times)
  have \star\cdot ?t \cdot (\star\cdot ?a) \cdot ?t = \star\cdot ?t \cdot (\star\cdot ?es) \cdot ?t \circ ?t\cdot (\star\cdot ?b) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t \circ ?t\cdot (\star\cdot ?ds) \cdot ?t \circ ?t\cdot (\star\cdot ?c) \cdot ?t \circ ?t\cdot (\star\cdot ?es) \cdot ?t \circ \star\cdot ?t \cdot (\star\cdot ?fs) \cdot ?t
    by (metis (no-types) restrict-times)
  also have ... = \star\cdot ?t \cdot (\star\cdot ?es) \cdot ?t \circ ?t\cdot (\star\cdot ?as) \cdot ?t \circ ?t\cdot (\star\cdot ?b) \cdot ?t \circ ?t\cdot (\star\cdot ?fs) \cdot ?t \circ ?t\cdot (\star\cdot ?ds) \cdot ?t \circ ?t\cdot (\star\cdot ?c) \cdot ?t \circ \star\cdot ?t \cdot (\star\cdot ?es) \cdot ?t \circ \star\cdot ?t \cdot (\star\cdot ?fs) \cdot ?t
by (simp add: restrict-sup)
using 2 3 4 5 by simp
also have ... = star-matrix' ?t g
by (metis star-matrix',simp(2))
finally show ?thesis
qed

lemma restrict-one:
assumes ~ List.member ks k
shows (k#ks)((mone::('a,'b::idempotent-semiring) square))(k#ks) = [k](mone)[k] ⊕ ks(mone)ks
by (subst restrict-nonempty) (simp add: assms member-rec one-disjoint)

lemma restrict-one-left-unit:
ks((mone::('a::finite,'b::idempotent-semiring) square))ks ⊗ ks(f)ls = ks(f)ls
proof (rule ext, rule prod-cases)
let ?o = mone::('a,'b::idempotent-semiring) square
fix i j
have (ks(?o)ks ⊗ ks(f)ls) (i,j) = (∪k (ks(?o)ks) (i,k) * (ks(f)ls) (k,j))
  by (simp add: times-matrix-def)
also have ... = (∪k (if List.member ks i ∧ List.member ks k then ?o (i,k) else bot) * (if List.member ks k ∧ List.member ls j then f (k,j) else bot))
  by (simp add: restrict-matrix-def)
also have ... = (∪k (if List.member ks i ∧ List.member ks k then (if i = k then 1 else bot) else bot) * (if List.member ks k ∧ List.member ls j then f (k,j) else bot))
  by (rule sup-matrix-def) simp-all
also have ... = (if List.member ks i then 1 else bot) * (if List.member ks i ∧ List.member ls j then f (i,j) else bot)
  by (simp add: sup-matrix-def) simp
also have ... = (ks(f)ls) (i,j)
  by (simp add: restrict-matrix-def)
finally show (ks(?o)ks ⊗ ks(f)ls) (i,j) = (ks(f)ls) (i,j)
  by simp

qed

The following lemmas consider restrictions to singleton index sets.

lemma restrict-singleton:
\[(\{k\}\{f\}\{l\}) \ (i,j) = (if \ i = k \land j = l \ then \ f \ (i,j) \ else \ bot)\]
by \(\text{simp add: restrict-matrix-def \ List.member-def}\)

**lemma** restrict-singleton-list:
\[(\{k\}\{f\}\{l\}\{s\}) \ (i,j) = (if \ i = k \land \ List.member \ ls \ j \ then \ f \ (i,j) \ else \ bot)\]
by \(\text{simp add: restrict-matrix-def \ List.member-def}\)

**lemma** restrict-list-singleton:
\[(\{k\}\{f\}\{l\}\{s\}) \ (i,j) = (if \ List.member \ ks \ i \land j = l \ then \ f \ (i,j) \ else \ bot)\]
by \(\text{simp add: restrict-matrix-def \ List.member-def}\)

**lemma** restrict-singleton-product:
\[\text{fixes} \ f \ g :: ('a::finite,'b::kleene-algebra) \ square\]
\[\text{shows} \ \((\{k\}\{f\}\{l\}) \odot \ [m]\{g\}\{n\}) \ (i,j) = (if \ i = k \land l = m \land j = n \ then \ f \ (i,l) \ast g \ (m,j) \ else \ bot)\]
\[\text{proof} -\]
\[\text{have} \ \((\{k\}\{f\}\{l\}) \odot \ [m]\{g\}\{n\}) \ (i,j) = (\bigcup_h (\{k\}\{f\}\{l\}) \ (i,h) \ast (\{m\}\{g\}\{n\}) \ (h,j))\]
by \(\text{simp add: times-matrix-def}\)
\[\text{also have} \ \ldots = (\bigcup_h (if \ i = k \land h = l \ then \ f \ (i,h) \ else \ bot) \ast (if \ h = m \land j = n \ then \ g \ (h,j) \ else \ bot))\]
by \(\text{simp add: restrict-singleton}\)
\[\text{also have} \ \ldots = (\bigcup_h (if \ i = k \land h = l \ then \ f \ (i,h) \ else \ bot) \ast (if \ h = m \land j = n \ then \ g \ (h,j) \ else \ bot))\]
by \(\text{rule sup-monoid.sum.cong} \ \text{auto}\)
\[\text{also have} \ \ldots = (if \ i = k \land h = l \ then \ f \ (i,l) \ else \ bot) \ast (if \ l = m \land j = n \ then \ g \ (l,j) \ else \ bot)\]
by \(\text{simp}\)
\[\text{finally show} \ \text{thesis}\]
.

**qed**

The Kleene star unfold law holds for matrices with a single entry on the diagonal.

**lemma** restrict-star-unfold:
\[\{l\}{\langle mone::('a::finite,'b::kleene-algebra) \ square\rangle} \{l\} \odot \ [l]\{f\}\{l\} \odot \ [l]\{\text{star \ o \ f}\}\{l\} = \ [l]\{\text{star \ o \ f}\}\{l\}\]
\[\text{proof} \ \text{(rule ext, rule prod-cases)}\]
\[\text{let} \ \otimes = \text{mone::('a,'b::kleene-algebra) \ square}\]
\[\text{fix} \ i \ j\]
\[\text{have} \ ((\{l\}{\langle \otimes\rangle}\{l\} \odot [l]\{f\}\{l\} \odot [l]\{\text{star \ o \ f}\}\{l\}) \ (i,j) = ([l]{\langle \otimes\rangle}\{l\} \ (i,j) \sqcup [l]\{f\}\{l\} \ (i,j))\]
by \(\text{simp add: sup-matrix-def}\)
\[\text{also have} \ \ldots = ([l]{\langle \otimes\rangle}\{l\}) \ (i,j) \sqcup (\bigcup_k ([l]\{f\}\{l\} \ (i,k) \ast ([l]\{\text{star \ o \ f}\}\{l\}) \ (k,j)))\]
by \(\text{simp add: times-matrix-def}\)
\[\text{also have} \ \ldots = ([l]{\langle \otimes\rangle}\{l\}) \ (i,j) \sqcup (\bigcup_k (if \ i = l \land k = l \ then \ f \ (i,k) \ else \ bot) \ast (if \ k = l \land j = l \ then \ (f \ (k,j)) \ else \ bot))\]
by \(\text{simp add: restrict-singleton \ o-def}\)

83
also have \( \{[l]\langle \mathcal{I} f \rangle[0]\} \) \((i,j) \sqcup \{\}\) k if \( k = l \) then (if \( i = l \) then \( f(i,k) \) else 
bo \) \( \star \) (if \( j = l \) then \( f(k,j) \) ) \star \) else bo 
apply (rule sup-monoid2[where \( f = \sup \)])
apply simp 
by (rule sup-monoid.sum.cong) auto
also have \( \{[l]\langle \mathcal{I} f \rangle[0]\} \) \((i,j) \sqcup \{\}\) i \( = \) \((\) \( f (l,l) \) \star \) else bo 
by (simp add: sup-monoid.sum.delta)
also have \( \{[l]\langle \mathcal{I} f \rangle[0]\} \) \((i,j) \sqcup \{\}\) i \( = \) \( \) \((\) \( f (l,l) \) \star \) else bo 
by (simp add: restrict-singleton one-matrix-def)
also have \( \{[l]\langle \mathcal{I} f \rangle[0]\} \) \((i,j) \sqcup \{\}\) i \( = \) \((\) \( f (l,l) \) \star \) else bo 
by (simp add: star-left-unfold-equal)
also have \( \{[l]\langle \mathcal{I} f \rangle[0]\} \) \((i,j) \sqcup \{\}\) (\( f (l,l) \) ) \star \) else bo 
by (simp add: restrict-singleton o-def)
finally show \( \{[l]\langle \mathcal{I} f \rangle[0]\} \sqcup \{\}\) \((i,j) \sqcup \{\}\) (\( f (l,l) \) ) \star \) else bo 
qeq

lemma restrict-all:
enum-class.enum(f) enum-class.enum = f
by (simp add: restrict-matrix-def List.member-def enum-UNIV)

