Stone-Kleene Relation Algebras

Walter Guttmann

October 11, 2017

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.

Contents

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].

theory *Iterings*

imports *Stone-Relation-Algebras.Semirings*

begin

2.1 Conway Semirings

In this section, we consider equational axioms for iteration. The algebraic structures are based on idempotent left semirings, which are expanded by a unary iteration operation. We start with an unfold property, one inequality of the sliding rule and distributivity over joins, which is similar to Conway's sumstar.

class *circ* =
  fixes *circ* :: $'a \Rightarrow 'a$ (-$^\circ$ [*100*] *100*)

class *left-conway-semiring* = *idempotent-left-semiring* + *circ* +
  assumes *circ-left-unfold*: $1 \sqcup x * x^\circ = x^\circ$
  assumes *circ-left-slide*: $(x * y)^\circ * x \leq x * (y * x)^\circ$
  assumes *circ-sup-1*: $(x \sqcup y)^\circ = x^\circ * (y * x^\circ)^\circ$
begin

We obtain one inequality of Conway's productstar, as well as of the other unfold rule.

lemma *circ-mult-sub*:
$1 \sqcup x * (y * x)^\circ * y \le (x * y)^\circ$
by (*metis sup-right-isotone circ-left-slide circ-left-unfold mult-assoc mult-right-isotone*)

lemma *circ-right-unfold-sub*:
$1 \sqcup x^\circ * x \le x^\circ$
by (*metis circ-mult-sub mult-1-left mult-1-right*)

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

lemma *circ-increasing*:
$x \le x^\circ$
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*:
$1 \le x^\circ$
by (*metis sup-left-divisibility circ-left-unfold*)

lemma *circ-mult-increasing*:
$x \le x * x^\circ$
by (*metis circ-reflexive mult-right-isotone mult-1-right*)

lemma *circ-mult-increasing-2*:
$x \le x^\circ * x$
by (*metis circ-reflexive mult-left-isotone mult-1-left*)

lemma *circ-transitive-equal*:
$x^\circ * x^\circ = x^\circ$
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*:
$x^{\circ\circ\circ} = x^{\circ\circ}$
by (*metis sup-idem circ-sup-1 circ-increasing circ-transitive-equal le-iff-sup*)

lemma *circ-one*:
$1^\circ = 1^{\circ\circ}$
by (*metis circ-circ-circ circ-zero*)

lemma *circ-sup-sub*:
$(x^\circ * y)^\circ * x^\circ \le (x \sqcup y)^\circ$
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 *circ-rtc-2*:
$1 \sqcup x \sqcup x^\circ * x^\circ = x^\circ$
by (*metis sup-assoc circ-increasing circ-plus-one circ-transitive-equal le-iff-sup*)

lemma *mult-zero-circ*:
$(x * bot)^\circ = 1 \sqcup x * bot$
by (*metis circ-left-unfold mult-assoc mult-left-zero*)

lemma *mult-zero-sup-circ*:
$(x \sqcup y * bot)^\circ = x^\circ * (y * bot)^\circ$
by (*metis circ-sup-1 mult-assoc mult-left-zero*)

lemma *circ-plus-sub*:
$x^\circ * x \le x * x^\circ$
by (*metis circ-left-slide mult-1-left mult-1-right*)

lemma *circ-loop-fixpoint*:
$y * (y^\circ * z) \sqcup z = y^\circ * z$
by (*metis sup-commute circ-left-unfold mult-assoc mult-1-left mult-right-dist-sup*)

lemma *left-plus-below-circ*:
$x * x^\circ \le x^\circ$
by (*metis sup.cobounded2 circ-left-unfold*)

lemma *right-plus-below-circ*:
$x^\circ * x \le x^\circ$
using *circ-right-unfold-sub* **by** *auto*

lemma *circ-sup-upper-bound*:
$x \le z^\circ \Longrightarrow y \le z^\circ \Longrightarrow x \sqcup y \le z^\circ$
by *simp*

lemma *circ-mult-upper-bound*:
$x \le z^\circ \Longrightarrow y \le z^\circ \Longrightarrow x * y \le z^\circ$
by (*metis mult-isotone circ-transitive-equal*)

lemma *circ-sub-dist*:
$x^\circ \le (x \sqcup y)^\circ$
by (*metis circ-sup-sub circ-plus-one mult-1-left mult-right-sub-dist-sup-left order-trans*)

lemma *circ-sub-dist-1*:
$x \le (x \sqcup y)^\circ$
using *circ-increasing le-supE* **by** *blast*

lemma *circ-sub-dist-2*:
$x * y \le (x \sqcup y)^\circ$
by (*metis sup-commute circ-mult-upper-bound circ-sub-dist-1*)

lemma *circ-sub-dist-3*:
$x^\circ * y^\circ \le (x \sqcup y)^\circ$
by (*metis sup-commute circ-mult-upper-bound circ-sub-dist*)

lemma *circ-isotone*:
$x \le y \Longrightarrow x^\circ \le y^\circ$
by (*metis circ-sub-dist le-iff-sup*)

lemma *circ-sup-2*:
$(x \sqcup y)^\circ \le (x^\circ * y^\circ)^\circ$
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*:
$1 \le x \Longrightarrow x * x^\circ = x^\circ$
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*:
$1 \le x \Longrightarrow x^\circ * x = x^\circ$
by (*metis antisym le-iff-sup mult-left-sub-dist-sup-left mult-1-right right-plus-below-circ*)

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

lemma *circ-decompose-6*:
$x^\circ * (y * x^\circ)^\circ = y^\circ * (x * y^\circ)^\circ$
by (*metis sup-commute circ-sup-1*)

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

lemma *circ-decompose-8*:
$(x \sqcup y)^\circ = (x \sqcup y)^\circ * x^\circ * y^\circ$
by (*metis antisym eq-refl mult-assoc mult-isotone mult-1-right*

circ-mult-upper-bound circ-reflexive circ-sub-dist-3)

lemma *circ-decompose-9*:
$(x^\circ * y^\circ)^\circ = x^\circ * y^\circ * (x^\circ * y^\circ)^\circ$
by (*metis circ-decompose-4 mult-assoc*)

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

lemma *circ-back-loop-prefixpoint*:
$(z * y^\circ) * y \sqcup z \leq z * y^\circ$
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*:
is-fixpoint $(\lambda x . y * x \sqcup z)$ $(y^\circ * z)$
by (*metis circ-loop-fixpoint is-fixpoint-def*)

lemma *circ-back-loop-is-prefixpoint*:
is-prefixpoint $(\lambda x . x * y \sqcup z)$ $(z * y^\circ)$
by (*metis circ-back-loop-prefixpoint is-prefixpoint-def*)

lemma *circ-circ-sup*:
$(1 \sqcup x)^\circ = x^{\circ\circ}$
by (*metis sup-commute circ-sup-1 circ-decompose-4 circ-zero mult-1-right*)

lemma *circ-circ-mult-sub*:
$x^\circ * 1^\circ \leq x^{\circ\circ}$
by (*metis circ-increasing circ-isotone circ-mult-upper-bound circ-reflexive*)

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

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

lemma *circ-square*:
$(x * x)^\circ \leq x^\circ$
by (*metis circ-increasing circ-isotone left-plus-circ mult-right-isotone*)

lemma *circ-mult-sub-sup*:

$(x * y)^\circ \le (x \sqcup y)^\circ$
by (*metis sup-ge1 sup-ge2 circ-isotone circ-square mult-isotone order-trans*)

lemma *circ-sup-mult-zero*:
$x^\circ * y = (x \sqcup y * bot)^\circ * y$
proof –
have $(x \sqcup y * bot)^\circ * y = x^\circ * (1 \sqcup y * bot) * y$
by (*metis mult-zero-sup-circ mult-zero-circ*)
also have ... $= x^\circ * (y \sqcup y * bot)$
by (*metis mult-assoc mult-1-left mult-left-zero mult-right-dist-sup*)
also have ... $= x^\circ * y$
by (*metis sup-commute le-iff-sup zero-right-mult-decreasing*)
finally show *?thesis*
by *simp*
qed

lemma *troeger-1*:
$(x \sqcup y)^\circ = x^\circ * (1 \sqcup y * (x \sqcup y)^\circ)$
by (*metis circ-sup-1 circ-left-unfold mult-assoc*)

lemma *troeger-2*:
$(x \sqcup y)^\circ * z = x^\circ * (y * (x \sqcup y)^\circ * z \sqcup z)$
by (*metis circ-sup-1 circ-loop-fixpoint mult-assoc*)

lemma *troeger-3*:
$(x \sqcup y * bot)^\circ = x^\circ * (1 \sqcup y * bot)$
by (*metis mult-zero-sup-circ mult-zero-circ*)

lemma *circ-sup-sub-sup-one-1*:
$x \sqcup y \le x^\circ * (1 \sqcup y)$
by (*metis circ-increasing circ-left-unfold mult-1-left mult-1-right mult-left-sub-dist-sup mult-right-sub-dist-sup-left order-trans sup-mono*)

lemma *circ-sup-sub-sup-one-2*:
$x^\circ * (x \sqcup y) \le x^\circ * (1 \sqcup y)$
by (*metis circ-sup-sub-sup-one-1 circ-transitive-equal mult-assoc mult-right-isotone*)

lemma *circ-sup-sub-sup-one*:
$x * x^\circ * (x \sqcup y) \le x * x^\circ * (1 \sqcup y)$
by (*metis circ-sup-sub-sup-one-2 mult-assoc mult-right-isotone*)

lemma *circ-square-2*:
$(x * x)^\circ * (x \sqcup 1) \le x^\circ$
by (*metis sup.bounded-iff circ-increasing circ-mult-upper-bound circ-reflexive circ-square*)

lemma *circ-extra-circ*:
$(y * x^\circ)^\circ = (y * y^\circ * x^\circ)^\circ$

by (*metis circ-decompose-6 circ-transitive-equal left-plus-circ mult-assoc*)

lemma *circ-circ-sub-mult*:
$1^\circ * x^\circ \leq x^{\circ\circ}$
by (*metis circ-increasing circ-isotone circ-mult-upper-bound circ-reflexive*)

lemma *circ-decompose-11*:
$(x^\circ * y^\circ)^\circ = (x^\circ * y^\circ)^\circ * x^\circ$
by (*metis circ-decompose-10 circ-decompose-4 circ-decompose-5 circ-decompose-9 left-plus-circ*)

lemma *circ-mult-below-circ-circ*:
$(x * y)^\circ \leq (x^\circ * y)^\circ * x^\circ$
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*:
$x^\circ * top = top$
by (*metis sup-right-top circ-loop-fixpoint*)

lemma *circ-left-top*:
$top * x^\circ = top$
by (*metis circ-right-top circ-top circ-decompose-11*)

lemma *mult-top-circ*:
$(x * top)^\circ = 1 \sqcup x * top$
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*:

$(x \sqcup y * bot)^\circ = x^\circ \sqcup x^\circ * y * bot$
by (*metis mult-assoc mult-left-dist-sup mult-1-right troeger-3*)

lemma *circ-unfold-sum*:
$(x \sqcup y)^\circ = x^\circ \sqcup x^\circ * y * (x \sqcup y)^\circ$
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*: $x * (y * x)^\circ \leq (x * y)^\circ * x$
begin

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

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

lemma *circ-right-unfold-1*:
$1 \sqcup x^\circ * x = x^\circ$
by (*metis circ-left-unfold circ-slide-1 mult-1-left mult-1-right*)

lemma *circ-mult-1*:
$(x * y)^\circ = 1 \sqcup x * (y * x)^\circ * y$
by (*metis circ-left-unfold circ-slide-1 mult-assoc*)

lemma *circ-sup-9*:
$(x \sqcup y)^\circ = (x^\circ * y)^\circ * x^\circ$
by (*metis circ-sup-1 circ-slide-1*)

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

lemma *circ-decompose-12*:
$x^\circ * y^\circ \leq (x^\circ * y)^\circ * x^\circ$
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*:
$(z * y^\circ) * y \sqcup z = z * y^\circ$
by (*metis sup-commute circ-left-unfold circ-plus-same mult-assoc*
mult-left-dist-sup mult-1-right)

lemma *circ-back-loop-is-fixpoint*:
is-fixpoint $(\lambda x . x * y \sqcup z)$ $(z * y^\circ)$
by (*metis circ-back-loop-fixpoint is-fixpoint-def*)

lemma *circ-elimination*:
$x * y = bot \Longrightarrow x * y^\circ \leq x$
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*: $z * x \leq y * z \longrightarrow z * x^\circ \leq y^\circ * z$
begin

lemma *circ-circ-mult*:
$1^\circ * x^\circ = x^{\circ\circ}$
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*:
$x * 1^\circ \leq 1^\circ * x$
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 $p \leq p * p$
and $p \leq 1$
and $p * x \leq x * p$
shows $p * x^\circ = p * (p * x)^\circ$
proof –
have $p * x \leq p * (p * x * p) * p$
by (*metis assms coreflexive-transitive eq-iff test-preserves-equation mult-assoc*)
hence $p * x^\circ \leq p * (p * x)^\circ$
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*: $z * x \le y * z \sqcup w \longrightarrow z * x^\circ \le y^\circ * (z \sqcup w * x^\circ)$
  assumes *circ-simulate-left*: $x * z \le z * y \sqcup w \longrightarrow x^\circ * z \le (z \sqcup x^\circ * w) * y^\circ$
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*:
  $x * z \le z * y \Longrightarrow x^\circ * z \le z * y^\circ \sqcup x^\circ * bot$
  by (*metis sup-monoid.add-0-right circ-simulate-left mult-assoc mult-left-zero mult-right-dist-sup*)

lemma *circ-separate-1*:
  assumes $y * x \le x * y$
    shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
proof −
  have $y^\circ * x \le x * y^\circ \sqcup y^\circ * bot$
    by (*metis assms circ-simulate-left-1*)
  hence $y^\circ * x * y^\circ \le x * y^\circ * y^\circ \sqcup y^\circ * bot * y^\circ$
    by (*metis mult-assoc mult-left-isotone mult-right-dist-sup*)
  also have ... $= x * y^\circ \sqcup y^\circ * bot$
    by (*metis circ-transitive-equal mult-assoc mult-left-zero*)
  finally have $y^\circ * (x * y^\circ)^\circ \le x^\circ * (y^\circ \sqcup y^\circ * bot)$
    using *circ-simulate-right mult-assoc* **by** *fastforce*
  also have ... $= x^\circ * y^\circ$
    by (*simp add*: *sup-absorb1 zero-right-mult-decreasing*)
  finally have $(x \sqcup y)^\circ \le x^\circ * y^\circ$
    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*:
  $x^\circ * 1^\circ = x^{\circ\circ}$
  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 $y * x \le x * y$

shows $y^\circ * x^\circ \leq x^\circ * y^\circ$
proof $-$
have $y * x^\circ \leq x^\circ * y$
by (*metis assms circ-simulate*)
hence $y^\circ * x^\circ \leq x^\circ * y^\circ \sqcup y^\circ * bot$
by (*metis circ-simulate-left-1*)
thus *?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 $r^\circ * q \leq q * r^\circ$
and $q \leq 1$
shows $s * (x \sqcup b \sqcup r \sqcup l)^\circ * q \leq s * (x * b^\circ * q \sqcup r \sqcup l)^\circ$
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 $s * (x \sqcup b \sqcup r \sqcup l)^\circ * q = s * l^\circ * (x \sqcup b \sqcup r)^\circ * q$
by (*metis sup-commute circ-separate-1 mult-assoc*)
also have ... $= s * l^\circ * b^\circ * r^\circ * q * (x * b^\circ * r^\circ * q)^\circ$
proof $-$
have $(b \sqcup r)^\circ = b^\circ * r^\circ$
by (*simp add*: *assms*(*4*) *circ-separate-1*)
hence $b^\circ * r^\circ * (q * (x * b^\circ * r^\circ))^\circ = (x \sqcup b \sqcup r)^\circ$
by (*metis* (*full-types*) *assms*(*2*) *circ-sup-1 sup-assoc sup-commute mult-assoc*)
thus *?thesis*
by (*metis circ-slide-1 mult-assoc*)
qed
also have ... $\leq s * l^\circ * b^\circ * r^\circ * q * (x * b^\circ * q * r^\circ)^\circ$
by (*metis assms*(*9*) *circ-isotone mult-assoc mult-right-isotone*)
also have ... $\leq s * q * l^\circ * b^\circ * r^\circ * (x * b^\circ * q * r^\circ)^\circ$
by (*metis assms*(*1*,*10*) *mult-left-isotone mult-right-isotone mult-1-right*)
also have ... $\leq s * l^\circ * q * b^\circ * r^\circ * (x * b^\circ * q * r^\circ)^\circ$
by (*metis assms*(*1*,*8*) *circ-simulate mult-assoc mult-left-isotone mult-right-isotone*)
also have ... $\leq s * l^\circ * r^\circ * (x * b^\circ * q * r^\circ)^\circ$
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-1-right)
also have ... $\leq s * (x * b^\circ * q \sqcup r \sqcup l)^\circ$
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 *itering* = *idempotent-left-zero-semiring* + *circ* +
assumes *circ-sup*: $(x \sqcup y)^\circ = (x^\circ * y)^\circ * x^\circ$
assumes *circ-mult*: $(x * y)^\circ = 1 \sqcup x * (y * x)^\circ * y$
assumes *circ-simulate-right-plus*: $z * x \leq y * y^\circ * z \sqcup w \longrightarrow z * x^\circ \leq y^\circ * (z \sqcup w * x^\circ)$
assumes *circ-simulate-left-plus*: $x * z \leq z * y^\circ \sqcup w \longrightarrow x^\circ * z \leq (z \sqcup x^\circ * w) * y^\circ$
begin

lemma *circ-right-unfold*:
$1 \sqcup x^\circ * x = x^\circ$
by (*metis circ-mult mult-1-left mult-1-right*)

lemma *circ-slide*:
$x * (y * x)^\circ = (x * y)^\circ * x$
proof −
have $x * (y * x)^\circ = Rf\ x\ (y * 1 \sqcup y * (x * (y * x)^\circ * y)) * x$
by (*metis* (*no-types*) *circ-mult mult-1-left mult-1-right mult-left-dist-sup mult-right-dist-sup mult-assoc*)
thus *?thesis*
by (*metis* (*no-types*) *circ-mult mult-1-right mult-left-dist-sup mult-assoc*)
qed

subclass *itering-3*
apply *unfold-locales*
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*)
by (*metis sup-commute sup-ge1 sup-right-isotone circ-mult mult-right-isotone mult-1-right order-trans circ-simulate-left-plus*)

lemma *circ-simulate-right-plus-1*:
$z * x \leq y * y^\circ * z \Longrightarrow z * x^\circ \leq y^\circ * z$
by (*metis sup-monoid.add-0-right circ-simulate-right-plus mult-left-zero*)

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

lemma *circ-simulate-2*:
$y * x^\circ \leq x^\circ * y^\circ \longleftrightarrow y^\circ * x^\circ \leq x^\circ * y^\circ$
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*:
$y * x \leq x \Longrightarrow y^\circ * x \leq x \sqcup y^\circ * bot$
by (*metis circ-simulate-left-plus-1 circ-zero mult-1-right*)

lemma *circ-simulate-3*:
$y * x^\circ \leq x^\circ \Longrightarrow y^\circ * x^\circ \leq x^\circ * y^\circ$
by (*metis sup.bounded-iff circ-reflexive circ-simulate-2 le-iff-sup mult-right-isotone mult-1-right*)

lemma *circ-separate-mult-1*:
$y * x \leq x * y \Longrightarrow (x * y)^\circ \leq x^\circ * y^\circ$
by (*metis circ-mult-sub-sup circ-separate-1*)

lemma *circ-separate-unfold*:
$(y * x^\circ)^\circ = y^\circ \sqcup y^\circ * y * x * x^\circ * (y * x^\circ)^\circ$
by (*metis circ-back-loop-fixpoint circ-plus-same circ-unfold-sum sup-commute mult-assoc*)

lemma *separation*:
assumes $y * x \leq x * y^\circ$
shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
proof –
have $y^\circ * x * y^\circ \leq x * y^\circ \sqcup y^\circ * bot$
by (*metis assms circ-simulate-left-plus-1 circ-transitive-equal mult-assoc mult-left-isotone*)
thus *?thesis*
by (*metis sup-commute circ-sup-1 circ-simulate-right circ-sub-dist-3 le-iff-sup mult-assoc mult-left-zero zero-right-mult-decreasing*)
qed

lemma *simulation*:
$y * x \leq x * y^\circ \Longrightarrow y^\circ * x^\circ \leq x^\circ * y^\circ$
by (*metis sup-ge2 circ-isotone circ-mult-upper-bound circ-sub-dist separation*)

lemma *circ-simulate-4*:
assumes $y * x \leq x * x^\circ * (1 \sqcup y)$

shows $y^\circ * x^\circ \le x^\circ * y^\circ$
proof –
have $x \sqcup (x * x^\circ * x * x \sqcup x * x) = x * x^\circ$
by (*metis* (*no-types*) *circ-back-loop-fixpoint mult-right-dist-sup sup-commute*)
hence $x \le x * x^\circ * 1 \sqcup x * x^\circ * y$
by (*metis mult-1-right sup-assoc sup-ge1*)
hence $(1 \sqcup y) * x \le x * x^\circ * (1 \sqcup y)$
using *assms mult-left-dist-sup mult-right-dist-sup* **by** *force*
hence $y * x^\circ \le x^\circ * y^\circ$
by (*metis circ-sup-upper-bound 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*:
$y * x \le x * x^\circ * (x \sqcup y) \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
by (*metis circ-sup-sub-sup-one circ-simulate-4 order-trans*)

lemma *circ-simulate-6*:
$y * x \le x * (x \sqcup y) \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
by (*metis sup-commute circ-back-loop-fixpoint circ-simulate-5 mult-right-sub-dist-sup-left order-trans*)

lemma *circ-separate-4*:
assumes $y * x \le x * x^\circ * (1 \sqcup y)$
shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
proof –
have $y * x * x^\circ \le x * x^\circ * (1 \sqcup y) * x^\circ$
by (*simp add*: *assms mult-left-isotone*)
also have ... $= x * x^\circ \sqcup x * x^\circ * y * x^\circ$
by (*simp add*: *circ-transitive-equal mult-left-dist-sup mult-right-dist-sup mult-assoc*)
also have ... $\le x * x^\circ \sqcup x * x^\circ * x^\circ * y^\circ$
by (*metis assms sup-right-isotone circ-simulate-2 circ-simulate-4 mult-assoc mult-right-isotone*)
finally have $y * x * x^\circ \le x * x^\circ * y^\circ$
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*:
$y * x \le x * x^\circ * (x \sqcup y) \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$
by (*metis circ-sup-sub-sup-one circ-separate-4 order-trans*)

lemma *circ-separate-6*:

$y * x \le x * (x \sqcup y) \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$
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* :: *'a*

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

lemma *L-def*:
$L = 1^\circ * bot$
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*:
$x * 1^\circ \le 1^\circ * x$
by (*metis L-split-sup sup-commute mult-1-right one-circ-mult-split*)

lemma *one-circ-mult-split-2*:
$1^\circ * x = x * 1^\circ \sqcup L$
proof –
have *1*: $x * 1^\circ \le L \sqcup x$
using *one-circ-mult-split sub-mult-one-circ* **by** *presburger*
have $x \sqcup x * 1^\circ = x * 1^\circ$
by (*meson circ-back-loop-prefixpoint le-iff-sup sup.boundedE*)
thus *?thesis*

using *1* **by** (*simp add*: *le-iff-sup one-circ-mult-split sup-assoc sup-commute*)
qed

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

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

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

lemma *L-left-zero*:
$L * x = L$
by (*metis L-def mult-assoc mult-left-zero*)

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

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

lemma *mult-L-circ-mult*:
$(x * L)^\circ * y = y \sqcup x * L$
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*:
$x^\circ * 1^\circ = x^{\circ\circ}$
by (*metis L-left-zero sup-commute circ-sup-1 circ-circ-sup mult-zero-circ one-circ-split*)

lemma *circ-circ-mult*:
$1^\circ * x^\circ = x^{\circ\circ}$
by (*metis antisym circ-circ-mult-1 circ-circ-sub-mult sub-mult-one-circ*)

lemma *circ-circ-split*:
$x^{\circ\circ} = L \sqcup x^{\circ}$
by (*metis circ-circ-mult one-circ-mult-split*)

lemma *circ-sup-6*:
$L \sqcup (x \sqcup y)^{\circ} = (x^{\circ} * y^{\circ})^{\circ}$
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*: $L = 1^{\circ} * bot$
begin

lemma *one-circ-split*:
$1^{\circ} = L \sqcup 1$
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*:
$1^{\circ} * x = L \sqcup x$
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*:
$x * 1^{\circ} \leq x \sqcup L$
by (*metis sup-commute one-circ-mult-split sub-mult-one-circ*)

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

lemma *L-split*:
$x * L \leq x * bot \sqcup L$
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*:
$L \leq x \Longrightarrow x * y \leq x \Longrightarrow x * y^{\circ} \leq x$
by (*metis circ-one circ-simulate le-iff-sup one-circ-mult-split*)

lemma *circ-left-induct-mult-iff-L*:
$L \leq x \Longrightarrow x * y \leq x \longleftrightarrow x * y^{\circ} \leq x$
by (*metis sup.bounded-iff circ-back-loop-fixpoint circ-left-induct-mult-L le-iff-sup*)

lemma *circ-left-induct-L*:
$L \leq x \Longrightarrow x * y \sqcup z \leq x \Longrightarrow z * y^{\circ} \leq x$
by (*metis sup.bounded-iff circ-left-induct-mult-L le-iff-sup mult-right-dist-sup*)

end

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* :: $'a \Rightarrow 'a$ ($\text{-}^{\star}$ [*100*] *100*)

class *left-kleene-algebra* = *idempotent-left-semiring* + *star* +
assumes *star-left-unfold* : $1 \sqcup y * y^{\star} \leq y^{\star}$
assumes *star-left-induct* : $z \sqcup y * x \leq x \longrightarrow y^{\star} * z \leq x$
begin

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

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

lemma *star-left-unfold-equal*:
$1 \sqcup x * x^{\star} = x^{\star}$
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*:
 $(x * y)^\star * x \leq x * (y * x)^\star$
 by (*metis mult-assoc mult-left-sub-dist-sup mult-1-right star-left-induct star-left-unfold-equal*)

lemma *star-isotone*:
 $x \leq y \Longrightarrow x^\star \leq y^\star$
 by (*metis sup-right-isotone mult-left-isotone order-trans star-left-unfold mult-1-right star-left-induct*)

lemma *star-sup-1*:
 $(x \sqcup y)^\star = x^\star * (y * x^\star)^\star$
proof (*rule antisym*)
 have $y * x^\star * (y * x^\star)^\star \leq (y * x^\star)^\star$
  using *sup-right-divisibility star-left-unfold-equal* **by** *auto*
 also have ... $\leq x^\star * (y * x^\star)^\star$
  using *mult-left-isotone sup-left-divisibility star-left-unfold-equal* **by** *fastforce*
 finally have $(x \sqcup y) * (x^\star * (y * x^\star)^\star) \leq x^\star * (y * x^\star)^\star$
  by (*metis le-supI mult-right-dist-sup mult-right-sub-dist-sup-right mult-assoc star-left-unfold-equal*)
 hence $1 \sqcup (x \sqcup y) * (x^\star * (y * x^\star)^\star) \leq x^\star * (y * x^\star)^\star$
  using *reflexive-mult-closed star-left-unfold* **by** *auto*
 thus $(x \sqcup y)^\star \leq x^\star * (y * x^\star)^\star$
  using *star-left-induct* **by** *force*
next
 have $x^\star * (y * x^\star)^\star \leq x^\star * (y * (x \sqcup y)^\star)^\star$
  by (*simp add*: *mult-right-isotone star-isotone*)
 also have ... $\leq x^\star * ((x \sqcup y) * (x \sqcup y)^\star)^\star$
  by (*simp add*: *mult-right-isotone mult-right-sub-dist-sup-right star-isotone*)
 also have ... $\leq x^\star * (x \sqcup y)^{\star\star}$
  using *mult-right-isotone star-left-unfold star-isotone* **by** *auto*
 also have ... $\leq (x \sqcup y)^\star * (x \sqcup y)^{\star\star}$
  by (*simp add*: *mult-left-isotone star-isotone*)
 also have ... $\leq (x \sqcup y)^\star$
  by (*metis sup.bounded-iff mult-1-right star-left-induct star-left-unfold*)
 finally show $x^\star * (y * x^\star)^\star \leq (x \sqcup y)^\star$
  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-locales*
 apply (*rule star-left-unfold-equal*)
 apply (*rule star-left-slide*)

by (*rule star-sup-1*)

context *left-kleene-algebra*
begin

A number of lemmas in this class are taken from Georg Struth's Kleene algebra theory [2].

