Stone-Kleene Relation Algebras

Walter Guttmann

February 23, 2021

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 and Kruskal's minimum spanning tree algorithms.

- 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* :: $'a \Rightarrow 'a$ (*-°* [*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*:
$1 \sqcup x * (y * x)^\circ * y \leq (x * y)^\circ$
⟨*proof*⟩

lemma *circ-right-unfold-sub*:
$1 \sqcup x^\circ * x \leq x^\circ$
⟨*proof*⟩

lemma *circ-zero*:
$bot^\circ = 1$
⟨*proof*⟩

lemma *circ-increasing*:
$x \leq x^\circ$
⟨*proof*⟩

lemma *circ-reflexive*:
$1 \leq x^\circ$
⟨*proof*⟩

lemma *circ-mult-increasing*:
$x \leq x * x^\circ$
⟨*proof*⟩

lemma *circ-mult-increasing-2*:
$x \leq x^\circ * x$
⟨*proof*⟩

lemma *circ-transitive-equal*:
$x^\circ * x^\circ = x^\circ$
⟨*proof*⟩

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

lemma *circ-circ-circ*:
$x^{\circ\circ\circ} = x^{\circ\circ}$
⟨*proof*⟩

lemma *circ-one*:
$1^\circ = 1^{\circ\circ}$
⟨*proof*⟩

lemma *circ-sup-sub*:
$(x^\circ * y)^\circ * x^\circ \leq (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-plus-one*:
$x^\circ = 1 \sqcup x^\circ$
⟨*proof*⟩

Iteration satisfies a characteristic property of reflexive transitive closures.

lemma *circ-rtc-2*:
$1 \sqcup x \sqcup x^\circ * x^\circ = x^\circ$
⟨*proof*⟩

lemma *mult-zero-circ*:
$(x * bot)^\circ = 1 \sqcup x * bot$
⟨*proof*⟩

lemma *mult-zero-sup-circ*:
$(x \sqcup y * bot)^\circ = x^\circ * (y * bot)^\circ$
⟨*proof*⟩

lemma *circ-plus-sub*:
$x^\circ * x \leq x * x^\circ$
⟨*proof*⟩

lemma *circ-loop-fixpoint*:
$y * (y^\circ * z) \sqcup z = y^\circ * z$
⟨*proof*⟩

lemma *left-plus-below-circ*:
$x * x^\circ \leq x^\circ$
⟨*proof*⟩

lemma *right-plus-below-circ*:
$x^\circ * x \leq x^\circ$
⟨*proof*⟩

lemma *circ-sup-upper-bound*:
$x \leq z^\circ \Longrightarrow y \leq z^\circ \Longrightarrow x \sqcup y \leq z^\circ$
⟨*proof*⟩

lemma *circ-mult-upper-bound*:
$x \leq z^\circ \Longrightarrow y \leq z^\circ \Longrightarrow x * y \leq z^\circ$
⟨*proof*⟩

lemma *circ-sub-dist*:
$x^\circ \leq (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-sub-dist-1*:
$x \le (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-sub-dist-2*:
$x * y \le (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-sub-dist-3*:
$x^\circ * y^\circ \le (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-isotone*:
$x \le y \Longrightarrow x^\circ \le y^\circ$
⟨*proof*⟩

lemma *circ-sup-2*:
$(x \sqcup y)^\circ \le (x^\circ * y^\circ)^\circ$
⟨*proof*⟩

lemma *circ-sup-one-left-unfold*:
$1 \le x \Longrightarrow x * x^\circ = x^\circ$
⟨*proof*⟩

lemma *circ-sup-one-right-unfold*:
$1 \le x \Longrightarrow x^\circ * x = x^\circ$
⟨*proof*⟩

lemma *circ-decompose-4*:
$(x^\circ * y^\circ)^\circ = x^\circ * (y^\circ * x^\circ)^\circ$
⟨*proof*⟩

lemma *circ-decompose-5*:
$(x^\circ * y^\circ)^\circ = (y^\circ * x^\circ)^\circ$
⟨*proof*⟩

lemma *circ-decompose-6*:
$x^\circ * (y * x^\circ)^\circ = y^\circ * (x * y^\circ)^\circ$
⟨*proof*⟩

lemma *circ-decompose-7*:
$(x \sqcup y)^\circ = x^\circ * y^\circ * (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-decompose-8*:
$(x \sqcup y)^\circ = (x \sqcup y)^\circ * x^\circ * y^\circ$
⟨*proof*⟩

lemma *circ-decompose-9*:
$(x^\circ * y^\circ)^\circ = x^\circ * y^\circ * (x^\circ * y^\circ)^\circ$
⟨*proof*⟩

lemma *circ-decompose-10*:
$(x^\circ * y^\circ)^\circ = (x^\circ * y^\circ)^\circ * x^\circ * y^\circ$
⟨*proof*⟩

lemma *circ-back-loop-prefixpoint*:
$(z * y^\circ) * y \sqcup z \leq z * y^\circ$
⟨*proof*⟩

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

lemma *circ-loop-is-fixpoint*:
is-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (y^\circ * z)$
⟨*proof*⟩

lemma *circ-back-loop-is-prefixpoint*:
is-prefixpoint $(\lambda x\ .\ x * y \sqcup z)\ (z * y^\circ)$
⟨*proof*⟩

lemma *circ-circ-sup*:
$(1 \sqcup x)^\circ = x^{\circ\circ}$
⟨*proof*⟩

lemma *circ-circ-mult-sub*:
$x^\circ * 1^\circ \leq x^{\circ\circ}$
⟨*proof*⟩

lemma *left-plus-circ*:
$(x * x^\circ)^\circ = x^\circ$
⟨*proof*⟩

lemma *right-plus-circ*:
$(x^\circ * x)^\circ = x^\circ$
⟨*proof*⟩

lemma *circ-square*:
$(x * x)^\circ \leq x^\circ$
⟨*proof*⟩

lemma *circ-mult-sub-sup*:
$(x * y)^\circ \leq (x \sqcup y)^\circ$
⟨*proof*⟩

lemma *circ-sup-mult-zero*:
$x^\circ * y = (x \sqcup y * bot)^\circ * y$
⟨*proof*⟩

lemma *troeger-1*:
$(x \sqcup y)^\circ = x^\circ * (1 \sqcup y * (x \sqcup y)^\circ)$
$\langle proof \rangle$

lemma *troeger-2*:
$(x \sqcup y)^\circ * z = x^\circ * (y * (x \sqcup y)^\circ * z \sqcup z)$
$\langle proof \rangle$

lemma *troeger-3*:
$(x \sqcup y * bot)^\circ = x^\circ * (1 \sqcup y * bot)$
$\langle proof \rangle$

lemma *circ-sup-sub-sup-one-1*:
$x \sqcup y \leq x^\circ * (1 \sqcup y)$
$\langle proof \rangle$

lemma *circ-sup-sub-sup-one-2*:
$x^\circ * (x \sqcup y) \leq x^\circ * (1 \sqcup y)$
$\langle proof \rangle$

lemma *circ-sup-sub-sup-one*:
$x * x^\circ * (x \sqcup y) \leq x * x^\circ * (1 \sqcup y)$
$\langle proof \rangle$

lemma *circ-square-2*:
$(x * x)^\circ * (x \sqcup 1) \leq x^\circ$
$\langle proof \rangle$

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

lemma *circ-circ-sub-mult*:
$1^\circ * x^\circ \leq x^{\circ\circ}$
$\langle proof \rangle$

lemma *circ-decompose-11*:
$(x^\circ * y^\circ)^\circ = (x^\circ * y^\circ)^\circ * x^\circ$
$\langle proof \rangle$

lemma *circ-mult-below-circ-circ*:
$(x * y)^\circ \leq (x^\circ * y)^\circ * x^\circ$
$\langle proof \rangle$