The following shows the various components of a matrix product. It is essentially a recursive implementation of the product.

lemma restrict-nonempty-product:
fixes \( f,g :: ('a::finite,'b::idempotent-semiring) square \)
assumes \( = \) List.member ls l
shows \( \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) = \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) o (\#m) / (\#m) o (\#m) / (\#m) \)
proof
also have \( \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) = \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) o (\#m) / (\#m) o (\#m) / (\#m) \)
by (metis restrict-nonempty)
also have \( \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) = \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) o (\#m) / (\#m) o (\#m) / (\#m) \)
by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
also have \( \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) = \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) o (\#m) / (\#m) o (\#m) / (\#m) \)
by (simp add: matrix-idempotent-semiring.mult-left-dist-sup)
also have \( \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) = \{\} \times (\#k) / (\#k) o (\#l) / (\#l) o (\#m) / (\#m) o (\#m) / (\#m) o (\#m) / (\#m) \)
by (simp add: List.member-def times-disjoint)

84
also have ... = (\lfloor f \rfloor l \odot \lfloor g \rfloor m) \oplus (\lfloor f \rfloor l \odot \lfloor g \rfloor m) \oplus (\lfloor f \rfloor l \odot \lfloor g \rfloor m) \oplus (\lfloor f \rfloor l \odot \lfloor g \rfloor m)

by (simp add: matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc
matrix-semilattice-sup.sup.sup-left-commute)
finally show \( \theta \)thesis

qed

Equality of matrices is componentwise.

lemma restrict-nonempty-eq:
\((k \# k s)(f)(l \# l s) = (k \# k s)(g)(l \# l s) \iff \lfloor k \rfloor f l = \lfloor k \rfloor g l \land (k \# f) l s = \lfloor k \rfloor g l s \land (k \# g) l s \land (k \# s)(f)(l \# l s))ls = (k \# s)(g)(l \# l s))ls)

by (simp add: member-def
restrict-matrix-def
restrict-sublist)
thus \((k \# k s)(f)(l \# l s) = (k \# k s)(g)(l \# l s))ls = (k \# k s)(g)(l \# l s))ls

by (simp add: restrict-sublist)

next
assume 3: \((k \# k s)(f)(l \# l s) = (k \# k s)(g)(l \# l s))ls = (k \# k s)(g)(l \# l s))ls

show \((k \# k s)(f)(l \# l s) = (k \# k s)(g)(l \# l s))ls

proof (rule ext, rule prod-cases)
fix \( i \) \( j \)

have 4: \( f (k, l) = g (k, l) \)

using 3 by (metis restrict-singleton)

have 5: \( \text{List.member} ls j \implies f (k, j) = g (k, j) \)

using 3 by (metis restrict-singleton-list)

have 6: \( \text{List.member} ks i \implies f (i, l) = g (i, l) \)

using 3 by (metis restrict-list-singleton)

have \((k \# s)(f) ls (i, j) = (k \# g)(ls) (i, j) \)

using 3 by simp

hence 7: \( \text{List.member} ks i \implies \text{List.member} ls j \implies f (i, j) = g (i, j) \)

by (simp add: restrict-matrix-def)

have \((k \# k s)(f)(l \# l s)) (i, j) = (if (i = k \lor \text{List.member} ks i) \land (j = l \lor \text{List.member} ls j) \then f (i, j) \else bot) \)

by (simp add: restrict-matrix-def List.member-def)

also have ... = (if i = k \land j = l \then g (i, j) \else if i = k \land \text{List.member} ls j \then f (i, j) \else if \text{List.member} ks i \land \text{List.member} ls j \then f (i, j) \else bot)

by auto

also have ... = (if i = k \land j = l \then g (i, j) \else if i = k \land \text{List.member} ls j \then g (i, j) \else if \text{List.member} ks i \land \text{List.member} ls j \then g (i, j) \else if \text{List.member} ks i \land \text{List.member} ls j \then g (i, j) \else bot)
∧ List.member ls j then g (i, j) else bot
  
  using 4 5 6 7 by simp
  also have ... = (if (i = k ∨ List.member ks i) ∧ (j = l ∨ List.member ls j)
  then g (i, j) else bot)
  by auto
  also have ... = ((k#ks)(g)(l#ls)) (i, j)
  by (simp add: restrict-matrix-def List.member-def)
  finally show ((k#ks)(f)(l#ls)) (i, j) = ((k#ks)(g)(l#ls)) (i, j)
    .
  qed

  qed

Inequality of matrices is componentwise.

lemma restrict-nonempty-less-eq:
  fixes f g :: ('a,'b::idempotent-semiring) square
  shows ((k#ks)(f)(l#ls) ≤ (k#ks)(g)(l#ls) ↔ [k](f)[l] ≤ [k](g)[l] ∧ [k](f)ls ≤ [k](g)ls
  by (unfold matrix-semilattice-sup.sup.sup-order-iff) (metis (no-types, lifting)
    restrict-nonempty-eq restrict-sup)

The following lemmas treat repeated restrictions to disjoint index sets.

lemma restrict-disjoint-left:
  assumes disjoint ks ms
  shows ms(ks(f)ls)ns = mbot
proof (rule ext, rule prod-cases)
  fix i j
  have (ms(ks(f)ls)ns) (i, j) = (if List.member ms i ∧ List.member ns j then if
  List.member ks i ∧ List.member ls j then f (i, j) else bot else bot)
  by (simp add: restrict-matrix-def)
  thus (ms(ks(f)ls)ns) (i, j) = mbot (i, j)
    using assms by (simp add: bot-matrix-def)
  qed

lemma restrict-disjoint-right:
  assumes disjoint ls ns
  shows ms(ks(f)ls)ns = mbot
proof (rule ext, rule prod-cases)
  fix i j
  have (ms(ks(f)ls)ns) (i, j) = (if List.member ms i ∧ List.member ns j then if
  List.member ks i ∧ List.member ls j then f (i, j) else bot else bot)
  by (simp add: restrict-matrix-def)
  thus (ms(ks(f)ls)ns) (i, j) = mbot (i, j)
    using assms by (simp add: bot-matrix-def)
  qed

The following lemma expresses the equality of a matrix and a product of two matrices componentwise.

lemma restrict-nonempty-product-eq:
  fixes f g h :: ('a::finite,'b::idempotent-semiring) square
assumes - List.member ks k
and - List.member ls l
and - List.member ms m

shows (k#ks)(f)(l#ls)(g)(m#ms) = (k#ks)(h)(m#ms) \iff
[k](f)[l] \odot [l](g)[m] \odot [k](f)ls \odot ls(g)[m] = [k]([h][m] \land [k](f)[l] \odot [l](g)[ms] \odot
[k](f)ls \odot ls(g)[ms] = [k](h)(m) \land ks(f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms] = ks(h)(m) \land ks(f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms] = ks(h)(m)

proof -

have 1: disjoint [k] ks \land disjoint [m] ms
by (simp add: assms(1,3) member-rec)

have 2: [k](k#ks)(f)(l#ls)(g)(m#ms)[m] = [k](f)[l] \odot [l](g)[m] \odot
[k](f)ls \odot ls(g)[m]

proof -

have [k](k#ks)(f)(l#ls)(g)(m#ms)[m] = [k](k(f)[l] \odot [l](g)[m] \odot
[k](f)ls \odot ls(g)[m] \odot [ks(f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms])
by (simp add: assms(2) restrict-nonempty-product)

also have ... = [k](ks(f)[l] \odot [l](g)[ms] \odot [k](ks(f)ls \odot ls(g)[m]) \odot
[k]([k](f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms])
by (simp add: restrict-times)

also have ... = [k](f)[l] \odot [l](g)[m] \odot [k](f)ls \odot ls(g)[m]

using / by (metis restrict-disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup)

finally show ?thesis

qed

have 3: [k](k#ks)(f)(l#ls)(g)(m#ms)ms = [k](f)[l] \odot [l](g)[ms] \odot
[k](f)ls \odot ls(g)[ms]

proof -

have [k](k#ks)(f)(l#ls)(g)(m#ms)ms = [k]([k](f)[l] \odot [l](g)[ms] \odot
[k](f)ls \odot ls(g)[ms] \odot [ks(f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms])
by (simp add: assms(2) restrict-nonempty-product)

also have ... = [k]([k](f)[l] \odot [l](g)[ms])ms \odot [k]([k](f)ls \odot ls(g)[ms])ms \odot
[k]([k](f)[l] \odot [l](g)[ms] \odot ks(f)ls \odot ls(g)[ms])ms \odot
[k](ks(f)[l] \odot [l](g)[ms])ms \odot
[k](ks(f)ls \odot ls(g)[ms])ms
by (simp add: matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup)

