Relation Algebra

Alasdair Armstrong, Simon Foster, Georg Struth, Tjark Weber

March 17, 2025

Abstract

Tarski's algebra of binary relations is formalised along the lines of the standard textbooks of Maddux and Schmidt and Ströhlein. This includes relation-algebraic concepts such as subidentities, vectors and a domain operation as well as various notions associated to functions. Relation algebras are also expanded by a reflexive transitive closure operation, and they are linked with Kleene algebras and models of binary relations and Boolean matrices.

Contents

1 Introductory Remarks

These theory files are only sparsely commented. Background information can be found in Tarski's original article [4] and in the books by Maddux [2] and Schmidt and Ströhlein [3]. We briefly discuss proof automation and the formalisation of direct products in [1].

2 (More) Boolean Algebra

theory *More-Boolean-Algebra*
 imports *Main*
begin

2.1 Laws of Boolean Algebra

The following laws of Boolean algebra support relational proofs. We might add laws for the binary minus since that would make certain theorems look more nicely. These are currently not so well supported.

context *boolean-algebra*
begin

no-notation
 times (**infixl** ‹·› *70*)
 and *plus* (**infixl** ‹+› *65*)
 and *Groups.zero-class.zero* (‹*0*›)
 and *Groups.one-class.one* (‹*1*›)

notation
 inf (**infixl** ‹·› *70*)
 and *sup* (**infixl** ‹+› *65*)
 and *bot* (‹*0*›)
 and *top* (‹*1*›)

lemma *meet-assoc*: $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
⟨*proof*⟩

lemma *aux4* [*simp*]: $x \cdot y + x \cdot -y = x$
⟨*proof*⟩

lemma *aux4-comm* [*simp*]: $x \cdot -y + x \cdot y = x$
⟨*proof*⟩

lemma *aux6* [*simp*]: $(x + y) \cdot -x = y \cdot -x$
⟨*proof*⟩

lemma *aux6-var* [*simp*]: $(-x + y) \cdot x = x \cdot y$
⟨*proof*⟩

lemma *aux9* [*simp*]: $x + -x \cdot y = x + y$
⟨*proof*⟩

lemma *join-iso*: $x \leq y \Longrightarrow x + z \leq y + z$
⟨*proof*⟩

lemma *join-isol*: $x \leq y \Longrightarrow z + x \leq z + y$
⟨*proof*⟩

lemma *join-double-iso*: $x \leq y \Longrightarrow w + x + z \leq w + y + z$
⟨*proof*⟩

lemma *comp-anti*: $x \leq y \longleftrightarrow -y \leq -x$
⟨*proof*⟩

lemma *meet-iso*: $x \leq y \Longrightarrow x \cdot z \leq y \cdot z$
⟨*proof*⟩

lemma *meet-isor*: $x \leq y \Longrightarrow z \cdot x \leq z \cdot y$
⟨*proof*⟩

lemma *meet-double-iso*: $x \leq y \Longrightarrow w \cdot x \cdot z \leq w \cdot y \cdot z$
⟨*proof*⟩

lemma *de-morgan-3* [*simp*]: $-(-x \cdot -y) = x + y$
⟨*proof*⟩

lemma *subdist-2-var*: $x + y \cdot z \leq x + y$
⟨*proof*⟩

lemma *dist-alt*: $[\![x + z = y + z; x \cdot z = y \cdot z]\!] \Longrightarrow x = y$
⟨*proof*⟩

Finally we prove the Galois connections for complementation.

lemma *galois-aux*: $x \cdot y = 0 \longleftrightarrow x \leq -y$
⟨*proof*⟩

lemma *galois-aux2*: $x \cdot -y = 0 \longleftrightarrow x \leq y$
⟨*proof*⟩

lemma *galois-1*: $x \cdot -y \leq z \longleftrightarrow x \leq y + z$
⟨*proof*⟩

lemma *galois-2*: $x \leq y + -z \longleftrightarrow x \cdot z \leq y$
⟨*proof*⟩

lemma *galois-aux3*: $x + y = 1 \longleftrightarrow -x \leq y$
⟨*proof*⟩

lemma *galois-aux4*: $-x + y = 1 \longleftrightarrow x \leq y$
⟨*proof*⟩

2.2 Boolean Algebras with Operators

We follow Jónsson and Tarski to define pairs of conjugate functions on Boolean algebras. We also consider material from Maddux's article. This gives rise to a Galois connection and the notion of Boolean algebras with operators.