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*:
$top^\circ = top$
⟨*proof*⟩

lemma *circ-right-top*:
$x^\circ * top = top$
⟨*proof*⟩

lemma *circ-left-top*:
$top * x^\circ = top$
⟨*proof*⟩

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

end

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

lemma *mult-zero-sup-circ-2*:
$(x \sqcup y * bot)^\circ = x^\circ \sqcup x^\circ * y * bot$
⟨*proof*⟩

lemma *circ-unfold-sum*:
$(x \sqcup y)^\circ = x^\circ \sqcup x^\circ * y * (x \sqcup y)^\circ$
⟨*proof*⟩

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*:
$x * (y * x)^\circ = (x * y)^\circ * x$
⟨*proof*⟩

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

lemma *circ-right-unfold-1*:
$1 \sqcup x^\circ * x = x^\circ$

$\langle proof \rangle$

lemma *circ-mult-1*:
$(x * y)^\circ = 1 \sqcup x * (y * x)^\circ * y$
$\langle proof \rangle$

lemma *circ-sup-9*:
$(x \sqcup y)^\circ = (x^\circ * y)^\circ * x^\circ$
$\langle proof \rangle$

lemma *circ-plus-same*:
$x^\circ * x = x * x^\circ$
$\langle proof \rangle$

lemma *circ-decompose-12*:
$x^\circ * y^\circ \leq (x^\circ * y)^\circ * x^\circ$
$\langle proof \rangle$

end

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

lemma *circ-back-loop-fixpoint*:
$(z * y^\circ) * y \sqcup z = z * y^\circ$
$\langle proof \rangle$

lemma *circ-back-loop-is-fixpoint*:
$\mathit{is\text{-}fixpoint}\ (\lambda x\ .\ x * y \sqcup z)\ (z * y^\circ)$
$\langle proof \rangle$

lemma *circ-elimination*:
$x * y = \mathit{bot} \Longrightarrow x * y^\circ \leq x$
$\langle proof \rangle$

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*:
$1^\circ * x^\circ = x^{\circ\circ}$
$\langle proof \rangle$

lemma *sub-mult-one-circ*:
$x * 1^\circ \le 1^\circ * x$
⟨*proof*⟩

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

lemma *circ-import*:
assumes $p \le p * p$
and $p \le 1$
and $p * x \le x * p$
shows $p * x^\circ = p * (p * x)^\circ$
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *circ-simulate-left-1*:
$x * z \le z * y \Longrightarrow x^\circ * z \le z * y^\circ \sqcup x^\circ * bot$
⟨*proof*⟩

lemma *circ-separate-1*:
assumes $y * x \le x * y$
shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
⟨*proof*⟩

lemma *circ-circ-mult-1*:
$x^\circ * 1^\circ = x^{\circ\circ}$
⟨*proof*⟩

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 $y^\circ * x^\circ \le x^\circ * y^\circ$
⟨*proof*⟩

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 $s * (x \sqcup b \sqcup r \sqcup l)^\circ * q \leq s * (x * b^\circ * q \sqcup r \sqcup l)^\circ$
$\langle proof \rangle$

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*:
$1 \sqcup x^\circ * x = x^\circ$
$\langle proof \rangle$

lemma *circ-slide*:
$x * (y * x)^\circ = (x * y)^\circ * x$
$\langle proof \rangle$

subclass *itering-3*
$\langle proof \rangle$

lemma *circ-simulate-right-plus-1*:
$z * x \leq y * y^\circ * z \Longrightarrow z * x^\circ \leq y^\circ * z$
$\langle proof \rangle$

lemma *circ-simulate-left-plus-1*:
$x * z \leq z * y^\circ \Longrightarrow x^\circ * z \leq z * y^\circ \sqcup x^\circ * bot$
$\langle proof \rangle$

lemma *circ-simulate-2*:
$y * x^\circ \leq x^\circ * y^\circ \longleftrightarrow y^\circ * x^\circ \leq x^\circ * y^\circ$

$\langle proof \rangle$

lemma *circ-simulate-absorb*:
$y * x \le x \Longrightarrow y^\circ * x \le x \sqcup y^\circ * bot$
$\langle proof \rangle$

lemma *circ-simulate-3*:
$y * x^\circ \le x^\circ \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-separate-mult-1*:
$y * x \le x * y \Longrightarrow (x * y)^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-separate-unfold*:
$(y * x^\circ)^\circ = y^\circ \sqcup y^\circ * y * x * x^\circ * (y * x^\circ)^\circ$
$\langle proof \rangle$

lemma *separation*:
assumes $y * x \le x * y^\circ$
shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
$\langle proof \rangle$

lemma *simulation*:
$y * x \le x * y^\circ \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-simulate-4*:
assumes $y * x \le x * x^\circ * (1 \sqcup y)$
shows $y^\circ * x^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-simulate-5*:
$y * x \le x * x^\circ * (x \sqcup y) \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-simulate-6*:
$y * x \le x * (x \sqcup y) \Longrightarrow y^\circ * x^\circ \le x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-separate-4*:
assumes $y * x \le x * x^\circ * (1 \sqcup y)$
shows $(x \sqcup y)^\circ = x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-separate-5*:
$y * x \le x * x^\circ * (x \sqcup y) \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$
$\langle proof \rangle$

lemma *circ-separate-6*:
$y * x \leq x * (x \sqcup y) \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$
⟨*proof*⟩

end

class *bounded-itering* = *bounded-idempotent-left-zero-semiring* + *itering*
begin

subclass *bounded-left-conway-semiring* ⟨*proof*⟩

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*: $x * (y \sqcup L) \leq x * y \sqcup L$
begin

lemma *L-def*:
$L = 1^\circ * bot$
⟨*proof*⟩

lemma *one-circ-split*:
$1^\circ = L \sqcup 1$
⟨*proof*⟩

lemma *one-circ-circ-split*:
$1^{\circ\circ} = L \sqcup 1$
⟨*proof*⟩

lemma *sub-mult-one-circ*:
$x * 1^\circ \leq 1^\circ * x$
⟨*proof*⟩

lemma *one-circ-mult-split-2*:
$1^\circ * x = x * 1^\circ \sqcup L$
⟨*proof*⟩

lemma *sub-mult-one-circ-split*:
$x * 1^\circ \leq x \sqcup L$
⟨*proof*⟩

lemma *sub-mult-one-circ-split-2*:
$x * 1^{\circ} \leq x \sqcup 1^{\circ}$
⟨*proof*⟩

lemma *L-split*:
$x * L \leq x * bot \sqcup L$
⟨*proof*⟩

lemma *L-left-zero*:
$L * x = L$
⟨*proof*⟩

lemma *one-circ-L*:
$1^{\circ} * L = L$
⟨*proof*⟩

lemma *mult-L-circ*:
$(x * L)^{\circ} = 1 \sqcup x * L$
⟨*proof*⟩

lemma *mult-L-circ-mult*:
$(x * L)^{\circ} * y = y \sqcup x * L$
⟨*proof*⟩

lemma *circ-L*:
$L^{\circ} = L \sqcup 1$
⟨*proof*⟩

lemma *L-below-one-circ*:
$L \leq 1^{\circ}$
⟨*proof*⟩

lemma *circ-circ-mult-1*:
$x^{\circ} * 1^{\circ} = x^{\circ\circ}$
⟨*proof*⟩

lemma *circ-circ-mult*:
$1^{\circ} * x^{\circ} = x^{\circ\circ}$
⟨*proof*⟩

lemma *circ-circ-split*:
$x^{\circ\circ} = L \sqcup x^{\circ}$
⟨*proof*⟩

lemma *circ-sup-6*:
$L \sqcup (x \sqcup y)^{\circ} = (x^{\circ} * y^{\circ})^{\circ}$
⟨*proof*⟩