lemma *star-sub-one*:
$x \leq 1 \Longrightarrow x^\star = 1$
by (*metis sup-right-isotone eq-iff le-iff-sup mult-1-right star.circ-plus-one star-left-induct*)

lemma *star-one*:
$1^\star = 1$
by (*simp add*: *star-sub-one*)

lemma *star-left-induct-mult*:
$x * y \leq y \Longrightarrow x^\star * y \leq y$
by (*simp add*: *star-left-induct*)

lemma *star-left-induct-mult-iff*:
$x * y \leq y \longleftrightarrow x^\star * y \leq y$
using *mult-left-isotone order-trans star.circ-increasing star-left-induct-mult* **by** *blast*

lemma *star-involutive*:
$x^\star = x^{\star\star}$
using *star.circ-circ-sup star-sup-1 star-one* **by** *auto*

lemma *star-sup-one*:
$(1 \sqcup x)^\star = x^\star$
using *star.circ-circ-sup star-involutive* **by** *auto*

lemma *star-left-induct-equal*:
$z \sqcup x * y = y \Longrightarrow x^\star * z \leq y$
by (*simp add*: *star-left-induct*)

lemma *star-left-induct-mult-equal*:
$x * y = y \Longrightarrow x^\star * y \leq y$
by (*simp add*: *star-left-induct-mult*)

lemma *star-star-upper-bound*:
$x^\star \leq z^\star \Longrightarrow x^{\star\star} \leq z^\star$
using *star-involutive* **by** *auto*

lemma *star-simulation-left*:
assumes $x * z \leq z * y$
shows $x^\star * z \leq z * y^\star$
proof –

have $x * z * y^\star \le z * y * y^\star$
  by (*simp add*: *assms mult-left-isotone*)
also have ... $\le z * y^\star$
  by (*simp add*: *mult-right-isotone star.left-plus-below-circ mult-assoc*)
finally have $z \sqcup x * z * y^\star \le z * y^\star$
  using *star.circ-back-loop-prefixpoint* **by** *auto*
thus *?thesis*
  by (*simp add*: *star-left-induct mult-assoc*)
qed

lemma *quasicomm-1*:
$y * x \le x * (x \sqcup y)^\star \longleftrightarrow y^\star * x \le x * (x \sqcup y)^\star$
by (*metis mult-isotone order-refl order-trans star.circ-increasing star-involutive star-simulation-left*)

lemma *star-rtc-3*:
$1 \sqcup x \sqcup y * y = y \Longrightarrow x^\star \le y$
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*)

lemma *star-decompose-1*:
$(x \sqcup y)^\star = (x^\star * y^\star)^\star$
apply (*rule antisym*)
apply (*simp add*: *star.circ-sup-2*)
using *star.circ-sub-dist-3 star-isotone star-involutive* **by** *fastforce*

lemma *star-sum*:
$(x \sqcup y)^\star = (x^\star \sqcup y^\star)^\star$
using *star-decompose-1 star-involutive* **by** *auto*

lemma *star-decompose-3*:
$(x^\star * y^\star)^\star = x^\star * (y * x^\star)^\star$
using *star-sup-1 star-decompose-1* **by** *auto*

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

lemma *star-loop-least-fixpoint*:
$y * x \sqcup z = x \Longrightarrow y^\star * z \le x$
by (*simp add*: *sup-commute star-left-induct-equal*)

lemma *star-loop-is-least-fixpoint*:
is-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (y^\star * z)$
by (*simp add*: *is-least-fixpoint-def star.circ-loop-fixpoint star-loop-least-fixpoint*)

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

lemma *affine-has-least-fixpoint*:

has-least-fixpoint (λx . $y * x \sqcup z$)
by (*metis has-least-fixpoint-def star-loop-is-least-fixpoint*)

lemma *star-outer-increasing*:
$x \leq y^\star * x * y^\star$
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*: $z \sqcup x * y \leq x \longrightarrow z * y^\star \leq x$
begin

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

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

lemma *star-simulation-right*:
assumes $z * x \leq y * z$
shows $z * x^\star \leq y^\star * z$
proof –
have $y^\star * z * x \leq y^\star * z$

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

end

Again we inherit results from the itering hierarchy.

sublocale *strong-left-kleene-algebra* < *star*: *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*:
$y * x \leq y \Longrightarrow y * x^\star \leq y$
by (*simp add*: *star-right-induct*)

lemma *star-right-induct-mult-iff*:
$y * x \leq y \longleftrightarrow y * x^\star \leq y$
using *mult-right-isotone order-trans star.circ-increasing star-right-induct-mult*
by *blast*

lemma *star-simulation-right-equal*:
$z * x = y * z \Longrightarrow z * x^\star = y^\star * z$
by (*metis eq-iff star-simulation-left star-simulation-right*)

lemma *star-simulation-star*:
$x * y \leq y * x \Longrightarrow x^\star * y^\star \leq y^\star * x^\star$
by (*simp add*: *star-simulation-left star-simulation-right*)

lemma *star-right-induct-equal*:
$z \sqcup y * x = y \Longrightarrow z * x^\star \leq y$
by (*simp add*: *star-right-induct*)

lemma *star-right-induct-mult-equal*:
$y * x = y \Longrightarrow y * x^\star \leq y$
by (*simp add*: *star-right-induct-mult*)

lemma *star-back-loop-least-fixpoint*:
$x * y \sqcup z = x \Longrightarrow z * y^\star \leq x$
by (*simp add*: *sup-commute star-right-induct-equal*)

lemma *star-back-loop-is-least-fixpoint*:
is-least-fixpoint $(\lambda x\ .\ x * y \sqcup z)\ (z * y^\star)$
proof (*unfold is-least-fixpoint-def* , *rule conjI*)

have $(z * y^\star * y \sqcup z) * y \leq z * y^\star * y \sqcup z$
using *le-supI1 mult-left-isotone star.circ-back-loop-prefixpoint* **by** *auto*
hence $z * y^\star \leq z * y^\star * y \sqcup z$
by (*simp add*: *star-right-induct*)
thus $z * y^\star * y \sqcup z = z * y^\star$
using *antisym star.circ-back-loop-prefixpoint* **by** *auto*
next
show $\forall x.\ x * y \sqcup z = x \longrightarrow z * y^\star \leq x$
by (*simp add*: *star-back-loop-least-fixpoint*)
qed

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

lemma *star-square*:
$x^\star = (1 \sqcup x) * (x * x)^\star$
proof −
let $?f = \lambda y\ .\ y * x \sqcup 1$
have *1*: *isotone ?f*
by (*metis sup-left-isotone isotone-def mult-left-isotone*)
have $?f \circ ?f = (\lambda y\ .\ y * (x * x) \sqcup (1 \sqcup x))$
by (*simp add*: *sup-assoc sup-commute mult-assoc mult-right-dist-sup o-def*)
thus *?thesis*
using *1* **by** (*metis mu-square mult-left-one star-back-loop-mu*
has-least-fixpoint-def star-back-loop-is-least-fixpoint)
qed

lemma *star-square-2*:
$x^\star = (x * x)^\star * (x \sqcup 1)$
proof −
have $(1 \sqcup x) * (x * x)^\star = (x * x)^\star * 1 \sqcup x * (x * x)^\star$
using *mult-right-dist-sup* **by** *force*
thus *?thesis*
by (*metis* (*no-types*) *antisym mult-left-sub-dist-sup star.circ-square-2*
star-slide sup-commute star-square)
qed

lemma *star-circ-simulate-right-plus*:
assumes $z * x \leq y * y^\star * z \sqcup w$
shows $z * x^\star \leq y^\star * (z \sqcup w * x^\star)$
proof −
have $(z \sqcup w * x^\star) * x \leq z * x \sqcup w * x^\star$
using *mult-right-dist-sup star.circ-back-loop-prefixpoint sup-right-isotone* **by**
auto
also have ... $\leq y * y^\star * z \sqcup w \sqcup w * x^\star$
using *assms sup-left-isotone* **by** *blast*
also have ... $\leq y * y^\star * z \sqcup w * x^\star$
using *le-supI1 star.circ-back-loop-prefixpoint sup-commute* **by** *auto*

also have ... $\leq y^\star * (z \sqcup w * x^\star)$
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 $y^\star * (z \sqcup w * x^\star) * x \leq y^\star * (z \sqcup w * x^\star)$
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*)
qed

lemma *transitive-star*:
$x * x \leq x \Longrightarrow x^\star = 1 \sqcup x$
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*:
$y^\star * (y^\star * x)^\star * y^\star = (y^\star * x)^\star * y^\star$
by (*metis star.circ-transitive-equal star-slide mult-assoc*)

lemma *star-circ-simulate-left-plus*:
assumes $x * z \leq z * y^\star \sqcup w$
shows $x^\star * z \leq (z \sqcup x^\star * w) * y^\star$
proof −
have $x * (x^\star * (w * y^\star)) \leq x^\star * (w * y^\star)$
by (*metis* (*no-types*) *mult-right-sub-dist-sup-left star.circ-loop-fixpoint mult-assoc*)
hence $x * ((z \sqcup x^\star * w) * y^\star) \leq x * z * y^\star \sqcup x^\star * w * y^\star$
using *mult-left-dist-sup mult-right-dist-sup sup-right-isotone mult-assoc* **by** *presburger*
also have ... $\leq (z * y^\star \sqcup w) * y^\star \sqcup x^\star * w * y^\star$
using *assms mult-isotone semiring.add-right-mono* **by** *blast*
also have ... $= z * y^\star \sqcup w * y^\star \sqcup x^\star * w * y^\star$
by (*simp add*: *mult-right-dist-sup star.circ-transitive-equal mult-assoc*)
also have ... $= (z \sqcup w \sqcup x^\star * w) * y^\star$
by (*simp add*: *mult-right-dist-sup*)
also have ... $= (z \sqcup x^\star * w) * y^\star$
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*

star.circ-right-unfold-1 star-left-induct)
qed

lemma *star-one-sup-below*:
$x * y^\star * (1 \sqcup z) \leq x * (y \sqcup z)^\star$
proof –
have $y^\star * z \leq (y \sqcup z)^\star$
using *sup-ge2 order-trans star.circ-increasing star.circ-mult-upper-bound star.circ-sub-dist* **by** *blast*
hence $y^\star \sqcup y^\star * z \leq (y \sqcup z)^\star$
by (*simp add*: *star.circ-sup-upper-bound star.circ-sub-dist*)
hence $y^\star * (1 \sqcup z) \leq (y \sqcup z)^\star$
by (*simp add*: *mult-left-dist-sup*)
thus *?thesis*
by (*metis mult-right-isotone mult-assoc*)
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.

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

lemma *star-separate*:

assumes $x * y = bot$
and $y * y = bot$
shows $(x \sqcup y)^\star = x^\star \sqcup y * x^\star$
proof –
have *1*: $y^\star = 1 \sqcup y$
using *assms*(*2*) **by** (*simp add*: *transitive-star*)
have $(x \sqcup y)^\star = y^\star * (x * y^\star)^\star$
by (*simp add*: *star.circ-decompose-6 star-sup-1*)
also have ... $= y^\star * (x * (1 \sqcup y * y^\star))^\star$
by (*simp add*: *star-left-unfold-equal*)
also have ... $= (1 \sqcup y) * x^\star$
using *1* **by** (*simp add*: *assms mult-left-dist-sup*)
also have ... $= x^\star \sqcup y * x^\star$
by (*simp add*: *mult-right-dist-sup*)
finally show *?thesis*
.
qed

end

We can now inherit from the strongest variant of iterings.

sublocale *left-zero-kleene-algebra* < *star*: *itering* **where** *circ* = *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*)

context *left-zero-kleene-algebra*
begin

lemma *star-absorb*:
$x * y = bot \Longrightarrow x * y^\star = x$
by (*metis sup.bounded-iff antisym-conv star.circ-back-loop-prefixpoint star.circ-elimination*)

lemma *star-separate-2*:
assumes $x * z^+ * y = bot$
and $y * z^+ * y = bot$
and $z * x = bot$
shows $(x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star = z^\star * (x^\star \sqcup y * x^\star) * z^\star$
proof –
have *1*: $x^\star * z^+ * y = z^+ * y$
by (*metis assms mult-assoc mult-1-left mult-left-zero star.circ-zero star-simulation-right-equal*)
have *2*: $z^\star * (x^\star \sqcup y * x^\star) * z^+ \le z^\star * (x^\star \sqcup y * x^\star) * z^\star$
by (*simp add*: *mult-right-isotone star.left-plus-below-circ*)
have $z^\star * z^+ * y * x^\star \le z^\star * y * x^\star$
by (*metis mult-left-isotone star.left-plus-below-circ star.right-plus-circ*

star-plus)
  also have ... $\leq z^\star * (x^\star \sqcup y * x^\star)$
    by (*simp add*: *mult-assoc mult-left-sub-dist-sup-right*)
  also have ... $\leq z^\star * (x^\star \sqcup y * x^\star) * z^\star$
    using *sup-right-divisibility star.circ-back-loop-fixpoint* **by** *blast*
  finally have *3*: $z^\star * z^+ * y * x^\star \leq z^\star * (x^\star \sqcup y * x^\star) * z^\star$
    .
  have $z^\star * (x^\star \sqcup y * x^\star) * z^\star * (z * (1 \sqcup y * x^\star)) = z^\star * (x^\star \sqcup y * x^\star) * z^+ \sqcup z^\star * (x^\star \sqcup y * x^\star) * z^+ * y * x^\star$
    by (*metis mult-1-right semiring.distrib-left star.circ-plus-same mult-assoc*)
  also have ... $= z^\star * (x^\star \sqcup y * x^\star) * z^+ \sqcup z^\star * (1 \sqcup y) * x^\star * z^+ * y * x^\star$
    by (*simp add*: *semiring.distrib-right mult-assoc*)
  also have ... $= z^\star * (x^\star \sqcup y * x^\star) * z^+ \sqcup z^\star * (1 \sqcup y) * z^+ * y * x^\star$
    using *1* **by** (*simp add*: *mult-assoc*)
  also have ... $= z^\star * (x^\star \sqcup y * x^\star) * z^+ \sqcup z^\star * z^+ * y * x^\star \sqcup z^\star * y * z^+ * y * x^\star$
    using *mult-left-dist-sup mult-right-dist-sup sup-assoc* **by** *auto*
  also have ... $= z^\star * (x^\star \sqcup y * x^\star) * z^+ \sqcup z^\star * z^+ * y * x^\star$
    by (*metis assms*(*2*) *mult-left-dist-sup mult-left-zero sup-commute sup-monoid.add-0-left mult-assoc*)
  also have ... $\leq z^\star * (x^\star \sqcup y * x^\star) * z^\star$
    using *2 3* **by** *simp*
  finally have $(x^\star \sqcup y * x^\star) \sqcup z^\star * (x^\star \sqcup y * x^\star) * z^\star * (z * (1 \sqcup y * x^\star)) \leq z^\star * (x^\star \sqcup y * x^\star) * z^\star$
    by (*simp add*: *star-outer-increasing*)
  hence *4*: $(x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star \leq z^\star * (x^\star \sqcup y * x^\star) * z^\star$
    by (*simp add*: *star-right-induct*)
  have *5*: $(x^\star \sqcup y * x^\star) * z^\star \leq (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star$
    by (*metis sup-ge1 mult-right-isotone mult-1-right star-isotone*)
  have $z * (x^\star \sqcup y * x^\star) = z * x^\star \sqcup z * y * x^\star$
    by (*simp add*: *mult-assoc mult-left-dist-sup*)
  also have ... $= z \sqcup z * y * x^\star$
    by (*simp add*: *assms star-absorb*)
  also have ... $= z * (1 \sqcup y * x^\star)$
    by (*simp add*: *mult-assoc mult-left-dist-sup*)
  also have ... $\leq (z * (1 \sqcup y * x^\star))^\star$
    by (*simp add*: *star.circ-increasing*)
  also have ... $\leq (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star$
    by (*metis le-supE mult-right-sub-dist-sup-left star.circ-loop-fixpoint*)
  finally have $z * (x^\star \sqcup y * x^\star) \leq (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star$
    .
  hence $z * (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star \leq (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star$
    by (*metis mult-assoc mult-left-isotone star.circ-transitive-equal*)
  hence $z^\star * (x^\star \sqcup y * x^\star) * z^\star \leq (x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star$
    using *5* **by** (*metis star-left-induct sup.bounded-iff mult-assoc*)
  thus *?thesis*
    using *4* **by** (*simp add*: *antisym*)
qed

end

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

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

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

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

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

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

lemma *circ-sup-3*:
$(x^\circ * y^\circ)^\star \leq (x \sqcup y)^\circ$
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*:
$x^\circ = x^\circ * bot \sqcup x^\star$
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*:
$x^\circ * y = x^\circ * bot \sqcup x^\star * y$
by (*metis circ-isolate mult-assoc mult-left-zero mult-right-dist-sup*)

lemma *circ-isolate-mult-sub*:
$x^\circ * y \leq x^\circ \sqcup x^\star * y$
by (*metis sup-left-isotone circ-isolate-mult zero-right-mult-decreasing*)

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

lemma *circ-sup-4*:
$(x \sqcup y)^\circ = (x^\star * y)^\circ * x^\circ$
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*:
$(x^\circ * y)^\circ * x^\circ = (x^\star * y)^\circ * x^\circ$
using *circ-sup-4 circ-sup-9* **by** *auto*

lemma *plus-circ*:
$(x^\star * x)^\circ = x^\circ$
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.

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* : $1 \sqcup y * y^\star \leq y^\star$
  assumes *star-left-induct-var* : $y * x \leq x \longrightarrow y^\star * x \leq x$
  assumes *star-right-induct-var* : $x * y \leq x \longrightarrow x * y^\star \leq x$
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*

begin

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

sublocale *bounded-distrib-lattice* < *comp-inf*: *bounded-kleene-algebra* **where** *star* = λx . *top* **and** *one* = *top* **and** *times* = *inf*
  apply *unfold-locales*
  apply (*simp add*: *inf.assoc*)
  apply *simp*
  apply *simp*
  apply (*simp add*: *le-infI2*)
  apply (*simp add*: *inf-sup-distrib2*)
  apply *simp*
  apply *simp*
  apply *simp*
  apply *simp*
  apply *simp*
  apply (*simp add*: *inf-sup-distrib1*)
  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.

class *stone-kleene-relation-algebra* = *stone-relation-algebra* + *kleene-algebra* +
  assumes *pp-dist-star*: $--(x^\star) = (--x)^\star$
begin

subclass *bounded-kleene-algebra* ..

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

lemma *conv-star-conv*:
  $x^\star \le x^{T\star T}$
proof −
  have $x^{T\star} * x^T \le x^{T\star}$
    by (*simp add*: *star.right-plus-below-circ*)
  hence *1*: $x * x^{T\star T} \le x^{T\star T}$
    using *conv-dist-comp conv-isotone* **by** *fastforce*
  have $1 \le x^{T\star T}$
    by (*simp add*: *reflexive-conv-closed star.circ-reflexive*)
  hence $1 \sqcup x * x^{T\star T} \le x^{T\star T}$
    using *1* **by** *simp*
  thus *?thesis*
    using *star-left-induct* **by** *fastforce*

qed

It follows that star and converse commute.

lemma *conv-star-commute*:
$x^{\star T} = x^{T\star}$
proof (*rule antisym*)
show $x^{\star T} \leq x^{T\star}$
using *conv-star-conv conv-isotone* **by** *fastforce*
next
show $x^{T\star} \leq x^{\star T}$
by (*metis conv-star-conv conv-involutive*)
qed

abbreviation *acyclic* :: *'a* ⇒ *bool* **where** $acyclic\ x \equiv x^{+} \leq -1$

abbreviation *forest* :: *'a* ⇒ *bool* **where** $forest\ x \equiv injective\ x \wedge acyclic\ x$

lemma *forest-bot*:
forest bot
by *simp*

lemma *acyclic-star-below-complement*:
$acyclic\ w \longleftrightarrow w^{T\star} \leq -w$
by (*simp add*: *conv-star-commute schroeder-4-p*)

lemma *acyclic-asymmetric*:
$acyclic\ w \Longrightarrow w^{T} \sqcap w = bot$
using *acyclic-star-below-complement inf.order-lesseq-imp pseudo-complement star.circ-increasing* **by** *blast*

lemma *vector-star-1*:
assumes *vector x*
shows $x^{T} * (x * x^{T})^{\star} \leq x^{T}$
proof −
have $x^{T} * (x * x^{T})^{\star} = (x^{T} * x)^{\star} * x^{T}$
by (*simp add*: *star-slide*)
also have ... $\leq top * x^{T}$
by (*simp add*: *mult-left-isotone*)
also have ... $= x^{T}$
using *assms vector-conv-covector* **by** *auto*
finally show *?thesis*
.
qed

lemma *vector-star-2*:
$vector\ x \Longrightarrow x^{T} * (x * x^{T})^{\star} \leq x^{T} * bot^{\star}$
by (*simp add*: *star-absorb vector-star-1*)

lemma *vector-vector-star*:

vector v $\Longrightarrow (v * v^T)^\star = 1 \sqcup v * v^T$
by (*simp add*: *transitive-star vv-transitive*)

lemma *forest-separate*:
assumes *forest x*
shows $x^\star * x^{T\star} \sqcap x^T * x \leq 1$
proof −
have $x^\star * 1 \leq -x^T$
using *assms schroeder-5-p* **by** *force*
hence *1*: $x^\star \sqcap x^T = bot$
by (*simp add*: *pseudo-complement*)
have $x^\star \sqcap x^T * x = (1 \sqcup x^\star * x) \sqcap x^T * x$
using *star.circ-right-unfold-1* **by** *simp*
also have ... $= (1 \sqcap x^T * x) \sqcup (x^\star * x \sqcap x^T * x)$
by (*simp add*: *inf-sup-distrib2*)
also have ... $\leq 1 \sqcup (x^\star * x \sqcap x^T * x)$
using *sup-left-isotone* **by** *simp*
also have ... $= 1 \sqcup (x^\star \sqcap x^T) * x$
by (*simp add*: *assms injective-comp-right-dist-inf*)
also have ... $= 1$
using *1* **by** *simp*
finally have *2*: $x^\star \sqcap x^T * x \leq 1$
.
hence *3*: $x^{T\star} \sqcap x^T * x \leq 1$
by (*metis* (*mono-tags*, *lifting*) *conv-star-commute conv-dist-comp conv-dist-inf conv-involutive coreflexive-symmetric*)
have $x^\star * x^{T\star} \sqcap x^T * x \leq (x^\star \sqcup x^{T\star}) \sqcap x^T * x$
using *assms cancel-separate inf.sup-left-isotone* **by** *simp*
also have ... ≤ 1
using *2 3* **by** (*simp add*: *inf-sup-distrib2*)
finally show *?thesis*
.
qed

lemma *cut-reachable*:
assumes $v^T = r^T * t^\star$
and $t \leq g$
shows $v * -v^T \sqcap g \leq (r^T * g^\star)^T * (r^T * g^\star)$
proof −
have $v * -v^T \sqcap g \leq v * top \sqcap g$
using *inf.sup-left-isotone mult-right-isotone top-greatest* **by** *blast*
also have ... $= (r^T * t^\star)^T * top \sqcap g$
by (*metis assms*(*1*) *conv-involutive*)
also have ... $\leq (r^T * g^\star)^T * top \sqcap g$
using *assms*(*2*) *conv-isotone inf.sup-left-isotone mult-left-isotone mult-right-isotone star-isotone* **by** *auto*
also have ... $\leq (r^T * g^\star)^T * ((r^T * g^\star) * g)$
by (*metis conv-involutive dedekind-1 inf-top.left-neutral*)
also have ... $\leq (r^T * g^\star)^T * (r^T * g^\star)$

by (*simp add*: *mult-assoc mult-right-isotone star.left-plus-below-circ star-plus*)
finally show *?thesis*
.
qed