also have ... = [k]([k](f)[l] \odot [l](g)[ms] \odot [k](f)ls \odot ls(g)[ms] \odot
[k](ks(f)[l] \odot [l](g)[ms])ms \odot
[k](ks(f)ls \odot ls(g)[ms])ms
\[(l(g)(m))ms \oplus [k](ks(ks(f)ls \circ ls(g)(m))ms \oplus [k](ks(ks(f)[l] \circ [l](g)(m))ms \oplus [k](ks(ks(f)ls \circ ls(g)(m))ms \ms
\]

by (simp add: restrict-times)

also have \[ ... = [k](f[l]) \circ [l](g)ms \oplus [k](f)ls \circ ls(g)ms\]

using \( f \) by (metis restrict-disjoint-left restrict-disjoint-right

matrix-bound-semilattice-sup-bot.sup-monoid.add-0-right

matrix-bound-semilattice-sup-bot.sup-monoid.add-0-left)

finally show \( \text{thesis} \)

qed

have 4: \( \text{ks}((k\#ks(f))(l\#ls) \circ (l\#ls)(g)(m\#ms))[m] = ks(f)[l] \circ [l](g)[m] \oplus ks(f)ls \circ ls(g)[m] \)

proof –

also have \( ... = ks([k](f)[l] \circ [l](g)[m])[m] \oplus ks([k](f)ls \circ ls(g)[m])[m] \oplus ks([k](f)[l] \circ [l](g)[m])ms \oplus ks([k](f)ls \circ ls(g)[m])ms \oplus ks([k](f)ls \circ ls(g)[m])ms \)

by (simp add: matrix-bound-semilattice-sup-bot.sup-monoid.add-assoc

restrict-sup)

also have \( ... = ks([k][k](f)[l] \circ [l](g)[m])[m] \oplus ks([k](k)(f)ls \circ ls(g)[m])[m] \oplus ks([k][k](f)ls \circ ls(g)[m])ms \oplus ks([k](k)(f)ls \circ ls(g)[m])ms \oplus ks([k](k)(f)ls \circ ls(g)[m])ms \)

by (simp add: assms(2) restrict-nonempty-product)

also have \( ... = ks([k](f)[l] \circ [l](g)[m])ms \oplus ks([k](f)ls \circ ls(g)[m])ms \)

using \( f \) by (metis restrict-disjoint-left restrict-disjoint-right

matrix-bound-semilattice-sup-bot.sup-monoid.add-0-right

matrix-bound-semilattice-sup-bot.sup-monoid.add-0-left)

finally show \( \text{thesis} \)

qed

have 5: \( \text{ks}((k\#ks(f))(l\#ls) \circ (l\#ls)(g)(m\#ms))ms = ks(f)[l] \circ [l](g)ms \oplus ks(f)ls \circ ls(g)ms \)

proof –

also have \( ... = ks([k](f)[l] \circ [l](g)[m])ms \oplus ks([k](f)ls \circ ls(g)[m])ms \oplus ks([k](f)[l] \circ [l](g)[m])ms \oplus ks([k](f)ls \circ ls(g)[m])ms \)

by (simp add: matrix-bound-semilattice-sup-bot.sup-monoid.add-assoc

restrict-sup)

also have \( ... = ks([k][k](f)[l] \circ [l](g)[m])[m] \oplus ks([k](k)(f)ls \circ ls(g)[m])[m] \)

by (simp add: restrict-times)

also have \( ... = ks([k][k](f)[l] \circ [l](g)[m])[m] \oplus ks([k](k)(f)ls \circ ls(g)[m])[m] \)

by (simp add: matrix-bound-semilattice-sup-bot.sup-monoid.add-assoc

restrict-sup)

also have \( ... = ks([k](k)(f)[l] \circ [l](g)[m])[m] \oplus ks([k](k)(f)ls \circ ls(g)[m])[m] \)

by (simp add: restrict-times)
\[
\begin{align*}
ls(g)[m] & \triangleq \text{ks}(l)[g] \sqcap \text{ls}(g)[m] \\
\text{by (simp add: restrict-times)} \\
\text{also have ...} & = \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \triangleright \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \\
\text{using f by (metis disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot sup-monoid.add-0-left)} \\
\text{finally show ?thesis} & \\
\text{qed}
\end{align*}
\]

The following lemma gives a componentwise characterisation of the inequality of a matrix and a product of two matrices.

**Lemma:** restrict-nonempty-product-less-eq:

**Fixes** \( f, g, h :: ('a::finite, 'b::idempotent-semiring) square \)

**Assumes** \( \vdash \text{List.member ks k} \)

**and** \( \vdash \text{List.member ls l} \)

**and** \( \vdash \text{List.member ms m} \)

**Shows** \( \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \preceq \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \)

**Proof**

1. \( \vdash \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \preceq \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \)

2. \( \vdash \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \preceq \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \)

3. \( \vdash \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \preceq \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \)

4. \( \vdash \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \preceq \text{ks}(f)[l] \sqcap \text{ls}(g)[m] \)
The Kleene star induction laws hold for matrices with a single entry on the diagonal. The matrix \( g \) can actually contain a whole row/colum at the appropriate index.

**Lemma restrict-star-left-induct:**
- **Fixes** \( f, g :: ('a::{finite, 'b::kleene-algebra}) square \)
- **Shows** distinct \( m s \implies [l](f)[l] \circ [l](g)[m] \circ [k](f)\ ls \circ [l](g)[m] \leq [k](h)[m] \land [k](f)[l] \circ [l](g)[m] \leq [k](h)[m] \land\)
  \[\begin{align*}
  [k](f)[l] \circ [l](g)\ ms & \circ [k](f)\ ls \circ [l](g)[m] \leq [k](h)[m] \land \ks(f)[l] \circ [l](g)[m] \leq [k](h)[m] \land \ks(f)\ ls \circ [l](g)[m] \leq [k](h)\ ms \leq [k](h)\ ms
  \end{align*}\]
- **Proof** (induct ms)
  - **Case** Nil thus \( \forall \)case
    - by (simp add: restrict-empty-right)
- **Next**
  - **Case** \( Cons \ (m s) \)
    - **Assume** 1: distinct \( m s \implies [l](f)[l] \circ [l](g)[m] \leq [l](g)[ms] \implies [l](star \circ f)[l] \circ [l](g)[ms] \leq [l](g)[ms] \)
      - **Assume** 2: distinct \( (m \#ms) \)
    - **Have** 4: \( [l](f)[l] \circ [l](g)[m] \leq [l](g)[m] \land [l](f)[l] \circ [l](g)[ms] \leq [l](g)[ms] \)
      - **Using** 2 3 by (meson restrict-nonempty-less-eq)
  - **Hence** 5: \( [l](star \circ f)[l] \circ [l](g)[ms] \leq [l](g)[ms] \)
    - **Using** 1 2 by simp
    - **Have** \( f \ (l,l) \ast g \ (l,m) \leq g \ (l,m) \)
      - **Using** 4 by (meson restrict-singleton-product restrict-singleton less-eq-matrix-def)
    - **Hence** 6: \( f \ (l,l) \ast g \ (l,m) \leq g \ (l,m) \)
      - **By** (simp add: star-left-induct-mult)
    - **Have** \( [l](star \circ f)[l] \circ [l](g)[m] \leq [l](g)[m] \)
  - **Proof** (unfold less-eq-matrix-def, rule allI, rule prod-cases)
  - **Fix** \( i \ j \)
    - **Have** \( ([l](star \circ f)[l] \circ [l](g)[m]) \ (i,j) = (\bigcup_k ([l](star \circ f)[l]) \ (i,k) *\)
proof
fun $i \cdot j$

have \( ([m](g)|l) \odot ([l](\langle \star o f \rangle)|l) \) \( (i,j) \) = \( (\bigcup_k ([m](g)|l)) \odot ([l](\langle \star o f \rangle)|l) \) \( (k,j) \)

by (simp add: times-matrix-def)
also have \( \ldots \) = \( (\bigcup_k \text{if} \ i = m \land k = l \text{then} \ g \ (i,k) \text{ else} \ ot) \odot (\text{if} \ k = l \land j = l \text{then} \ (f \ (k,j))\star \text{ else} \ ot) \)

by (simp add: restrict-singleton o-def)
also have \( \ldots \) = \( (\bigcup_k \text{if} \ k = l \text{then} \ (i = m \text{ then} \ g \ (i,k) \text{ else} \ ot) \odot (\text{if} \ j = l \text{then} \ (f \ (k,j))\star \text{ else} \ ot) \)

by (rule sup-monoid.sum.cong) auto
also have \( \ldots \) = \( (\text{if} \ i = m \text{ then} \ g \ (i,l) \text{ else} \ ot) \odot (\text{if} \ j = l \text{then} \ (f \ (l,j))\star \text{ else} \ ot) \)