end

class *itering-L* = *itering* + *L* +
  assumes *L-def*: $L = 1^\circ * bot$
begin

lemma *one-circ-split*:
  $1^\circ = L \sqcup 1$
  ⟨*proof*⟩

lemma *one-circ-mult-split*:
  $1^\circ * x = L \sqcup x$
  ⟨*proof*⟩

lemma *sub-mult-one-circ-split*:
  $x * 1^\circ \le x \sqcup L$
  ⟨*proof*⟩

lemma *sub-mult-one-circ-split-2*:
  $x * 1^\circ \le x \sqcup 1^\circ$
  ⟨*proof*⟩

lemma *L-split*:
  $x * L \le x * bot \sqcup L$
  ⟨*proof*⟩

subclass *left-conway-semiring-L*
  ⟨*proof*⟩

lemma *circ-left-induct-mult-L*:
  $L \le x \Longrightarrow x * y \le x \Longrightarrow x * y^\circ \le x$
  ⟨*proof*⟩

lemma *circ-left-induct-mult-iff-L*:
  $L \le x \Longrightarrow x * y \le x \longleftrightarrow x * y^\circ \le x$
  ⟨*proof*⟩

lemma *circ-left-induct-L*:
  $L \le x \Longrightarrow x * y \sqcup z \le x \Longrightarrow z * y^\circ \le x$
  ⟨*proof*⟩

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* : $1 \sqcup y * y^\star \le y^\star$
  assumes *star-left-induct* : $z \sqcup y * x \le x \longrightarrow y^\star * z \le x$
begin

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

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

lemma *star-left-unfold-equal*:
  $1 \sqcup x * x^\star = x^\star$
  ⟨*proof*⟩

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*:
  $(x * y)^\star * x \le x * (y * x)^\star$
  ⟨*proof*⟩

lemma *star-isotone*:
  $x \le y \Longrightarrow x^\star \le y^\star$
  ⟨*proof*⟩

lemma *star-sup-1*:
  $(x \sqcup y)^\star = x^\star * (y * x^\star)^\star$
⟨*proof*⟩

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*

⟨*proof*⟩

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*:
$x \leq 1 \Longrightarrow x^\star = 1$
⟨*proof*⟩

lemma *star-one*:
$1^\star = 1$
⟨*proof*⟩

lemma *star-left-induct-mult*:
$x * y \leq y \Longrightarrow x^\star * y \leq y$
⟨*proof*⟩

lemma *star-left-induct-mult-iff*:
$x * y \leq y \longleftrightarrow x^\star * y \leq y$
⟨*proof*⟩

lemma *star-involutive*:
$x^\star = x^{\star\star}$
⟨*proof*⟩

lemma *star-sup-one*:
$(1 \sqcup x)^\star = x^\star$
⟨*proof*⟩

lemma *star-plus-loops*:
$x^\star \sqcup 1 = x^+ \sqcup 1$
⟨*proof*⟩

lemma *star-left-induct-equal*:
$z \sqcup x * y = y \Longrightarrow x^\star * z \leq y$
⟨*proof*⟩

lemma *star-left-induct-mult-equal*:
$x * y = y \Longrightarrow x^\star * y \leq y$
⟨*proof*⟩

lemma *star-star-upper-bound*:
$x^\star \leq z^\star \Longrightarrow x^{\star\star} \leq z^\star$
⟨*proof*⟩

lemma *star-simulation-left*:
assumes $x * z \leq z * y$

shows $x^\star * z \leq z * y^\star$
$\langle proof \rangle$

lemma *quasicomm-1*:
$y * x \leq x * (x \sqcup y)^\star \longleftrightarrow y^\star * x \leq x * (x \sqcup y)^\star$
$\langle proof \rangle$

lemma *star-rtc-3*:
$1 \sqcup x \sqcup y * y = y \Longrightarrow x^\star \leq y$
$\langle proof \rangle$

lemma *star-decompose-1*:
$(x \sqcup y)^\star = (x^\star * y^\star)^\star$
$\langle proof \rangle$

lemma *star-sum*:
$(x \sqcup y)^\star = (x^\star \sqcup y^\star)^\star$
$\langle proof \rangle$

lemma *star-decompose-3*:
$(x^\star * y^\star)^\star = x^\star * (y * x^\star)^\star$
$\langle proof \rangle$

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

lemma *star-loop-least-fixpoint*:
$y * x \sqcup z = x \Longrightarrow y^\star * z \leq x$
$\langle proof \rangle$

lemma *star-loop-is-least-fixpoint*:
is-least-fixpoint $(\lambda x . y * x \sqcup z)$ $(y^\star * z)$
$\langle proof \rangle$

lemma *star-loop-mu*:
μ $(\lambda x . y * x \sqcup z) = y^\star * z$
$\langle proof \rangle$

lemma *affine-has-least-fixpoint*:
has-least-fixpoint $(\lambda x . y * x \sqcup z)$
$\langle proof \rangle$

lemma *star-outer-increasing*:
$x \leq y^\star * x * y^\star$
$\langle proof \rangle$

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*⟩

lemma *star-slide*:
  $(x * y)^\star * x = x * (y * x)^\star$
⟨*proof*⟩

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

end

Again we inherit results from the itering hierarchy.

sublocale *strong-left-kleene-algebra* < *star*: *itering-1* **where** *circ* = *star*
  ⟨*proof*⟩

context *strong-left-kleene-algebra*
begin

lemma *star-right-induct-mult*:
  $y * x \leq y \Longrightarrow y * x^\star \leq y$
  ⟨*proof*⟩

lemma *star-right-induct-mult-iff*:
  $y * x \leq y \longleftrightarrow y * x^\star \leq y$
  ⟨*proof*⟩

lemma *star-simulation-right-equal*:
  $z * x = y * z \Longrightarrow z * x^\star = y^\star * z$
  ⟨*proof*⟩

lemma *star-simulation-star*:
  $x * y \leq y * x \Longrightarrow x^\star * y^\star \leq y^\star * x^\star$
  ⟨*proof*⟩

lemma *star-right-induct-equal*:
  $z \sqcup y * x = y \Longrightarrow z * x^\star \leq y$
  ⟨*proof*⟩

lemma *star-right-induct-mult-equal*:
  $y * x = y \Longrightarrow y * x^\star \leq y$

$\langle proof \rangle$

lemma *star-back-loop-least-fixpoint*:
$x * y \sqcup z = x \Longrightarrow z * y^\star \le x$
$\langle proof \rangle$

lemma *star-back-loop-is-least-fixpoint*:
is-least-fixpoint $(\lambda x . x * y \sqcup z)$ $(z * y^\star)$
$\langle proof \rangle$

lemma *star-back-loop-mu*:
μ $(\lambda x . x * y \sqcup z) = z * y^\star$
$\langle proof \rangle$

lemma *star-square*:
$x^\star = (1 \sqcup x) * (x * x)^\star$
$\langle proof \rangle$

lemma *star-square-2*:
$x^\star = (x * x)^\star * (x \sqcup 1)$
$\langle proof \rangle$

lemma *star-circ-simulate-right-plus*:
assumes $z * x \le y * y^\star * z \sqcup w$
shows $z * x^\star \le y^\star * (z \sqcup w * x^\star)$
$\langle proof \rangle$

lemma *transitive-star*:
$x * x \le x \Longrightarrow x^\star = 1 \sqcup x$
$\langle proof \rangle$

lemma *star-sup-2*:
assumes $x * x \le x$
and $x * y \le x$
shows $(x \sqcup y)^\star = y^\star * (x \sqcup 1)$
$\langle proof \rangle$

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*:
$y^\star * (y^\star * x)^\star * y^\star = (y^\star * x)^\star * y^\star$