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 $r^T * g^\star = r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star$
proof −
have *1*: $r^T \leq r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star$
using *mult-right-isotone mult-1-right star.circ-reflexive* **by** *fastforce*
have *2*: *covector* $(r^T * g^\star)$
using *assms covector-mult-closed vector-conv-covector* **by** *auto*
have $r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * g \leq r^T * g^\star * g$
by (*simp add*: *mult-left-isotone mult-right-isotone star-isotone*)
also have ... $\leq r^T * g^\star$
by (*simp add*: *mult-assoc mult-right-isotone star.left-plus-below-circ star-plus*)
finally have $r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * g = r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * g \sqcap r^T * g^\star$
by (*simp add*: *le-iff-inf*)
also have ... $= r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * (g \sqcap r^T * g^\star)$
using *assms covector-comp-inf covector-mult-closed vector-conv-covector* **by** *auto*
also have ... $= (r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star \sqcap r^T * g^\star) * (g \sqcap r^T * g^\star)$
by (*simp add*: *inf.absorb2 inf-commute mult-right-isotone star-isotone*)
also have ... $= r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * (g \sqcap r^T * g^\star \sqcap (r^T * g^\star)^T)$
using *2* **by** (*metis comp-inf-vector-1*)
also have ... $= r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * ((r^T * g^\star)^T \sqcap r^T * g^\star \sqcap g)$
using *inf-commute inf-assoc* **by** *simp*
also have ... $= r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)$
using *2* **by** (*metis covector-conv-vector inf-top.right-neutral vector-inf-comp*)
also have ... $\leq r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star$
by (*simp add*: *mult-assoc mult-right-isotone star.left-plus-below-circ star-plus*)
finally have $r^T * g^\star \leq r^T * ((r^T * g^\star)^T * (r^T * g^\star) \sqcap g)^\star$
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 ≤ w
    and w ≤ (r^T * g^⋆)^T * (r^T * g^⋆) ⊓ g
    and r^T * g^⋆ ≤ r^T * w^⋆
  shows w * v ≤ v
proof –
  have w * r ≤ (r^T * g^⋆)^T * (r^T * g^⋆) * r
    using assms(6) mult-left-isotone by auto
  also have ... ≤ (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 ... ≤ (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 ... ≤ –w * r
    using assms(4) by (simp add: mult-left-isotone
acyclic-star-below-complement)
  also have ... ≤ –(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 ⊔ r
    by (simp add: calculation star.circ-loop-fixpoint)
  also have ... ≤ w^T * v ⊔ r
    using assms(5) comp-isotone conv-isotone semiring.add-right-mono by auto
  finally have w * v ≤ w * w^T * v ⊔ 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 ... ≤ 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 *reachable-inv*:
assumes *vector* v
and $e \leq v * -v^T$
and $e * t = bot$
and $v^T = r^T * t^\star$
shows $(v \sqcup e^T * top)^T = r^T * (t \sqcup e)^\star$
proof −
have *1*: $v^T \leq r^T * (t \sqcup e)^\star$
by (*simp add*: *assms*(*4*) *mult-right-isotone star.circ-sub-dist*)
have *2*: $(e^T * top)^T = top * e$
by (*simp add*: *conv-dist-comp*)
also have ... $= top * (v * -v^T \sqcap e)$
by (*simp add*: *assms*(*2*) *inf-absorb2*)
also have ... $\leq top * (v * top \sqcap e)$
using *inf.sup-left-isotone mult-right-isotone top-greatest* **by** *blast*
also have ... $= top * v^T * e$
by (*simp add*: *comp-inf-vector inf.sup-monoid.add-commute*)
also have ... $= v^T * e$
using *assms*(*1*) *vector-conv-covector* **by** *auto*
also have ... $\leq r^T * (t \sqcup e)^\star * e$
using *1* **by** (*simp add*: *mult-left-isotone*)
also have ... $\leq r^T * (t \sqcup e)^\star * (t \sqcup e)$
by (*simp add*: *mult-right-isotone*)
also have ... $\leq r^T * (t \sqcup e)^\star$
by (*simp add*: *comp-associative mult-right-isotone star.right-plus-below-circ*)
finally have *3*: $(v \sqcup e^T * top)^T \leq r^T * (t \sqcup e)^\star$
using *1* **by** (*simp add*: *conv-dist-sup*)
have $r^T \leq r^T * t^\star$
using *sup.bounded-iff star.circ-back-loop-prefixpoint* **by** *blast*
also have ... $\leq (v \sqcup e^T * top)^T$
by (*metis assms*(*4*) *conv-isotone sup-ge1*)
finally have *4*: $r^T \leq (v \sqcup e^T * top)^T$
.
have $(v \sqcup e^T * top)^T * (t \sqcup e) = (v \sqcup e^T * top)^T * t \sqcup (v \sqcup e^T * top)^T * e$
by (*simp add*: *mult-left-dist-sup*)
also have ... $\leq (v \sqcup e^T * top)^T * t \sqcup top * e$
using *comp-isotone semiring.add-left-mono* **by** *auto*
also have ... $= v^T * t \sqcup top * e * t \sqcup top * e$
using *2* **by** (*simp add*: *conv-dist-sup mult-right-dist-sup*)
also have ... $= v^T * t \sqcup top * e$
by (*simp add*: *assms*(*3*) *comp-associative*)
also have ... $\leq r^T * t^\star \sqcup top * e$
by (*metis assms*(*4*) *star.circ-back-loop-fixpoint sup-ge1 sup-left-isotone*)
also have ... $= v^T \sqcup top * e$
by (*simp add*: *assms*(*4*))
finally have *5*: $(v \sqcup e^T * top)^T * (t \sqcup e) \leq (v \sqcup e^T * top)^T$
using *2* **by** (*simp add*: *conv-dist-sup*)
have $r^T * (t \sqcup e)^\star \leq (v \sqcup e^T * top)^T * (t \sqcup e)^\star$
using *4* **by** (*simp add*: *mult-left-isotone*)

also have ... $\leq (v \sqcup e^T * top)^T$
  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 $e \leq v * -v^T$
    and $t \leq v * v^T$
  shows *acyclic* $(t \sqcup e)$
proof –
  have $t^+ * e \leq t^+ * v * -v^T$
    by (*simp add*: *assms*(*3*) *comp-associative mult-right-isotone*)
  also have ... $\leq v * v^T * t^\star * v * -v^T$
    by (*simp add*: *assms*(*4*) *mult-left-isotone*)
  also have ... $\leq v * top * -v^T$
    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*: $t^+ * e \leq -1$
    .
  have *2*: $e * t^\star = e$
    using *assms*(*2*–*4*) *et*(*1*) *star-absorb* **by** *blast*
  have $e^\star = 1 \sqcup e \sqcup e * e * e^\star$
    by (*metis star.circ-loop-fixpoint star-square-2 sup-commute*)
  also have ... $= 1 \sqcup e$
    using *assms*(*2*,*3*) *ee comp-left-zero bot-least sup-absorb1* **by** *simp*
  finally have *3*: $e^\star = 1 \sqcup e$
    .
  have $e \leq v * -v^T$
    by (*simp add*: *assms*(*3*))
  also have ... ≤ -1
    by (*simp add*: *pp-increasing schroeder-3-p*)
  finally have *4*: $t^+ * e \sqcup e \leq -1$
    using *1* **by** *simp*
  have $(t \sqcup e)^+ = (t \sqcup e) * t^\star * (e * t^\star)^\star$
    using *star-sup-1 mult-assoc* **by** *simp*
  also have ... $= (t \sqcup e) * t^\star * (1 \sqcup e)$
    using *2 3* **by** *simp*
  also have ... $= t^+ * (1 \sqcup e) \sqcup e * t^\star * (1 \sqcup e)$
    by (*simp add*: *comp-right-dist-sup*)
  also have ... $= t^+ * (1 \sqcup e) \sqcup e * (1 \sqcup e)$
    using *2* **by** *simp*

also have ... $= t^+ * (1 \sqcup e) \sqcup e$
  using *3* **by** (*metis star-absorb assms*(*2*,*3*) *ee*)
also have ... $= t^+ \sqcup t^+ * e \sqcup e$
  by (*simp add*: *mult-left-dist-sup*)
also have ... ≤ -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* $(v * v^T)$
    and $t \leq v * v^T \sqcap --g$
    and $v^T = r^T * t^\star$
    and $e \leq v * -v^T \sqcap --g$
    and *vector* v
    and *regular* $((v \sqcup e^T * top) * (v \sqcup e^T * top)^T)$
  shows $t \sqcup e \leq (r^T * (--((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g))^\star)^T * (r^T * (--((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g))^\star)$
proof −
  let $?v = v \sqcup e^T * top$
  let $?G = ?v * ?v^T \sqcap g$
  let $?c = r^T * (--?G)^\star$
  have $v^T \leq r^T * (--(v * v^T \sqcap g))^\star$
    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*: $v^T \leq ?c \wedge v \leq ?c^T$
    by (*metis conv-isotone conv-involutive*)
  have $t \leq v * v^T$
    using *assms*(*2*) **by** *auto*
  hence *3*: $t \leq ?c^T * ?c$
    using *2 order-trans mult-isotone* **by** *blast*
  have $e \leq v * top \sqcap --g$
    by (*metis assms*(*4*,*5*) *inf.bounded-iff inf.sup-left-divisibility mult-right-isotone top.extremum*)
  hence $e \leq v * top \sqcap top * e \sqcap --g$
    by (*simp add*: *top-left-mult-increasing inf.boundedI*)
  hence $e \leq v * top * e \sqcap --g$
    by (*metis comp-inf-covector inf.absorb2 mult-assoc top.extremum*)
  hence $t \sqcup e \leq (v * v^T \sqcap --g) \sqcup (v * top * e \sqcap --g)$
    using *assms*(*2*) *sup-mono* **by** *blast*
  also have ... $= v * ?v^T \sqcap --g$
    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*: $t \sqcup e \leq --?G$
.
have $e \leq e * e^T * e$
by (*simp add*: *ex231c*)
also have ... $\leq v * -v^T * -v * v^T * e$
by (*metis assms*(*4*) *mult-left-isotone conv-isotone conv-dist-comp mult-assoc mult-isotone conv-involutive conv-complement inf.boundedE*)
also have ... $\leq v * top * v^T * e$
by (*metis mult-assoc mult-left-isotone mult-right-isotone top.extremum*)
also have ... $= v * r^T * t^\star * e$
using *assms*(*3*,*5*) **by** (*simp add*: *mult-assoc*)
also have ... $\leq v * r^T * (t \sqcup e)^\star$
by (*simp add*: *comp-associative mult-right-isotone star.circ-mult-upper-bound star.circ-sub-dist-1 star-isotone sup-commute*)
also have ... $\leq v * ?c$
using *4* **by** (*simp add*: *mult-assoc mult-right-isotone star-isotone*)
also have ... $\leq ?c^T * ?c$
using *2* **by** (*simp add*: *mult-left-isotone*)
finally show *?thesis*
using *3* **by** *simp*
qed

lemma *span-inv*:
assumes $e \leq v * -v^T$
and *vector v*
and *atom e*
and $t \leq (v * v^T) \sqcap g$
and $g^T = g$
and $v^T = r^T * t^\star$
and *injective r*
and $r^T \leq v^T$
and $r^T * ((v * v^T) \sqcap g)^\star \leq r^T * t^\star$
shows $r^T * (((v \sqcup e^T * top) * (v \sqcup e^T * top)^T) \sqcap g)^\star \leq r^T * (t \sqcup e)^\star$
proof −
let $?d = (v * v^T) \sqcap g$
have *1*: $(v \sqcup e^T * top) * (v \sqcup e^T * top)^T = v * v^T \sqcup v * v^T * e \sqcup e^T * v * v^T \sqcup e^T * e$
using *assms*(*1*−*3*) *ve-dist* **by** *simp*
have $t^T \leq ?d^T$
using *assms*(*4*) *conv-isotone* **by** *simp*
also have ... $= (v * v^T) \sqcap g^T$
by (*simp add*: *conv-dist-comp conv-dist-inf*)
also have ... $= ?d$
by (*simp add*: *assms*(*5*))
finally have *2*: $t^T \leq ?d$
.
have $v * v^T = (r^T * t^\star)^T * (r^T * t^\star)$

by (*metis assms*(*6*) *conv-involutive*)
also have ... = $t^{T\star}$ ∗ (*r* ∗ r^T) ∗ $t^\star$
by (*simp add*: *comp-associative conv-dist-comp conv-star-commute*)
also have ... ≤ $t^{T\star}$ ∗ *1* ∗ $t^\star$
by (*simp add*: *assms*(*7*) *mult-left-isotone star-right-induct-mult-iff star-sub-one*)
also have ... = $t^{T\star}$ ∗ $t^\star$
by *simp*
also have ... ≤ *?d** ∗ $t^\star$
using *2* **by** (*simp add*: *comp-left-isotone star.circ-isotone*)
also have ... ≤ *?d** ∗ *?d**
using *assms*(*4*) *mult-right-isotone star-isotone* **by** *simp*
also have *3*: ... = *?d**
by (*simp add*: *star.circ-transitive-equal*)
finally have *4*: *v* ∗ v^T ≤ *?d**
.
have *5*: r^T ∗ *?d** ∗ (*v* ∗ v^T ⊓ *g*) ≤ r^T ∗ *?d**
by (*simp add*: *comp-associative mult-right-isotone star.circ-plus-same star.left-plus-below-circ*)
have r^T ∗ *?d** ∗ (*v* ∗ v^T ∗ *e* ⊓ *g*) ≤ r^T ∗ *?d** ∗ *v* ∗ v^T ∗ *e*
by (*simp add*: *comp-associative comp-right-isotone*)
also have ... ≤ r^T ∗ *?d** ∗ *e*
using *3 4* **by** (*metis comp-associative comp-isotone eq-refl*)
finally have *6*: r^T ∗ *?d** ∗ (*v* ∗ v^T ∗ *e* ⊓ *g*) ≤ r^T ∗ *?d** ∗ *e*
.
have *7*: ∀ *x* . r^T ∗ (*1* ⊔ *v* ∗ v^T) ∗ e^T ∗ *x* = *bot*
proof
fix *x*
have r^T ∗ (*1* ⊔ *v* ∗ v^T) ∗ e^T ∗ *x* ≤ r^T ∗ (*1* ⊔ *v* ∗ v^T) ∗ e^T ∗ *top*
by (*simp add*: *mult-right-isotone*)
also have ... = r^T ∗ e^T ∗ *top* ⊔ r^T ∗ *v* ∗ v^T ∗ e^T ∗ *top*
by (*simp add*: *comp-associative mult-left-dist-sup mult-right-dist-sup*)
also have ... = r^T ∗ e^T ∗ *top*
by (*metis assms*(*1*,*2*) *mult-assoc mult-right-dist-sup mult-right-zero sup-bot-right vTeT*)
also have ... ≤ v^T ∗ e^T ∗ *top*
by (*simp add*: *assms*(*8*) *comp-isotone*)
also have ... = *bot*
using *vTeT assms*(*1*,*2*) **by** *simp*
finally show r^T ∗ (*1* ⊔ *v* ∗ v^T) ∗ e^T ∗ *x* = *bot*
by (*simp add*: *le-bot*)
qed
have r^T ∗ *?d** ∗ (e^T ∗ *v* ∗ v^T ⊓ *g*) ≤ r^T ∗ *?d** ∗ e^T ∗ *v* ∗ v^T
by (*simp add*: *comp-associative comp-right-isotone*)
also have ... ≤ r^T ∗ (*1* ⊔ *v* ∗ v^T) ∗ e^T ∗ *v* ∗ 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*: $r^T * ?d^\star * (e^T * v * v^T \sqcap g) = bot$
by (*simp add*: *le-bot*)
have $r^T * ?d^\star * (e^T * e \sqcap g) \leq r^T * ?d^\star * e^T * e$
by (*simp add*: *comp-associative comp-right-isotone*)
also have ... $\leq r^T * (1 \sqcup v * v^T) * e^T * e$
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*: $r^T * ?d^\star * (e^T * e \sqcap g) = bot$
by (*simp add*: *le-bot*)
have $r^T * ?d^\star * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) = r^T * ?d^\star * ((v * v^T \sqcup v * v^T * e \sqcup e^T * v * v^T \sqcup e^T * e) \sqcap g)$
using *1* **by** *simp*
also have ... $= r^T * ?d^\star * ((v * v^T \sqcap g) \sqcup (v * v^T * e \sqcap g) \sqcup (e^T * v * v^T \sqcap g) \sqcup (e^T * e \sqcap g))$
by (*simp add*: *inf-sup-distrib2*)
also have ... $= r^T * ?d^\star * (v * v^T \sqcap g) \sqcup r^T * ?d^\star * (v * v^T * e \sqcap g) \sqcup r^T * ?d^\star * (e^T * v * v^T \sqcap g) \sqcup r^T * ?d^\star * (e^T * e \sqcap g)$
by (*simp add*: *comp-left-dist-sup*)
also have ... $= r^T * ?d^\star * (v * v^T \sqcap g) \sqcup r^T * ?d^\star * (v * v^T * e \sqcap g)$
using *8 9* **by** *simp*
also have ... $\leq r^T * ?d^\star \sqcup r^T * ?d^\star * e$
using *5 6 sup.mono* **by** *simp*
also have ... $= r^T * ?d^\star * (1 \sqcup e)$
by (*simp add*: *mult-left-dist-sup*)
finally have *10*: $r^T * ?d^\star * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \leq r^T * ?d^\star * (1 \sqcup e)$
by *simp*
have $r^T * ?d^\star * e * (v * v^T \sqcap g) \leq r^T * ?d^\star * e * v * v^T$
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*: $r^T * ?d^\star * e * (v * v^T \sqcap g) = bot$
by (*simp add*: *le-bot*)
have $r^T * ?d^\star * e * (v * v^T * e \sqcap g) \leq r^T * ?d^\star * e * v * v^T * e$
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*: $r^T * ?d^\star * e * (v * v^T * e \sqcap g) = bot$
by (*simp add*: *le-bot*)
have $r^T * ?d^\star * e * (e^T * v * v^T \sqcap g) \leq r^T * ?d^\star * e * e^T * v * v^T$
by (*simp add*: *comp-associative comp-right-isotone*)
also have ... $\leq r^T * ?d^\star * 1 * v * v^T$
by (*metis assms*(*3*) *atom-injective comp-associative comp-left-isotone comp-right-isotone*)
also have ... $= r^T * ?d^\star * v * v^T$
by *simp*
also have ... $\leq r^T * ?d^\star * ?d^\star$

using *4* **by** (*simp add*: *mult-right-isotone mult-assoc*)
also have ... $= r^T * ?d^\star$
by (*simp add*: *star.circ-transitive-equal comp-associative*)
finally have *13*: $r^T * ?d^\star * e * (e^T * v * v^T \sqcap g) \leq r^T * ?d^\star$
.
have $r^T * ?d^\star * e * (e^T * e \sqcap g) \leq r^T * ?d^\star * e * e^T * e$
by (*simp add*: *comp-associative comp-right-isotone*)
also have ... $\leq r^T * ?d^\star * 1 * e$
by (*metis assms*(*3*) *atom-injective comp-associative comp-left-isotone comp-right-isotone*)
also have ... $= r^T * ?d^\star * e$
by *simp*
finally have *14*: $r^T * ?d^\star * e * (e^T * e \sqcap g) \leq r^T * ?d^\star * e$
.
have $r^T * ?d^\star * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) = r^T * ?d^\star * e * ((v * v^T \sqcup v * v^T * e \sqcup e^T * v * v^T \sqcup e^T * e) \sqcap g)$
using *1* **by** *simp*
also have ... $= r^T * ?d^\star * e * ((v * v^T \sqcap g) \sqcup (v * v^T * e \sqcap g) \sqcup (e^T * v * v^T \sqcap g) \sqcup (e^T * e \sqcap g))$
by (*simp add*: *inf-sup-distrib2*)
also have ... $= r^T * ?d^\star * e * (v * v^T \sqcap g) \sqcup r^T * ?d^\star * e * (v * v^T * e \sqcap g) \sqcup r^T * ?d^\star * e * (e^T * v * v^T \sqcap g) \sqcup r^T * ?d^\star * e * (e^T * e \sqcap g)$
by (*simp add*: *comp-left-dist-sup*)
also have ... $= r^T * ?d^\star * e * (e^T * v * v^T \sqcap g) \sqcup r^T * ?d^\star * e * (e^T * e \sqcap g)$
using *11 12* **by** *simp*
also have ... $\leq r^T * ?d^\star \sqcup r^T * ?d^\star * e$
using *13 14 sup-mono* **by** *simp*
also have ... $= r^T * ?d^\star * (1 \sqcup e)$
by (*simp add*: *mult-left-dist-sup*)
finally have *15*: $r^T * ?d^\star * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \leq r^T * ?d^\star * (1 \sqcup e)$
by *simp*
have $r^T \leq r^T * ?d^\star$
using *mult-right-isotone star.circ-reflexive* **by** *fastforce*
also have ... $\leq r^T * ?d^\star * (1 \sqcup e)$
by (*simp add*: *semiring.distrib-left*)
finally have *16*: $r^T \leq r^T * ?d^\star * (1 \sqcup e)$
.
have $r^T * ?d^\star * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) = r^T * ?d^\star * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \sqcup r^T * ?d^\star * e * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g)$
by (*simp add*: *semiring.distrib-left semiring.distrib-right*)
also have ... $\leq r^T * ?d^\star * (1 \sqcup e)$
using *10 15 le-supI* **by** *simp*
finally have $r^T * ?d^\star * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \leq r^T * ?d^\star * (1 \sqcup e)$
.
hence $r^T \sqcup r^T * ?d^\star * (1 \sqcup e) * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g) \leq r^T * ?d^\star * (1 \sqcup e)$

using *16 sup-least* **by** *simp*
hence $r^T * ((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g)^\star \leq r^T * ?d^\star * (1 \sqcup e)$
by (*simp add*: *star-right-induct*)
also have ... $\leq r^T * t^\star * (1 \sqcup e)$
by (*simp add*: *assms*(*9*) *mult-left-isotone*)
also have ... $\leq r^T * (t \sqcup e)^\star$
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** $E\ w\ v\ e \equiv w \sqcap --v * -v^T \sqcap top * e * w^{T\star}$
abbreviation $P :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $P\ w\ v\ e \equiv w \sqcap -v * -v^T \sqcap top * e * w^{T\star}$
abbreviation $EP :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $EP\ w\ v\ e \equiv w \sqcap -v^T \sqcap top * e * w^{T\star}$
abbreviation $W :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $W\ w\ v\ e \equiv (w \sqcap -(EP\ w\ v\ e)) \sqcup (P\ w\ v\ e)^T \sqcup e$

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 $e \leq v * -v^T$
and *vector* v
shows $(w \sqcap -(EP\ w\ v\ e)) * e^T = bot$
proof –
have *1*: $top * e \leq -v^T$
by (*simp add*: *assms schroeder-4-p vTeT*)
have $top * e \leq top * e * w^{T\star}$
using *sup-right-divisibility star.circ-back-loop-fixpoint* **by** *blast*
hence $top * e \leq -v^T \sqcap top * e * w^{T\star}$
using *1* **by** *simp*
hence $top * e \leq -(w \sqcap -EP\ w\ v\ e)$
by (*metis inf.assoc inf-import-p le-infI2 p-antitone p-antitone-iff*)
hence $(w \sqcap -(EP\ w\ v\ e)) * e^T \leq bot$
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 $(P\ w\ v\ e)^T * e^T \leq (w \sqcap top * e * w^{T\star})^T * e^T$
    using *comp-inf.mult-left-isotone conv-dist-inf mult-left-isotone* **by** *simp*
  also have ... $= (w^T \sqcap w^{T\star T} * e^T * top) * e^T$
    by (*simp add*: *comp-associative conv-dist-comp conv-dist-inf*)
  also have ... $= w^\star * e^T * top \sqcap w^T * e^T$
    by (*simp add*: *conv-star-commute inf-vector-comp*)
  also have ... $\leq (w^T \sqcap w^\star * e^T * top * e) * (e^T \sqcap w^+ * e^T * top)$
    by (*metis dedekind mult-assoc conv-involutive inf-commute*)
  also have ... $\leq (w^\star * e^T * top * e) * (w^+ * e^T * top)$
    by (*simp add*: *mult-isotone*)
  also have ... $\leq (top * e) * (w^+ * e^T * top)$
    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*: $(w \sqcap -(EP\ w\ v\ e)) * (w \sqcap -(EP\ w\ v\ e))^T \leq 1$
  proof −
    have $(w \sqcap -(EP\ w\ v\ e)) * (w \sqcap -(EP\ w\ v\ e))^T \leq w * w^T$
      by (*simp add*: *comp-isotone conv-isotone*)
    also have ... ≤ 1
      by (*simp add*: *assms(3)*)
    finally show *?thesis*
      .
  qed

have *2*: $(w \sqcap -(EP\ w\ v\ e)) * (P\ w\ v\ e)^{TT} \leq 1$
proof −
have $top * (P\ w\ v\ e)^T = top * (w^T \sqcap -v * -v^T \sqcap w^{T\star T} * e^T * top)$
by (*simp add*: *comp-associative conv-complement conv-dist-comp conv-dist-inf*)
also have ... $= top * e * w^{T\star} * (w^T \sqcap -v * -v^T)$
by (*metis comp-inf-vector conv-dist-comp conv-involutive inf-top-left mult-assoc*)
also have ... $\leq top * e * w^{T\star} * (w^T \sqcap top * -v^T)$
using *comp-inf.mult-right-isotone mult-left-isotone mult-right-isotone* **by** *simp*
also have ... $= top * e * w^{T\star} * w^T \sqcap -v^T$
by (*metis assms*(*4*) *comp-inf-covector vector-conv-compl*)
also have ... $\leq -v^T \sqcap top * e * w^{T\star}$
by (*simp add*: *comp-associative comp-isotone inf.coboundedI1 star.circ-plus-same star.left-plus-below-circ*)
finally have $top * (P\ w\ v\ e)^T \leq -(w \sqcap -EP\ w\ v\ e)$
by (*metis inf.assoc inf-import-p le-infI2 p-antitone p-antitone-iff*)
hence $(w \sqcap -(EP\ w\ v\ e)) * (P\ w\ v\ e)^{TT} \leq bot$
using *p-top schroeder-4-p* **by** *blast*
thus *?thesis*
by (*simp add*: *bot-unique*)
qed
have *3*: $(w \sqcap -(EP\ w\ v\ e)) * e^T \leq 1$
by (*metis assms*(*2*,*4*) *exchange-injective-3 bot-least*)
have *4*: $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^T \leq 1$
using *2 conv-dist-comp coreflexive-symmetric* **by** *fastforce*
have *5*: $(P\ w\ v\ e)^T * (P\ w\ v\ e)^{TT} \leq 1$
proof −
have $(P\ w\ v\ e)^T * (P\ w\ v\ e)^{TT} \leq (top * e * w^{T\star})^T * (top * e * w^{T\star})$
by (*simp add*: *conv-dist-inf mult-isotone*)
also have ... $= w^\star * e^T * top * top * e * w^{T\star}$
using *conv-star-commute conv-dist-comp conv-involutive conv-top mult-assoc* **by** *presburger*
also have ... $= w^\star * e^T * top * e * w^{T\star}$
by (*simp add*: *comp-associative*)
also have ... $\leq w^\star * 1 * w^{T\star}$
by (*metis comp-left-isotone comp-right-isotone mult-assoc assms*(*1*) *point-injective*)
finally have $(P\ w\ v\ e)^T * (P\ w\ v\ e)^{TT} \leq w^\star * w^{T\star} \sqcap w^T * w$
by (*simp add*: *conv-isotone inf.left-commute inf.sup-monoid.add-commute mult-isotone*)
also have ... ≤ 1
by (*simp add*: *assms*(*3*) *forest-separate*)
finally show *?thesis*
.
qed
have *6*: $(P\ w\ v\ e)^T * e^T \leq 1$
using *assms exchange-injective-6 bot-least* **by** *simp*

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

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

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

by (*simp add*: *assms*(*4*))
finally have *1*: $(W\ w\ v\ e) * e^T * top \le v$
.
have $(W\ w\ v\ e) * v = (w \sqcap -(EP\ w\ v\ e)) * v \sqcup (P\ w\ v\ e)^T * v \sqcup e * v$
by (*simp add*: *mult-right-dist-sup*)
also have ... $= (w \sqcap -(EP\ w\ v\ e)) * v$
by (*metis assms*(*2*,*4*) *pv ev sup-bot-right*)
also have ... $\le w * v$
by (*simp add*: *mult-left-isotone*)
finally have *2*: $(W\ w\ v\ e) * v \le v$
using *assms*(*5*) *order-trans* by *blast*
have $(W\ w\ v\ e) * (v \sqcup e^T * top) = (W\ w\ v\ e) * v \sqcup (W\ w\ v\ e) * e^T * top$
by (*simp add*: *semiring.distrib-left mult-assoc*)
also have ... $\le v$
using *1 2* by *simp*
also have ... $\le v \sqcup e^T * top$
by *simp*
finally show *?thesis*
.
qed

The graph after exchanging is acyclic.

lemma *exchange-acyclic*:
assumes *vector v*
and $e \le v * -v^T$
and $w * v \le v$
and *acyclic w*
shows *acyclic* $(W\ w\ v\ e)$
proof −
have *1*: $(P\ w\ v\ e)^T * e = bot$
proof −
have $(P\ w\ v\ e)^T * e \le (-v * -v^T)^T * e$
by (*meson conv-order dual-order.trans inf.coboundedl inf.cobounded2 mult-left-isotone*)
also have ... $= -v * -v^T * e$
by (*simp add*: *conv-complement conv-dist-comp*)
also have ... $\le -v * -v^T * v * -v^T$
by (*simp add*: *assms*(*2*) *comp-associative mult-right-isotone*)
also have ... $= bot$
by (*simp add*: *assms*(*1*) *covector-vector-comp mult-assoc*)
finally show *?thesis*
by (*simp add*: *bot-unique*)
qed
have *2*: $e * e = bot$
using *assms*(*1*,*2*) *ee* by *auto*
have *3*: $(w \sqcap -(EP\ w\ v\ e)) * (P\ w\ v\ e)^T = bot$
proof −
have $top * (P\ w\ v\ e) \le top * (-v * -v^T \sqcap top * e * w^{T\star})$
using *comp-inf.mult-semi-associative mult-right-isotone* by *auto*

also have ... $\leq$ *top* $*$ $-v$ $*$ $-v^T \sqcap$ *top* $*$ *top* $*$ e $*$ $w^{T\star}$
by (*simp add*: *comp-inf-covector mult-assoc*)
also have ... $\leq$ *top* $*$ $-v^T \sqcap$ *top* $*$ e $*$ $w^{T\star}$
by (*meson comp-inf.comp-isotone mult-left-isotone top.extremum*)
also have ... $= -v^T \sqcap$ *top* $*$ e $*$ $w^{T\star}$
by (*simp add*: *assms*(*1*) *vector-conv-compl*)
finally have *top* $*$ (P w v e) $\leq -(w \sqcap -EP\ w\ v\ e)$
by (*metis inf.assoc inf-import-p le-infI2 p-antitone p-antitone-iff*)
hence $(w \sqcap -(EP\ w\ v\ e)) * (P\ w\ v\ e)^T \leq bot$
using *p-top schroeder-4-p* **by** *blast*
thus *?thesis*
using *bot-unique* **by** *blast*
qed
hence *4*: $(w \sqcap -(EP\ w\ v\ e)) * (P\ w\ v\ e)^{T\star} = w \sqcap -(EP\ w\ v\ e)$
using *star-absorb* **by** *blast*
hence *5*: $(w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T\star} = (w \sqcap -(EP\ w\ v\ e))^+$
by (*metis star-plus mult-assoc*)
hence *6*: $(w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} = (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v$
$e)^{T\star}$
by (*metis star.circ-loop-fixpoint mult-assoc*)
have *7*: $(w \sqcap -(EP\ w\ v\ e))^+ * e \leq v * top$
proof –
have $e \leq v * top$
using *assms*(*2*) *dual-order.trans mult-right-isotone top-greatest* **by** *blast*
hence *8*: $e \sqcup w * v * top \leq v * top$
by (*simp add*: *assms*(*1*,*3*) *comp-associative*)
have $(w \sqcap -(EP\ w\ v\ e))^+ * e \leq w^+ * e$
by (*simp add*: *comp-isotone star-isotone*)
also have ... $\leq w^\star * e$
by (*simp add*: *mult-left-isotone star.left-plus-below-circ*)
also have ... $\leq v * top$
using *8* **by** (*simp add*: *comp-associative star-left-induct*)
finally show *?thesis*
.
qed
have *9*: $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ * e = bot$
proof –
have $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ * e \leq (P\ w\ v\ e)^T * v * top$
using *7* **by** (*simp add*: *mult-assoc mult-right-isotone*)
also have ... $= bot$
by (*simp add*: *assms*(*1*) *pv*)
finally show *?thesis*
using *bot-unique* **by** *blast*
qed
have *10*: $e * (w \sqcap -(EP\ w\ v\ e))^+ * e = bot$
proof –
have $e * (w \sqcap -(EP\ w\ v\ e))^+ * e \leq e * v * top$
using *7* **by** (*simp add*: *mult-assoc mult-right-isotone*)
also have ... $\leq v * -v^T * v * top$