by (simp add: sup-monoid.sum.delta)
also have \( \ldots \) = \( (\text{if} \ i = m \land j = l \text{ then} \ g \ (m,l) \star (f \ (l,l))\star \text{ else} \ ot) \)

by simp
also have \( \ldots \) \text{ \leq} \( ([m](g)|l) \odot ([l](\langle \star o f \rangle)|l) \) \( (i,j) \)

using 6 by (simp add: restrict-singleton)
finally show \( ([m](g)|l) \odot ([l](\langle \star o f \rangle)|l) \) \( (i,j) \) \leq \( ([m](g)|l) \odot ([l](\langle \star o f \rangle)|l) \) \( (i,j) \)

qed

thus \( (m\#ms)(g)|l) \odot ([l](\langle \star o f \rangle)|l) \leq (m\#ms)(g)|l) \)

using 2 5 by (metis (no-types, hide-lams)
matrix-idempotent-semiring.mult-right-dist-sup matrix-semilattice-sup.sup.sup_mono restrict-nonempty-left)

qed

lemma restrict-pp:
fixes \( f :: \langle 'a,'b::p-algebra \rangle \) square
shows \( \text{ks}\odot (f)|l) = (\text{ks}\odot (f)|l) \)

by (unfold restrict-matrix-def uminus-matrix-def) auto

lemma pp-star-commute:
fixes \( f :: \langle 'a,'b::stone-kleene-relation-algebra \rangle \) square
shows \( \odot (\star o f) = \star o \odot f \)

by (simp add: uminus-matrix-def o-def pp-dist-star)

6.2 Matrices form a Kleene Algebra

Matrices over Kleene algebras form a Kleene algebra using Conway's construction. It remains to prove one unfold and two induction axioms of the Kleene star. Each proof is by induction over the size of the matrix represented by an index list.

interpretation matrix-kleene-algebra: kleene-algebra-var where sup = sup-matrix and less-eq = less-eq-matrix and less = less-matrix and bot = bot-matrix::\((a::enum,'b::kleene-algebra)\) square and one = one-matrix and times = times-matrix and star = star-matrix

proof
fix \( g :: \langle 'a,'b \rangle \) square
let \( ?e = \text{enum-class.enum}::'a\ list \)
let \( \exists o = \text{mone} :: ('a', 'b) \text{ square} \)

have \( \forall g :: ('a', 'b) \text{ square} \cdot \text{distinct } ?e \rightarrow (?e(?o) ?e \circ ?e(?g) ?e \circ \text{ star-matrix}' ?e g) = (\text{ star-matrix}' ?e g) \)

proof (induct rule: list.induct)
  case Nil thus \( ? \)case
    by (simp add: restrict-empty-left)
next
  case (Cons k s)
  let \( ?t = k \# s \)
  assume 1: \( \forall g :: ('a', 'b) \text{ square} \cdot \text{distinct } s \rightarrow (s(?o) s \circ s(?g) s \circ \text{ star-matrix}' s g) = (\text{ star-matrix}' s g) \)
  show \( \forall g :: ('a', 'b) \text{ square} \cdot \text{distinct } ?t \rightarrow (?t(?o) ?t \circ ?t(?g) ?t \circ \text{ star-matrix}' ?t g) = (\text{ star-matrix}' ?t g) \)
proof (rule allI, rule impI)
  fix \( g :: ('a', 'b) \text{ square} \)
  assume 2: \( \text{distinct } ?t \)
  let \( ?r = [k] \)
  let \( ?a = ?r(?g) ?r \)
  let \( ?b = ?r(?g) s \)
  let \( ?c = s(?g) ?r \)
  let \( ?d = s(?g) s \)
  let \( ?as = ?r(?o ?a) ?r \)
  let \( ?ds = \text{ star-matrix}' s ?d \)
  let \( ?e = ?a \oplus ?b \circ ?ds \circ ?c \)
  let \( ?es = ?r(?o ?e) ?r \)
  let \( ?f = ?d \oplus ?c \circ ?as \circ ?b \)
  let \( ?fs = \text{ star-matrix}' s ?f \)
  have \( s(?ds) s = ?ds \land s(?fs) s = ?fs \)
    by (simp add: restrict-star)
  hence 3: \( ?r(?e) ?r = ?e \land s(?ff) s = ?ff \)
    by (meson (no-types, lifting) restrict-one-left-unit restrict-sup restrict-times)
  have 4: \( \text{disjoint } s ?r \land \text{ disjoint } ?r s \)
    using 2 by (simp add: in-set-member member-rec)
  hence 5: \( ?t(?o) ?t = ?r(?o) ?r \circ s(?o) s \)
    by (meson member-rec(1) restrict-one)
  have 6: \( ?t(?g) ?t \circ ?es = ?a \circ ?es \circ ?c \circ ?es \)
proof --
  have \( ?t(?g) ?t \circ ?es = (?a \circ ?b \circ ?c \circ ?d) \circ ?es \)
    by (meson restrict-nonempty)
  also have \( = ?a \circ ?es \circ ?b \circ ?es \circ ?c \circ ?es \circ ?d \circ ?es \)
    by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
  also have \( \circ ?c \circ ?es \)
    using 4 by (simp add: times-disjoint)
  finally show \( ?\text{thesis} \)
qod

have 7: \( ?t(?g) ?t \circ ?as \circ ?b \circ ?fs = ?a \circ ?as \circ ?b \circ ?fs \circ ?c \circ ?as \circ \)
\( ?b \circ ?fs \)
proof --
have \( \exists t \) \( \exists \mathcal{C} \) \( \exists \mathcal{B} \) \( \exists \mathcal{F}_s \) = (\( \exists a \oplus \exists b \oplus \exists c \oplus \exists d \)) \( \oplus \exists a \oplus \exists b \oplus \exists f_s \\
\text{by (metis restrict-nonempty)} \\
\text{also have } \ldots = \exists a \oplus \exists b \oplus \exists f_s \oplus \exists b \oplus \exists a \oplus \exists b \oplus \exists f_s \oplus \exists c \oplus \exists a \\
\text{by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)} \\
\text{also have } \ldots = \exists a \oplus \exists b \oplus \exists f_s \oplus \exists c \oplus \exists a \oplus \exists b \oplus \exists f_s \\
\text{using 4 by (simp add: times-disjoint)} \\
\text{finally show } \exists t \\
\text{. qed} \\
\text{have 9: } \exists t \langle g \rangle \exists t \oplus \exists d_s \oplus \exists c \oplus \exists e_s = \exists b \oplus \exists d_s \oplus \exists c \oplus \exists e_s \oplus \exists d \oplus \exists d_s \\
\text{\( \exists c \oplus \exists e_s \) proof } - \\
\text{have } \exists t \langle g \rangle \exists t \oplus \exists f_s = (\exists a \oplus \exists b \oplus \exists c \oplus \exists d) \oplus \exists f_s \\
\text{by (metis restrict-nonempty)} \\
\text{also have } \ldots = \exists a \oplus \exists f_s \oplus \exists b \oplus \exists f_s \oplus \exists c \oplus \exists f_s \oplus \exists d \oplus \exists f_s \\
\text{by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)} \\
\text{also have } \ldots = \exists b \oplus \exists f_s \oplus \exists d \oplus \exists f_s \\
\text{using 4 by (metis (no-types, lifting) times-disjoint)} \\
\text{matrix-idempotent-semiring.mult-left-zero restrict-star} \\
\text{matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right} \\
\text{matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left) } \\
\text{finally show } \exists \text{thesis} \\
\text{. qed} \\
\text{have 9: } \exists t \langle g \rangle \exists t \oplus \exists f_s = \exists b \oplus \exists f_s \oplus \exists d \oplus \exists f_s \\
\text{proof } - \\
\text{have } \exists t \langle g \rangle \exists t \oplus \exists f_s = (\exists a \oplus \exists b \oplus \exists c \oplus \exists d) \oplus \exists f_s \\
\text{by (metis restrict-nonempty)} \\
\text{also have } \ldots = \exists a \oplus \exists f_s \oplus \exists b \oplus \exists f_s \oplus \exists c \oplus \exists f_s \oplus \exists d \oplus \exists f_s \\
\text{by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)} \\
\text{also have } \ldots = \exists b \oplus \exists f_s \oplus \exists d \oplus \exists f_s \\
\text{using 4 by (metis (no-types, lifting) times-disjoint restrict-star)} \\
\text{matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right} \\
\text{matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left) } \\
\text{finally show } \exists \text{thesis} \\
\text{. qed} \\
\text{have } \exists t \langle \exists a \rangle \exists t \oplus \exists t \langle g \rangle \exists t \oplus \exists \text{star-matrix'} \exists t \langle g \rangle = \exists \langle \exists t \langle \exists a \rangle \oplus \exists t \langle g \rangle \rangle \langle \exists t \langle \exists a \rangle \oplus \exists t \langle g \rangle \rangle \\
\text{by (metis star-matrix'.simp(2))} \\
\text{also have } \ldots = \exists \langle \exists t \langle \exists a \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \oplus \exists \langle \exists t \langle g \rangle \rangle \exists t \\
\text{by (simp add: matrix-idempotent-semiring.mult-left-dist-sup} \\
\text{matrix-monoid.mult-assoc matrix-semilattice-sup.sup-assoc}) \\
\text{also have } \ldots = \exists \langle \exists t \langle \exists a \rangle \rangle \exists r \oplus \exists s \langle \exists t \langle \exists a \rangle \rangle \exists a \oplus \exists \langle \exists t \langle g \rangle \rangle \exists a \oplus \exists \langle \exists t \langle g \rangle \rangle \exists a \oplus \exists \langle \exists t \langle g \rangle \rangle \exists a \\
\text{\( \oplus \exists a \oplus \exists f_s \oplus \exists c \oplus \exists a \oplus \exists b \oplus \exists f_s \oplus \exists b \oplus \exists d_s \oplus \exists c \oplus \exists e_s \oplus \exists d \oplus \exists d_s \oplus \exists c \oplus \exists e_s \oplus \exists d \oplus \exists d_s \oplus \exists c \oplus \exists e_s \oplus \exists d \oplus \exists d_s \) 94
using 5 6 7 8 9 by (simp add: matrix-semilattice-sup.assoc)
also have ... = (\fr(\{o\})\fr + (\fa o \frs o \frb o \frds o \frc o \frs)) o (\frb o \frs o \fpa o \fras o \frb o \frfs) @ (\fc o \frs o \frd o \frds o \frc o \frs) @ (s(\{fo\} s o (\frd o \frfs o \frc o \fras o \frb o \frfs))
by (simp only: matrix-semilattice-sup.assoc)