$\langle proof \rangle$

lemma *star-circ-simulate-left-plus*:
assumes $x * z \leq z * y^\star \sqcup w$
shows $x^\star * z \leq (z \sqcup x^\star * w) * y^\star$
$\langle proof \rangle$

lemma *star-one-sup-below*:
$x * y^\star * (1 \sqcup z) \leq x * (y \sqcup z)^\star$
$\langle proof \rangle$

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 $x^\star * y^\star \leq x^\star \sqcup y^\star$
$\langle proof \rangle$

lemma *star-separate*:
assumes $x * y = bot$
and $y * y = bot$
shows $(x \sqcup y)^\star = x^\star \sqcup y * x^\star$
$\langle proof \rangle$

end

We can now inherit from the strongest variant of iterings.

sublocale *left-zero-kleene-algebra* < *star*: *itering* **where** $circ = star$
$\langle proof \rangle$

context *left-zero-kleene-algebra*
begin

lemma *star-absorb*:
$x * y = bot \Longrightarrow x * y^\star = x$
$\langle proof \rangle$

lemma *star-separate-2*:
assumes $x * z^+ * y = bot$
and $y * z^+ * y = bot$
and $z * x = bot$
shows $(x^\star \sqcup y * x^\star) * (z * (1 \sqcup y * x^\star))^\star = z^\star * (x^\star \sqcup y * x^\star) * z^\star$
$\langle proof \rangle$

lemma *cancel-separate-eq*:
$x * y \leq 1 \Longrightarrow x^\star * y^\star = x^\star \sqcup y^\star$
$\langle proof \rangle$

lemma *cancel-separate-1*:
assumes $x * y \leq 1$
shows $(x \sqcup y)^\star = y^\star * x^\star$
⟨*proof*⟩

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

lemma *plus-half-bot*:
$x * y * x = bot \Longrightarrow (x * y)^+ = x * y$
⟨*proof*⟩

lemma *cancel-separate-1-sup*:
assumes $x * y \leq 1$
and $y * x \leq 1$
shows $(x \sqcup y)^\star = x^\star \sqcup y^\star$
⟨*proof*⟩

Lemma *star-separate-3* was contributed by Nicolas Robinson-O'Brien.

lemma *star-separate-3*:
assumes $y * x^\star * y \leq y$
shows $(x \sqcup y)^\star = x^\star \sqcup x^\star * y * x^\star$
⟨*proof*⟩

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*:
$x^\star \leq x^\circ$
⟨*proof*⟩

lemma *star-zero-below-circ-mult*:
$x^\star * bot \leq x^\circ * y$
⟨*proof*⟩

lemma *star-mult-circ*:
$x^\star * x^\circ = x^\circ$
⟨*proof*⟩

lemma *circ-mult-star*:
$x^\circ * x^\star = x^\circ$
⟨*proof*⟩

lemma *circ-star*:
$x^{\circ\star} = x^\circ$
⟨*proof*⟩

lemma *star-circ*:
$x^{\star\circ} = x^{\circ\circ}$
⟨*proof*⟩

lemma *circ-sup-3*:
$(x^\circ * y^\circ)^\star \leq (x \sqcup y)^\circ$
⟨*proof*⟩

end

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

subclass *left-kleene-conway-semiring* ⟨*proof*⟩

lemma *circ-isolate*:
$x^\circ = x^\circ * bot \sqcup x^\star$
⟨*proof*⟩

lemma *circ-isolate-mult*:
$x^\circ * y = x^\circ * bot \sqcup x^\star * y$
⟨*proof*⟩

lemma *circ-isolate-mult-sub*:
$x^\circ * y \leq x^\circ \sqcup x^\star * y$
⟨*proof*⟩

lemma *circ-sub-decompose*:
$(x^\circ * y)^\circ \leq (x^\star * y)^\circ * x^\circ$
⟨*proof*⟩

lemma *circ-sup-4*:
$(x \sqcup y)^\circ = (x^\star * y)^\circ * x^\circ$
⟨*proof*⟩

lemma *circ-sup-5*:
$(x^\circ * y)^\circ * x^\circ = (x^\star * y)^\circ * x^\circ$
⟨*proof*⟩

lemma *plus-circ*:

$(x^\star * x)^\circ = x^\circ$
⟨*proof*⟩

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** *circc* = *star* ⟨*proof*⟩

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* ⟨*proof*⟩

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

sublocale *bounded-kleene-algebra* < *star*: *bounded-itering* **where** *circ* = *star* ⟨*proof*⟩

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*
⟨*proof*⟩

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* $= \lambda x$. *top* **and** *one* $=$ *top* **and** *times* $=$ *inf*
⟨*proof*⟩

We add the Kleene star operation to each of bounded distributive allegories, pseudocomplemented distributive allegories and Stone relation algebras. We start with single-object bounded distributive allegories.

class *bounded-distrib-kleene-allegory* $=$ *bounded-distrib-allegory* $+$ *kleene-algebra*
begin

subclass *bounded-kleene-algebra* ⟨*proof*⟩

lemma *conv-star-conv*:
$x^\star \leq x^{T\star T}$
⟨*proof*⟩

It follows that star and converse commute.

lemma *conv-star-commute*:
$x^{\star T} = x^{T\star}$
⟨*proof*⟩

lemma *conv-plus-commute*:
$x^{+T} = x^{T+}$
⟨*proof*⟩

Lemma *reflexive-inf-star* was contributed by Nicolas Robinson-O'Brien.

lemma *reflexive-inf-star*:
assumes *reflexive* y
shows $y \sqcap x^\star = 1 \sqcup (y \sqcap x^+)$
⟨*proof*⟩

The following results are variants of a separation lemma of Kleene algebras.

lemma *cancel-separate-2*:
assumes $x * y \leq 1$
shows $((w \sqcap x) \sqcup (z \sqcap y))^\star = (z \sqcap y)^\star * (w \sqcap x)^\star$
⟨*proof*⟩

lemma *cancel-separate-3*:
  assumes $x * y \leq 1$
    shows $(w \sqcap x)^\star * (z \sqcap y)^\star = (w \sqcap x)^\star \sqcup (z \sqcap y)^\star$
$\langle proof \rangle$

lemma *cancel-separate-4*:
  assumes $z * y \leq 1$
      and $w \leq y \sqcup z$
      and $x \leq y \sqcup z$
    shows $w^\star * x^\star = (w \sqcap y)^\star * ((w \sqcap z)^\star \sqcup (x \sqcap y)^\star) * (x \sqcap z)^\star$
$\langle proof \rangle$

lemma *cancel-separate-5*:
  assumes $w * z^T \leq 1$
    shows $w \sqcap x * (y \sqcap z) \leq y$
$\langle proof \rangle$

lemma *cancel-separate-6*:
  assumes $z * y \leq 1$
      and $w \leq y \sqcup z$
      and $x \leq y \sqcup z$
      and $v * z^T \leq 1$
      and $v \sqcap y^\star = bot$
    shows $v \sqcap w^\star * x^\star \leq x \sqcup w$
$\langle proof \rangle$

We show several results about the interaction of vectors and the Kleene star.

lemma *vector-star-1*:
  assumes *vector* x
    shows $x^T * (x * x^T)^\star \leq x^T$
$\langle proof \rangle$

lemma *vector-star-2*:
  vector $x \Longrightarrow x^T * (x * x^T)^\star \leq x^T * bot^\star$
  $\langle proof \rangle$

lemma *vector-vector-star*:
  vector $v \Longrightarrow (v * v^T)^\star = 1 \sqcup v * v^T$
  $\langle proof \rangle$

lemma *equivalence-star-closed*:
  equivalence $x \Longrightarrow$ *equivalence* $(x^\star)$
  $\langle proof \rangle$

lemma *equivalence-plus-closed*:
  equivalence $x \Longrightarrow$ *equivalence* (x^+)
  $\langle proof \rangle$