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*: $e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \le v * -v^T$
proof −
have *12*: $-v^T * w \le -v^T$
by (*metis assms*(*3*) *conv-complement order-lesseq-imp pp-increasing schroeder-6-p*)
have $v * -v^T * (w \sqcap -(EP\ w\ v\ e)) \le v * -v^T * w$
by (*simp add*: *comp-isotone star-isotone*)
also have ... $\le v * -v^T$
using *12* **by** (*simp add*: *comp-isotone comp-associative*)
finally have *13*: $v * -v^T * (w \sqcap -(EP\ w\ v\ e)) \le v * -v^T$
.
have *14*: $(P\ w\ v\ e)^T \le -v * -v^T$
by (*metis conv-complement conv-dist-comp conv-involutive conv-order inf-le1 inf-le2 order-trans*)
have $e * (P\ w\ v\ e)^{T\star} \le v * -v^T * (P\ w\ v\ e)^{T\star}$
by (*simp add*: *assms*(*2*) *mult-left-isotone*)
also have ... $= v * -v^T \sqcup v * -v^T * (P\ w\ v\ e)^{T+}$
by (*metis mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute*)
also have ... $= v * -v^T \sqcup v * -v^T * (P\ w\ v\ e)^{T\star} * (P\ w\ v\ e)^T$
by (*simp add*: *mult-assoc star-plus*)
also have ... $\le v * -v^T \sqcup v * -v^T * (P\ w\ v\ e)^{T\star} * -v * -v^T$
using *14 mult-assoc mult-right-isotone sup-right-isotone* **by** *simp*
also have ... $\le v * -v^T \sqcup v * top * -v^T$
by (*metis top-greatest mult-right-isotone mult-left-isotone mult-assoc sup-right-isotone*)
also have ... $= v * -v^T$
by (*simp add*: *assms*(*1*))
finally have $e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \le v * -v^T * (w \sqcap -(EP\ w\ v\ e))^\star$
by (*simp add*: *mult-left-isotone*)
also have ... $\le v * -v^T$
using *13* **by** (*simp add*: *star-right-induct-mult*)
finally show *?thesis*
.
qed
have *15*: $(w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \le -1$
proof −
have $(w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star = (w \sqcap -(EP\ w\ v\ e))^+ * (w \sqcap -(EP\ w\ v\ e))^\star$
using *5* **by** *simp*
also have ... $= (w \sqcap -(EP\ w\ v\ e))^+$
by (*simp add*: *mult-assoc star.circ-transitive-equal*)
also have ... $\le w^+$

by (*simp add*: *comp-isotone star-isotone*)
finally show *?thesis*
using *assms*(*4*) **by** *simp*
qed
have *16*: $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$
proof −
have $(w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T+} \leq (w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T\star}$
by (*simp add*: *mult-right-isotone star.left-plus-below-circ*)
also have ... $= (w \sqcap -(EP\ w\ v\ e))^+$
using *5* **by** *simp*
also have ... $\leq w^+$
by (*simp add*: *comp-isotone star-isotone*)
finally have $(w \sqcap -(EP\ w\ v\ e))^+ * (P\ w\ v\ e)^{T+} \leq -1$
using *assms*(*4*) **by** *simp*
hence *17*: $(P\ w\ v\ e)^{T+} * (w \sqcap -(EP\ w\ v\ e))^+ \leq -1$
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 ... $= w^{+T}$
by (*simp add*: *conv-dist-comp conv-star-commute star-plus*)
also have ... ≤ -1
using *assms*(*4*) *conv-complement conv-isotone* **by** *force*
finally have *18*: $(P\ w\ v\ e)^{T+} \leq -1$
.
have $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star = (P\ w\ v\ e)^T * ((w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T\star}) * (w \sqcap -(EP\ w\ v\ e))^\star$
using *6* **by** (*simp add*: *comp-associative*)
also have ... $= (P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ * (w \sqcap -(EP\ w\ v\ e))^\star \sqcup (P\ w\ v\ e)^{T+} * (w \sqcap -(EP\ w\ v\ e))^\star$
by (*simp add*: *mult-left-dist-sup mult-right-dist-sup*)
also have ... $= (P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+} * (w \sqcap -(EP\ w\ v\ e))^\star$
by (*simp add*: *mult-assoc star.circ-transitive-equal*)
also have ... $= (P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+} * (1 \sqcup (w \sqcap -(EP\ w\ v\ e))^+)$
using *star-left-unfold-equal* **by** *simp*
also have ... $= (P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+} * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+}$
by (*simp add*: *mult-left-dist-sup sup.left-commute sup-commute*)
also have ... $= ((P\ w\ v\ e)^T \sqcup (P\ w\ v\ e)^{T+}) * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+}$
by (*simp add*: *mult-right-dist-sup*)
also have ... $= (P\ w\ v\ e)^{T+} * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T+}$
using *star.circ-mult-increasing* **by** (*simp add*: *le-iff-sup*)
also have ... ≤ -1
using *17 18* **by** *simp*

finally show *?thesis*

.

qed

have *19*: $e * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$

proof −

have $e * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star = e * ((w \sqcap -(EP\ w\ v\ e))^+ \sqcup (P\ w\ v\ e)^{T\star}) * (w \sqcap -(EP\ w\ v\ e))^\star$

using *6* **by** (*simp add*: *mult-assoc*)

also have ... $= e * (w \sqcap -(EP\ w\ v\ e))^+ * (w \sqcap -(EP\ w\ v\ e))^\star \sqcup e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$

by (*simp add*: *mult-left-dist-sup mult-right-dist-sup*)

also have ... $= e * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$

by (*simp add*: *mult-assoc star.circ-transitive-equal*)

also have ... $\leq e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^+ \sqcup e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$

by (*metis mult-right-sub-dist-sup-right semiring.add-right-mono star.circ-back-loop-fixpoint*)

also have ... $\leq e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$

using *mult-right-isotone star.left-plus-below-circ* **by** *auto*

also have ... $\leq v * -v^T$

using *11* **by** *simp*

also have ... ≤ -1

by (*simp add*: *pp-increasing schroeder-3-p*)

finally show *?thesis*

.

qed

have *20*: $(W\ w\ v\ e) * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$

using *15 16 19* **by** (*simp add*: *comp-right-dist-sup*)

have *21*: $(w \sqcap -(EP\ w\ v\ e))^+ * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$

proof −

have $(w \sqcap -(EP\ w\ v\ e)) * v * -v^T \leq w * v * -v^T$

by (*simp add*: *comp-isotone star-isotone*)

also have ... $\leq v * -v^T$

by (*simp add*: *assms*(*3*) *mult-left-isotone*)

finally have *22*: $(w \sqcap -(EP\ w\ v\ e)) * v * -v^T \leq v * -v^T$

.

have $(w \sqcap -(EP\ w\ v\ e))^+ * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq (w \sqcap -(EP\ w\ v\ e))^+ * v * -v^T$

using *11* **by** (*simp add*: *mult-right-isotone mult-assoc*)

also have ... $\leq (w \sqcap -(EP\ w\ v\ e))^\star * v * -v^T$

using *mult-left-isotone star.left-plus-below-circ* **by** *blast*

also have ... $\leq v * -v^T$

using *22* **by** (*simp add*: *star-left-induct-mult mult-assoc*)

also have ... ≤ -1

by (*simp add*: *pp-increasing schroeder-3-p*)

finally show *?thesis*

.

qed
have *23*: $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$
proof −
have $(P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^\star * e = (P\ w\ v\ e)^T * e \sqcup (P\ w\ v\ e)^T * (w \sqcap -(EP\ w\ v\ e))^+ * e$
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*: $e * (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$
proof −
have $e * (w \sqcap -(EP\ w\ v\ e))^\star * e = e * e \sqcup e * (w \sqcap -(EP\ w\ v\ e))^+ * e$
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*: $(W\ w\ v\ e) * (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \leq -1$
using *21 23 24* **by** (*simp add*: *comp-right-dist-sup*)
have $(W\ w\ v\ e)^\star = ((P\ w\ v\ e)^T \sqcup e)^\star * ((w \sqcap -(EP\ w\ v\ e)) * ((P\ w\ v\ e)^T \sqcup e)^\star)^\star$
by (*metis star-sup-1 sup.left-commute sup-commute*)
also have ... $= ((P\ w\ v\ e)^{T\star} \sqcup e * (P\ w\ v\ e)^{T\star}) * ((w \sqcap -(EP\ w\ v\ e)) * ((P\ w\ v\ e)^{T\star} \sqcup e * (P\ w\ v\ e)^{T\star}))^\star$
using *1 2 star-separate* **by** *auto*
also have ... $= ((P\ w\ v\ e)^{T\star} \sqcup e * (P\ w\ v\ e)^{T\star}) * ((w \sqcap -(EP\ w\ v\ e)) * (1 \sqcup e * (P\ w\ v\ e)^{T\star}))^\star$
using *4 mult-left-dist-sup* **by** *auto*
also have ... $= (w \sqcap -(EP\ w\ v\ e))^\star * ((P\ w\ v\ e)^{T\star} \sqcup e * (P\ w\ v\ e)^{T\star}) * (w \sqcap -(EP\ w\ v\ e))^\star$
using *3 9 10 star-separate-2* **by** *blast*
also have ... $= (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \sqcup (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$
by (*simp add*: *semiring.distrib-left semiring.distrib-right mult-assoc*)
finally have $(W\ w\ v\ e)^+ = (W\ w\ v\ e) * ((w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \sqcup (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star)$
by *simp*
also have ... $= (W\ w\ v\ e) * (w \sqcap -(EP\ w\ v\ e))^\star * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star \sqcup (W\ w\ v\ e) * (w \sqcap -(EP\ w\ v\ e))^\star * e * (P\ w\ v\ e)^{T\star} * (w \sqcap -(EP\ w\ v\ e))^\star$

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 $e \leq v * -v^T \sqcap g$
and $t \leq g$
and $v^T = r^T * t^\star$
shows $e \leq (r^T * g^\star)^T * (r^T * g^\star) \sqcap g$
proof –
have $e \leq v * -v^T \sqcap g$
by (*rule assms*(*1*))
also have ... $\leq v * (-v^T \sqcap v^T * g) \sqcap g$
by (*simp add*: *dedekind-1*)
also have ... $\leq v * v^T * g \sqcap g$
by (*simp add*: *comp-associative comp-right-isotone inf-commute le-infI2*)
also have ... $= v * (r^T * t^\star) * g \sqcap g$
by (*simp add*: *assms*(*3*))
also have ... $= (r^T * t^\star)^T * (r^T * t^\star) * g \sqcap g$
by (*metis assms*(*3*) *conv-involutive*)
also have ... $\leq (r^T * t^\star)^T * (r^T * g^\star) * g \sqcap g$
using *assms*(*2*) *comp-inf.mult-left-isotone comp-isotone star-isotone* **by** *auto*
also have ... $\leq (r^T * t^\star)^T * (r^T * g^\star) \sqcap g$
using *inf.sup-right-isotone inf-commute mult-assoc mult-right-isotone star.left-plus-below-circ star-plus* **by** *presburger*
also have ... $\leq (r^T * g^\star)^T * (r^T * g^\star) \sqcap g$
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 $t \leq w$
and $t \leq v * v^T$
and *vector v*
shows $t \leq W\ w\ v\ e$
proof –
have $t \sqcap EP\ w\ v\ e \leq t \sqcap -v^T$
by (*simp add*: *inf.coboundedI2 inf.sup-monoid.add-assoc*)
also have ... $\leq v * v^T \sqcap -v^T$
by (*simp add*: *assms*(*2*) *inf.coboundedI1*)

also have ... $\leq$ *bot*
by (*simp add*: *assms*(*3*) *covector-vector-comp eq-refl schroeder-2*)
finally have $t \leq -(EP\ w\ v\ e)$
using *le-bot pseudo-complement* **by** *blast*
hence $t \leq w \sqcap -(EP\ w\ v\ e)$
using *assms*(*1*) **by** *simp*
thus *?thesis*
by (*simp add*: *le-supI2 sup-commute*)
qed

lemma *mst-extends-new-tree*:
$t \leq w \Longrightarrow t \leq v * v^T \Longrightarrow vector\ v \Longrightarrow t \sqcup e \leq W\ w\ v\ e$
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 *mst-reachable-inv*:
assumes *regular* $(EP\ w\ v\ e)$
and *vector* r
and $e \leq v * -v^T$
and *vector* v
and $v^T = r^T * t^\star$
and $t \leq w$
and $t \leq v * v^T$
and $w * v \leq v$
shows $r^T * w^\star \leq r^T * (W\ w\ v\ e)^\star$
proof −
have *1*: $r^T \leq r^T * (W\ w\ v\ e)^\star$
using *sup.bounded-iff star.circ-back-loop-prefixpoint* **by** *blast*
have $top * e * (w^T \sqcap -v^T)^\star * w^T \sqcap -v^T = top * e * (w^T \sqcap -v^T)^\star * (w^T \sqcap -v^T)$
by (*simp add*: *assms*(*4*) *covector-comp-inf vector-conv-compl*)
also have ... $\leq top * e * (w^T \sqcap -v^T)^\star$
by (*simp add*: *comp-isotone mult-assoc star.circ-plus-same star.left-plus-below-circ*)
finally have *2*: $top * e * (w^T \sqcap -v^T)^\star * w^T \leq top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T$
by (*simp add*: *shunting-var-p*)
have *3*: $--v^T * w^T \leq top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T$
by (*metis assms*(*8*) *conv-dist-comp conv-order mult-assoc order.trans pp-comp-semi-commute pp-isotone sup.coboundedI1 sup-commute*)
have *4*: $top * e \leq top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T$
using *sup-right-divisibility star.circ-back-loop-fixpoint le-supI1* **by** *blast*
have $(top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T) * w^T = top * e * (w^T \sqcap -v^T)^\star * w^T \sqcup --v^T * w^T$
by (*simp add*: *comp-right-dist-sup*)
also have ... $\leq top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T$
using *2 3* **by** *simp*
finally have $top * e \sqcup (top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T) * w^T \leq top * e *$

$(w^T \sqcap -v^T)^\star \sqcup --v^T$
  using *4* **by** *simp*
 hence *5*: $top * e * w^{T\star} \leq top * e * (w^T \sqcap -v^T)^\star \sqcup --v^T$
  by (*simp add*: *star-right-induct*)
 have *6*: $top * e \leq top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star$
  using *sup-right-divisibility star.circ-back-loop-fixpoint* **by** *blast*
 have $(top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)^T \leq (top * e * w^{T\star})^T$
  by (*simp add*: *star-isotone mult-right-isotone conv-isotone inf-assoc*)
 also have ... $= w^\star * e^T * top$
  by (*simp add*: *conv-dist-comp conv-star-commute mult-assoc*)
 finally have *7*: $(top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)^T \leq w^\star * e^T * top$
  .
 have $(top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)^T \leq (top * e * (-v * -v^T)^\star)^T$
  by (*simp add*: *conv-isotone inf-commute mult-right-isotone star-isotone le-infI2*)
 also have ... $\leq (top * v * -v^T * (-v * -v^T)^\star)^T$
  by (*metis assms*(*3*) *conv-isotone mult-left-isotone mult-right-isotone mult-assoc*)
 also have ... $= (top * v * (-v^T * -v)^\star * -v^T)^T$
  by (*simp add*: *mult-assoc star-slide*)
 also have ... $\leq (top * -v^T)^T$
  using *conv-order mult-left-isotone* **by** *auto*
 also have ... $= -v$
  by (*simp add*: *assms*(*4*) *conv-complement vector-conv-compl*)
 finally have *8*: $(top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)^T \leq w^\star * e^T * top \sqcap -v$
  using *7* **by** *simp*
 have *covector* $(top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)$
  by (*simp add*: *covector-mult-closed*)
 hence $top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T \sqcap -v^T) = top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T \sqcap -v^T \sqcap (top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star)^T)$
  by (*metis comp-inf-vector-1 inf.idem*)
 also have ... $\leq top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T \sqcap -v^T \sqcap w^\star * e^T * top \sqcap -v)$
  using *8 mult-right-isotone inf.sup-right-isotone inf-assoc* **by** *simp*
 also have ... $= top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T \sqcap (-v \sqcap -v^T) \sqcap w^\star * e^T * top)$
  using *inf-assoc inf-commute* **by** (*simp add*: *inf-assoc*)
 also have ... $= top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)$
  using *assms*(*4*) *conv-complement vector-complement-closed vector-covector* **by** *fastforce*
 also have ... $\leq top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star$
  by (*simp add*: *comp-associative comp-isotone star.circ-plus-same star.left-plus-below-circ*)
 finally have *9*: $top * e \sqcup top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star * (w^T$

$\sqcap -v^T) \leq top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star$
using *6* **by** *simp*
have $EP\ w\ v\ e \leq -v^T \sqcap top * e * w^{T\star}$
using *inf.sup-left-isotone* **by** *auto*
also have ... $\leq top * e * (w^T \sqcap -v^T)^\star$
using *5* **by** (*metis inf-commute shunting-var-p*)
also have ... $\leq top * e * (w^T \sqcap -v * -v^T \sqcap w^\star * e^T * top)^\star$
using *9* **by** (*simp add*: *star-right-induct*)
finally have *10*: $EP\ w\ v\ e \leq top * e * (P\ w\ v\ e)^{T\star}$
by (*simp add*: *conv-complement conv-dist-comp conv-dist-inf conv-star-commute mult-assoc*)
have $top * e = top * (v * -v^T \sqcap e)$
by (*simp add*: *assms*(*3*) *inf.absorb2*)
also have ... $\leq top * (v * top \sqcap e)$
using *inf.sup-right-isotone inf-commute mult-right-isotone top-greatest* **by** *presburger*
also have ... $= (top \sqcap (v * top)^T) * e$
using *assms*(*4*) *covector-inf-comp-3* **by** *presburger*
also have ... $= top * v^T * e$
by (*simp add*: *conv-dist-comp*)
also have ... $= top * r^T * t^\star * e$
by (*simp add*: *assms*(*5*) *comp-associative*)
also have ... $\leq top * r^T * (W\ w\ v\ e)^\star * e$
by (*metis assms*(*4*,*6*,*7*) *mst-extends-old-tree star-isotone mult-left-isotone mult-right-isotone*)
finally have *11*: $top * e \leq top * r^T * (W\ w\ v\ e)^\star * e$
.
have $r^T * (W\ w\ v\ e)^\star * (EP\ w\ v\ e) \leq r^T * (W\ w\ v\ e)^\star * (top * e * (P\ w\ v\ e)^{T\star})$
using *10 mult-right-isotone* **by** *blast*
also have ... $= r^T * (W\ w\ v\ e)^\star * top * e * (P\ w\ v\ e)^{T\star}$
by (*simp add*: *mult-assoc*)
also have ... $\leq top * e * (P\ w\ v\ e)^{T\star}$
by (*metis comp-associative comp-inf-covector inf.idem inf.sup-right-divisibility*)
also have ... $\leq top * r^T * (W\ w\ v\ e)^\star * e * (P\ w\ v\ e)^{T\star}$
using *11* **by** (*simp add*: *mult-left-isotone*)
also have ... $= r^T * (W\ w\ v\ e)^\star * e * (P\ w\ v\ e)^{T\star}$
using *assms*(*2*) *vector-conv-covector* **by** *auto*
also have ... $\leq r^T * (W\ w\ v\ e)^\star * (W\ w\ v\ e) * (P\ w\ v\ e)^{T\star}$
by (*simp add*: *mult-left-isotone mult-right-isotone*)
also have ... $\leq r^T * (W\ w\ v\ e)^\star * (W\ w\ v\ e) * (W\ w\ v\ e)^\star$
by (*meson dual-order.trans mult-right-isotone star-isotone sup-ge1 sup-ge2*)
also have ... $\leq r^T * (W\ w\ v\ e)^\star$
by (*metis mult-assoc mult-right-isotone star.circ-transitive-equal star.left-plus-below-circ*)
finally have *12*: $r^T * (W\ w\ v\ e)^\star * (EP\ w\ v\ e) \leq r^T * (W\ w\ v\ e)^\star$
.
have $r^T * (W\ w\ v\ e)^\star * w \leq r^T * (W\ w\ v\ e)^\star * (w \sqcup EP\ w\ v\ e)$

by (*simp add*: *inf-assoc*)
also have ... $= r^T * (W\ w\ v\ e)^\star * ((w \sqcup EP\ w\ v\ e) \sqcap (-(EP\ w\ v\ e) \sqcup EP\ w\ v\ e))$
by (*metis assms*(*1*) *inf-top-right stone*)
also have ... $= r^T * (W\ w\ v\ e)^\star * ((w \sqcap -(EP\ w\ v\ e)) \sqcup EP\ w\ v\ e)$
by (*simp add*: *sup-inf-distrib2*)
also have ... $= r^T * (W\ w\ v\ e)^\star * (w \sqcap -(EP\ w\ v\ e)) \sqcup r^T * (W\ w\ v\ e)^\star * (EP\ w\ v\ e)$
by (*simp add*: *comp-left-dist-sup*)
also have ... $\leq r^T * (W\ w\ v\ e)^\star * (W\ w\ v\ e) \sqcup r^T * (W\ w\ v\ e)^\star * (EP\ w\ v\ e)$
using *mult-right-isotone sup-left-isotone* **by** *auto*
also have ... $\leq r^T * (W\ w\ v\ e)^\star \sqcup r^T * (W\ w\ v\ e)^\star * (EP\ w\ v\ e)$
using *mult-assoc mult-right-isotone star.circ-plus-same star.left-plus-below-circ sup-left-isotone* **by** *auto*
also have ... $= r^T * (W\ w\ v\ e)^\star$
using *12 sup.absorb1* **by** *blast*
finally have $r^T \sqcup r^T * (W\ w\ v\ e)^\star * w \leq r^T * (W\ w\ v\ e)^\star$
using *1* **by** *simp*
thus *?thesis*
by (*simp add*: *star-right-induct*)
qed

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 $v \Longrightarrow E\ w\ v\ e \sqcup P\ w\ v\ e = EP\ w\ v\ e$
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*:

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 $w^\star * v \leq v$
by (*simp add*: *assms*(*3*) *star-left-induct-mult*)
hence *1*: $v^T * w^{T\star} \leq v^T$
using *conv-star-commute conv-dist-comp conv-isotone* **by** *fastforce*
have $e * w^T \leq w * w^T \sqcap e * w^T$
by (*simp add*: *assms*(*1*) *mult-left-isotone*)
also have ... $\leq 1 \sqcap e * w^T$
using *assms*(*2*) *inf.sup-left-isotone* **by** *auto*
also have ... $= 1 \sqcap w * e^T$
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 $top * e * w^{T+} \leq top * v^T * w^{T\star}$
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 $P\ w\ v\ e = bot$
proof $-$
have *1*: $e = w \sqcap top * e$
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*: $-v * -v^T \sqcap top * v^T = bot$
by (*simp add*: *assms*(*5*) *comp-associative covector-vector-comp inf.sup-monoid.add-commute schroeder-2*)
have $P\ w\ v\ e = (w \sqcap -v * -v^T \sqcap top * e) \sqcup (w \sqcap -v * -v^T \sqcap top * e * w^{T+})$

by (*metis inf-sup-distrib1 mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute*)
also have ... $\leq (e \sqcap -v * -v^T) \sqcup (w \sqcap -v * -v^T \sqcap top * e * w^{T+})$
using *1* **by** (*metis comp-inf.mult-semi-associative inf.sup-monoid.add-commute semiring.add-right-mono*)
also have ... $\leq (e \sqcap -v * -v^T) \sqcup (w \sqcap -v * -v^T \sqcap top * v^T)$
using *2* **by** (*metis sup-right-isotone inf.sup-right-isotone*)
also have ... $\leq (e \sqcap -v * -v^T) \sqcup (-v * -v^T \sqcap top * v^T)$
using *inf.assoc le-infI2* **by** *auto*
also have ... $\leq v * -v^T \sqcap -v * -v^T$
using *3 assms(4) inf.sup-left-isotone* **by** *auto*
also have ... $\leq v * top \sqcap -v * top$
using *inf.sup-mono mult-right-isotone top-greatest* **by** *blast*
also have ... $= bot$
using *assms(5) inf-compl-bot vector-complement-closed* **by** *auto*
finally show *?thesis*
by (*simp add: le-iff-inf*)
qed

lemma *epm-6*:
assumes $e \leq w$
and *injective* w
and $w * v \leq v$
and $e \leq v * -v^T$
and *vector* v
shows $E\ w\ v\ e = e$
proof –
have *1*: $e \leq --v * -v^T$
using *assms(4) mult-isotone order-lesseq-imp pp-increasing* **by** *blast*
have *2*: $top * e * w^{T+} \leq top * v^T$
by (*simp add: assms(1–4) epm-4*)
have *3*: $e = w \sqcap top * e$
by (*simp add: assms(1,2) epm-3*)
hence $e \leq top * e * w^{T\star}$
by (*metis le-infI2 star.circ-back-loop-fixpoint sup.commute sup-ge1*)
hence *4*: $e \leq E\ w\ v\ e$
using *1* **by** (*simp add: assms(1)*)
have *5*: $--v * -v^T \sqcap top * v^T = bot$
by (*simp add: assms(5) comp-associative covector-vector-comp inf.sup-monoid.add-commute schroeder-2*)
have $E\ w\ v\ e = (w \sqcap --v * -v^T \sqcap top * e) \sqcup (w \sqcap --v * -v^T \sqcap top * e * w^{T+})$
by (*metis inf-sup-distrib1 mult-assoc star.circ-back-loop-fixpoint star-plus sup-commute*)
also have ... $\leq (e \sqcap --v * -v^T) \sqcup (w \sqcap --v * -v^T \sqcap top * e * w^{T+})$
using *3* **by** (*metis comp-inf.mult-semi-associative inf.sup-monoid.add-commute semiring.add-right-mono*)
also have ... $\leq (e \sqcap --v * -v^T) \sqcup (w \sqcap --v * -v^T \sqcap top * v^T)$
using *2* **by** (*metis sup-right-isotone inf.sup-right-isotone*)

also have ... $\leq (e \sqcap --v * -v^T) \sqcup (--v * -v^T \sqcap top * v^T)$
using *inf.assoc le-infI2* **by** *auto*
also have ... $\leq e$
by (*simp add*: *5*)
finally show *?thesis*
using *4* **by** (*simp add*: *antisym*)
qed

lemma *epm-7*:
regular $(EP\ w\ v\ e) \Longrightarrow e \leq w \Longrightarrow$ *injective* $w \Longrightarrow w * v \leq v \Longrightarrow e \leq v * -v^T \Longrightarrow$ *vector* $v \Longrightarrow W\ w\ v\ e = w$
by (*metis conv-bot epm-2 epm-5 epm-6*)

lemma *epm-8*:
assumes *acyclic w*
shows $(w \sqcap -(EP\ w\ v\ e)) \sqcap (P\ w\ v\ e)^T = bot$
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*)
qed

lemma *epm-9*:
assumes $e \leq v * -v^T$
and *vector v*
shows $(w \sqcap -(EP\ w\ v\ e)) \sqcap e = bot$
proof –
have *1*: $e \leq -v^T$
by (*metis assms complement-conv-sub vector-conv-covector ev p-antitone-iff p-bot*)
have $(w \sqcap -(EP\ w\ v\ e)) \sqcap e = (w \sqcap --v^T \sqcap e) \sqcup (w \sqcap -(top * e * w^{T\star}) \sqcap e)$
by (*simp add*: *inf-commute inf-sup-distrib1*)
also have ... $\leq (--v^T \sqcap e) \sqcup (-(top * e * w^{T\star}) \sqcap e)$
using *comp-inf.mult-left-isotone inf.coboundeḋ2 semiring.add-mono* **by** *blast*
also have ... $= -(top * e * w^{T\star}) \sqcap e$
using *1* **by** (*metis inf.sup-relative-same-increasing inf-commute inf-sup-distrib1 maddux-3-13 regular-closed-p*)
also have ... $= bot$
by (*metis inf.sup-relative-same-increasing inf-bot-right inf-commute inf-p mult-left-isotone star-outer-increasing top-greatest*)
finally show *?thesis*
by (*simp add*: *le-iff-inf*)
qed