matrix-semilattice-sup.sup-commute matrix-semilattice-sup.sup-left-commute
also have ... = (\fr(\{o\})\fr + (\fa o \frs o \frb o \frds o \frc o \frs)) o (\frb o \frs o \fpa o \fras o \frb o \frfs) @ (s(\{fo\} s o \frc o \fras o \frd o \frs) @ \frd o \frs o \frc o \fras o \frb o \frfs)
by (simp add: restrict-one-left-unit)
also have ... = (\fr(\{o\})\fr + \frc o \frs) @ (s(\{fo\} s o \frd o \frds) o \frc o \frs) @ (s(\{fo\} s o \frf o \frs)
using 1 2 3 by (metis distinct.simps(2))
also have ... = (\fr(\{o\})\fr + \frc o \frs) @ (s(\{fo\} s o \frd o \frds) o \frc o \frs) @ \frfs
using 1 2 by (metis (no-types, lifting) distinct.simps(2) restrict-superlist)
also have ... = \frs o (s(\{fo\} \frr o \fa o \fras) o \frb o \frfs) @ (\frds o \frc o \frs) o \frfs
using 3 by (metis restrict-star-unfold)
also have ... = \frs o \fras o \frb o \frfs @ \frds o \frc o \frs o \frfs
by (metis (no-types, lifting) restrict-one-left-unit restrict-star-unfold restrict-times)
also have ... = star-matrix' ?t g
by (metis star-matrix'.simpss(2))
finally show \fr(\{fo\} ?t + ?t(g)) ?t o star-matrix' ?t g = star-matrix' ?t g
.
qed
qed
thus \fo o y o y^\circ \leq y^\circ
by (simp add: enum-distinct restrict-all)
next
fix x y z :: ('a,'b) square
let ?e = enum-class.enum::'a list
have \forall g h :: ('a,'b) square . \forall zs . distinct ?e \And distinct zs -> (\fr(e) ?e o \frh(zs) ?e <= \fr(e) ?e(zs) star-matrix' ?g o \frh(zs) ?e <= \fr(e) ?e(zs)
proof (induct rule: list.induct)
case Nil thus ?case
by (simp add: restrict-empty-left)
case (Cons k s)
let ?t = k#s
assume 1: \forall g h :: ('a,'b) square . \forall zs . distinct s \And distinct zs -> (s(g) s o s(h) zs <= s(h) zs star-matrix' s g o s(h) zs <= s(h) zs)
show \forall g h :: ('a,'b) square . \forall zs . distinct ?t \And distinct zs -> (?t(g) ?t o ?t(h) zs <= ?t(h) zs star-matrix' ?t g o ?t(h) zs <= ?t(h) zs)
proof (intro all)
fix g h :: ('a,'b) square
fix zs :: 'a list

show distinct ?t ∧ distinct zs → (?t(g) g ∩ ?t(h)zs ≤ ?t(h)zs →
star-matrix' ?t g ∩ ?t(h)zs ≤ ?t(h)zs)

proof (cases zs)
  case Nil thus thesis
  by (metis restrict-empty-right restrict-star restrict-times)

next
  case (Cons y ys)
  assume 2: zs = y#ys
  show distinct ?t ∧ distinct zs → (?t(g) g ∩ ?t(h)zs ≤ ?t(h)zs →
star-matrix' ?t g ∩ ?t(h)zs ≤ ?t(h)zs)
proof (intro impI)
  let ?y = [g]
  assume 3: distinct ?t ∧ distinct zs
  hence 4: distinct s ∧ distinct ys ∧ ¬ List.member s k ∧ ¬ List.member
ys y

  using 2 by (simp add: List.member-def)
  let ?r = [k]
  let ?a = ?r(g) g ∩ ?r(h)
  let ?b = ?r(g)s
  let ?c = s(g) g ∩ ?r(h)
  let ?d = s(g)s
  let ?as = ?r(star o ?a) g ∩ ?r(h)
  let ?ds = star-matrix' s ?d
  let ?es = ?r(star o ?e) g ∩ ?r(h)
  let ?f = ?d ∪ ?c ∩ ?as ∩ ?b
  let ?fs = star-matrix' s ?f
  let ?ha = ?r(h) g ∩ ?r(h)
  let ?hb = ?r(h)s
  let ?hc = s(h) g ∩ ?r(h)
  let ?hd = s(h)s
  assume ?t(g) g ∩ ?t(h)zs ≤ ?t(h)zs

  using 2 3 4 by (simp add: restrict-nonempty-product-less-eq)
  have 6: s(?ds)s = ?ds ∧ s(?fs)s = ?fs
  by (simp add: restrict-star)
  hence 7: ?r(?c) g ∩ ?e ∧ s(?f)s = ?f
  by (metis (no-types, lifting) restrict-one-left-unit restrict-sup
restrict-times)
  have 8: disjoint s ?r ∧ disjoint ?r s
  using 3 by (simp add: in-set-member member-rec(1) member-rec(2))
proof –
  have ?es ∩ ?t(h)zs = ?es ∩ (?ha ⊕ ?hb ⊕ ?hc ⊕ ?hd)
    using 2 by (metis restrict-nonempty)
    by (simp add: matrix-idempotent-semiring mult-left-dist-supp)

96
also have ... = ?es o ?ha o ?es o ?hb
  using 8 by (simp add: times-disjoint)
finally show ?thesis
.
qed

proof -
  have ?as o ?b o ?fs o ?t(h)zs = ?as o ?b o ?fs o (?ha o ?hb o ?hc o ?hd)
using 2 by (metis restrict-nonempty)
also have ... = ?as o ?b o ?fs o ?ha o ?as o ?b o ?fs o ?hb o ?as o ?b o ?fs o ?hd
  by (simp add: matrix-idempotent-semiring.multi-left-dist-sup)
also have ... = ?as o ?b o (?fs o ?ha) o ?as o ?b o (?fs o ?hb) o ?as o ?b o ?fs o ?hd
  by (simp add: matrix-monoid.multi-assoc)
also have ... = ?as o ?b o mbot o ?as o ?b o mbot o ?as o ?b o ?fs o ?hd
using 6 8 by (metis (no-types) times-disjoint)
also have ... = ?as o ?b o ?fs o ?hc o ?as o ?b o ?fs o ?hd
  by simp
finally show ?thesis
.
qed

  by (simp add: times-disjoint)
  by (simp add: matrix-idempotent-semiring.multi-left-dist-sup)
  by (simp add: matrix-monoid.multi-assoc)
using 8 by (metis times-disjoint)
also have ... = ?ds o ?c o ?es o ?ha o ?ds o ?c o ?es o ?hb
  by simp
finally show ?thesis
.
qed

have 12: ?fs o ?t(h)zs = ?fs o ?hc o ?fs o ?hd
proof -
  have ?fs o ?t(h)zs = ?fs o (?ha o ?hb o ?hc o ?hd)
using 2 by (metis restrict-nonempty)