The following equivalence relation characterises the component trees of a forest. This is a special case of undirected reachability in a directed graph.

abbreviation *forest-components* $f \equiv f^{T\star} * f^\star$

lemma *forest-components-equivalence*:
injective $x \Longrightarrow$ *equivalence* (*forest-components* x)
⟨*proof*⟩

lemma *forest-components-increasing*:
$x \leq$ *forest-components* x
⟨*proof*⟩

lemma *forest-components-isotone*:
$x \leq y \Longrightarrow$ *forest-components* $x \leq$ *forest-components* y
⟨*proof*⟩

lemma *forest-components-idempotent*:
injective $x \Longrightarrow$ *forest-components* (*forest-components* x) $=$ *forest-components* x
⟨*proof*⟩

lemma *forest-components-star*:
injective $x \Longrightarrow$ (*forest-components* x)* $=$ *forest-components* x
⟨*proof*⟩

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*⟩

lemma *kruskal-acyclic-inv-1*:
assumes *injective* f
and $e *$ *forest-components* $f * e = bot$
shows $(f \sqcap top * e * f^{T\star})^T * f^\star * e = bot$
⟨*proof*⟩

lemma *kruskal-forest-components-inf-1*:
assumes $f \leq w \sqcup w^T$
and *injective* w
and $f \leq$ *forest-components* g
shows $f *$ *forest-components* (*forest-components* $g \sqcap w$) $\leq$ *forest-components* (*forest-components* $g \sqcap w$)
⟨*proof*⟩

lemma *kruskal-forest-components-inf*:
assumes $f \leq w \sqcup w^T$
and *injective* w
shows *forest-components* $f \leq$ *forest-components* (*forest-components* $f \sqcap w$)
⟨*proof*⟩

end

We next add the Kleene star to single-object pseudocomplemented distributive allegories.

class *pd-kleene-allegory* = *pd-allegory* + *bounded-distrib-kleene-allegory*
begin

The following definitions and results concern acyclic graphs and forests.

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

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

lemma *forest-bot*:
forest bot
⟨*proof*⟩

lemma *acyclic-down-closed*:
$x \leq y \Longrightarrow$ *acyclic* $y \Longrightarrow$ *acyclic* x
⟨*proof*⟩

lemma *forest-down-closed*:
$x \leq y \Longrightarrow$ *forest* $y \Longrightarrow$ *forest* x
⟨*proof*⟩

lemma *acyclic-star-below-complement*:
acyclic $w \longleftrightarrow w^{T\star} \leq -w$
⟨*proof*⟩

lemma *acyclic-star-below-complement-1*:
acyclic $w \longleftrightarrow w^\star \sqcap w^T = bot$
⟨*proof*⟩

lemma *acyclic-star-inf-conv*:
assumes *acyclic* w
shows $w^\star \sqcap w^{T\star} = 1$
⟨*proof*⟩

lemma *acyclic-asymmetric*:
acyclic $w \Longrightarrow$ *asymmetric* w
⟨*proof*⟩

lemma *forest-separate*:
assumes *forest* x
shows $x^\star * x^{T\star} \sqcap x^T * x \leq 1$
⟨*proof*⟩

The following definition captures the components of undirected weighted graphs.

abbreviation *components* $g \equiv (--g)^\star$

lemma *components-equivalence*:
$symmetric\ x \Longrightarrow equivalence\ (components\ x)$
⟨*proof*⟩

lemma *components-increasing*:
$x \leq components\ x$
⟨*proof*⟩

lemma *components-isotone*:
$x \leq y \Longrightarrow components\ x \leq components\ y$
⟨*proof*⟩

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*⟩

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 $v^T = r^T * t^\star$
and *forest* w
and $t \leq w$
and $w \leq (r^T * g^\star)^T * (r^T * g^\star) \sqcap g$
and $r^T * g^\star \leq r^T * w^\star$
shows $w * v \leq v$
⟨*proof*⟩

4.1 Prim's Algorithm

The following results are used for proving the correctness of Prim's minimum spanning tree algorithm.

4.1.1 Preservation of Invariant

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*⟩

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 $e \le v * -v^T$
and $t \le v * v^T$
shows *acyclic* $(t \sqcup e)$
⟨*proof*⟩

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 $t \le v * v^T \sqcap --g$
and $v^T = r^T * t^\star$
and $e \le v * -v^T \sqcap --g$
and *vector* v
and *regular* $((v \sqcup e^T * top) * (v \sqcup e^T * top)^T)$
shows $t \sqcup e \le (r^T * (--((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g))^\star)^T * (r^T * (--((v \sqcup e^T * top) * (v \sqcup e^T * top)^T \sqcap g))^\star)$
⟨*proof*⟩

lemma *span-inv*:
assumes $e \le v * -v^T$
and *vector* v
and *arc* e
and $t \le (v * v^T) \sqcap g$
and $g^T = g$
and $v^T = r^T * t^\star$
and *injective* r
and $r^T \le v^T$
and $r^T * ((v * v^T) \sqcap g)^\star \le r^T * t^\star$
shows $r^T * (((v \sqcup e^T * top) * (v \sqcup e^T * top)^T) \sqcap g)^\star \le r^T * (t \sqcup e)^\star$
⟨*proof*⟩

4.1.2 Exchange gives Spanning Trees

The following abbreviations are used in the spanning tree application using Prim's algorithm 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 $prim\text{-}E$ :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $prim\text{-}E\ w\ v\ e \equiv w \sqcap --v * -v^T \sqcap top * e * w^{T\star}$
abbreviation $prim\text{-}P$ :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $prim\text{-}P\ w\ v\ e \equiv w \sqcap -v * -v^T \sqcap top * e * w^{T\star}$
abbreviation $prim\text{-}EP$ :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $prim\text{-}EP\ w\ v\ e \equiv w \sqcap -v^T \sqcap top * e * w^{T\star}$
abbreviation $prim\text{-}W$ :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ **where** $prim\text{-}W\ w\ v\ e \equiv (w \sqcap -(prim\text{-}EP\ w\ v\ e)) \sqcup (prim\text{-}P\ w\ v\ e)^T \sqcup e$

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 $(w \sqcap -(prim\text{-}EP\ w\ v\ e)) * e^T = bot$
⟨*proof*⟩

lemma *exchange-injective-6*:
  assumes *arc* e
    and *forest* w
  shows $(prim\text{-}P\ w\ v\ e)^T * e^T = bot$
⟨*proof*⟩

The graph after exchanging is injective.

lemma *exchange-injective*:
  assumes *arc* e
    and $e \leq v * -v^T$
    and *forest* w
    and *vector* v
  shows *injective* $(prim\text{-}W\ w\ v\ e)$
⟨*proof*⟩

lemma *pv*:
  assumes *vector* v
  shows $(prim\text{-}P\ w\ v\ e)^T * v = bot$
⟨*proof*⟩

lemma *vector-pred-inv*:
  assumes *arc* e
    and $e \leq v * -v^T$
    and *forest* w
    and *vector* v
    and $w * v \leq v$
  shows $(prim\text{-}W\ w\ v\ e) * (v \sqcup e^T * top) \leq v \sqcup e^T * top$
⟨*proof*⟩

The graph after exchanging is acyclic.

lemma *exchange-acyclic*:

assumes *vector* v
and $e \leq v * -v^T$
and $w * v \leq v$
and *acyclic* w
shows *acyclic* (*prim-W* $w\ v\ e$)
⟨*proof*⟩