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 $(P\ w\ v\ e)^T \sqcap e \leq -v * -v^T \sqcap v * -v^T$
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*)

qed

The following lemmas show that the relation characterising the edge across the cut is an atom.

lemma *atom-edge-1*:
assumes $e \leq v * -v^T \sqcap g$
and *vector* v
and $v^T = r^T * t^\star$
and $t \leq g$
and $r^T * g^\star \leq r^T * w^\star$
shows $top * e \leq v^T * w^\star$
proof –
have $top * e \leq top * (v * -v^T \sqcap g)$
using *assms(1) mult-right-isotone* **by** *auto*
also have ... $\leq top * (v * top \sqcap g)$
using *inf.sup-right-isotone inf-commute mult-right-isotone top-greatest* **by** *presburger*
also have ... $= v^T * g$
by (*metis assms(2) covector-inf-comp-3 inf-top.left-neutral*)
also have ... $= r^T * t^\star * g$
by (*simp add: assms(3)*)
also have ... $\leq r^T * g^\star * g$
by (*simp add: assms(4) mult-left-isotone mult-right-isotone star-isotone*)
also have ... $\leq r^T * g^\star$
by (*simp add: mult-assoc mult-right-isotone star.right-plus-below-circ*)
also have ... $\leq r^T * w^\star$
by (*simp add: assms(5)*)
also have ... $\leq v^T * w^\star$
by (*metis assms(3) mult-left-isotone mult-right-isotone mult-1-right star.circ-reflexive*)
finally show *?thesis*
.
qed

lemma *atom-edge-2*:
assumes $e \leq v * -v^T \sqcap g$
and *vector* v
and $v^T = r^T * t^\star$
and $t \leq g$
and $r^T * g^\star \leq r^T * w^\star$
and $w * v \leq v$
and *injective* w
shows $top * e * w^{T\star} \leq v^T * w^\star$
proof –
have *1*: $top * e \leq v^T * w^\star$
using *assms(1–5) atom-edge-1* **by** *blast*
have $v^T * w^\star * w^T = v^T * w^T \sqcup v^T * w^+ * w^T$
by (*metis mult-assoc mult-left-dist-sup star.circ-loop-fixpoint sup-commute*)
also have ... $\leq v^T \sqcup v^T * w^+ * w^T$

by (*metis assms*(*6*) *conv-dist-comp conv-isotone sup-left-isotone*)
also have ... $= v^T \sqcup v^T * w^\star * (w * w^T)$
by (*metis mult-assoc star-plus*)
also have ... $\leq v^T \sqcup v^T * w^\star$
by (*metis assms*(*7*) *mult-right-isotone mult-1-right sup-right-isotone*)
also have ... $= v^T * w^\star$
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 $e \leq v * -v^T \sqcap g$
and *vector v*
and $v^T = r^T * t^\star$
and $t \leq g$
and $r^T * g^\star \leq r^T * w^\star$
and $w * v \leq v$
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 ... $= w \sqcap --v * top \sqcap top * -v^T \sqcap top * e * w^{T\star}$
by (*metis assms*(*2*) *comp-inf-covector inf.assoc inf-top.left-neutral vector-conv-compl*)
also have ... $= w \sqcap top * e * w^{T\star} \sqcap -v^T \sqcap --v$
using *assms*(*2*) *inf.assoc inf.commute vector-conv-compl vector-complement-closed* **by** (*simp add*: *inf-assoc*)
finally have *1*: $w \sqcap top * e * w^{T\star} \sqcap -v^T \leq -v$
using *shunting-1-pp* **by** *force*
have $w^\star * e^T * top = (top * e * w^{T\star})^T$
by (*simp add*: *conv-star-commute comp-associative conv-dist-comp*)
also have ... $\leq (v^T * w^\star)^T$
using *assms*(*1*−*7*) *atom-edge-2* **by** (*simp add*: *conv-isotone*)
also have ... $= w^{T\star} * v$
by (*simp add*: *conv-star-commute conv-dist-comp*)
finally have *2*: $w^\star * e^T * top \leq w^{T\star} * v$
.
have $(w^T \sqcap w^\star * e^T * top)^T * -v = (w \sqcap top * e * w^{T\star}) * -v$
by (*simp add*: *conv-dist-comp conv-dist-inf conv-star-commute mult-assoc*)
also have ... $= (w \sqcap top * e * w^{T\star} \sqcap -v^T) * top$
by (*metis assms*(*2*) *conv-complement covector-inf-comp-3 inf-top.right-neutral vector-complement-closed*)
also have ... $\leq -v * top$
using *1* **by** (*simp add*: *comp-isotone*)
also have ... $= -v$
using *assms*(*2*) *vector-complement-closed* **by** *auto*

finally have $(w^T \sqcap w^\star * e^T * top) * --v \le --v$
  using *p-antitone-iff schroeder-3-p* **by** *auto*
hence $w^\star * e^T * top \sqcap w^T * --v \le --v$
  by (*simp add*: *inf-vector-comp*)
hence *3*: $w^T * --v \le --v \sqcup -(w^\star * e^T * top)$
  by (*simp add*: *inf.commute shunting-p*)
have $w^T * -(w^\star * e^T * top) \le -(w^\star * e^T * top)$
  by (*metis mult-assoc p-antitone p-antitone-iff schroeder-3-p star.circ-loop-fixpoint sup-commute sup-right-divisibility*)
also have ... $\le --v \sqcup -(w^\star * e^T * top)$
  by *simp*
finally have $w^T * (--v \sqcup -(w^\star * e^T * top)) \le --v \sqcup -(w^\star * e^T * top)$
  using *3* **by** (*simp add*: *mult-left-dist-sup*)
hence $w^{T\star} * (--v \sqcup -(w^\star * e^T * top)) \le --v \sqcup -(w^\star * e^T * top)$
  using *star-left-induct-mult-iff* **by** *blast*
hence $w^{T\star} * --v \le --v \sqcup -(w^\star * e^T * top)$
  by (*simp add*: *semiring.distrib-left*)
hence $w^\star * e^T * top \sqcap w^{T\star} * --v \le --v$
  by (*simp add*: *inf-commute shunting-p*)
hence $w^\star * e^T * top \le --v$
  using *2* **by** (*metis inf.absorb1 p-antitone-iff p-comp-pp vector-export-comp*)
hence *4*: $e^T * top \le --v$
  by (*metis mult-assoc star.circ-loop-fixpoint sup.bounded-iff*)
have $e^T * top \le (v * -v^T)^T * top$
  using *assms*(*1*) *comp-isotone conv-isotone* **by** *auto*
also have ... $\le -v * top$
  by (*simp add*: *conv-complement conv-dist-comp mult-assoc mult-right-isotone*)
also have ... $= -v$
  using *assms*(*2*) *vector-complement-closed* **by** *auto*
finally have $e^T * top \le bot$
  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

lemma *atom-edge-4*:
  assumes $e \le v * -v^T \sqcap g$
    and *vector v*
    and $v^T = r^T * t^\star$
    and $t \le g$
    and $r^T * g^\star \le r^T * w^\star$
    and *atom e*
  shows $top * E\ w\ v\ e * top = top$
proof −
  have $--v^T * w = (--v^T * w \sqcap -v^T) \sqcup (--v^T * w \sqcap --v^T)$
    by (*simp add*: *maddux-3-11-pp*)
  also have ... $\le (--v^T * w \sqcap -v^T) \sqcup --v^T$

using *sup-right-isotone* **by** *auto*
also have ... $= --v^T * (w \sqcap -v^T) \sqcup --v^T$
using *assms*(*2*) *covector-comp-inf covector-complement-closed vector-conv-covector* **by** *auto*
also have ... $\leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
by (*metis star.circ-back-loop-fixpoint sup.cobounded2 sup-left-isotone*)
finally have *1*: $--v^T * w \leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
.
have $--v^T * (w \sqcap -v^T) * w^\star * w \leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
by (*simp add*: *le-supI1 mult-assoc mult-right-isotone star.circ-plus-same star.left-plus-below-circ*)
hence *2*: $(--v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T) * w \leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
using *1* **by** (*simp add*: *inf.orderE mult-right-dist-sup*)
have $v^T \leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
by (*simp add*: *pp-increasing sup.coboundedI2*)
hence $v^T * w^\star \leq --v^T * (w \sqcap -v^T) * w^\star \sqcup --v^T$
using *2* **by** (*simp add*: *star-right-induct*)
hence *3*: $-v^T \sqcap v^T * w^\star \leq --v^T * (w \sqcap -v^T) * w^\star$
by (*metis inf-commute shunting-var-p*)
have $top * e = top * e \sqcap v^T * w^\star$
by (*meson assms*(*1*−*5*) *atom-edge-1 inf.orderE*)
also have ... $\leq top * v * -v^T \sqcap v^T * w^\star$
using *assms*(*1*) *inf.sup-left-isotone mult-assoc mult-right-isotone* **by** *auto*
also have ... $\leq top * -v^T \sqcap v^T * w^\star$
using *inf.sup-left-isotone mult-left-isotone top-greatest* **by** *blast*
also have ... $= -v^T \sqcap v^T * w^\star$
by (*simp add*: *assms*(*2*) *vector-conv-compl*)
also have ... $\leq --v^T * (w \sqcap -v^T) * w^\star$
using *3* **by** *simp*
also have ... $= (top \sqcap (--v)^T) * (w \sqcap -v^T) * w^\star$
by (*simp add*: *conv-complement*)
also have ... $= top * (w \sqcap --v \sqcap -v^T) * w^\star$
using *assms*(*2*) *covector-inf-comp-3 inf-assoc inf-left-commute vector-complement-closed* **by** *presburger*
also have ... $= top * (w \sqcap --v * -v^T) * w^\star$
by (*metis assms*(*2*) *vector-complement-closed conv-complement inf-assoc vector-covector*)
finally have $top * (e^T * top)^T \leq top * (w \sqcap --v * -v^T) * w^\star$
by (*metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top*)
hence $top \leq top * (w \sqcap --v * -v^T) * w^\star * (e^T * top)$
using *assms*(*6*) *shunt-bijective* **by** *blast*
also have ... $= top * (w \sqcap --v * -v^T) * (top * e * w^{\star T})^T$
by (*simp add*: *conv-dist-comp mult-assoc*)
also have ... $= top * (w \sqcap --v * -v^T \sqcap top * e * w^{\star T}) * top$
by (*simp add*: *comp-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 $w * v \leq v$
and *injective w*
and *atom e*
shows $(E\ w\ v\ e)^T * top * E\ w\ v\ e \leq 1$
proof −
have *1*: $e^T * top * e \leq 1$
by (*simp add*: *assms*(*4*) *point-injective*)
have $E\ w\ v\ e \leq --v * top$
by (*simp add*: *inf-commute le-infI2 mult-right-isotone*)
hence *2*: $E\ w\ v\ e \leq --v$
by (*simp add*: *assms*(*1*) *vector-complement-closed*)
have *3*: $w * --v \leq --v$
by (*simp add*: *assms*(*2*) *p-antitone p-antitone-iff*)
have $w \sqcap top * E\ w\ v\ e \leq w * (E\ w\ v\ e)^T * E\ w\ v\ e$
by (*metis dedekind-2 inf.commute inf-top.left-neutral*)
also have ... $\leq w * w^T * E\ w\ v\ e$
by (*simp add*: *conv-isotone le-infI1 mult-left-isotone mult-right-isotone*)
also have ... $\leq E\ w\ v\ e$
by (*metis assms*(*3*) *mult-left-isotone mult-left-one*)
finally have *4*: $w \sqcap top * E\ w\ v\ e \leq E\ w\ v\ e$
.
have $w^+ \sqcap top * E\ w\ v\ e = w^\star * (w \sqcap top * E\ w\ v\ e)$
by (*simp add*: *comp-inf-covector star-plus*)
also have ... $\leq w^\star * E\ w\ v\ e$
using *4* **by** (*simp add*: *mult-right-isotone*)
also have ... $\leq --v$
using *2 3 star-left-induct sup.bounded-iff* **by** *blast*
finally have *5*: $w^+ \sqcap top * E\ w\ v\ e \sqcap -v = bot$
using *shunting-1-pp* **by** *blast*
hence *6*: $w^{+T} \sqcap (E\ w\ v\ e)^T * top \sqcap -v^T = bot$
using *conv-complement conv-dist-comp conv-dist-inf conv-top conv-bot* **by** *force*
have $(E\ w\ v\ e)^T * top * E\ w\ v\ e \leq (top * e * w^{T\star})^T * top * (top * e * w^{T\star})$
by (*simp add*: *conv-isotone mult-isotone*)
also have ... $= w^\star * e^T * top * e * w^{T\star}$
by (*metis conv-star-commute conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top*)
also have ... $\leq w^\star * w^{T\star}$
using *1* **by** (*metis mult-assoc mult-1-right mult-right-isotone mult-left-isotone*)
also have ... $= w^\star \sqcup w^{T\star}$
by (*metis assms*(*3*) *cancel-separate inf.eq-iff star.circ-sup-sub-sup-one-1 star.circ-plus-one star-involutive*)
also have ... $= w^+ \sqcup w^{T+} \sqcup 1$
by (*metis star.circ-plus-one star-left-unfold-equal sup.assoc sup.commute*)
finally have *7*: $(E\ w\ v\ e)^T * top * E\ w\ v\ e \leq w^+ \sqcup w^{T+} \sqcup 1$
.

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*: $E\ w\ v\ e \leq -v^T$
.
hence *9*: $(E\ w\ v\ e)^T \leq -v$
by (*metis conv-complement conv-involutive conv-isotone*)
have $(E\ w\ v\ e)^T * top * E\ w\ v\ e = (w^+ \sqcup w^{T+} \sqcup 1) \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e$
using *7* **by** (*simp add*: *inf.absorb-iff2*)
also have ... $= (1 \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e) \sqcup (w^+ \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e) \sqcup (w^{T+} \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e)$
using *comp-inf.mult-right-dist-sup sup-assoc sup-commute* **by** *auto*
also have ... $\leq 1 \sqcup (w^+ \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e) \sqcup (w^{T+} \sqcap (E\ w\ v\ e)^T * top * E\ w\ v\ e)$
using *inf-le1 sup-left-isotone* **by** *blast*
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 *8 inf.sup-right-isotone 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 ... $= 1 \sqcup (w^+ \sqcap -v * top \sqcap top * E\ w\ v\ e) \sqcup (w^{T+} \sqcap (E\ w\ v\ e)^T * top \sqcap top * -v^T)$
by (*metis* (*no-types*) *vector-export-comp inf-top-right inf-assoc*)
also have ... $= 1 \sqcup (w^+ \sqcap -v \sqcap top * E\ w\ v\ e) \sqcup (w^{T+} \sqcap (E\ w\ v\ e)^T * top \sqcap -v^T)$
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 *?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 ... ≤ 1

```
    by (simp add: assms(3))
  finally have 1: E w v e * 1 * (E w v e)^T ≤ 1
    .
  have (E w v e)^T * top * E w v e ≤ 1
    by (simp add: assms atom-edge-5)
  also have ... ≤ --1
    by (simp add: pp-increasing)
  finally have 2: E w v e * -1 * (E w v e)^T ≤ bot
    by (metis conv-involutive regular-closed-bot regular-dense-top
triple-schroeder-p)
  have E w v e * top * (E w v e)^T = E w v e * 1 * (E w v e)^T ⊔ E w v e * -1 *
(E w v e)^T
    by (metis mult-left-dist-sup mult-right-dist-sup regular-complement-top
regular-one-closed)
  also have ... ≤ 1
    using 1 2 by (simp add: bot-unique)
  finally show ?thesis
    .
qed

lemma atom-edge:
  assumes e ≤ v * -v^T ⊓ g
      and vector v
      and v^T = r^T * t^⋆
      and t ≤ g
      and r^T * g^⋆ ≤ r^T * w^⋆
      and w * v ≤ v
      and injective w
      and atom e
    shows atom (E w v e)
proof (intro conjI)
  have E w v e * top * (E w v e)^T ≤ 1
    using assms(2,6-8) atom-edge-6 by simp
  thus injective (E w v e * top)
    by (metis conv-dist-comp conv-top mult-assoc top-mult-top)
next
  show surjective (E w v e * top)
    using assms(1-5,8) atom-edge-4 mult-assoc by simp
next
  have (E w v e)^T * top * E w v e ≤ 1
    using assms(2,6-8) atom-edge-5 by simp
  thus injective ((E w v e)^T * top)
    by (metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top)
next
  have top * E w v e * top = top
    using assms(1-5,8) atom-edge-4 by simp
  thus surjective ((E w v e)^T * 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 $v^T = r^T * t^\star$
    and $v * -v^T \sqcap g = bot$
    and $t \leq v * v^T \sqcap g$
    and $r^T * (v * v^T \sqcap g)^\star \leq r^T * t^\star$
   shows $v^T = r^T * g^\star$
proof –
  let $?vv = v * v^T \sqcap g$
  have *1*: $r^T \leq v^T$
   using *assms*(*3*) *mult-right-isotone mult-1-right star.circ-reflexive* **by** *fastforce*
  have $v * top \sqcap g = (v * v^T \sqcup v * -v^T) \sqcap g$
   by (*metis assms*(*1*) *conv-complement mult-left-dist-sup regular-complement-top*)
  also have ... $= ?vv \sqcup (v * -v^T \sqcap g)$
   by (*simp add*: *inf-sup-distrib2*)
  also have ... $= ?vv$
   by (*simp add*: *assms*(*4*))
  finally have *2*: $v * top \sqcap g = ?vv$
   by *simp*
  have $r^T * ?vv^\star \leq v^T * ?vv^\star$
   using *1* **by** (*simp add*: *comp-left-isotone*)
  also have ... $\leq v^T * (v * v^T)^\star$
   by (*simp add*: *comp-right-isotone star.circ-isotone*)
  also have ... $\leq v^T$
   by (*simp add*: *assms*(*2*) *vector-star-1*)
  finally have $r^T * ?vv^\star \leq v^T$
   by *simp*
  hence $r^T * ?vv^\star * g = (r^T * ?vv^\star \sqcap v^T) * g$
   by (*simp add*: *inf.absorb1*)
  also have ... $= r^T * ?vv^\star * (v * top \sqcap g)$
   by (*simp add*: *assms*(*2*) *covector-inf-comp-3*)
  also have ... $= r^T * ?vv^\star * ?vv$
   using *2* **by** *simp*
  also have ... $\leq r^T * ?vv^\star$
   by (*simp add*: *comp-associative comp-right-isotone star.left-plus-below-circ star-plus*)
  finally have $r^T \sqcup r^T * ?vv^\star * g \leq r^T * ?vv^\star$
   using *star.circ-back-loop-prefixpoint* **by** *auto*
  hence $r^T * g^\star \leq r^T * ?vv^\star$
   using *star-right-induct* **by** *blast*
  hence $r^T * g^\star = r^T * ?vv^\star$

by (*simp add*: *antisym mult-right-isotone star-isotone*)
also have ... $= r^T * t^\star$
using *assms*(*5*,*6*) *antisym mult-right-isotone star-isotone* **by** *auto*
also have ... $= v^T$
by (*simp add*: *assms*(*3*))
finally show *?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 *vector r*
and *injective r*
and $v^T = r^T * t^\star$
and *forest w*
and $t \leq w$
and $w \leq v * v^T$
shows $w = t$
proof −
have *1*: *vector v*
using *assms*(*1*,*3*) *covector-mult-closed vector-conv-covector* **by** *auto*
have $w * v \leq v * v^T * v$
by (*simp add*: *assms*(*6*) *mult-left-isotone*)
also have ... $\leq v$
using *1* **by** (*metis mult-assoc mult-right-isotone top-greatest*)
finally have *2*: $w * v \leq v$
.
have *3*: $r \leq v$
by (*metis assms*(*3*) *conv-order mult-right-isotone mult-1-right star.circ-reflexive*)
have *4*: $v \sqcap -r = t^{T\star} * r \sqcap -r$
by (*metis assms*(*3*) *conv-dist-comp conv-involutive conv-star-commute*)
also have ... $= (r \sqcup t^{T+} * r) \sqcap -r$
using *mult-assoc star.circ-loop-fixpoint sup-commute* **by** *auto*
also have ... $\leq t^{T+} * r$
by (*simp add*: *shunting*)
also have ... $\leq t^T * top$
by (*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 *conv-order inf.sup-right-isotone mult-isotone* **by** *auto*
also have ... $= 1 \sqcap t^T * top * t$
by (*metis conv-dist-comp conv-involutive conv-top mult-assoc top-mult-top*)
also have ... $\leq t^T * (top * t \sqcap t * 1)$
by (*metis conv-involutive dedekind-1 inf.commute mult-assoc*)
also have ... $\leq t^T * t$
by (*simp add*: *mult-right-isotone*)
finally have *5*: $1 \sqcap (v \sqcap -r) * (v \sqcap -r)^T \leq t^T * t$
.

have $w * w^+ \leq -1$
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 $w * r \sqcap w^{T+} * r = (w \sqcap w^{T+}) * r$
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*: $w * r \sqcap w^{T+} * r = bot$
.
have $-1 * r \leq -r$
using *assms*(*2*) *schroeder-4-p* **by** *force*
hence $-1 * r * top \leq -r$
by (*simp add*: *assms*(*1*) *comp-associative*)
hence *8*: $r^T * -1 * r \leq bot$
by (*simp add*: *mult-assoc schroeder-6-p*)
have $r^T * w * r \leq r^T * w^+ * r$
by (*simp add*: *mult-left-isotone mult-right-isotone star.circ-mult-increasing*)
also have ... $\leq r^T * -1 * r$
by (*simp add*: *assms*(*4*) *comp-isotone*)
finally have $r^T * w * r \leq bot$
using *8* **by** *simp*
hence $w * r * top \leq -r$
by (*simp add*: *mult-assoc schroeder-6-p*)
hence $w * r \leq -r$
by (*simp add*: *assms*(*1*) *comp-associative*)
hence $w * r \leq -r \sqcap w * v$
using *3* **by** (*simp add*: *mult-right-isotone*)
also have ... $\leq -r \sqcap v$
using *2* **by** (*simp add*: *le-infI2*)
also have ... $= -r \sqcap t^{T\star} * r$
using *4* **by** (*simp add*: *inf-commute*)
also have ... $\leq -r \sqcap w^{T\star} * r$
using *assms*(*5*) *comp-inf.mult-right-isotone conv-isotone mult-left-isotone star-isotone* **by** *auto*
also have ... $= -r \sqcap (r \sqcup w^{T+} * r)$
using *mult-assoc star.circ-loop-fixpoint sup-commute* **by** *auto*
also have ... $\leq w^{T+} * r$
using *inf.commute maddux-3-13* **by** *auto*
finally have $w * r = bot$
using *7* **by** (*simp add*: *le-iff-inf*)
hence $w = w \sqcap top * -r^T$
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 ... $= w \sqcap v * v^T \sqcap top * -r^T$
by (*simp add*: *assms*(*6*) *inf-absorb1*)
also have ... $\leq w \sqcap top * v^T \sqcap top * -r^T$
using *comp-inf.mult-left-isotone comp-inf.mult-right-isotone mult-left-isotone*

```
by auto
  also have ... = w ⊓ top * (v^T ⊓ -r^T)
    using 1 assms(1) covector-inf-closed inf-assoc vector-conv-compl
vector-conv-covector by auto
  also have ... = w * (1 ⊓ (v ⊓ -r) * top)
    by (simp add: comp-inf-vector conv-complement conv-dist-inf)
  also have ... = w * (1 ⊓ (v ⊓ -r) * (v ⊓ -r)^T)
    by (metis conv-top dedekind-eq inf-commute inf-top-left mult-1-left
mult-1-right)
  also have ... ≤ w * t^T * t
    using 5 by (simp add: comp-isotone mult-assoc)
  also have ... ≤ w * w^T * t
    by (simp add: assms(5) comp-isotone conv-isotone)
  also have ... ≤ 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 =
λx . 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 unfold-locales
  apply simp
  apply (simp add: min.coboundedI1 min.commute)
  by (simp add: min.coboundedI1)

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

end
```

A Kleene relation algebra is based on a relation algebra.

class *kleene-relation-algebra* = *relation-algebra* + *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*

imports *Stone-Relation-Algebras.Relation-Subalgebras Kleene-Relation-Algebras*

begin

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

lift-definition *star-regular* :: *'a regular* ⇒ *'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*

imports *Stone-Relation-Algebras.Matrix-Relation-Algebras*
Kleene-Relation-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* ⇒ (*'a*,*'b*::*bot*) *square* ⇒ *'a list* ⇒ (*'a*,*'b*) *square* (- ⟨-⟩ - [*90*,*41*,*90*] *91*)
  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* ⇒ (*'a*,*'b*::{*star*,*times*,*bounded-semilattice-sup-bot*}) *square* ⇒ (*'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* ($(-^{\odot}$ *[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*⟩*ns* = *mbot*
proof (*rule ext*, *rule prod-cases*)
fix *i j*