We do not explicitly define families of functions over Boolean algebras as a type class.

This development should certainly be expanded do deal with complete Boolean algebras one the one hand and other lattices on the other hand.

Boolean algebras with operators and their variants can be applied in various ways. The prime example are relation algebras. The modular laws, for instance, can be derived by instantiation. Other applications are antidomain semirings where modal operators satisfy conjugations and Galois connections, and algebras of predicate transformers.

We define conjugation as a predicate which holds if a pair of functions are conjugates.

definition *is-conjugation* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow bool$
  where *is-conjugation* $f\ g \equiv (\forall x\ y\ .\ f\ x \cdot y = 0 \longleftrightarrow x \cdot g\ y = 0)$

We now prove the standard lemmas. First we show that conjugation is symmetric and that conjugates are uniqely defined.

lemma *is-conjugation-sym*: *is-conjugation* $f\ g \longleftrightarrow$ *is-conjugation* $g\ f$
⟨*proof*⟩

lemma *is-conjugation-unique*: ⟦*is-conjugation* $f\ g$; *is-conjugation* $f\ h$⟧ $\Longrightarrow g = h$
⟨*proof*⟩

Next we show that conjugates give rise to adjoints in a Galois connection.

lemma *conj-galois-1*:
  assumes *is-conjugation* $f\ g$
  shows $f\ x \leq y \longleftrightarrow x \leq -g\ (-y)$
⟨*proof*⟩

lemma *conj-galois-2*:
  assumes *is-conjugation* $f\ g$

shows $g\ x \leq y \longleftrightarrow x \leq -f\ (-y)$
⟨*proof*⟩

Now we prove some of the standard properties of adjoints and conjugates. In fact, conjugate functions even distribute over all existing suprema. We display the next proof in detail because it is elegant.

lemma *f-pre-additive*:
assumes *is-conjugation f g*
shows $f\ (x + y) \leq z \longleftrightarrow f\ x + f\ y \leq z$
⟨*proof*⟩

lemma *f-additive*:
assumes *is-conjugation f g*
shows $f\ (sup\ x\ y) = sup\ (f\ x)\ (f\ y)$
⟨*proof*⟩

lemma *g-pre-additive*:
assumes *is-conjugation f g*
shows $g\ (sup\ x\ y) \leq z \longleftrightarrow sup\ (g\ x)\ (g\ y) \leq z$
⟨*proof*⟩

lemma *g-additive*:
assumes *is-conjugation f g*
shows $g\ (sup\ x\ y) = sup\ (g\ x)\ (g\ y)$
⟨*proof*⟩

Additivity of adjoints obviously implies their isotonicity.

lemma *f-iso*:
assumes *is-conjugation f g*
shows $x \leq y \longrightarrow f\ x \leq f\ y$
⟨*proof*⟩

lemma *g-iso*:
assumes *is-conjugation f g*
shows $x \leq y \longrightarrow g\ x \leq g\ y$
⟨*proof*⟩

lemma *f-subdist*:
assumes *is-conjugation f g*
shows $f\ (x \cdot y) \leq f\ x$
⟨*proof*⟩

lemma *g-subdist*:
assumes *is-conjugation f g*
shows $g\ (x \cdot y) \leq g\ x$
⟨*proof*⟩

Next we prove cancellation and strictness laws.

lemma *cancellation-1*:

assumes *is-conjugation f g*
shows $f\ (-g\ x) \leq -x$
⟨*proof*⟩

lemma *cancellation-2*:
assumes *is-conjugation f g*
shows $g\ (-f\ x) \leq -x$
⟨*proof*⟩

lemma *f-strict*:
assumes *is-conjugation f g*
shows $f\ 0 = 0$
⟨*proof*⟩

lemma *g-strict*:
assumes *is-conjugation f g*
shows $g\ 0 = 0$
⟨*proof*⟩

The following variants of modular laws have more concrete counterparts in relation algebra.