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*⟩

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 $t \leq$ *prim-W* $w\ v\ e$
⟨*proof*⟩

lemma *mst-extends-new-tree*:
$t \leq w \Longrightarrow t \leq v * v^T \Longrightarrow$ *vector* $v \Longrightarrow t \sqcup e \leq$ *prim-W* $w\ v\ e$
⟨*proof*⟩

Lemmas *forests-bot-1*, *forests-bot-2*, *forests-bot-3* and *fc-comp-eq-fc* were contributed by Nicolas Robinson-O'Brien.

lemma *forests-bot-1*:
assumes *equivalence* e
and *forest* f
shows $(-e \sqcap f) * (e \sqcap f)^T = bot$
⟨*proof*⟩

lemma *forests-bot-2*:
assumes *equivalence* e
and *forest* f
shows $(-e \sqcap f^T) * x \sqcap (e \sqcap f^T) * y = bot$
⟨*proof*⟩

lemma *forests-bot-3*:
assumes *equivalence* e
and *forest* f
shows $x * (-e \sqcap f) \sqcap y * (e \sqcap f) = bot$
⟨*proof*⟩

end

We finally add the Kleene star to Stone relation algebras. 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* + *pd-kleene-allegory* +
  assumes *pp-dist-star*: $--(x^\star) = (--x)^\star$
begin

lemma *reachable-without-loops*:
  $x^\star = (x \sqcap -1)^\star$
⟨*proof*⟩

lemma *plus-reachable-without-loops*:
  $x^+ = (x \sqcap -1)^+ \sqcup (x \sqcap 1)$
  ⟨*proof*⟩

lemma *star-plus-without-loops*:
  $x^\star \sqcap -1 = x^+ \sqcap -1$
  ⟨*proof*⟩

lemma *regular-closed-star*:
  regular $x \Longrightarrow$ *regular* $(x^\star)$
  ⟨*proof*⟩

lemma *components-idempotent*:
  components (*components* x) = *components* x
  ⟨*proof*⟩

lemma *fc-comp-eq-fc*:
  −*forest-components* $(--f)$ = −*forest-components* f
  ⟨*proof*⟩

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* (*prim-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 $r^T * w^\star \leq r^T * (\textit{prim-W}\ w\ v\ e)^\star$
⟨*proof*⟩

Some of the following lemmas already hold in pseudocomplemented distributive Kleene allegories.

4.1.3 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*:
$vector\ v \Longrightarrow prim\text{-}E\ w\ v\ e \sqcup prim\text{-}P\ w\ v\ e = prim\text{-}EP\ w\ v\ e$
⟨*proof*⟩

lemma *epm-2*:
assumes $regular\ (prim\text{-}EP\ w\ v\ e)$
and $vector\ v$
shows $(w \sqcap -(prim\text{-}EP\ w\ v\ e)) \sqcup prim\text{-}P\ w\ v\ e \sqcup prim\text{-}E\ w\ v\ e = w$
⟨*proof*⟩

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*⟩

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 $prim\text{-}P\ w\ v\ e = bot$
⟨*proof*⟩

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 $prim\text{-}E\ w\ v\ e = e$
⟨*proof*⟩

lemma *epm-7*:
$regular\ (prim\text{-}EP\ w\ v\ e) \Longrightarrow e \leq w \Longrightarrow injective\ w \Longrightarrow w * v \leq v \Longrightarrow e \leq v * -v^T \Longrightarrow vector\ v \Longrightarrow prim\text{-}W\ w\ v\ e = w$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *epm-9*:
assumes $e \le v * -v^T$
and *vector* v
shows $(w \sqcap -(\mathit{prim\text{-}EP}\ w\ v\ e)) \sqcap e = \mathit{bot}$
⟨*proof*⟩

lemma *epm-10*:
assumes $e \le v * -v^T$
and *vector* v
shows $(\mathit{prim\text{-}P}\ w\ v\ e)^T \sqcap e = \mathit{bot}$
⟨*proof*⟩

lemma *epm-11*:
assumes *vector* v
shows $(w \sqcap -(\mathit{prim\text{-}EP}\ w\ v\ e)) \sqcap \mathit{prim\text{-}P}\ w\ v\ e = \mathit{bot}$
⟨*proof*⟩

lemma *epm-12*:
assumes *vector* v
shows $(w \sqcap -(\mathit{prim\text{-}EP}\ w\ v\ e)) \sqcap \mathit{prim\text{-}E}\ w\ v\ e = \mathit{bot}$
⟨*proof*⟩

lemma *epm-13*:
assumes *vector* v
shows $\mathit{prim\text{-}P}\ w\ v\ e \sqcap \mathit{prim\text{-}E}\ w\ v\ e = \mathit{bot}$
⟨*proof*⟩

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

lemma *arc-edge-1*:
assumes $e \le v * -v^T \sqcap g$
and *vector* v
and $v^T = r^T * t^\star$
and $t \le g$
and $r^T * g^\star \le r^T * w^\star$
shows $\mathit{top} * e \le v^T * w^\star$
⟨*proof*⟩

lemma *arc-edge-2*:
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 $w * v \le v$
and *injective* w
shows $\mathit{top} * e * w^{T\star} \le v^T * w^\star$

⟨*proof*⟩

lemma *arc-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 *prim-E w v e* $=$ *bot*
shows $e =$ *bot*
⟨*proof*⟩

lemma *arc-edge-4*:
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 *arc e*
shows *top* $*$ *prim-E w v e* $*$ *top* $=$ *top*
⟨*proof*⟩

lemma *arc-edge-5*:
assumes *vector v*
and $w * v \leq v$
and *injective w*
and *arc e*
shows $($*prim-E w v e*$)^T *$ *top* $*$ *prim-E w v e* $\leq$ *1*
⟨*proof*⟩

lemma *arc-edge-6*:
assumes *vector v*
and $w * v \leq v$
and *injective w*
and *arc e*
shows *prim-E w v e* $*$ *top* $* ($*prim-E w v e*$)^T \leq$ *1*
⟨*proof*⟩

lemma *arc-edge*:
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 *arc e*

shows *arc* (*prim-E* w v e)
⟨*proof*⟩

4.1.4 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*⟩

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*⟩

4.2 Kruskal's Algorithm

The following results are used for proving the correctness of Kruskal's minimum spanning tree algorithm.

4.2.1 Preservation of Invariant

We first treat the preservation of the invariant. The following lemmas show conditions necessary for preserving that f is a forest.

lemma *kruskal-injective-inv-2*:
assumes *arc* e
and *acyclic* f
shows $top * e * f^{T\star} * f^T \leq -e$
⟨*proof*⟩

lemma *kruskal-injective-inv-3*:

assumes *arc* e
and *forest* f
shows $(top * e * f^{T\star})^T * (top * e * f^{T\star}) \sqcap f^T * f \leq 1$
$\langle proof \rangle$

lemma *kruskal-acyclic-inv*:
assumes *acyclic* f
and *covector* q
and $(f \sqcap q)^T * f^\star * e = bot$
and $e * f^\star * e = bot$
and $f^{T\star} * f^\star \leq -e$
shows *acyclic* $((f \sqcap -q) \sqcup (f \sqcap q)^T \sqcup e)$
$\langle proof \rangle$

lemma *kruskal-exchange-acyclic-inv-1*:
assumes *acyclic* f
and *covector* q
shows *acyclic* $((f \sqcap -q) \sqcup (f \sqcap q)^T)$
$\langle proof \rangle$

lemma *kruskal-exchange-acyclic-inv-2*:
assumes *acyclic* w
and *injective* w
and $d \leq w$
and *bijective* $(d^T * top)$
and *bijective* $(e * top)$
and $d \leq top * e^T * w^{T\star}$
and $w * e^T * top = bot$
shows *acyclic* $((w \sqcap -d) \sqcup e)$
$\langle proof \rangle$