```
  have (ks⟨f⟩ls ⊙ ms⟨g⟩ns) (i,j) = (⨆k (ks⟨f⟩ls) (i,k) * (ms⟨g⟩ns) (k,j))
    by (simp add: times-matrix-def)
  also have ... = (⨆k (if List.member ks i ∧ List.member ls k then f (i,k) else
bot) * (if List.member ms k ∧ List.member ns j then g (k,j) else bot))
    by (simp add: restrict-matrix-def)
  also have ... = (⨆k if List.member ms k ∧ List.member ns j then bot * g (k,j)
else (if List.member ks i ∧ List.member ls k then f (i,k) else bot) * bot)
    using assms by (auto intro: sup-monoid.sum.cong)
  also have ... = (⨆(k::'a) bot)
    by (simp add: sup-monoid.sum.neutral)
  also have ... = bot
    by (simp add: eq-iff le-funI)
  also have ... = mbot (i,j)
    by (simp add: bot-matrix-def)
  finally show (ks⟨f⟩ls ⊙ ms⟨g⟩ns) (i,j) = mbot (i,j)
    .
qed

lemma one-disjoint:
  assumes disjoint ks ls
    shows ks⟨(mone::('a,'b::idempotent-semiring) square)⟩ls = mbot
proof (rule ext, rule prod-cases)
  let ?o = mone::('a,'b) square
  fix i j
  have (ks⟨?o⟩ls) (i,j) = (if List.member ks i ∧ List.member ls j then if i = j
then 1 else bot else bot)
    by (simp add: restrict-matrix-def one-matrix-def)
  also have ... = bot
    using assms by auto
  also have ... = mbot (i,j)
    by (simp add: bot-matrix-def)
  finally show (ks⟨?o⟩ls) (i,j) = 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 ≡ ∀ x . List.member ks x ⟶ List.member ls x

lemma restrict-sublist:
  assumes is-sublist ls ks
      and is-sublist ms ns
    shows ls⟨ks⟨f⟩ns⟩ms = ls⟨f⟩ms
proof (rule ext, rule prod-cases)
  fix i j
  show (ls⟨ks⟨f⟩ns⟩ms) (i,j) = (ls⟨f⟩ms) (i,j)
  proof (cases List.member ls i ∧ List.member ms j)
    case True thus ?thesis
      by (simp add: assms restrict-matrix-def)
```

next
case *False* **thus** *?thesis*
by (*unfold restrict-matrix-def*) *auto*
qed
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 $(ks\langle ls\langle f\rangle ms\rangle ns)\ (i,j) = (ls\langle f\rangle ms)\ (i,j)$
proof (*cases List.member ls i* ∧ *List.member ms j*)
case *True* **thus** *?thesis*
by (*simp add*: *assms restrict-matrix-def*)
next
case *False* **thus** *?thesis*
by (*unfold restrict-matrix-def*) *auto*
qed
qed

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

lemma *restrict-sup*:
fixes *f g* :: ('*a*,'*b*::*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* :: ('*a*,'*b*::*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 ks i* ∧ *List.member ms j then* ($\bigsqcup_k$ $(ks\langle f\rangle ls)\ (i,k) * (ls\langle g\rangle ms)\ (k,j)$) *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 ks i* ∧ *List.member ls k then f* (*i*,*k*) *else bot*) ∗ (*if List.member ls k* ∧ *List.member ms j then g* (*k*,*j*) *else bot*)) *else bot*)
by (*simp add*: *restrict-matrix-def*)
also have ... = (*if List.member ks i* ∧ *List.member ms j then* ($\bigsqcup_k$ *if List.member ls k then f* (*i*,*k*) ∗ *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* (*if List.member ls k then f* (*i*,*k*) ∗ *g* (*k*,*j*) *else bot*) *else bot*)
by *auto*
also have ... = ($\bigsqcup_k$ (*if List.member ks i* ∧ *List.member ls k then f* (*i*,*k*) *else bot*) ∗ (*if List.member ls k* ∧ *List.member ms j then g* (*k*,*j*) *else bot*))

```
    by (auto intro: sup-monoid.sum.cong)
  also have ... = (⨆k (ks⟨f⟩ls) (i,k) ∗ (ls⟨g⟩ms) (k,j))
    by (simp add: restrict-matrix-def)
  also have ... = (ks⟨f⟩ls ⊙ ls⟨g⟩ms) (i,j)
    by (simp add: times-matrix-def)
  finally show (ks⟨(ks⟨f⟩ls ⊙ ls⟨g⟩ms)⟩ms) (i,j) = (ks⟨f⟩ls ⊙ ls⟨g⟩ms) (i,j)
    .
qed