97
also have \( \ldots = ?fs \odot ?ha \odot \ldots \)
by (simp add: matrix-idempotent-semiring.mult-left-dist-sup)
also have \( \ldots = ?fs \odot ?hc \odot \ldots \)
using 6 8 by (metis (no-types) times-disjoint
matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left)
finally show \( \ldots \)
qed
have 13: ?es \odot ?ha \leq \ldots
proof –
  have ?b \odot ?ds \odot ?c \odot ?ha \leq \ldots
  using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
matrix-monoid.mult-assoc)
also have \( \ldots \leq \ldots \)
using 1 3 5 by (simp add:
matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc
member-rec(2) restrict-sublist)
also have \( \ldots \leq \ldots \)
using 5 by simp
finally have ?c \odot ?ha \leq \ldots
using 5 by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
thus \( \ldots \)
using 7 by (simp add: restrict-star-left-induct)
qed
have 14: ?es \odot \ldots \leq \ldots
proof –
  have ?b \odot ?ds \odot ?c \odot \ldots \leq \ldots
  using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
matrix-monoid.mult-assoc)
also have \( \ldots \leq \ldots \)
using 1 4 5 by (simp add:
matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc
restrict-sublist)
also have \( \ldots \leq \ldots \)
using 5 by simp
finally have ?c \odot \ldots \leq \ldots
using 5 by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
thus \( \ldots \)
using 4 7 by (simp add: restrict-star-left-induct)
qed
have 15: ?fs \odot \ldots \leq \ldots
proof –
  have ?c \odot ?as \odot \ldots \leq \ldots
  using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
matrix-monoid.mult-assoc)
also have \( \ldots \leq \ldots \)
using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
matrix-monoid.mult-assoc restrict-star-left-induct restrict-sublist)
also have \( \ldots \leq \ldots \)
98
using 5 by simp
finally have \( ?f \odot ?hc \leq ?hc \)
using 5 by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
thus \( ?\text{thesis} \)
using 1 3 7 by simp
qed
have 16: \( ?fs \odot ?hd \leq ?hd \)
proof -
  have \( ?c \odot ?as \odot ?b \odot ?hd \leq ?c \odot ?as \odot ?hb \)
    using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
                   matrix-monoid.mult-assoc)
  also have ... \( \leq ?c \odot ?hb \)
    using 4 5 by (simp add:
                   matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc
                   restrict-star-left-induct restrict-sublist)
  also have ... \( \leq ?hd \)
    using 5 by simp
finally have \( ?f \odot ?hd \leq ?hd \)
using 5 by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)
thus \( ?\text{thesis} \)
using 1 4 7 by simp
qed
have 17: \( ?as \odot ?b \odot ?fs \odot ?hc \leq ?ha \)
proof -
  have \( ?as \odot ?b \odot ?fs \odot ?hc \leq ?as \odot ?b \odot ?hc \)
    using 15 by (simp add:
                 matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc)
  also have ... \( \leq ?as \odot ?ha \)
    using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
                   matrix-monoid.mult-assoc)
  also have ... \( \leq ?ha \)
    using 5 by (simp add: restrict-star-left-induct restrict-sublist)
finally show \( ?\text{thesis} \)
.
qed
have 18: \( ?as \odot ?b \odot ?fs \odot ?hd \leq ?hb \)
proof -
  have \( ?as \odot ?b \odot ?fs \odot ?hd \leq ?as \odot ?b \odot ?hd \)
    using 16 by (simp add:
                 matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc)
  also have ... \( \leq ?as \odot ?hb \)
    using 5 by (simp add: matrix-idempotent-semiring.mult-right-isotone
                   matrix-monoid.mult-assoc)
  also have ... \( \leq ?hb \)
    using 4 5 by (simp add: restrict-star-left-induct restrict-sublist)
finally show \( ?\text{thesis} \)
.
qed
have 19: \( ?ds \odot ?c \odot ?es \odot ?ha \leq ?hc \)

99
proof –
  using 13 by (simp add: matrix-idempotent-semiring mult-right-isotone matrix-monoid mult-assoc)
also have ... ≤ ?ds ⊕ ?hc
  using 5 by (simp add: matrix-idempotent-semiring mult-right-isotone matrix-monoid mult-assoc)
also have ... ≤ ?hc
  using 1 3 5 by (simp add: restrict-sublist)
finally show ?thesis
.
qed

proof –
  using 14 by (simp add: matrix-idempotent-semiring mult-right-isotone matrix-monoid mult-assoc)
also have ... ≤ ?ds ⊕ ?hd
  using 5 by (simp add: matrix-idempotent-semiring mult-right-isotone matrix-monoid mult-assoc)
also have ... ≤ ?hd
  using 1 4 5 by (simp add: restrict-sublist)
finally show ?thesis
.
qed

  using 13 17 matrix-semilattice-sup le-sup/ by blast
  using 14 18 matrix-semilattice-sup le-sup/ by blast
  using 15 19 matrix-semilattice-sup le-sup/ by blast
  using 16 20 matrix-semilattice-sup le-sup/ by blast
  by (metis star-matrix'..simps(2))
also have ... = ?es ⊕ ?t(h)zs ⊕ ?as ⊕ ?b ⊕ ?fs ⊕ ?t(h)zs ⊕ ?ds ⊕ ?c ⊕ ?es ⊕ ?t(h)zs ⊕ ?fs ⊕ ?t(h)zs
  by (simp add: matrix-idempotent-semiring mult-right-dist-sup)
  using 9 10 11 12 by (simp only: matrix-semilattice-sup sup-assoc)
  by (simp only: matrix-semilattice-sup sup-assoc matrix-semilattice-sup sup-commute matrix-semilattice-sup sup-left-commute)
also have ... ≤ ?ha ⊕ ?hb ⊕ ?hc ⊕ ?hd
using 21 22 23 24 \text{matrix-semilattice-sup.sup.mono} by blast
also have \ldots = \exists t(h) zs
using 2 by (metis restrict-nonempty)
finally show \text{star-matrix'} \forall g \circ \exists t(h) zs \leq \exists t(h) zs
.

\text{qed}
\text{qed}
\text{qed}
\text{hence} \ \forall zs . \ \text{distinct} \ zs \rightarrow (y \circ \exists e(x) zs \leq \exists e(x) zs \rightarrow y^0 \circ \exists e(x) zs \leq \exists e(x)zs)
by (simp add: enum-distinct restrict-all)
\text{thus} \ y \odot x \leq x \rightarrow y^0 \circ x \leq x
by (metis restrict-all enum-distinct)
next
\text{fix} \ x \ y \ z :: ('a,'b) square
\text{let} \ \forall e = \text{enum-class.enum.}'a list
\text{have} \ \forall g h :: ('a,'b) square . \ \forall zs . \ \text{distinct} \ s \land \text{distinct} \ zs \rightarrow (zs(h)s \circ s(g)s \leq zs(h)s) \rightarrow zs(h)s \circ \text{star-matrix'} \ s \leq zs(h)s)
\text{show} \ \forall g h :: ('a,'b) square . \ \forall zs . \ \text{distinct} \ ?t \land \text{distinct} \ zs \rightarrow (zs(h)?t \circ ?t(g)?t \leq zs(h)?t) \rightarrow zs(h)?t \circ \text{star-matrix'} \ ?t \ g \leq zs(h)?t)
\text{proof (intro allI)}
\text{fix} \ g h :: ('a,'b) square
\text{fix} \ zs :: 'a list
\text{show} \ \text{distinct} \ ?t \land \text{distinct} \ zs \rightarrow (zs(h)?t \circ ?t(g)?t \leq zs(h)?t) \rightarrow zs(h)?t
\circ \text{star-matrix'} \ ?t \ g \leq zs(h)?t)
\text{proof (cases zs)}
\text{case Nil thus \ ?thesis}
\text{by (metis restrict-empty-left restrict-star restrict-times)}
next
\text{case (Cons k s)}
\text{assume 1:} \ \forall g h :: ('a,'b) square . \ \forall zs . \ \text{distinct} \ s \land \text{distinct} \ zs \rightarrow (zs(h)s \circ s(g)s \leq zs(h)s) \rightarrow zs(h)s \circ \text{star-matrix'} \ s \leq zs(h)s)
\text{show} \ \forall g h :: ('a,'b) square . \ \forall zs . \ \text{distinct} \ ?t \land \text{distinct} \ zs \rightarrow (zs(h)?t \circ ?t(g)?t \leq zs(h)?t) \rightarrow zs(h)?t
\circ \text{star-matrix'} \ ?t \ g \leq zs(h)?t)
\text{proof (intro implI)}
\text{let} \ ?y = [y]
\text{assume 3:} \ \text{distinct} \ ?t \land \text{distinct} \ zs
\text{hence 4:} \ \text{distinct} \ s \land \text{distinct} \ ys \land \neg \text{List.member} \ s \ k \land \neg \text{List.member} \ ys \ y
\text{using 2 by (simp add: List.member-def)}
\text{let} \ ?r = [k]
\text{let} \ ?a = ?r(g)\ ?r