4.2.2 Exchange gives Spanning Trees

The lemmas in this section are used to show that the relation after exchange represents a spanning tree.

lemma *inf-star-import*:
assumes $x \leq z$
and *univalent* z
and *reflexive* y
and *regular* z
shows $x^\star * y \sqcap z^\star \leq x^\star * (y \sqcap z^\star)$
$\langle proof \rangle$

lemma *kruskal-exchange-forest-components-inv*:
assumes *injective* $((w \sqcap -d) \sqcup e)$
and *regular* d
and $e * top * e = e$
and $d \leq top * e^T * w^{T\star}$
and $w * e^T * top = bot$

and *injective* w
and $d \leq w$
and $d \leq (w \sqcap -d)^{T\star} * e^T * top$
shows *forest-components* $w \leq$ *forest-components* $((w \sqcap -d) \sqcup e)$
⟨*proof*⟩

lemma *kruskal-spanning-inv*:
assumes *injective* $((f \sqcap -q) \sqcup (f \sqcap q)^T \sqcup e)$
and *regular* q
and *regular* e
and $(-h \sqcap --g)^\star \leq$ *forest-components* f
shows *components* $(-(h \sqcap -e \sqcap -e^T) \sqcap g) \leq$ *forest-components* $((f \sqcap -q) \sqcup (f \sqcap q)^T \sqcup e)$
⟨*proof*⟩

lemma *kruskal-exchange-spanning-inv-1*:
assumes *injective* $((w \sqcap -q) \sqcup (w \sqcap q)^T)$
and *regular* $(w \sqcap q)$
and *components* $g \leq$ *forest-components* w
shows *components* $g \leq$ *forest-components* $((w \sqcap -q) \sqcup (w \sqcap q)^T)$
⟨*proof*⟩

lemma *kruskal-exchange-spanning-inv-2*:
assumes *injective* w
and $w^\star * e^T = e^T$
and $f \sqcup f^T \leq (w \sqcap -d \sqcap -d^T) \sqcup (w^T \sqcap -d \sqcap -d^T)$
and $d \leq$ *forest-components* $f * e^T * top$
shows $d \leq (w \sqcap -d)^{T\star} * e^T * top$
⟨*proof*⟩

lemma *kruskal-spanning-inv-1*:
assumes $e \leq F$
and *regular* e
and *components* $(-h \sqcap g) \leq F$
and *equivalence* F
shows *components* $(-(h \sqcap -e \sqcap -e^T) \sqcap g) \leq F$
⟨*proof*⟩

lemma *kruskal-reroot-edge*:
assumes *injective* $(e^T * top)$
and *acyclic* w
shows $((w \sqcap -(top * e * w^{T\star})) \sqcup (w \sqcap top * e * w^{T\star})^T) * e^T = bot$
⟨*proof*⟩

4.2.3 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 that the relation characterising the edge across the cut is an arc.

lemma *kruskal-edge-arc*:
assumes *equivalence* F
and *forest* w
and *arc* e
and *regular* F
and $F \leq$ *forest-components* $(F \sqcap w)$
and *regular* w
and $w * e^T = bot$
and $e * F * e = bot$
and $e^T \leq w^\star$
shows *arc* $(w \sqcap top * e^T * w^{T\star} \sqcap F * e^T * top \sqcap top * e * -F)$
$\langle proof \rangle$

lemma *kruskal-edge-arc-1*:
assumes $e \leq --h$
and $h \leq g$
and *symmetric* g
and *components* $g \leq$ *forest-components* w
and $w * e^T = bot$
shows $e^T \leq w^\star$
$\langle proof \rangle$

lemma *kruskal-edge-between-components-1*:
assumes *equivalence* F
and *mapping* $(top * e)$
shows $F \leq -(w \sqcap top * e^T * w^{T\star} \sqcap F * e^T * top \sqcap top * e * -F)$
$\langle proof \rangle$

lemma *kruskal-edge-between-components-2*:
assumes *forest-components* $f \leq -d$
and *injective* f
and $f \sqcup f^T \leq w \sqcup w^T$
shows $f \sqcup f^T \leq (w \sqcap -d \sqcap -d^T) \sqcup (w^T \sqcap -d \sqcap -d^T)$
$\langle proof \rangle$

end

4.3 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*
$\langle proof \rangle$

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*]: $x^\star$ = *top*
begin

subclass *kleene-algebra*
⟨*proof*⟩

subclass *stone-kleene-relation-algebra*
⟨*proof*⟩

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*
⟨*proof*⟩

instance
⟨*proof*⟩

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 $c = s\langle g\rangle r$ *in*
let $d = s\langle g\rangle s$ *in*
let $as = r\langle star\ o\ a\rangle r$ *in*
let $ds = star\text{-}matrix'\ s\ d$ *in*
let $e = a \oplus b \odot ds \odot c$ *in*
let $es = r\langle star\ o\ e\rangle r$ *in*
let $f = d \oplus c \odot as \odot b$ *in*
let $fs = star\text{-}matrix'\ s\ f$ *in*
$es \oplus as \odot b \odot fs \oplus ds \odot c \odot es \oplus fs$
)

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 $star\text{-}matrix :: ('a::enum,'b::\{star,times,bounded\text{-}semilattice\text{-}sup\text{-}bot\})\ square \Rightarrow ('a,'b)\ square\ (\text{-}^{\odot}\ [100]\ 100)$ **where** $star\text{-}matrix\ f = star\text{-}matrix'\ (enum\text{-}class.enum::'a\ list)\ f$

The following lemmas deconstruct matrices with non-empty restrictions.

lemma *restrict-empty-left*:
$[]\langle f\rangle ls = mbot$
⟨*proof*⟩

lemma *restrict-empty-right*:
$ks\langle f\rangle[] = mbot$
⟨*proof*⟩

lemma *restrict-nonempty-left*:
fixes $f :: ('a,'b::bounded\text{-}semilattice\text{-}sup\text{-}bot)\ square$
shows $(k\#ks)\langle f\rangle ls = [k]\langle f\rangle ls \oplus ks\langle f\rangle ls$
⟨*proof*⟩

lemma *restrict-nonempty-right*:
fixes $f :: ('a,'b::bounded\text{-}semilattice\text{-}sup\text{-}bot)\ square$
shows $ks\langle f\rangle(l\#ls) = ks\langle f\rangle[l] \oplus ks\langle f\rangle ls$
⟨*proof*⟩

lemma *restrict-nonempty*:
fixes $f :: ('a,'b::bounded\text{-}semilattice\text{-}sup\text{-}bot)\ square$
shows $(k\#ks)\langle f\rangle(l\#ls) = [k]\langle f\rangle[l] \oplus [k]\langle f\rangle ls \oplus ks\langle f\rangle[l] \oplus ks\langle f\rangle ls$
⟨*proof*⟩

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

abbreviation $disjoint\ ks\ ls \equiv \neg(\exists x\ .\ List.member\ ks\ x \wedge List.member\ ls\ x)$

lemma *times-disjoint*:
fixes $f\ g :: ('a,'b::idempotent\text{-}semiring)\ square$
assumes $disjoint\ ls\ ms$

shows $ks\langle f\rangle ls \odot ms\langle g\rangle ns = mbot$
$\langle proof\rangle$

lemma *one-disjoint*:
assumes *disjoint ks ls*
shows $ks\langle (mone::('a,'b::idempotent\text{-}semiring)\ square)\rangle ls = mbot$
$\langle proof\rangle$

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

abbreviation *is-sublist ks ls* $\equiv \forall x\ .\ List.member\ ks\ x \longrightarrow List.member\ ls\ x$

lemma *restrict-sublist*:
assumes *is-sublist ls ks*
and *is-sublist ms ns*
shows $ls\langle ks\langle f\rangle ns\rangle ms = ls\langle f\rangle ms$
$\langle proof\rangle$

lemma *restrict-superlist*:
assumes *is-sublist ls ks*
and *is-sublist ms ns*
shows $ks\langle ls\langle f\rangle ms\rangle ns = ls\langle f\rangle ms$
$\langle proof\rangle$