lemma restrict-star:
  fixes g :: ('a,'b::kleene-algebra) square
  shows t⟨star-matrix' t g⟩t = star-matrix' t 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#s
  assume ⋀g::('a,'b) square . s⟨star-matrix' s g⟩s = star-matrix' s g
  hence 1: ⋀g::('a,'b) square . ?t⟨star-matrix' s g⟩?t = star-matrix' s g
    by (metis member-rec(1) restrict-superlist)
  show ?t⟨star-matrix' ?t g⟩?t = star-matrix' ?t g
  proof −
    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⟨star o ?a⟩?r
    let ?ds = star-matrix' s ?d
    let ?e = ?a ⊕ ?b ⊙ ?ds ⊙ ?c
    let ?es = ?r⟨star o ?e⟩?r
    let ?f = ?d ⊕ ?c ⊙ ?as ⊙ ?b
    let ?fs = star-matrix' s ?f
    have 2: ?t⟨?as⟩?t = ?as ∧ ?t⟨?b⟩?t = ?b ∧ ?t⟨?c⟩?t = ?c ∧ ?t⟨?es⟩?t = ?es
      by (simp add: restrict-superlist member-def)
    have 3: ?t⟨?ds⟩?t = ?ds ∧ ?t⟨?fs⟩?t = ?fs
      using 1 by simp
    have 4: ?t⟨?t⟨?as⟩?t ⊙ ?t⟨?b⟩?t ⊙ ?t⟨?fs⟩?t⟩?t = ?t⟨?as⟩?t ⊙ ?t⟨?b⟩?t ⊙
?t⟨?fs⟩?t
      by (metis (no-types) restrict-times)
    have 5: ?t⟨?t⟨?ds⟩?t ⊙ ?t⟨?c⟩?t ⊙ ?t⟨?es⟩?t⟩?t = ?t⟨?ds⟩?t ⊙ ?t⟨?c⟩?t ⊙
?t⟨?es⟩?t
      by (metis (no-types) restrict-times)
    have ?t⟨star-matrix' ?t g⟩?t = ?t⟨?es ⊕ ?as ⊙ ?b ⊙ ?fs ⊕ ?ds ⊙ ?c ⊙ ?es ⊕
?fs⟩?t
      by (metis star-matrix'.simps(2))
    also have ... = ?t⟨?es⟩?t ⊕ ?t⟨?as ⊙ ?b ⊙ ?fs⟩?t ⊕ ?t⟨?ds ⊙ ?c ⊙ ?es⟩?t ⊕
?t⟨?fs⟩?t
```

by (*simp add*: *restrict-sup*)
also have ... = *?es* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?fs*
using *2 3 4 5* **by** *simp*
also have ... = *star-matrix' ?t g*
by (*metis star-matrix'.simps*(*2*))
finally show *?thesis*
.
qed
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*) = ($\bigsqcup_k$ (*ks*⟨*?o*⟩*ks*) (*i*,*k*) ∗ (*ks*⟨*f*⟩*ls*) (*k*,*j*))
by (*simp add*: *times-matrix-def*)
also have ... = ($\bigsqcup_k$ (*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 ... = ($\bigsqcup_k$ (*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 (*unfold one-matrix-def*) *auto*
also have ... = ($\bigsqcup_k$ (*if i* = *k then* (*if List.member ks i then 1 else bot*) *else bot*) ∗ (*if List.member ks k* ∧ *List.member ls j then f* (*k*,*j*) *else bot*))
by (*auto intro*: *sup-monoid.sum.cong*)
also have ... = ($\bigsqcup_k$ *if i* = *k then* (*if List.member ks i then 1 else bot*) ∗ (*if List.member ks i* ∧ *List.member ls j then f* (*i*,*j*) *else bot*) *else bot*)
by (*rule sup-monoid.sum.cong*) *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-monoid.sum.delta'*)
also have ... = (*if List.member ks i* ∧ *List.member ls j then f* (*i*,*j*) *else bot*)
by *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*)
.
qed

The following lemmas consider restrictions to singleton index sets.

lemma *restrict-singleton*:

$([k]\langle f\rangle[l])\ (i,j) = (\textit{if } i = k \wedge j = l \textit{ then } f\ (i,j) \textit{ else bot})$
by (*simp add*: *restrict-matrix-def List.member-def*)

lemma *restrict-singleton-list*:
$([k]\langle f\rangle ls)\ (i,j) = (\textit{if } i = k \wedge \textit{List.member ls j then } f\ (i,j) \textit{ else bot})$
by (*simp add*: *restrict-matrix-def List.member-def*)

lemma *restrict-list-singleton*:
$(ks\langle f\rangle[l])\ (i,j) = (\textit{if List.member ks i} \wedge j = l \textit{ then } f\ (i,j) \textit{ else bot})$
by (*simp add*: *restrict-matrix-def List.member-def*)

lemma *restrict-singleton-product*:
fixes *f g* :: (*'a::finite,'b::kleene-algebra*) *square*
shows $([k]\langle f\rangle[l] \odot [m]\langle g\rangle[n])\ (i,j) = (\textit{if } i = k \wedge l = m \wedge j = n \textit{ then } f\ (i,l) * g\ (m,j) \textit{ else bot})$
proof –
have $([k]\langle f\rangle[l] \odot [m]\langle g\rangle[n])\ (i,j) = (\bigsqcup_h ([k]\langle f\rangle[l])\ (i,h) * ([m]\langle g\rangle[n])\ (h,j))$
by (*simp add*: *times-matrix-def*)
also have ... $= (\bigsqcup_h (\textit{if } i = k \wedge h = l \textit{ then } f\ (i,h) \textit{ else bot}) * (\textit{if } h = m \wedge j = n \textit{ then } g\ (h,j) \textit{ else bot}))$
by (*simp add*: *restrict-singleton*)
also have ... $= (\bigsqcup_h \textit{if } h = l \textit{ then } (\textit{if } i = k \textit{ then } f\ (i,h) \textit{ else bot}) * (\textit{if } h = m \wedge j = n \textit{ then } g\ (h,j) \textit{ else bot}) \textit{ else bot})$
by (*rule sup-monoid.sum.cong*) *auto*
also have ... $= (\textit{if } i = k \textit{ then } f\ (i,l) \textit{ else bot}) * (\textit{if } l = m \wedge j = n \textit{ then } g\ (l,j) \textit{ else bot})$
by (*simp add*: *sup-monoid.sum.delta*)
also have ... $= (\textit{if } i = k \wedge l = m \wedge j = n \textit{ then } f\ (i,l) * g\ (m,j) \textit{ else bot})$
by *simp*
finally show *?thesis*
.
qed

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

lemma *restrict-star-unfold*:
$[l]\langle(\textit{mone}::(\textit{'a::finite,'b::kleene-algebra}) \textit{ square})\rangle[l] \oplus [l]\langle f\rangle[l] \odot [l]\langle \textit{star o } f\rangle[l] = [l]\langle \textit{star o } f\rangle[l]$
proof (*rule ext*, *rule prod-cases*)
let $?o = \textit{mone}::(\textit{'a,'b::kleene-algebra}) \textit{ square}$
fix *i j*
have $([l]\langle ?o\rangle[l] \oplus [l]\langle f\rangle[l] \odot [l]\langle \textit{star o } f\rangle[l])\ (i,j) = ([l]\langle ?o\rangle[l])\ (i,j) \sqcup ([l]\langle f\rangle[l] \odot [l]\langle \textit{star o } f\rangle[l])\ (i,j)$
by (*simp add*: *sup-matrix-def*)
also have ... $= ([l]\langle ?o\rangle[l])\ (i,j) \sqcup (\bigsqcup_k ([l]\langle f\rangle[l])\ (i,k) * ([l]\langle \textit{star o } f\rangle[l])\ (k,j))$
by (*simp add*: *times-matrix-def*)
also have ... $= ([l]\langle ?o\rangle[l])\ (i,j) \sqcup (\bigsqcup_k (\textit{if } i = l \wedge k = l \textit{ then } f\ (i,k) \textit{ else bot}) * (\textit{if } k = l \wedge j = l \textit{ then } (f\ (k,j))^\star \textit{ else bot}))$
by (*simp add*: *restrict-singleton o-def*)

also have ... = ([l]⟨?o⟩[l]) (i,j) ⊔ (⨆$_k$ *if k = l then (if i = l then f (i,k) else bot) ∗ (if j = l then (f (k,j))** *else bot) else bot*)
apply (*rule arg-cong2*[**where** *f=sup*])
apply *simp*
by (*rule sup-monoid.sum.cong*) *auto*
also have ... = ([l]⟨?o⟩[l]) (i,j) ⊔ (*if i = l then f (i,l) else bot*) ∗ (*if j = l then (f (l,j))** *else bot*)
by (*simp add*: *sup-monoid.sum.delta*)
also have ... = (*if i = l ∧ j = l then 1 ⊔ f (l,l) ∗ (f (l,l))** *else bot*)
by (*simp add*: *restrict-singleton one-matrix-def*)
also have ... = (*if i = l ∧ j = l then (f (l,l))** *else bot*)
by (*simp add*: *star-left-unfold-equal*)
also have ... = ([l]⟨*star o f*⟩[l]) (i,j)
by (*simp add*: *restrict-singleton o-def*)
finally show ([l]⟨?o⟩[l] ⊕ [l]⟨f⟩[l] ⊙ [l]⟨*star o f*⟩[l]) (i,j) = ([l]⟨*star o f*⟩[l]) (i,j)
.
qed

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 (k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms) = ([k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ ([k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms)
proof −
have (k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms) = ([k]⟨f⟩[l] ⊕ [k]⟨f⟩ls ⊕ ks⟨f⟩[l] ⊕ ks⟨f⟩ls) ⊙ ([l]⟨g⟩[m] ⊕ [l]⟨g⟩ms ⊕ ls⟨g⟩[m] ⊕ ls⟨g⟩ms)
by (*metis restrict-nonempty*)
also have ... = [k]⟨f⟩[l] ⊙ ([l]⟨g⟩[m] ⊕ [l]⟨g⟩ms ⊕ ls⟨g⟩[m] ⊕ ls⟨g⟩ms) ⊕ [k]⟨f⟩ls ⊙ ([l]⟨g⟩[m] ⊕ [l]⟨g⟩ms ⊕ ls⟨g⟩[m] ⊕ ls⟨g⟩ms) ⊕ ks⟨f⟩[l] ⊙ ([l]⟨g⟩[m] ⊕ [l]⟨g⟩ms ⊕ ls⟨g⟩[m] ⊕ ls⟨g⟩ms) ⊕ ks⟨f⟩ls ⊙ ([l]⟨g⟩[m] ⊕ [l]⟨g⟩ms ⊕ ls⟨g⟩[m] ⊕ ls⟨g⟩ms)
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... = ([k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩[l] ⊙ ls⟨g⟩[m] ⊕ [k]⟨f⟩[l] ⊙ ls⟨g⟩ms) ⊕ ([k]⟨f⟩ls ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩[l] ⊙ ls⟨g⟩[m] ⊕ ks⟨f⟩[l] ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩ls ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms)
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup*)
also have ... = ([k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms) ⊕ ([k]⟨f⟩ls ⊙ ls⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩[l] ⊙ [l]⟨g⟩ms) ⊕ (ks⟨f⟩ls ⊙ ls⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms)
using *assms* **by** (*simp add*: *List.member-def times-disjoint*)

also have ... = $([k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus ([k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms)$
by (*simp add*: *matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc matrix-semilattice-sup.sup-left-commute*)
finally show *?thesis*
.
qed

Equality of matrices is componentwise.

lemma *restrict-nonempty-eq*:
$(k\#ks)\langle f\rangle(l\#ls) = (k\#ks)\langle g\rangle(l\#ls) \longleftrightarrow [k]\langle f\rangle[l] = [k]\langle g\rangle[l] \wedge [k]\langle f\rangle ls = [k]\langle g\rangle ls \wedge ks\langle f\rangle[l] = ks\langle g\rangle[l] \wedge ks\langle f\rangle ls = ks\langle g\rangle ls$
proof
assume *1*: $(k\#ks)\langle f\rangle(l\#ls) = (k\#ks)\langle g\rangle(l\#ls)$
have *2*: *is-sublist* $[k]$ $(k\#ks)$ $\wedge$ *is-sublist ks* $(k\#ks)$ $\wedge$ *is-sublist* $[l]$ $(l\#ls)$ $\wedge$ *is-sublist ls* $(l\#ls)$
by (*simp add*: *member-rec*)
hence $[k]\langle f\rangle[l] = [k]\langle(k\#ks)\langle f\rangle(l\#ls)\rangle[l] \wedge [k]\langle f\rangle ls = [k]\langle(k\#ks)\langle f\rangle(l\#ls)\rangle ls \wedge ks\langle f\rangle[l] = ks\langle(k\#ks)\langle f\rangle(l\#ls)\rangle[l] \wedge ks\langle f\rangle ls = ks\langle(k\#ks)\langle f\rangle(l\#ls)\rangle ls$
by (*simp add*: *restrict-sublist*)
thus $[k]\langle f\rangle[l] = [k]\langle g\rangle[l] \wedge [k]\langle f\rangle ls = [k]\langle g\rangle ls \wedge ks\langle f\rangle[l] = ks\langle g\rangle[l] \wedge ks\langle f\rangle ls = ks\langle g\rangle ls$
using *1 2* **by** (*simp add*: *restrict-sublist*)
next
assume *3*: $[k]\langle f\rangle[l] = [k]\langle g\rangle[l] \wedge [k]\langle f\rangle ls = [k]\langle g\rangle ls \wedge ks\langle f\rangle[l] = ks\langle g\rangle[l] \wedge ks\langle f\rangle ls = ks\langle g\rangle ls$
show $(k\#ks)\langle f\rangle(l\#ls) = (k\#ks)\langle g\rangle(l\#ls)$
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*: *List.member ls j* $\Longrightarrow$ *f* (*k*,*j*) = *g* (*k*,*j*)
using *3* **by** (*metis restrict-singleton-list*)
have *6*: *List.member ks i* $\Longrightarrow$ *f* (*i*,*l*) = *g* (*i*,*l*)
using *3* **by** (*metis restrict-list-singleton*)
have $(ks\langle f\rangle ls)$ (*i*,*j*) = $(ks\langle g\rangle ls)$ (*i*,*j*)
using *3* **by** *simp*
hence *7*: *List.member ks i* $\Longrightarrow$ *List.member ls j* $\Longrightarrow$ *f* (*i*,*j*) = *g* (*i*,*j*)
by (*simp add*: *restrict-matrix-def*)
have $((k\#ks)\langle f\rangle(l\#ls))$ (*i*,*j*) = (*if* (*i* = *k* $\vee$ *List.member ks i*) $\wedge$ (*j* = *l* $\vee$ *List.member ls j*) *then f* (*i*,*j*) *else bot*)
by (*simp add*: *restrict-matrix-def List.member-def*)
also have ... = (*if i* = *k* $\wedge$ *j* = *l then f* (*i*,*j*) *else if i* = *k* $\wedge$ *List.member ls j then f* (*i*,*j*) *else if List.member ks i* $\wedge$ *j* = *l then f* (*i*,*j*) *else if List.member ks i* $\wedge$ *List.member ls j then f* (*i*,*j*) *else bot*)
by *auto*
also have ... = (*if i* = *k* $\wedge$ *j* = *l then g* (*i*,*j*) *else if i* = *k* $\wedge$ *List.member ls j then g* (*i*,*j*) *else if List.member ks i* $\wedge$ *j* = *l then g* (*i*,*j*) *else if List.member ks i*

∧ *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* ∧ *ks*⟨*f*⟩[*l*] ≼ *ks*⟨*g*⟩[*l*] ∧ *ks*⟨*f*⟩*ls* ≼ *ks*⟨*g*⟩*ls*
by (*unfold matrix-semilattice-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 $\neg$ *List.member ks k*
and $\neg$ *List.member ls l*
and $\neg$ *List.member ms m*
shows $(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms) = (k\#ks)\langle h\rangle(m\#ms) \longleftrightarrow [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m] = [k]\langle h\rangle[m] \wedge [k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms = [k]\langle h\rangle ms \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m] = ks\langle h\rangle[m] \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms = ks\langle h\rangle ms$
proof −
have *1*: *disjoint* $[k]$ *ks* $\wedge$ *disjoint* $[m]$ *ms*
by (*simp add*: *assms*(*1*,*3*) *member-rec*)
have *2*: $[k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle[m] = [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]$
proof −
have $[k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle[m] = [k]\langle([k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus ([k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms)\rangle[m]$
by (*simp add*: *assms*(*2*) *restrict-nonempty-product*)
also have ... $= [k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle[m] \oplus [k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle[m] \oplus [k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle[m] \oplus [k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle ms\rangle[m] \oplus [k]\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle[m] \oplus [k]\langle ks\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle[m] \oplus [k]\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle[m] \oplus [k]\langle ks\langle f\rangle ls \odot ls\langle g\rangle ms\rangle[m]$
by (*simp add*: *matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup*)
also have ... $= [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m] \oplus [k]\langle[k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle ms\rangle[m] \oplus [k]\langle[k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle ms\rangle ms\rangle[m] \oplus [k]\langle ks\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle[m]\rangle[m] \oplus [k]\langle ks\langle ks\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle[m]\rangle[m] \oplus [k]\langle ks\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle ms\rangle[m] \oplus [k]\langle ks\langle ks\langle f\rangle ls \odot ls\langle g\rangle ms\rangle ms\rangle[m]$
by (*simp add*: *restrict-times*)
also have ... $= [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]$
using *1* **by** (*metis restrict-disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right*)
finally show *?thesis*
.
qed
have *3*: $[k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle ms = [k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms$
proof −
have $[k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle ms = [k]\langle([k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus ([k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms)\rangle ms$
by (*simp add*: *assms*(*2*) *restrict-nonempty-product*)
also have ... $= [k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle ms \oplus [k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle ms \oplus [k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle ms \oplus [k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle ms\rangle ms \oplus [k]\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle ms \oplus [k]\langle ks\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle ms \oplus [k]\langle ks\langle f\rangle[l] \odot [l]\langle g\rangle ms\rangle ms \oplus [k]\langle ks\langle f\rangle ls \odot ls\langle g\rangle ms\rangle ms$
by (*simp add*: *matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup*)
also have ... $= [k]\langle[k]\langle[k]\langle f\rangle[l] \odot [l]\langle g\rangle[m]\rangle[m]\rangle ms \oplus [k]\langle[k]\langle[k]\langle f\rangle ls \odot ls\langle g\rangle[m]\rangle[m]\rangle ms \oplus [k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms \oplus [k]\langle ks\langle ks\langle f\rangle[l] \odot$

[l]⟨g⟩[m]⟩[m]⟩ms ⊕ [k]⟨ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩[m]⟩[m]⟩ms ⊕ [k]⟨ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms⟩ms ⊕ [k]⟨ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms⟩ms
    by (*simp add*: *restrict-times*)
  also have ... = [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms
    using *1* **by** (*metis restrict-disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left*)
  finally show *?thesis*
    .
qed
have *4*: ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]
proof –
  have ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = ks⟨([k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ ([k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms)⟩[m]
    by (*simp add*: *assms*(*2*) *restrict-nonempty-product*)
  also have ... = ks⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩[m] ⊕ ks⟨[k]⟨f⟩ls ⊙ ls⟨g⟩[m]⟩[m] ⊕ ks⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩[m] ⊕ ks⟨[k]⟨f⟩ls ⊙ ls⟨g⟩ms⟩[m] ⊕ ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩[m] ⊕ ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩[m]⟩[m] ⊕ ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩[m] ⊕ ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩ms⟩[m]
    by (*simp add*: *matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup*)
  also have ... = ks⟨[k]⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩[m]⟩[m] ⊕ ks⟨[k]⟨[k]⟨f⟩ls ⊙ ls⟨g⟩[m]⟩[m]⟩[m] ⊕ ks⟨[k]⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms⟩[m] ⊕ ks⟨[k]⟨[k]⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms⟩[m] ⊕ ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m] ⊕ ks⟨ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms⟩[m] ⊕ ks⟨ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms⟩[m]
    by (*simp add*: *restrict-times*)
  also have ... = ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]
    using *1* **by** (*metis restrict-disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left*)
  finally show *?thesis*
    .
qed
have *5*: ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms
proof –
  have ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = ks⟨([k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ ([k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]) ⊕ (ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms)⟩ms
    by (*simp add*: *assms*(*2*) *restrict-nonempty-product*)
  also have ... = ks⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩ms ⊕ ks⟨[k]⟨f⟩ls ⊙ ls⟨g⟩[m]⟩ms ⊕ ks⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms ⊕ ks⟨[k]⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms ⊕ ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩ms ⊕ ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩[m]⟩ms ⊕ ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms ⊕ ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms
    by (*simp add*: *matrix-bounded-semilattice-sup-bot.sup-monoid.add-assoc restrict-sup*)
  also have ... = ks⟨[k]⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩[m]⟩ms ⊕ ks⟨[k]⟨[k]⟨f⟩ls ⊙

ls⟨g⟩[m]⟩[m]⟩ms ⊕ ks⟨[k]⟨[k]⟨f⟩[l] ⊙ [l]⟨g⟩ms⟩ms⟩ms ⊕ ks⟨[k]⟨[k]⟨f⟩ls ⊙ ls⟨g⟩ms⟩ms⟩ms ⊕ ks⟨ks⟨ks⟨f⟩[l] ⊙ [l]⟨g⟩[m]⟩[m]⟩ms ⊕ ks⟨ks⟨ks⟨f⟩ls ⊙ ls⟨g⟩[m]⟩[m]⟩ms ⊕ ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms
by (*simp add*: *restrict-times*)
also have ... = ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms
using *1* **by** (*metis restrict-disjoint-left restrict-disjoint-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left*)
finally show *?thesis*
.
qed
have (k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms) = (k#ks)⟨h⟩(m#ms) ⟷ (k#ks)⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩(m#ms) = (k#ks)⟨h⟩(m#ms)
by (*simp add*: *restrict-times*)
also have ... ⟷ [k]⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = [k]⟨h⟩[m] ∧ [k]⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = [k]⟨h⟩ms ∧ ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = ks⟨h⟩[m] ∧ ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = ks⟨h⟩ms
by (*meson restrict-nonempty-eq*)
also have ... ⟷ [k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m] = [k]⟨h⟩[m] ∧ [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms = [k]⟨h⟩ms ∧ ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m] = ks⟨h⟩[m] ∧ ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms = ks⟨h⟩ms
using *2 3 4 5* **by** *simp*
finally show *?thesis*
by *simp*
qed

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 ¬ List.member ks k
and ¬ List.member ls l
and ¬ List.member ms m
shows (k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms) ⪯ (k#ks)⟨h⟩(m#ms) ⟷ [k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m] ⪯ [k]⟨h⟩[m] ∧ [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms ⪯ [k]⟨h⟩ms ∧ ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m] ⪯ ks⟨h⟩[m] ∧ ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩ms ⪯ ks⟨h⟩ms
proof –
have *1*: [k]⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = [k]⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩[m]
by (*metis assms restrict-nonempty-product-eq restrict-times*)
have *2*: [k]⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = [k]⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕ [k]⟨f⟩ls ⊙ ls⟨g⟩ms
by (*metis assms restrict-nonempty-product-eq restrict-times*)
have *3*: ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩[m] = ks⟨f⟩[l] ⊙ [l]⟨g⟩[m] ⊕ ks⟨f⟩ls ⊙ ls⟨g⟩[m]
by (*metis assms restrict-nonempty-product-eq restrict-times*)
have *4*: ks⟨(k#ks)⟨f⟩(l#ls) ⊙ (l#ls)⟨g⟩(m#ms)⟩ms = ks⟨f⟩[l] ⊙ [l]⟨g⟩ms ⊕

$ks\langle f\rangle ls \odot ls\langle g\rangle ms$
  by (*metis assms restrict-nonempty-product-eq restrict-times*)
 have $(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms) \preceq (k\#ks)\langle h\rangle(m\#ms) \longleftrightarrow$
$(k\#ks)\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle(m\#ms) \preceq (k\#ks)\langle h\rangle(m\#ms)$
  by (*simp add*: *restrict-times*)
 also have ... $\longleftrightarrow [k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle[m] \preceq [k]\langle h\rangle[m] \wedge$
$[k]\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle ms \preceq [k]\langle h\rangle ms \wedge ks\langle(k\#ks)\langle f\rangle(l\#ls) \odot$
$(l\#ls)\langle g\rangle(m\#ms)\rangle[m] \preceq ks\langle h\rangle[m] \wedge ks\langle(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms)\rangle ms$
$\preceq ks\langle h\rangle ms$
  by (*meson restrict-nonempty-less-eq*)
 also have ... $\longleftrightarrow [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m] \preceq [k]\langle h\rangle[m] \wedge$
$[k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms \preceq [k]\langle h\rangle ms \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus$
$ks\langle f\rangle ls \odot ls\langle g\rangle[m] \preceq ks\langle h\rangle[m] \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms \preceq$
$ks\langle h\rangle ms$
  using *1 2 3 4* **by** *simp*
 finally show *?thesis*
  by *simp*
qed

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 ms* $\Longrightarrow [l]\langle f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms \Longrightarrow [l]\langle star\ o\ f\rangle[l] \odot$
$[l]\langle g\rangle ms \preceq [l]\langle g\rangle ms$
proof (*induct ms*)
 case *Nil* **thus** *?case*
  by (*simp add*: *restrict-empty-right*)
next
 case (*Cons m ms*)
 assume *1*: *distinct ms* $\Longrightarrow [l]\langle f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms \Longrightarrow [l]\langle star\ o\ f\rangle[l] \odot$
$[l]\langle g\rangle ms \preceq [l]\langle g\rangle ms$
 assume *2*: *distinct* $(m\#ms)$
 assume *3*: $[l]\langle f\rangle[l] \odot [l]\langle g\rangle(m\#ms) \preceq [l]\langle g\rangle(m\#ms)$
 have *4*: $[l]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \preceq [l]\langle g\rangle[m] \wedge [l]\langle f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms$
  using *2 3* **by** (*metis distinct.simps*(*2*) *matrix-semilattice-sup.sup.bounded-iff*
member-def member-rec(*2*) *restrict-nonempty-product-less-eq*)
 hence *5*: $[l]\langle star\ o\ f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms$
  using *1 2* **by** *simp*
 have $f\ (l,l) * g\ (l,m) \leq g\ (l,m)$
  using *4* **by** (*metis restrict-singleton-product restrict-singleton*
less-eq-matrix-def)
 hence *6*: $(f\ (l,l))^\star * g\ (l,m) \leq g\ (l,m)$
  by (*simp add*: *star-left-induct-mult*)
 have $[l]\langle star\ o\ f\rangle[l] \odot [l]\langle g\rangle[m] \preceq [l]\langle g\rangle[m]$
 proof (*unfold less-eq-matrix-def*, *rule allI*, *rule prod-cases*)
  fix $i\ j$
  have $([l]\langle star\ o\ f\rangle[l] \odot [l]\langle g\rangle[m])\ (i,j) = (\bigsqcup_k ([l]\langle star\ o\ f\rangle[l])\ (i,k) *$

$([l]\langle g\rangle[m])\ (k,j))$
by (*simp add*: *times-matrix-def*)
also have ... $= (\bigsqcup_k\ (\mathit{if}\ i = l \wedge k = l\ \mathit{then}\ (f\ (i,k))^\star\ \mathit{else}\ \mathit{bot}) * (\mathit{if}\ k = l \wedge j = m\ \mathit{then}\ g\ (k,j)\ \mathit{else}\ \mathit{bot}))$
by (*simp add*: *restrict-singleton o-def*)
also have ... $= (\bigsqcup_k\ \mathit{if}\ k = l\ \mathit{then}\ (\mathit{if}\ i = l\ \mathit{then}\ (f\ (i,k))^\star\ \mathit{else}\ \mathit{bot}) * (\mathit{if}\ j = m\ \mathit{then}\ g\ (k,j)\ \mathit{else}\ \mathit{bot})\ \mathit{else}\ \mathit{bot})$
by (*rule sup-monoid.sum.cong*) *auto*
also have ... $= (\mathit{if}\ i = l\ \mathit{then}\ (f\ (i,l))^\star\ \mathit{else}\ \mathit{bot}) * (\mathit{if}\ j = m\ \mathit{then}\ g\ (l,j)\ \mathit{else}\ \mathit{bot})$
by (*simp add*: *sup-monoid.sum.delta*)
also have ... $= (\mathit{if}\ i = l \wedge j = m\ \mathit{then}\ (f\ (l,l))^\star * g\ (l,m)\ \mathit{else}\ \mathit{bot})$
by *simp*
also have ... $\leq ([l]\langle g\rangle[m])\ (i,j)$
using *6* **by** (*simp add*: *restrict-singleton*)
finally show $([l]\langle \mathit{star}\ o\ f\rangle[l] \odot [l]\langle g\rangle[m])\ (i,j) \leq ([l]\langle g\rangle[m])\ (i,j)$
.
qed
thus $[l]\langle \mathit{star}\ o\ f\rangle[l] \odot [l]\langle g\rangle(m\#ms) \preceq [l]\langle g\rangle(m\#ms)$
using *2 5* **by** (*metis* (*no-types*, *hide-lams*) *matrix-idempotent-semiring.mult-left-dist-sup matrix-semilattice-sup.sup.mono restrict-nonempty-right*)
qed

lemma *restrict-star-right-induct*:
fixes *f g* :: (*'a*::*finite*,*'b*::*kleene-algebra*) *square*
shows $\mathit{distinct}\ ms \Longrightarrow ms\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq ms\langle g\rangle[l] \Longrightarrow ms\langle g\rangle[l] \odot [l]\langle \mathit{star}\ o\ f\rangle[l] \preceq ms\langle g\rangle[l]$
proof (*induct ms*)
case *Nil* **thus** *?case*
by (*simp add*: *restrict-empty-left*)
next
case (*Cons m ms*)
assume *1*: $\mathit{distinct}\ ms \Longrightarrow ms\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq ms\langle g\rangle[l] \Longrightarrow ms\langle g\rangle[l] \odot [l]\langle \mathit{star}\ o\ f\rangle[l] \preceq ms\langle g\rangle[l]$
assume *2*: *distinct* (*m#ms*)
assume *3*: $(m\#ms)\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq (m\#ms)\langle g\rangle[l]$
have *4*: $[m]\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq [m]\langle g\rangle[l] \wedge ms\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq ms\langle g\rangle[l]$
using *2 3* **by** (*metis distinct.simps*(*2*) *matrix-semilattice-sup.sup.bounded-iff member-def member-rec*(*2*) *restrict-nonempty-product-less-eq*)
hence *5*: $ms\langle g\rangle[l] \odot [l]\langle \mathit{star}\ o\ f\rangle[l] \preceq ms\langle g\rangle[l]$
using *1 2* **by** *simp*
have $g\ (m,l) * f\ (l,l) \leq g\ (m,l)$
using *4* **by** (*metis restrict-singleton-product restrict-singleton less-eq-matrix-def*)
hence *6*: $g\ (m,l) * (f\ (l,l))^\star \leq g\ (m,l)$
by (*simp add*: *star-right-induct-mult*)
have $[m]\langle g\rangle[l] \odot [l]\langle \mathit{star}\ o\ f\rangle[l] \preceq [m]\langle g\rangle[l]$
proof (*unfold less-eq-matrix-def*, *rule allI*, *rule prod-cases*)

fix *i j*
have $([m]\langle g\rangle[l] \odot [l]\langle star\ o\ f\rangle[l])\ (i,j) = (\bigsqcup_k\ ([m]\langle g\rangle[l])\ (i,k) * ([l]\langle star\ o\ f\rangle[l])\ (k,j))$
by (*simp add*: *times-matrix-def*)
also have ... $= (\bigsqcup_k\ (\mathit{if}\ i = m \wedge k = l\ \mathit{then}\ g\ (i,k)\ \mathit{else}\ bot) * (\mathit{if}\ k = l \wedge j = l\ \mathit{then}\ (f\ (k,j))^\star\ \mathit{else}\ bot))$
by (*simp add*: *restrict-singleton o-def*)
also have ... $= (\bigsqcup_k\ \mathit{if}\ k = l\ \mathit{then}\ (\mathit{if}\ i = m\ \mathit{then}\ g\ (i,k)\ \mathit{else}\ bot) * (\mathit{if}\ j = l\ \mathit{then}\ (f\ (k,j))^\star\ \mathit{else}\ bot)\ \mathit{else}\ bot)$
by (*rule sup-monoid.sum.cong*) *auto*
also have ... $= (\mathit{if}\ i = m\ \mathit{then}\ g\ (i,l)\ \mathit{else}\ bot) * (\mathit{if}\ j = l\ \mathit{then}\ (f\ (l,j))^\star\ \mathit{else}\ bot)$
by (*simp add*: *sup-monoid.sum.delta*)
also have ... $= (\mathit{if}\ i = m \wedge j = l\ \mathit{then}\ g\ (m,l) * (f\ (l,l))^\star\ \mathit{else}\ bot)$
by *simp*
also have ... $\le ([m]\langle g\rangle[l])\ (i,j)$
using *6* **by** (*simp add*: *restrict-singleton*)
finally show $([m]\langle g\rangle[l] \odot [l]\langle star\ o\ f\rangle[l])\ (i,j) \le ([m]\langle g\rangle[l])\ (i,j)$
.
qed
thus $(m\#ms)\langle g\rangle[l] \odot [l]\langle star\ o\ f\rangle[l] \preceq (m\#ms)\langle g\rangle[l]$
using *2 5* **by** (*metis* (*no-types*, *hide-lams*) *matrix-idempotent-semiring.mult-right-dist-sup matrix-semilattice-sup.sup.mono restrict-nonempty-left*)
qed

lemma *restrict-pp*:
fixes *f* :: (*'a*,*'b*::*p-algebra*) *square*
shows $ks\langle\ominus\ominus f\rangle ls = \ominus\ominus(ks\langle f\rangle ls)$
by (*unfold restrict-matrix-def uminus-matrix-def*) *auto*

lemma *pp-star-commute*:
fixes *f* :: (*'a*,*'b*::*stone-kleene-relation-algebra*) *square*
shows $\ominus\ominus(star\ o\ f) = star\ o\ \ominus\ominus f$
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 *y* :: (*'a*,*'b*) *square*
let *?e* = *enum-class.enum*::*'a list*

let *?o* = *mone* :: (*'a*,*'b*) *square*
have ∀ *g* :: (*'a*,*'b*) *square* . *distinct ?e* ⟶ (*?e*⟨*?o*⟩*?e* ⊕ *?e*⟨*g*⟩*?e* ⊙ *star-matrix' ?e g*) = (*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*: ∀ *g* :: (*'a*,*'b*) *square* . *distinct s* ⟶ (*s*⟨*?o*⟩*s* ⊕ *s*⟨*g*⟩*s* ⊙ *star-matrix' s g*) = (*star-matrix' s g*)
show ∀ *g* :: (*'a*,*'b*) *square* . *distinct ?t* ⟶ (*?t*⟨*?o*⟩*?t* ⊕ *?t*⟨*g*⟩*?t* ⊙ *star-matrix' ?t g*) = (*star-matrix' ?t g*)
proof (*rule allI*, *rule impI*)
fix *g* :: (*'a*,*'b*) *square*
assume *2*: *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*⟨*star o ?a*⟩*?r*
let *?ds* = *star-matrix' s ?d*
let *?e* = *?a* ⊕ *?b* ⊙ *?ds* ⊙ *?c*
let *?es* = *?r*⟨*star o ?e*⟩*?r*
let *?f* = *?d* ⊕ *?c* ⊙ *?as* ⊙ *?b*
let *?fs* = *star-matrix' s ?f*
have *s*⟨*?ds*⟩*s* = *?ds* ∧ *s*⟨*?fs*⟩*s* = *?fs*
by (*simp add*: *restrict-star*)
hence *3*: *?r*⟨*?e*⟩*?r* = *?e* ∧ *s*⟨*?f*⟩*s* = *?f*
by (*metis* (*no-types*, *lifting*) *restrict-one-left-unit restrict-sup restrict-times*)
have *4*: *disjoint s ?r* ∧ *disjoint ?r s*
using *2* **by** (*simp add*: *in-set-member member-rec*)
hence *5*: *?t*⟨*?o*⟩*?t* = *?r*⟨*?o*⟩*?r* ⊕ *s*⟨*?o*⟩*s*
by (*meson member-rec*(*1*) *restrict-one*)
have *6*: *?t*⟨*g*⟩*?t* ⊙ *?es* = *?a* ⊙ *?es* ⊕ *?c* ⊙ *?es*
proof −
have *?t*⟨*g*⟩*?t* ⊙ *?es* = (*?a* ⊕ *?b* ⊕ *?c* ⊕ *?d*) ⊙ *?es*
by (*metis restrict-nonempty*)
also have ... = *?a* ⊙ *?es* ⊕ *?b* ⊙ *?es* ⊕ *?c* ⊙ *?es* ⊕ *?d* ⊙ *?es*
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... = *?a* ⊙ *?es* ⊕ *?c* ⊙ *?es*
using *4* **by** (*simp add*: *times-disjoint*)
finally show *?thesis*
.
qed
have *7*: *?t*⟨*g*⟩*?t* ⊙ *?as* ⊙ *?b* ⊙ *?fs* = *?a* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?c* ⊙ *?as* ⊙ *?b* ⊙ *?fs*
proof −

have $?t\langle g\rangle ?t \odot ?as \odot ?b \odot ?fs = (?a \oplus ?b \oplus ?c \oplus ?d) \odot ?as \odot ?b \odot ?fs$