101
let \(?b = ?r(g)\) s
let \(?c = s(g)\) r
let \(?d = s(g)\) s
let \(?as = ?r(star o ?a)\) r
let \(?ds = star-matrix'\ s\ ?d\)
let \(\ell c = ?a + ?b\ o\ ?ds\ ?c\)
let \(?es = ?r(star o ?e)\) r
let \(?fs = star-matrix'\ s\ ?f\)
let \(?ha = ?y(h)\) r
let \(?hb = ?y(h)\) s
let \(?hc = ys(h)\) r
let \(?hd = ys(h)\) s
assume \(zs(h)\) ?t \(\circ\) \(?t(g)\) ?t \(\leq\) \(zs(h)\) ?t
hence 5: \(?ha\ o\ ?a\ o\ \neg\ ?hb\ o\ ?c\ \leq\ \neg\ ?ha\ o\ \neg\ ?b\ o\ \neg\ ?hb\ o\ ?d\ \leq\ \neg\ ?hb\ \wedge\ ?hc\ o\ \neg\ ?a\ o\ \neg\ ?hd\ o\ ?c\ \leq\ \neg\ ?hc\ \wedge\ ?hc\ o\ \neg\ ?b\ o\ \neg\ ?hd\ o\ ?d\ \leq\ \neg\ ?hd\)
using 2\ 3\ 4 by (simp add: restrict-nonempty-product-less-eq)
have 6: \(s(?ds)s = ?ds\ \wedge\ s(?fs)s = ?fs\)
by (simp add: restrict-star)
 hence 7: \(\neg\ ?r(?e)\ ?r\ =\ ?e\ \wedge\ s(?f)s = ?f\)
by (metis (no-types, lifting) restrict-one-left-unit restrict-sup restrict-times)

have 8: disjoint s ?r \land disjoint ?r s

using 3 by (simp add: in-set-member member-rec)

have 9: \(zs(h)\) ?t \(\circ\) \(?es = ?ha\ o\ ?es\ o\ \neg\ ?hc\ o\ ?es\)
proof -

have \(zs(h)\) ?t \(\circ\) \(?es = (\neg\ ?ha\ o\ ?hb\ o\ \neg\ ?hc\ o\ ?hd)\) \(\circ\) \(?es\)

using 2 by (metis restrict-nonempty)

also have \(\ldots = ?ha\ o\ ?es\ o\ \neg\ ?hb\ o\ \neg\ ?es\ o\ ?es\ o\ \neg\ ?hd\ o\ ?es\)

by (simp add: matrix-idempotent-semiring.mult-right-dist-sup)

also have \(\ldots = ?ha\ o\ ?es\ o\ \neg\ ?hc\ o\ ?es\)

using 8 by (simp add: times-disjoint)

finally show ?thesis

qed

have 10: \(zs(h)\) ?t \(\circ\) \(?as\ o\ \neg\ ?b\ o\ ?fs = ?ha\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\ o\ ?hc\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\)

proof -

have \(zs(h)\) ?t \(\circ\) \(?as\ o\ \neg\ ?b\ o\ ?fs = (\neg\ ?ha\ o\ ?hb\ o\ \neg\ ?hc\ o\ ?hd)\) \(\circ\) \(?as\ o\ \neg\ ?b\ o\ ?fs\)

using 2 by (metis restrict-nonempty)

also have \(\ldots = ?ha\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\ o\ ?hb\ o\ \neg\ ?as\ o\ \neg\ ?b\ o\ ?fs\ o\ ?hc\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\ o\ mbot\ o\ \neg\ ?fs\ o\ ?hc\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\)

using 8 by (metis (no-types) times-disjoint)

also have \(\ldots = ?ha\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\ o\ ?hc\ o\ ?as\ o\ \neg\ ?b\ o\ ?fs\)

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

\[\text{proof} - \]

have 11: \( \langle h \rangle \cdot t \circ ds \circ c \circ es = \langle hb \circ ds \circ c \circ es \rangle \circ \langle hd \circ c \circ es \rangle \)

\[\text{proof} - \]

have \( \langle h \rangle \cdot t \circ ds \circ c \circ es = (\langle ha \circ hb \circ hc \circ hd \rangle) \circ ds \circ c \circ es \)

\[\text{proof} - \]

using 2 by (metis restrict-nonempty)

also have \( \ldots = \langle ha \circ ds \circ c \circ es \rangle \langle hb \circ ds \circ c \circ es \rangle \langle hc \circ ds \circ c \circ es \rangle \langle hd \circ ds \circ c \circ es \rangle \)

\[\text{proof} - \]

by (simp add: matrix-idempotent-semiring, mult-right-dist-sup)

also have \( \ldots = \langle mbot \circ c \circ es \rangle \langle hb \circ ds \circ c \circ es \rangle \langle mbot \circ c \circ es \rangle \langle hd \circ ds \circ c \circ es \rangle \)

\[\text{proof} - \]

by simp

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

\[\text{proof} - \]

have 12: \( \langle h \rangle \cdot t \circ fs = \langle hb \circ fs \circ hd \circ fs \rangle \)

\[\text{proof} - \]

have \( \langle h \rangle \cdot t \circ fs = (\langle ha \circ hb \circ hc \circ hd \rangle) \circ fs \)

\[\text{proof} - \]

using 2 by (metis restrict-nonempty)

also have \( \ldots = \langle ha \circ fs \circ hb \circ fs \circ hc \circ fs \circ hd \circ fs \rangle \)

\[\text{proof} - \]

by (simp add: matrix-idempotent-semiring, mult-right-dist-sup)

also have \( \ldots = \langle hb \circ fs \circ hd \circ fs \rangle \)

\[\text{proof} - \]

using 6 8 by (metis (no-types) times-disjoint)

matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right

matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left

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

\[\text{proof} - \]

have 13: \( \langle ha \circ es \rangle \leq \langle ha \rangle \)

\[\text{proof} - \]

have \( \langle ha \circ b \circ ds \circ c \leq \langle hb \circ ds \circ c \rangle \)

\[\text{proof} - \]

using 5 by (simp add: matrix-idempotent-semiring, mult-left-isotone)

also have \( \ldots \leq \langle hb \circ c \rangle \)

\[\text{proof} - \]

using 1 4 5 by (simp add:

matrix-idempotent-semiring, mult-left-isotone restrict-sublist)

also have \( \ldots \leq \langle ha \rangle \)

\[\text{proof} - \]

finally have \( \langle ha \circ e \leq \langle ha \rangle \)

\[\text{proof} - \]

using 5 by (simp add: matrix-idempotent-semiring, mult-left-dist-sup

matrix-monoid, mult-assoc)

thus \( \text{thesis} \)

\[\text{proof} - \]

using 7 by (simp add: restrict-star-right-induct)

qed

have 14: \( \langle hb \circ fs \rangle \leq \langle hb \rangle \)
proof

have \( ?hb \odot ?c \odot ?as \odot ?b \preceq ?ha \odot ?as \odot ?b \)
using 5 by (metis matrix-semilattice-sup.le-supE
matrix-idempotent-semiring.mult-left-isotone)
also have \( \ldots \preceq ?ha \odot ?b \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
restrict-star-right-induct restrict-sublist)
also have \( \ldots \preceq ?hb \)
using 5 by simp
finally have \( ?hb \odot ?e \preceq ?hb \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-dist-sup
matrix-monoid.mult-assoc)
thus \( \Box \)
using 1 3 7 by simp
qed

have 15: \( ?hc \odot ?es \preceq ?hc \)
proof

have \( ?hc \odot ?b \odot ?ds \odot ?c \preceq ?hd \odot ?ds \odot ?c \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
also have \( \ldots \preceq ?hd \odot ?c \)
using 1 4 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
restrict-star-right-induct restrict-sublist)
also have \( \ldots \preceq ?hc \)
using 5 by simp
finally have \( ?hc \odot ?e \preceq ?hc \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-dist-sup
matrix-monoid.mult-assoc)
thus \( \Box \)
using 4 7 by (simp add: restrict-star-right-induct
qed

have 16: \( ?hd \odot ?fs \preceq ?hd \)
proof

have \( ?hd \odot ?c \odot ?as \odot ?b \preceq ?hc \odot ?as \odot ?b \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
also have \( \ldots \preceq ?hc \odot ?b \)
using 4 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
restrict-star-right-induct restrict-sublist)
also have \( \ldots \preceq ?hd \)
using 5 by simp
finally have \( ?hd \odot ?f \preceq ?hd \)
using 5 by (simp add: matrix-idempotent-semiring.mult-left-dist-sup
matrix-monoid.mult-assoc)
thus \( \Box \)
using 1 4 7 by simp
qed

have 17: \( ?hb \odot ?ds \odot ?c \odot ?es \preceq ?ha \)
proof

have \( ?hb \odot ?ds \odot ?c \odot ?es \preceq ?hb \odot ?c \odot ?es \)
using 1 4 5 by (simp add:
\textbf{matrix-idempotent-semiring} \multleft-isotone \textbf{restrict-sublist}

also have ... \preceq \texttt{ha} \odot \texttt{as} \odot \texttt{c} \odot \texttt{es} \preceq \texttt{hb}

using 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone)
also have ... \preceq \texttt{ha}
using 13 by simp
finally show \texttt{thesis}
\begin{itemize}
\item qed
\end{itemize}

have 18: \texttt{ha} \odot \texttt{as} \odot \texttt{b} \odot \texttt{fs} \preceq \texttt{hb}