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

lemma *restrict-sup*:
fixes $f\ g :: ('a,'b::bounded\text{-}semilattice\text{-}sup\text{-}bot)\ square$
shows $ks\langle f \oplus g\rangle ls = ks\langle f\rangle ls \oplus ks\langle g\rangle ls$
$\langle proof\rangle$

lemma *restrict-times*:
fixes $f\ g :: ('a,'b::idempotent\text{-}semiring)\ square$
shows $ks\langle ks\langle f\rangle ls \odot ls\langle g\rangle ms\rangle ms = ks\langle f\rangle ls \odot ls\langle g\rangle ms$
$\langle proof\rangle$

lemma *restrict-star*:
fixes $g :: ('a,'b::kleene\text{-}algebra)\ square$
shows $t\langle star\text{-}matrix'\ t\ g\rangle t = star\text{-}matrix'\ t\ g$
$\langle proof\rangle$

lemma *restrict-one*:
assumes $\neg$ *List.member ks k*
shows $(k\#ks)\langle (mone::('a,'b::idempotent\text{-}semiring)\ square)\rangle (k\#ks) = [k]\langle mone\rangle [k] \oplus ks\langle mone\rangle ks$
$\langle proof\rangle$

lemma *restrict-one-left-unit*:
$ks\langle (mone::('a::finite,'b::idempotent\text{-}semiring)\ square)\rangle ks \odot ks\langle f\rangle ls = ks\langle f\rangle ls$

$\langle proof \rangle$

The following lemmas consider restrictions to singleton index sets.

lemma *restrict-singleton*:
$([k]\langle f\rangle[l])\ (i,j) = (\mathit{if}\ i = k \wedge j = l\ \mathit{then}\ f\ (i,j)\ \mathit{else}\ \mathit{bot})$
$\langle proof \rangle$

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

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

lemma *restrict-singleton-product*:
fixes $f\ g$ :: ($'a$::*finite*,$'b$::*kleene-algebra*) *square*
shows $([k]\langle f\rangle[l] \odot [m]\langle g\rangle[n])\ (i,j) = (\mathit{if}\ i = k \wedge l = m \wedge j = n\ \mathit{then}\ f\ (i,l) * g\ (m,j)\ \mathit{else}\ \mathit{bot})$
$\langle proof \rangle$

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

lemma *restrict-star-unfold*:
$[l]\langle(\mathit{mone}::('a::\mathit{finite},'b::\mathit{kleene\text{-}algebra})\ \mathit{square})\rangle[l] \oplus [l]\langle f\rangle[l] \odot [l]\langle \mathit{star}\ o\ f\rangle[l] = [l]\langle \mathit{star}\ o\ f\rangle[l]$
$\langle proof \rangle$

lemma *restrict-all*:
$\mathit{enum\text{-}class.enum}\langle f\rangle \mathit{enum\text{-}class.enum} = f$
$\langle proof \rangle$

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 $\neg$ *List.member ls l*
shows $(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms) = ([k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus ([k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m]) \oplus (ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms)$
$\langle proof \rangle$

Equality of matrices is componentwise.

lemma *restrict-nonempty-eq*:
$(k\#ks)\langle f\rangle(l\#ls) = (k\#ks)\langle g\rangle(l\#ls) \longleftrightarrow [k]\langle f\rangle[l] = [k]\langle g\rangle[l] \wedge [k]\langle f\rangle ls = [k]\langle g\rangle ls \wedge ks\langle f\rangle[l] = ks\langle g\rangle[l] \wedge ks\langle f\rangle ls = ks\langle g\rangle ls$
$\langle proof \rangle$

Inequality of matrices is componentwise.

lemma *restrict-nonempty-less-eq*:
fixes $f\ g$ :: $('a,'b::idempotent\text{-}semiring)\ square$
shows $(k\#ks)\langle f\rangle(l\#ls) \preceq (k\#ks)\langle g\rangle(l\#ls) \longleftrightarrow [k]\langle f\rangle[l] \preceq [k]\langle g\rangle[l] \wedge [k]\langle f\rangle ls \preceq [k]\langle g\rangle ls \wedge ks\langle f\rangle[l] \preceq ks\langle g\rangle[l] \wedge ks\langle f\rangle ls \preceq ks\langle g\rangle ls$
⟨*proof*⟩

The following lemmas treat repeated restrictions to disjoint index sets.

lemma *restrict-disjoint-left*:
assumes *disjoint ks ms*
shows $ms\langle ks\langle f\rangle ls\rangle ns = mbot$
⟨*proof*⟩

lemma *restrict-disjoint-right*:
assumes *disjoint ls ns*
shows $ms\langle ks\langle f\rangle ls\rangle ns = mbot$
⟨*proof*⟩

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

lemma *restrict-nonempty-product-eq*:
fixes $f\ g\ h$ :: $('a::finite,'b::idempotent\text{-}semiring)\ square$
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*⟩

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 $f\ g\ h$ :: $('a::finite,'b::idempotent\text{-}semiring)\ square$
assumes $\neg$ *List.member ks k*
and $\neg$ *List.member ls l*
and $\neg$ *List.member ms m*
shows $(k\#ks)\langle f\rangle(l\#ls) \odot (l\#ls)\langle g\rangle(m\#ms) \preceq (k\#ks)\langle h\rangle(m\#ms) \longleftrightarrow [k]\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle[m] \preceq [k]\langle h\rangle[m] \wedge [k]\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus [k]\langle f\rangle ls \odot ls\langle g\rangle ms \preceq [k]\langle h\rangle ms \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle[m] \oplus ks\langle f\rangle ls \odot ls\langle g\rangle[m] \preceq ks\langle h\rangle[m] \wedge ks\langle f\rangle[l] \odot [l]\langle g\rangle ms \oplus ks\langle f\rangle ls \odot ls\langle g\rangle ms \preceq ks\langle h\rangle ms$
⟨*proof*⟩

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 $f\ g$ :: $('a::finite,'b::kleene\text{-}algebra)\ square$

shows *distinct ms* $\Longrightarrow$ $[l]\langle f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms \Longrightarrow [l]\langle star\ o\ f\rangle[l] \odot [l]\langle g\rangle ms \preceq [l]\langle g\rangle ms$
$\langle proof\rangle$

lemma *restrict-star-right-induct*:
fixes *f g* :: (*'a*::*finite*,*'b*::*kleene-algebra*) *square*
shows *distinct ms* $\Longrightarrow$ $ms\langle g\rangle[l] \odot [l]\langle f\rangle[l] \preceq ms\langle g\rangle[l] \Longrightarrow ms\langle g\rangle[l] \odot [l]\langle star\ o\ f\rangle[l] \preceq ms\langle g\rangle[l]$
$\langle proof\rangle$

lemma *restrict-pp*:
fixes *f* :: (*'a*,*'b*::*p-algebra*) *square*
shows $ks\langle \ominus\ominus f\rangle ls = \ominus\ominus(ks\langle f\rangle ls)$
$\langle proof\rangle$

lemma *pp-star-commute*:
fixes *f* :: (*'a*,*'b*::*stone-kleene-relation-algebra*) *square*
shows $\ominus\ominus(star\ o\ f) = star\ o\ \ominus\ominus f$
$\langle proof\rangle$

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*
$\langle proof\rangle$

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*
$\langle proof\rangle$

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, *Theoretical Aspects of Computing – ICTAC 2016*, volume 9965 of *Lecture Notes in Computer Science*, pages 51–68. 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.