by (*metis restrict-nonempty*)
also have ... $= ?a \odot ?as \odot ?b \odot ?fs \oplus ?b \odot ?as \odot ?b \odot ?fs \oplus ?c \odot ?as \odot ?b \odot ?fs \oplus ?d \odot ?as \odot ?b \odot ?fs$
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... $= ?a \odot ?as \odot ?b \odot ?fs \oplus ?c \odot ?as \odot ?b \odot ?fs$
using *4* **by** (*simp add*: *times-disjoint*)
finally show *?thesis*
.
qed
have *8*: $?t\langle g\rangle ?t \odot ?ds \odot ?c \odot ?es = ?b \odot ?ds \odot ?c \odot ?es \oplus ?d \odot ?ds \odot ?c \odot ?es$
proof −
have $?t\langle g\rangle ?t \odot ?ds \odot ?c \odot ?es = (?a \oplus ?b \oplus ?c \oplus ?d) \odot ?ds \odot ?c \odot ?es$
by (*metis restrict-nonempty*)
also have ... $= ?a \odot ?ds \odot ?c \odot ?es \oplus ?b \odot ?ds \odot ?c \odot ?es \oplus ?c \odot ?ds \odot ?c \odot ?es \oplus ?d \odot ?ds \odot ?c \odot ?es$
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... $= ?b \odot ?ds \odot ?c \odot ?es \oplus ?d \odot ?ds \odot ?c \odot ?es$
using *4* **by** (*metis* (*no-types*, *lifting*) *times-disjoint matrix-idempotent-semiring.mult-left-zero restrict-star matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left*)
finally show *?thesis*
.
qed
have *9*: $?t\langle g\rangle ?t \odot ?fs = ?b \odot ?fs \oplus ?d \odot ?fs$
proof −
have $?t\langle g\rangle ?t \odot ?fs = (?a \oplus ?b \oplus ?c \oplus ?d) \odot ?fs$
by (*metis restrict-nonempty*)
also have ... $= ?a \odot ?fs \oplus ?b \odot ?fs \oplus ?c \odot ?fs \oplus ?d \odot ?fs$
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... $= ?b \odot ?fs \oplus ?d \odot ?fs$
using *4* **by** (*metis* (*no-types*, *lifting*) *times-disjoint restrict-star matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-right matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left*)
finally show *?thesis*
.
qed
have $?t\langle ?o\rangle ?t \oplus ?t\langle g\rangle ?t \odot star\text{-}matrix'\ ?t\ g = ?t\langle ?o\rangle ?t \oplus ?t\langle g\rangle ?t \odot (?es \oplus ?as \odot ?b \odot ?fs \oplus ?ds \odot ?c \odot ?es \oplus ?fs)$
by (*metis star-matrix'.simps*(*2*))
also have ... $= ?t\langle ?o\rangle ?t \oplus ?t\langle g\rangle ?t \odot ?es \oplus ?t\langle g\rangle ?t \odot ?as \odot ?b \odot ?fs \oplus ?t\langle g\rangle ?t \odot ?ds \odot ?c \odot ?es \oplus ?t\langle g\rangle ?t \odot ?fs$
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc matrix-semilattice-sup.sup-assoc*)
also have ... $= ?r\langle ?o\rangle ?r \oplus s\langle ?o\rangle s \oplus ?a \odot ?es \oplus ?c \odot ?es \oplus ?a \odot ?as \odot ?b \odot ?fs \oplus ?c \odot ?as \odot ?b \odot ?fs \oplus ?b \odot ?ds \odot ?c \odot ?es \oplus ?d \odot ?ds \odot ?c \odot ?es \oplus ?b \odot ?fs \oplus ?d \odot ?fs$

using *5 6 7 8 9* **by** (*simp add*: *matrix-semilattice-sup.sup.assoc*)
also have ... = (*?r*⟨*?o*⟩*?r* ⊕ (*?a* ⊙ *?es* ⊕ *?b* ⊙ *?ds* ⊙ *?c* ⊙ *?es*)) ⊕ (*?b* ⊙ *?fs* ⊕ *?a* ⊙ *?as* ⊙ *?b* ⊙ *?fs*) ⊕ (*?c* ⊙ *?es* ⊕ *?d* ⊙ *?ds* ⊙ *?c* ⊙ *?es*) ⊕ (*s*⟨*?o*⟩*s* ⊕ (*?d* ⊙ *?fs* ⊕ *?c* ⊙ *?as* ⊙ *?b* ⊙ *?fs*))
by (*simp only*: *matrix-semilattice-sup.sup-assoc* *matrix-semilattice-sup.sup-commute* *matrix-semilattice-sup.sup-left-commute*)
also have ... = (*?r*⟨*?o*⟩*?r* ⊕ (*?a* ⊙ *?es* ⊕ *?b* ⊙ *?ds* ⊙ *?c* ⊙ *?es*)) ⊕ (*?r*⟨*?o*⟩*?r* ⊙ *?b* ⊙ *?fs* ⊕ *?a* ⊙ *?as* ⊙ *?b* ⊙ *?fs*) ⊕ (*s*⟨*?o*⟩*s* ⊙ *?c* ⊙ *?es* ⊕ *?d* ⊙ *?ds* ⊙ *?c* ⊙ *?es*) ⊕ (*s*⟨*?o*⟩*s* ⊕ (*?d* ⊙ *?fs* ⊕ *?c* ⊙ *?as* ⊙ *?b* ⊙ *?fs*))
by (*simp add*: *restrict-one-left-unit*)
also have ... = (*?r*⟨*?o*⟩*?r* ⊕ *?e* ⊙ *?es*) ⊕ ((*?r*⟨*?o*⟩*?r* ⊕ *?a* ⊙ *?as*) ⊙ *?b* ⊙ *?fs*) ⊕ ((*s*⟨*?o*⟩*s* ⊕ *?d* ⊙ *?ds*) ⊙ *?c* ⊙ *?es*) ⊕ (*s*⟨*?o*⟩*s* ⊕ *?f* ⊙ *?fs*)
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... = (*?r*⟨*?o*⟩*?r* ⊕ *?e* ⊙ *?es*) ⊕ ((*?r*⟨*?o*⟩*?r* ⊕ *?a* ⊙ *?as*) ⊙ *?b* ⊙ *?fs*) ⊕ ((*s*⟨*?o*⟩*s* ⊕ *?d* ⊙ *?ds*) ⊙ *?c* ⊙ *?es*) ⊕ *?fs*
using *1 2 3* **by** (*metis distinct.simps*(*2*))
also have ... = (*?r*⟨*?o*⟩*?r* ⊕ *?e* ⊙ *?es*) ⊕ ((*?r*⟨*?o*⟩*?r* ⊕ *?a* ⊙ *?as*) ⊙ *?b* ⊙ *?fs*) ⊕ (*?ds* ⊙ *?c* ⊙ *?es*) ⊕ *?fs*
using *1 2* **by** (*metis* (*no-types*, *lifting*) *distinct.simps*(*2*) *restrict-superlist*)
also have ... = *?es* ⊕ ((*?r*⟨*?o*⟩*?r* ⊕ *?a* ⊙ *?as*) ⊙ *?b* ⊙ *?fs*) ⊕ (*?ds* ⊙ *?c* ⊙ *?es*) ⊕ *?fs*
using *3* **by** (*metis restrict-star-unfold*)
also have ... = *?es* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?fs*
by (*metis* (*no-types*, *lifting*) *restrict-one-left-unit restrict-star-unfold restrict-times*)
also have ... = *star-matrix′ ?t g*
by (*metis star-matrix′.simps*(*2*))
finally show *?t*⟨*?o*⟩*?t* ⊕ *?t*⟨*g*⟩*?t* ⊙ *star-matrix′ ?t g* = *star-matrix′ ?t g*
.
qed
qed
thus *?o* ⊕ *y* ⊙ $y^{\odot}$ ⪯ $y^{\odot}$
by (*simp add*: *enum-distinct restrict-all*)
next
fix *x y z* :: (′*a*,′*b*) *square*
let *?e* = *enum-class.enum*::′*a list*
have ∀ *g h* :: (′*a*,′*b*) *square* . ∀ *zs* . *distinct ?e* ∧ *distinct zs* ⟶ (*?e*⟨*g*⟩*?e* ⊙ *?e*⟨*h*⟩*zs* ⪯ *?e*⟨*h*⟩*zs* ⟶ *star-matrix′ ?e g* ⊙ *?e*⟨*h*⟩*zs* ⪯ *?e*⟨*h*⟩*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*: ∀ *g h* :: (′*a*,′*b*) *square* . ∀ *zs* . *distinct s* ∧ *distinct zs* ⟶ (*s*⟨*g*⟩*s* ⊙ *s*⟨*h*⟩*zs* ⪯ *s*⟨*h*⟩*zs* ⟶ *star-matrix′ s g* ⊙ *s*⟨*h*⟩*zs* ⪯ *s*⟨*h*⟩*zs*)
show ∀ *g h* :: (′*a*,′*b*) *square* . ∀ *zs* . *distinct ?t* ∧ *distinct zs* ⟶ (*?t*⟨*g*⟩*?t* ⊙ *?t*⟨*h*⟩*zs* ⪯ *?t*⟨*h*⟩*zs* ⟶ *star-matrix′ ?t g* ⊙ *?t*⟨*h*⟩*zs* ⪯ *?t*⟨*h*⟩*zs*)
proof (*intro allI*)
fix *g h* :: (′*a*,′*b*) *square*

fix *zs* :: *'a list*
show *distinct ?t* $\wedge$ *distinct zs* $\longrightarrow$ (*?t*$\langle g\rangle$*?t* $\odot$ *?t*$\langle h\rangle$*zs* $\preceq$ *?t*$\langle h\rangle$*zs* $\longrightarrow$ *star-matrix' ?t g* $\odot$ *?t*$\langle h\rangle$*zs* $\preceq$ *?t*$\langle h\rangle$*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* $\wedge$ *distinct zs* $\longrightarrow$ (*?t*$\langle g\rangle$*?t* $\odot$ *?t*$\langle h\rangle$*zs* $\preceq$ *?t*$\langle h\rangle$*zs* $\longrightarrow$ *star-matrix' ?t g* $\odot$ *?t*$\langle h\rangle$*zs* $\preceq$ *?t*$\langle h\rangle$*zs*)
proof (*intro impI*)
let *?y* = [*y*]
assume *3*: *distinct ?t* $\wedge$ *distinct zs*
hence *4*: *distinct s* $\wedge$ *distinct ys* $\wedge$ $\neg$ *List.member s k* $\wedge$ $\neg$ *List.member ys y*
using *2* **by** (*simp add*: *List.member-def*)
let *?r* = [*k*]
let *?a* = *?r*$\langle g\rangle$*?r*
let *?b* = *?r*$\langle g\rangle$*s*
let *?c* = *s*$\langle g\rangle$*?r*
let *?d* = *s*$\langle g\rangle$*s*
let *?as* = *?r*$\langle$*star o ?a*$\rangle$*?r*
let *?ds* = *star-matrix' s ?d*
let *?e* = *?a* $\oplus$ *?b* $\odot$ *?ds* $\odot$ *?c*
let *?es* = *?r*$\langle$*star o ?e*$\rangle$*?r*
let *?f* = *?d* $\oplus$ *?c* $\odot$ *?as* $\odot$ *?b*
let *?fs* = *star-matrix' s ?f*
let *?ha* = *?r*$\langle h\rangle$*?y*
let *?hb* = *?r*$\langle h\rangle$*ys*
let *?hc* = *s*$\langle h\rangle$*?y*
let *?hd* = *s*$\langle h\rangle$*ys*
assume *?t*$\langle g\rangle$*?t* $\odot$ *?t*$\langle h\rangle$*zs* $\preceq$ *?t*$\langle h\rangle$*zs*
hence *5*: *?a* $\odot$ *?ha* $\oplus$ *?b* $\odot$ *?hc* $\preceq$ *?ha* $\wedge$ *?a* $\odot$ *?hb* $\oplus$ *?b* $\odot$ *?hd* $\preceq$ *?hb* $\wedge$ *?c* $\odot$ *?ha* $\oplus$ *?d* $\odot$ *?hc* $\preceq$ *?hc* $\wedge$ *?c* $\odot$ *?hb* $\oplus$ *?d* $\odot$ *?hd* $\preceq$ *?hd*
using *2 3 4* **by** (*simp add*: *restrict-nonempty-product-less-eq*)
have *6*: *s*$\langle$*?ds*$\rangle$*s* = *?ds* $\wedge$ *s*$\langle$*?fs*$\rangle$*s* = *?fs*
by (*simp add*: *restrict-star*)
hence *7*: *?r*$\langle$*?e*$\rangle$*?r* = *?e* $\wedge$ *s*$\langle$*?f*$\rangle$*s* = *?f*
by (*metis* (*no-types*, *lifting*) *restrict-one-left-unit restrict-sup restrict-times*)
have *8*: *disjoint s ?r* $\wedge$ *disjoint ?r s*
using *3* **by** (*simp add*: *in-set-member member-rec*(*1*) *member-rec*(*2*))
have *9*: *?es* $\odot$ *?t*$\langle h\rangle$*zs* = *?es* $\odot$ *?ha* $\oplus$ *?es* $\odot$ *?hb*
proof –
have *?es* $\odot$ *?t*$\langle h\rangle$*zs* = *?es* $\odot$ (*?ha* $\oplus$ *?hb* $\oplus$ *?hc* $\oplus$ *?hd*)
using *2* **by** (*metis restrict-nonempty*)
also have ... = *?es* $\odot$ *?ha* $\oplus$ *?es* $\odot$ *?hb* $\oplus$ *?es* $\odot$ *?hc* $\oplus$ *?es* $\odot$ *?hd*
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup*)

also have ... = *?es* ⊙ *?ha* ⊕ *?es* ⊙ *?hb*
using *8* **by** (*simp add*: *times-disjoint*)
finally show *?thesis*
.
qed
have *10*: *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?t*⟨*h*⟩*zs* = *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*
proof −
have *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?t*⟨*h*⟩*zs* = *?as* ⊙ *?b* ⊙ *?fs* ⊙ (*?ha* ⊕ *?hb* ⊕ *?hc* ⊕ *?hd*)
using *2* **by** (*metis restrict-nonempty*)
also have ... = *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?ha* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hb* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup*)
also have ... = *?as* ⊙ *?b* ⊙ (*?fs* ⊙ *?ha*) ⊕ *?as* ⊙ *?b* ⊙ (*?fs* ⊙ *?hb*) ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*
by (*simp add*: *matrix-monoid.mult-assoc*)
also have ... = *?as* ⊙ *?b* ⊙ *mbot* ⊕ *?as* ⊙ *?b* ⊙ *mbot* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*
using *6 8* **by** (*metis* (*no-types*) *times-disjoint*)
also have ... = *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*
by *simp*
finally show *?thesis*
.
qed
have *11*: *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?t*⟨*h*⟩*zs* = *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb*
proof −
have *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?t*⟨*h*⟩*zs* = *?ds* ⊙ *?c* ⊙ *?es* ⊙ (*?ha* ⊕ *?hb* ⊕ *?hc* ⊕ *?hd*)
using *2* **by** (*metis restrict-nonempty*)
also have ... = *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hc* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hd*
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup*)
also have ... = *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?ds* ⊙ *?c* ⊙ (*?es* ⊙ *?hc*) ⊕ *?ds* ⊙ *?c* ⊙ (*?es* ⊙ *?hd*)
by (*simp add*: *matrix-monoid.mult-assoc*)
also have ... = *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?ds* ⊙ *?c* ⊙ *mbot* ⊕ *?ds* ⊙ *?c* ⊙ *mbot*
using *8* **by** (*metis times-disjoint*)
also have ... = *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb*
by *simp*
finally show *?thesis*
.
qed
have *12*: *?fs* ⊙ *?t*⟨*h*⟩*zs* = *?fs* ⊙ *?hc* ⊕ *?fs* ⊙ *?hd*
proof −
have *?fs* ⊙ *?t*⟨*h*⟩*zs* = *?fs* ⊙ (*?ha* ⊕ *?hb* ⊕ *?hc* ⊕ *?hd*)
using *2* **by** (*metis restrict-nonempty*)

also have ... = *?fs* ⊙ *?ha* ⊕ *?fs* ⊙ *?hb* ⊕ *?fs* ⊙ *?hc* ⊕ *?fs* ⊙ *?hd*
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup*)
also have ... = *?fs* ⊙ *?hc* ⊕ *?fs* ⊙ *?hd*
using *6 8* **by** (*metis* (*no-types*) *times-disjoint*
matrix-bounded-semilattice-sup-bot.sup-monoid.add-0-left)
finally show *?thesis*
.
qed
have *13*: *?es* ⊙ *?ha* ⪯ *?ha*
proof −
have *?b* ⊙ *?ds* ⊙ *?c* ⊙ *?ha* ⪯ *?b* ⊙ *?ds* ⊙ *?hc*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone*
matrix-monoid.mult-assoc)
also have ... ⪯ *?b* ⊙ *?hc*
using *1 3 5* **by** (*simp add*:
matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc
member-rec(*2*) *restrict-sublist*)
also have ... ⪯ *?ha*
using *5* **by** *simp*
finally have *?e* ⊙ *?ha* ⪯ *?ha*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
thus *?thesis*
using *7* **by** (*simp add*: *restrict-star-left-induct*)
qed
have *14*: *?es* ⊙ *?hb* ⪯ *?hb*
proof −
have *?b* ⊙ *?ds* ⊙ *?c* ⊙ *?hb* ⪯ *?b* ⊙ *?ds* ⊙ *?hd*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone*
matrix-monoid.mult-assoc)
also have ... ⪯ *?b* ⊙ *?hd*
using *1 4 5* **by** (*simp add*:
matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc
restrict-sublist)
also have ... ⪯ *?hb*
using *5* **by** *simp*
finally have *?e* ⊙ *?hb* ⪯ *?hb*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
thus *?thesis*
using *4 7* **by** (*simp add*: *restrict-star-left-induct*)
qed
have *15*: *?fs* ⊙ *?hc* ⪯ *?hc*
proof −
have *?c* ⊙ *?as* ⊙ *?b* ⊙ *?hc* ⪯ *?c* ⊙ *?as* ⊙ *?ha*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone*
matrix-monoid.mult-assoc)
also have ... ⪯ *?c* ⊙ *?ha*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone*
matrix-monoid.mult-assoc restrict-star-left-induct restrict-sublist)
also have ... ⪯ *?hc*

using *5* **by** *simp*
finally have *?f* ⊙ *?hc* ≼ *?hc*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
thus *?thesis*
using *1 3 7* **by** *simp*
qed
have *16*: *?fs* ⊙ *?hd* ≼ *?hd*
proof −
have *?c* ⊙ *?as* ⊙ *?b* ⊙ *?hd* ≼ *?c* ⊙ *?as* ⊙ *?hb*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc*)
also have ... ≼ *?c* ⊙ *?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 ... ≼ *?hd*
using *5* **by** *simp*
finally have *?f* ⊙ *?hd* ≼ *?hd*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
thus *?thesis*
using *1 4 7* **by** *simp*
qed
have *17*: *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ≼ *?ha*
proof −
have *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ≼ *?as* ⊙ *?b* ⊙ *?hc*
using *15* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc*)
also have ... ≼ *?as* ⊙ *?ha*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc*)
also have ... ≼ *?ha*
using *5* **by** (*simp add*: *restrict-star-left-induct restrict-sublist*)
finally show *?thesis*
.
qed
have *18*: *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd* ≼ *?hb*
proof −
have *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd* ≼ *?as* ⊙ *?b* ⊙ *?hd*
using *16* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc*)
also have ... ≼ *?as* ⊙ *?hb*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-right-isotone matrix-monoid.mult-assoc*)
also have ... ≼ *?hb*
using *4 5* **by** (*simp add*: *restrict-star-left-induct restrict-sublist*)
finally show *?thesis*
.
qed
have *19*: *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ≼ *?hc*

proof –
have *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ≼ *?ds* ⊙ *?c* ⊙ *?ha*
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
have *20*: *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ≼ *?hd*
proof –
have *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ≼ *?ds* ⊙ *?c* ⊙ *?hb*
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
have *21*: *?es* ⊙ *?ha* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ≼ *?ha*
using *13 17 matrix-semilattice-sup.le-supI* **by** *blast*
have *22*: *?es* ⊙ *?hb* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd* ≼ *?hb*
using *14 18 matrix-semilattice-sup.le-supI* **by** *blast*
have *23*: *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?fs* ⊙ *?hc* ≼ *?hc*
using *15 19 matrix-semilattice-sup.le-supI* **by** *blast*
have *24*: *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?fs* ⊙ *?hd* ≼ *?hd*
using *16 20 matrix-semilattice-sup.le-supI* **by** *blast*
have *star-matrix′ ?t g* ⊙ *?t⟨h⟩zs* = (*?es* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?fs*) ⊙ *?t⟨h⟩zs*
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*)
also have ... = *?es* ⊙ *?ha* ⊕ *?es* ⊙ *?hb* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?fs* ⊙ *?hc* ⊕ *?fs* ⊙ *?hd*
using *9 10 11 12* **by** (*simp only*: *matrix-semilattice-sup.sup-assoc*)
also have ... = (*?es* ⊙ *?ha* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hc*) ⊕ (*?es* ⊙ *?hb* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊙ *?hd*) ⊕ (*?ds* ⊙ *?c* ⊙ *?es* ⊙ *?ha* ⊕ *?fs* ⊙ *?hc*) ⊕ (*?ds* ⊙ *?c* ⊙ *?es* ⊙ *?hb* ⊕ *?fs* ⊙ *?hd*)
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 matrix-semilattice-sup.sup.mono* **by** *blast*
also have ... = *?t⟨h⟩zs*
using *2* **by** (*metis restrict-nonempty*)
finally show *star-matrix' ?t g* ⊙ *?t⟨h⟩zs* ⪯ *?t⟨h⟩zs*
.
qed
qed
qed
qed
hence ∀ *zs* . *distinct zs* ⟶ (*y* ⊙ *?e⟨x⟩zs* ⪯ *?e⟨x⟩zs* ⟶ *y*^⊙ ⊙ *?e⟨x⟩zs* ⪯
?e⟨x⟩zs)
by (*simp add*: *enum-distinct restrict-all*)
thus *y* ⊙ *x* ⪯ *x* ⟶ *y*^⊙ ⊙ *x* ⪯ *x*
by (*metis restrict-all enum-distinct*)
next
fix *x y z* :: (*'a*,*'b*) *square*
let *?e* = *enum-class.enum*::*'a list*
have ∀ *g h* :: (*'a*,*'b*) *square* . ∀ *zs* . *distinct ?e* ∧ *distinct zs* ⟶ (*zs⟨h⟩?e* ⊙
?e⟨g⟩?e ⪯ *zs⟨h⟩?e* ⟶ *zs⟨h⟩?e* ⊙ *star-matrix' ?e g* ⪯ *zs⟨h⟩?e*)
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*: ∀ *g h* :: (*'a*,*'b*) *square* . ∀ *zs* . *distinct s* ∧ *distinct zs* ⟶ (*zs⟨h⟩s* ⊙
s⟨g⟩s ⪯ *zs⟨h⟩s* ⟶ *zs⟨h⟩s* ⊙ *star-matrix' s g* ⪯ *zs⟨h⟩s*)
show ∀ *g h* :: (*'a*,*'b*) *square* . ∀ *zs* . *distinct ?t* ∧ *distinct zs* ⟶ (*zs⟨h⟩?t* ⊙
?t⟨g⟩?t ⪯ *zs⟨h⟩?t* ⟶ *zs⟨h⟩?t* ⊙ *star-matrix' ?t g* ⪯ *zs⟨h⟩?t*)
proof (*intro allI*)
fix *g h* :: (*'a*,*'b*) *square*
fix *zs* :: *'a list*
show *distinct ?t* ∧ *distinct zs* ⟶ (*zs⟨h⟩?t* ⊙ *?t⟨g⟩?t* ⪯ *zs⟨h⟩?t* ⟶ *zs⟨h⟩?t*
⊙ *star-matrix' ?t g* ⪯ *zs⟨h⟩?t*)
proof (*cases zs*)
case *Nil* **thus** *?thesis*
by (*metis restrict-empty-left restrict-star restrict-times*)
next
case (*Cons y ys*)
assume *2*: *zs* = *y#ys*
show *distinct ?t* ∧ *distinct zs* ⟶ (*zs⟨h⟩?t* ⊙ *?t⟨g⟩?t* ⪯ *zs⟨h⟩?t* ⟶
zs⟨h⟩?t ⊙ *star-matrix' ?t g* ⪯ *zs⟨h⟩?t*)
proof (*intro impI*)
let *?y* = [*y*]
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⟩?r*

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 *?e* = *?a* ⊕ *?b* ⊙ *?ds* ⊙ *?c*
let *?es* = *?r⟨star o ?e⟩?r*
let *?f* = *?d* ⊕ *?c* ⊙ *?as* ⊙ *?b*
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* ⊙ *?t⟨g⟩?t* ⪯ *zs⟨h⟩?t*
hence *5*: *?ha* ⊙ *?a* ⊕ *?hb* ⊙ *?c* ⪯ *?ha* ∧ *?ha* ⊙ *?b* ⊕ *?hb* ⊙ *?d* ⪯ *?hb* ∧ *?hc* ⊙ *?a* ⊕ *?hd* ⊙ *?c* ⪯ *?hc* ∧ *?hc* ⊙ *?b* ⊕ *?hd* ⊙ *?d* ⪯ *?hd*
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⟨?e⟩?r* = *?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*)
have *9*: *zs⟨h⟩?t* ⊙ *?es* = *?ha* ⊙ *?es* ⊕ *?hc* ⊙ *?es*
proof −
have *zs⟨h⟩?t* ⊙ *?es* = (*?ha* ⊕ *?hb* ⊕ *?hc* ⊕ *?hd*) ⊙ *?es*
using *2* **by** (*metis restrict-nonempty*)
also have ... = *?ha* ⊙ *?es* ⊕ *?hb* ⊙ *?es* ⊕ *?hc* ⊙ *?es* ⊕ *?hd* ⊙ *?es*
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... = *?ha* ⊙ *?es* ⊕ *?hc* ⊙ *?es*
using *8* **by** (*simp add*: *times-disjoint*)
finally show *?thesis*
.
qed
have *10*: *zs⟨h⟩?t* ⊙ *?as* ⊙ *?b* ⊙ *?fs* = *?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs*
proof −
have *zs⟨h⟩?t* ⊙ *?as* ⊙ *?b* ⊙ *?fs* = (*?ha* ⊕ *?hb* ⊕ *?hc* ⊕ *?hd*) ⊙ *?as* ⊙ *?b* ⊙ *?fs*
using *2* **by** (*metis restrict-nonempty*)
also have ... = *?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hb* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hd* ⊙ *?as* ⊙ *?b* ⊙ *?fs*
by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)
also have ... = *?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *mbot* ⊙ *?b* ⊙ *?fs* ⊕ *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *mbot* ⊙ *?b* ⊙ *?fs*
using *8* **by** (*metis* (*no-types*) *times-disjoint*)
also have ... = *?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs*
by *simp*

finally show *?thesis*

.

qed

have *11*: $zs\langle h\rangle ?t \odot ?ds \odot ?c \odot ?es = ?hb \odot ?ds \odot ?c \odot ?es \oplus ?hd \odot ?ds \odot ?c \odot ?es$

proof –

have $zs\langle h\rangle ?t \odot ?ds \odot ?c \odot ?es = (?ha \oplus ?hb \oplus ?hc \oplus ?hd) \odot ?ds \odot ?c \odot ?es$

using *2* **by** (*metis restrict-nonempty*)

also have ... $= ?ha \odot ?ds \odot ?c \odot ?es \oplus ?hb \odot ?ds \odot ?c \odot ?es \oplus ?hc \odot ?ds \odot ?c \odot ?es \oplus ?hd \odot ?ds \odot ?c \odot ?es$

by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)

also have ... $= mbot \odot ?c \odot ?es \oplus ?hb \odot ?ds \odot ?c \odot ?es \oplus mbot \odot ?c \odot ?es \oplus ?hd \odot ?ds \odot ?c \odot ?es$

using *6 8* **by** (*metis* (*no-types*) *times-disjoint*)

also have ... $= ?hb \odot ?ds \odot ?c \odot ?es \oplus ?hd \odot ?ds \odot ?c \odot ?es$

by *simp*

finally show *?thesis*

.

qed

have *12*: $zs\langle h\rangle ?t \odot ?fs = ?hb \odot ?fs \oplus ?hd \odot ?fs$

proof –

have $zs\langle h\rangle ?t \odot ?fs = (?ha \oplus ?hb \oplus ?hc \oplus ?hd) \odot ?fs$

using *2* **by** (*metis restrict-nonempty*)

also have ... $= ?ha \odot ?fs \oplus ?hb \odot ?fs \oplus ?hc \odot ?fs \oplus ?hd \odot ?fs$

by (*simp add*: *matrix-idempotent-semiring.mult-right-dist-sup*)

also have ... $= ?hb \odot ?fs \oplus ?hd \odot ?fs$

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 *?thesis*

.

qed

have *13*: $?ha \odot ?es \preceq ?ha$

proof –

have $?ha \odot ?b \odot ?ds \odot ?c \preceq ?hb \odot ?ds \odot ?c$

using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)

also have ... $\preceq ?hb \odot ?c$

using *1 4 5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-sublist*)

also have ... $\preceq ?ha$

using *5* **by** *simp*

finally have $?ha \odot ?e \preceq ?ha$

using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc*)

thus *?thesis*

using *7* **by** (*simp add*: *restrict-star-right-induct*)

qed

have *14*: $?hb \odot ?fs \preceq ?hb$

proof –
have *?hb* ⊙ *?c* ⊙ *?as* ⊙ *?b* ⪯ *?ha* ⊙ *?as* ⊙ *?b*
using *5* **by** (*metis matrix-semilattice-sup.le-supE matrix-idempotent-semiring.mult-left-isotone*)
also have ... ⪯ *?ha* ⊙ *?b*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-star-right-induct restrict-sublist*)
also have ... ⪯ *?hb*
using *5* **by** *simp*
finally have *?hb* ⊙ *?f* ⪯ *?hb*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc*)
thus *?thesis*
using *1 3 7* **by** *simp*
qed
have *15*: *?hc* ⊙ *?es* ⪯ *?hc*
proof –
have *?hc* ⊙ *?b* ⊙ *?ds* ⊙ *?c* ⪯ *?hd* ⊙ *?ds* ⊙ *?c*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
also have ... ⪯ *?hd* ⊙ *?c*
using *1 4 5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-sublist*)
also have ... ⪯ *?hc*
using *5* **by** *simp*
finally have *?hc* ⊙ *?e* ⪯ *?hc*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc*)
thus *?thesis*
using *4 7* **by** (*simp add*: *restrict-star-right-induct*)
qed
have *16*: *?hd* ⊙ *?fs* ⪯ *?hd*
proof –
have *?hd* ⊙ *?c* ⊙ *?as* ⊙ *?b* ⪯ *?hc* ⊙ *?as* ⊙ *?b*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
also have ... ⪯ *?hc* ⊙ *?b*
using *4 5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-star-right-induct restrict-sublist*)
also have ... ⪯ *?hd*
using *5* **by** *simp*
finally have *?hd* ⊙ *?f* ⪯ *?hd*
using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc*)
thus *?thesis*
using *1 4 7* **by** *simp*
qed
have *17*: *?hb* ⊙ *?ds* ⊙ *?c* ⊙ *?es* ⪯ *?ha*
proof –
have *?hb* ⊙ *?ds* ⊙ *?c* ⊙ *?es* ⪯ *?hb* ⊙ *?c* ⊙ *?es*
using *1 4 5* **by** (*simp add*:

matrix-idempotent-semiring.mult-left-isotone restrict-sublist)
 also have ... $\preceq$ *?ha* $\odot$ *?es*
 using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
 also have ... $\preceq$ *?ha*
 using *13* **by** *simp*
 finally show *?thesis*
 .
 qed
 have *18*: *?ha* $\odot$ *?as* $\odot$ *?b* $\odot$ *?fs* $\preceq$ *?hb*
 proof −
 have *?ha* $\odot$ *?as* $\odot$ *?b* $\odot$ *?fs* $\preceq$ *?ha* $\odot$ *?b* $\odot$ *?fs*
 using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-star-right-induct restrict-sublist*)
 also have ... $\preceq$ *?hb* $\odot$ *?fs*
 using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
 also have ... $\preceq$ *?hb*
 using *14* **by** *simp*
 finally show *?thesis*
 by *simp*
 qed
 have *19*: *?hd* $\odot$ *?ds* $\odot$ *?c* $\odot$ *?es* $\preceq$ *?hc*
 proof −
 have *?hd* $\odot$ *?ds* $\odot$ *?c* $\odot$ *?es* $\preceq$ *?hd* $\odot$ *?c* $\odot$ *?es*
 using *1 4 5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-sublist*)
 also have ... $\preceq$ *?hc* $\odot$ *?es*
 using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
 also have ... $\preceq$ *?hc*
 using *15* **by** *simp*
 finally show *?thesis*
 by *simp*
 qed
 have *20*: *?hc* $\odot$ *?as* $\odot$ *?b* $\odot$ *?fs* $\preceq$ *?hd*
 proof −
 have *?hc* $\odot$ *?as* $\odot$ *?b* $\odot$ *?fs* $\preceq$ *?hc* $\odot$ *?b* $\odot$ *?fs*
 using *4 5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone restrict-star-right-induct restrict-sublist*)
 also have ... $\preceq$ *?hd* $\odot$ *?fs*
 using *5* **by** (*simp add*: *matrix-idempotent-semiring.mult-left-isotone*)
 also have ... $\preceq$ *?hd*
 using *16* **by** *simp*
 finally show *?thesis*
 by *simp*
 qed
 have *21*: *?ha* $\odot$ *?es* $\oplus$ *?hb* $\odot$ *?ds* $\odot$ *?c* $\odot$ *?es* $\preceq$ *?ha*
 using *13 17 matrix-semilattice-sup.le-supI* **by** *blast*
 have *22*: *?ha* $\odot$ *?as* $\odot$ *?b* $\odot$ *?fs* $\oplus$ *?hb* $\odot$ *?fs* $\preceq$ *?hb*
 using *14 18 matrix-semilattice-sup.le-supI* **by** *blast*
 have *23*: *?hc* $\odot$ *?es* $\oplus$ *?hd* $\odot$ *?ds* $\odot$ *?c* $\odot$ *?es* $\preceq$ *?hc*

using *15 19 matrix-semilattice-sup.le-supI* **by** *blast*
have *24*: *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hd* ⊙ *?fs* ≼ *?hd*
using *16 20 matrix-semilattice-sup.le-supI* **by** *blast*
have *zs*⟨*h*⟩*?t* ⊙ *star-matrix' ?t g* = *zs*⟨*h*⟩*?t* ⊙ (*?es* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?fs*)
by (*metis star-matrix'.simps*(*2*))
also have ... = *zs*⟨*h*⟩*?t* ⊙ *?es* ⊕ *zs*⟨*h*⟩*?t* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *zs*⟨*h*⟩*?t* ⊙ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *zs*⟨*h*⟩*?t* ⊙ *?fs*
by (*simp add*: *matrix-idempotent-semiring.mult-left-dist-sup matrix-monoid.mult-assoc*)
also have ... = *?ha* ⊙ *?es* ⊕ *?hc* ⊙ *?es* ⊕ *?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hb* ⊙ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?hd* ⊙ *?ds* ⊙ *?c* ⊙ *?es* ⊕ *?hb* ⊙ *?fs* ⊕ *?hd* ⊙ *?fs*
using *9 10 11 12* **by** (*simp add*: *matrix-semilattice-sup.sup-assoc*)
also have ... = (*?ha* ⊙ *?es* ⊕ *?hb* ⊙ *?ds* ⊙ *?c* ⊙ *?es*) ⊕ (*?ha* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hb* ⊙ *?fs*) ⊕ (*?hc* ⊙ *?es* ⊕ *?hd* ⊙ *?ds* ⊙ *?c* ⊙ *?es*) ⊕ (*?hc* ⊙ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?hd* ⊙ *?fs*)
using *9 10 11 12* **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 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
qed
hence ∀ *zs* . *distinct zs* ⟶ (*zs*⟨*x*⟩*?e* ⊙ *y* ≼ *zs*⟨*x*⟩*?e* ⟶ *zs*⟨*x*⟩*?e* ⊙ $y^{\odot}$ ≼ *zs*⟨*x*⟩*?e*)
by (*simp add*: *enum-distinct restrict-all*)
thus *x* ⊙ *y* ≼ *x* ⟶ *x* ⊙ $y^{\odot}$ ≼ *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*

let *?e* = *enum-class.enum*::*'a list*
let *?o* = *mone* :: (*'a*,*'b*) *square*
show $\ominus\ominus(x^{\odot}) = (\ominus\ominus x)^{\odot}$
proof (*rule matrix-order.antisym*)
have ∀ *g* :: (*'a*,*'b*) *square* . *distinct ?e* ⟶ ⊖⊖(*star-matrix' ?e* (⊖⊖*g*)) = *star-matrix' ?e* (⊖⊖*g*)
proof (*induct rule*: *list.induct*)
case *Nil* **thus** *?case*
by *simp*
next
case (*Cons k s*)
let *?t* = *k*#*s*
assume *1*: ∀ *g* :: (*'a*,*'b*) *square* . *distinct s* ⟶ ⊖⊖(*star-matrix' s* (⊖⊖*g*)) = *star-matrix' s* (⊖⊖*g*)
show ∀ *g* :: (*'a*,*'b*) *square* . *distinct ?t* ⟶ ⊖⊖(*star-matrix' ?t* (⊖⊖*g*)) = *star-matrix' ?t* (⊖⊖*g*)
proof (*rule allI*, *rule impI*)
fix *g* :: (*'a*,*'b*) *square*
assume *2*: *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*⟨*star o ?a*⟩*?r*
let *?ds* = *star-matrix' s ?d*
let *?e* = *?a* ⊕ *?b* ⊙ *?ds* ⊙ *?c*
let *?es* = *?r*⟨*star o ?e*⟩*?r*
let *?f* = *?d* ⊕ *?c* ⊙ *?as* ⊙ *?b*
let *?fs* = *star-matrix' s ?f*
have *s*⟨*?ds*⟩*s* = *?ds* ∧ *s*⟨*?fs*⟩*s* = *?fs*
by (*simp add*: *restrict-star*)
have *3*: ⊖⊖*?a* = *?a* ∧ ⊖⊖*?b* = *?b* ∧ ⊖⊖*?c* = *?c* ∧ ⊖⊖*?d* = *?d*
by (*metis matrix-p-algebra.regular-closed-p restrict-pp*)
hence *4*: ⊖⊖*?as* = *?as*
by (*metis pp-star-commute restrict-pp*)
hence ⊖⊖*?f* = *?f*
using *3* **by** (*metis matrix-stone-algebra.regular-closed-sup matrix-stone-relation-algebra.regular-mult-closed*)
hence *5*: ⊖⊖*?fs* = *?fs*
using *1 2* **by** (*metis distinct.simps*(*2*))
have *6*: ⊖⊖*?ds* = *?ds*
using *1 2* **by** (*simp add*: *restrict-pp*)
hence ⊖⊖*?e* = *?e*
using *3* **by** (*metis matrix-stone-algebra.regular-closed-sup matrix-stone-relation-algebra.regular-mult-closed*)
hence *7*: ⊖⊖*?es* = *?es*
by (*metis pp-star-commute restrict-pp*)
have ⊖⊖(*star-matrix' ?t* (⊖⊖*g*)) = ⊖⊖(*?es* ⊕ *?as* ⊙ *?b* ⊙ *?fs* ⊕ *?ds* ⊙ *?c*

$\odot$ *?es* $\oplus$ *?fs*)
by (*metis star-matrix'.simps*(*2*))
also have ... = $\ominus\ominus$*?es* $\oplus$ $\ominus\ominus$*?as* $\odot$ $\ominus\ominus$*?b* $\odot$ $\ominus\ominus$*?fs* $\oplus$ $\ominus\ominus$*?ds* $\odot$ $\ominus\ominus$*?c* $\odot$ $\ominus\ominus$*?es* $\oplus$ $\ominus\ominus$*?fs*
by (*simp add*: *matrix-stone-relation-algebra.pp-dist-comp*)
also have ... = *?es* $\oplus$ *?as* $\odot$ *?b* $\odot$ *?fs* $\oplus$ *?ds* $\odot$ *?c* $\odot$ *?es* $\oplus$ *?fs*
using *3 4 5 6 7* **by** *simp*
finally show $\ominus\ominus$(*star-matrix' ?t* ($\ominus\ominus g$)) = *star-matrix' ?t* ($\ominus\ominus g$)
by (*metis star-matrix'.simps*(*2*))
qed
qed
hence $(\ominus\ominus x)^{\odot} = \ominus\ominus((\ominus\ominus x)^{\odot})$
by (*simp add*: *enum-distinct restrict-all*)
thus $\ominus\ominus(x^{\odot}) \preceq (\ominus\ominus x)^{\odot}$
by (*metis matrix-kleene-algebra.star.circ-isotone matrix-p-algebra.pp-increasing matrix-p-algebra.pp-isotone*)
next
have *?o* $\oplus$ $\ominus\ominus x \odot \ominus\ominus(x^{\odot}) \preceq \ominus\ominus(x^{\odot})$
by (*metis matrix-kleene-algebra.star-left-unfold-equal matrix-p-algebra.sup-pp-semi-commute matrix-stone-relation-algebra.pp-dist-comp*)
thus $(\ominus\ominus x)^{\odot} \preceq \ominus\ominus(x^{\odot})$
using *matrix-kleene-algebra.star-left-induct* **by** *fastforce*
qed
qed

end

References

[1] A. Armstrong, S. Foster, G. Struth, and T. Weber. Relation algebra. *Archive of Formal Proofs*, 2016, first version 2014.

[2] A. Armstrong, V. B. F. Gomes, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2016, first version 2013.

[3] T. Asplund. Formalizing the Kleene star for square matrices. Bachelor Thesis IT 14 002, Uppsala Universitet, Department of Information Technology, 2014.

[4] R. J. R. Back and J. von Wright. Reasoning algebraically about loops. *Acta Inf.*, 36(4):295–334, 1999.

[5] S. L. Bloom and Z. Ésik. *Iteration Theories: The Equational Logic of Iterative Processes.* Springer, 1993.

[6] E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, pages 45–59. Springer, 2000.

[7] J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, 1971.

[8] S. Foster and G. Struth. Regular algebras. *Archive of Formal Proofs*, 2016, first version 2014.

[9] W. Guttmann. Algebras for iteration and infinite computations. *Acta Inf.*, 49(5):343–359, 2012.

[10] W. Guttmann. Relation-algebraic verification of Prim's minimum spanning tree algorithm. In A. Sampaio and F. Wang, editors, *International Colloquium on Theoretical Aspects of Computing*, volume 9965 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2016.

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

[12] W. Guttmann. Stone relation algebras. In P. Höfner, D. Pous, and G. Struth, editors, *Relational and Algebraic Methods in Computer Science*, volume 10226 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2017.

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

[14] D. Kozen. Typed Kleene algebra. Technical Report TR98-1669, Cornell University, 1998.

[15] B. Möller. Kleene getting lazy. *Sci. Comput. Programming*, 65(2):195–214, 2007.

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

[17] J. von Wright. Towards a refinement algebra. *Sci. Comput. Programming*, 51(1–2):23–45, 2004.