\textbf{proof} –

have \texttt{ha} \odot \texttt{as} \odot \texttt{b} \odot \texttt{fs} \preceq \texttt{ha} \odot \texttt{b} \odot \texttt{fs}
using 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
\textbf{restrict-star-right-induct} \textbf{restrict-sublist})
also have ... \preceq \texttt{hb} \odot \texttt{fs}
using 14 by simp
finally show \texttt{thesis}
by simp
qed

have 19: \texttt{hd} \odot \texttt{ds} \odot \texttt{c} \odot \texttt{es} \preceq \texttt{hc}

\textbf{proof} –

have \texttt{hd} \odot \texttt{ds} \odot \texttt{c} \odot \texttt{es} \preceq \texttt{hd} \odot \texttt{c} \odot \texttt{es}
using 1 4 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
\textbf{restrict-star-right-induct} \textbf{restrict-sublist})
also have ... \preceq \texttt{hc} \odot \texttt{es}
using 15 by simp
finally show \texttt{thesis}
by simp
qed

have 20: \texttt{hc} \odot \texttt{as} \odot \texttt{b} \odot \texttt{fs} \preceq \texttt{hd}

\textbf{proof} –

have \texttt{hc} \odot \texttt{as} \odot \texttt{b} \odot \texttt{fs} \preceq \texttt{hc} \odot \texttt{b} \odot \texttt{fs}
using 4 5 by (simp add: matrix-idempotent-semiring.mult-left-isotone
\textbf{restrict-star-right-induct} \textbf{restrict-sublist})
also have ... \preceq \texttt{hd} \odot \texttt{fs}
using 16 by simp
finally show \texttt{thesis}
by simp
qed

have 21: \texttt{ha} \odot \texttt{es} \odot \texttt{hb} \odot \texttt{ds} \odot \texttt{c} \odot \texttt{es} \preceq \texttt{ha}
using 13 17 matrix-semilattice-sup.le-supI by blast

have 22: \texttt{ha} \odot \texttt{as} \odot \texttt{b} \odot \texttt{fs} \odot \texttt{hb} \odot \texttt{fs} \preceq \texttt{hb}
using 14 18 matrix-semilattice-sup.le-supI by blast

have 23: \texttt{hc} \odot \texttt{es} \odot \texttt{hd} \odot \texttt{ds} \odot \texttt{c} \odot \texttt{es} \preceq \texttt{hc}
using 15 19 matrix-semilattice-sup.le-sup1 by blast
using 16 20 matrix-semilattice-sup.le-sup1 by blast
by (metis star-matrix'.simsps(2))
also have ... = zs(h)?t ⊗ ?es ⊗ zs(h)?t ⊗ ?as ⊗ ?b ⊗ ?fs ⊗ zs(h)?t ⊗ ?fs
by (simp add: matrix-idempotent-semiring.mult-left-dist-sup
matrix-monoid.mult-associac)
using 9 10 11 12 by (simp add: matrix-semilattice-sup-sup-associac)
using 9 10 11 12 by (simp only: matrix-semilattice-sup.sup-associac
matrix-semilattice-sup.sup-commute matrix-semilattice-sup.sup-left-commute)
also have ... ≤ ?ha ⊗ ?hb ⊗ ?hc ⊗ ?hd
using 21 22 23 24 matrix-semilattice-sup.sup-mono by blast
also have ... = zs(h)?t
using 2 by (metis restrict-nonempty)
finally show zs(h)?t ⊗ star-matrix'* ?t g ≤ zs(h)?t
.
qed
qed
qed
hence ∀ zs. distinct zs → (zs(x)?e ⊔ y ≤ zs(x)?e → zs(x)?e ⊔ yΟ ≤ 
zs(x)?e)
by (simp add: enum-distinct restrict-all)
thus x ⊔ y ≤ x → x ⊔ yΟ ≤ x
by (metis restrict-all enum-distinct)
qed

6.3 Matrices form a Stone-Kleene Relation Algebra

Matrices over Stone-Kleene relation algebras form a Stone-Kleene relation algebra. It remains to prove the axiom about the interaction of Kleene star and double complement.

interpretation matrix-stone-kleene-relation-algebra: stone-kleene-relation-algebra
where sup = sup-matrix and inf = inf-matrix and less-eq = less-eq-matrix
and less = less-matrix and bot =
bot-matrix::('a::enum,'b::stone-kleene-relation-algebra) square and top =
top-matrix and uminus = uminus-matrix and one = one-matrix and times =
times-matrix and conv = conv-matrix and star = star-matrix
proof
fix x :: ('a,'b) square

106
let ?e = enum-class enum::'a list
let ?o = mone :: ('a,'b) square
show oo(xo) = (oox)o

proof (rule matrix-order.antisym)
  have \forall g :: ('a,'b) square . distinct ?e \rightarrow oo(star-matrix' ?e (oog)) = star-matrix' ?e (oo g)
proof (induct rule: list.induct)
  case Nil thus \case
  by simp
next
  case (Cons k s)
  fix g :: ('a,'b) square
  assume 1: \forall g :: ('a,'b) square . distinct s \rightarrow oo(star-matrix' s (oo g)) = star-matrix' s (oo g)
  show \forall g :: ('a,'b) square . distinct ?t \rightarrow oo(star-matrix' ?t (oo g)) = star-matrix' ?t (oo g)
proof (rule allI, rule impI)
  fix g :: ('a,'b) square
  assume 2: distinct ?t
let ?t = k#s
  show \forall g :: ('a,'b) square . distinct ?t \rightarrow oo(star-matrix' ?t (oo g)) = star-matrix' ?t (oo g)
proof (rule matrix-order.antisym)
  have oo\star ?e = oo ?as \oplus ?a \ominus ?as \ominus ?c = oo oo ?c = oo oo ?d = oo ?e
  by (simp add: restrict-star)
  have oo\star ?e = oo ?a \ominus ?a = oo\star ?b = oo ?b \ominus ?b = oo\star ?c = oo\star ?c = oo\star ?d = oo \star
  by (metis matrix-p-algebra.regular-closed-p restrict-pp)
  hence oo\star ?e = oo ?e
  by (metis pp-star-commute restrict-pp)
  hence oo ?f = ?f
  using 1 2 by (metis matrix-stone-algebra.regular-closed-sup matrix-stone-relation-algebra.regular-mul-closed)
  hence oo ?fs = oo\star
  using 1 2 by (metis distinct.simps(2))
  have oo\star ?ds = oo\star
  using 1 2 by (simp add: restrict-pp)
  hence oo?e = oo ?e
  using 1 2 by (metis matrix-stone-algebra.regular-closed-sup matrix-stone-relation-algebra.regular-mul-closed)
  have oo\star ?e = oo ?a \ominus oo ?b = oo ?b \ominus oo ?c = oo \star
  by (metis matrix-stone-algebra.regular-closed-sup matrix-stone-relation-algebra.regular-mul-closed)
  hence oo ?es = oo ?e
  by (metis pp-star-commute restrict-pp)
  have oo\star(star-matrix' ?t (oo g)) = oo\star(oo ?es \ominus oo ?as \ominus oo ?b \ominus oo\star ?ds \ominus oo ?c

107
also have ... = \circ\circ ?es \odot \circ\circ ?as \odot \circ\circ ?b \odot \circ\circ ?fs \odot \circ\circ ?ds \odot \circ\circ ?c \odot 
\circ\circ ?es \odot \circ\circ ?fs
by (simp add: matrix-stone-relation-algebra.pp-dist-comp)
also have ... = ?es \odot ?as \odot ?b \odot ?fs \odot ?ds \odot ?c \odot ?es \odot ?fs
using 3 4 5 6 7 by simp
finally show \circ\circ (\star-matrix'(?t (\circ\circ g))) = \star-matrix'(?t (\circ\circ g))
by (metis star-matrix'.simp(2))
qed
qed

next
have \ni \odot \circ\circ x \odot \circ\circ (x^0) \preceq \circ\circ (x^0)
by (metis matrix-kleene-algebra.star-left-unfold-equal
matrix-p-algebra.pp-increasing matrix-p-algebra.pp-isotone)
thus (\circ\circ x)^0 \preceq \circ\circ (x^0)
using matrix-kleene-algebra.star-left-induct by fastforce
qed
qed
end

References