lemma *modular-1-aux*:
assumes *is-conjugation f g*
shows $f\ (x \cdot -g\ y) \cdot y = 0$
⟨*proof*⟩

lemma *modular-2-aux*:
assumes *is-conjugation f g*
shows $g\ (x \cdot -f\ y) \cdot y = 0$
⟨*proof*⟩

lemma *modular-1*:
assumes *is-conjugation f g*
shows $f\ x \cdot y = f\ (x \cdot g\ y) \cdot y$
⟨*proof*⟩

lemma *modular-2*:
assumes *is-conjugation f g*
shows $g\ x \cdot y = g\ (x \cdot f\ y) \cdot y$
⟨*proof*⟩

lemma *conjugate-eq-aux*:
is-conjugation f g $\Longrightarrow f\ (x \cdot -g\ y) \leq f\ x \cdot -y$
⟨*proof*⟩

lemma *conjugate-eq*:
is-conjugation f g $\longleftrightarrow (\forall\, x\ y.\ f\ (x \cdot -g\ y) \leq f\ x \cdot -y \wedge g\ (y \cdot -f\ x) \leq g\ y \cdot -x)$
(**is** $?l \longleftrightarrow ?r$)
⟨*proof*⟩

lemma *conjugation-prop1*: *is-conjugation f g* $\Longrightarrow$ *f y* $\cdot$ *z* $\leq$ *f* (*y* $\cdot$ *g z*)
⟨*proof*⟩

lemma *conjugation-prop2*: *is-conjugation f g* $\Longrightarrow$ *g z* $\cdot$ *y* $\leq$ *g* (*z* $\cdot$ *f y*)
⟨*proof*⟩

end

end

3 Relation Algebra

theory *Relation-Algebra*
  imports *More-Boolean-Algebra Kleene-Algebra.Kleene-Algebra*
begin

We follow Tarski's original article and Maddux's book, in particular we use their notation. In contrast to Schmidt and Ströhlein we do not assume that the Boolean algebra is complete and we do not consider the Tarski rule in this development.

A main reason for using complete Boolean algebras seems to be that the Knaster-Tarski fixpoint theorem becomes available for defining notions of iteration. In fact, several chapters of Schmidt and Ströhlein's book deal with iteration.

We capture iteration in an alternative way by linking relation algebras with Kleene algebras (cf. *relation-algebra-rtc*).

class *relation-algebra* = *boolean-algebra* +
  fixes *composition* :: ′*a* ⇒ ′*a* ⇒ ′*a* (**infixl** ‹;› *75*)
    and *converse* :: ′*a* ⇒ ′*a* (‹(-˘)› [*1000*] *999*)
    and *unit* :: ′*a* (‹*1*′′›)
  assumes *comp-assoc*: (*x* ; *y*) ; *z* = *x* ; (*y* ; *z*)
    and *comp-unitr* [*simp*]: *x* ; *1*′ = *x*
    and *comp-distr*: (*x* + *y*) ; *z* = *x* ; *z* + *y* ; *z*
    and *conv-invol* [*simp*]: (*x*˘)˘ = *x*
    and *conv-add* [*simp*]: (*x* + *y*)˘ = *x*˘ + *y*˘
    and *conv-contrav* [*simp*]: (*x* ; *y*)˘ = *y*˘ ; *x*˘
    and *comp-res*: *x*˘ ; −(*x* ; *y*) ≤ −*y*

We first show that every relation algebra is a dioid. We do not yet treat the zero (the minimal element of the boolean reduct) since the proof of the annihilation laws is rather tricky to automate. Following Maddux we derive them from properties of Boolean algebras with operators.

sublocale *relation-algebra* ⊆ *dioid-one* (+) (;) (≤) (<) *1*′
⟨*proof*⟩

context *relation-algebra*
begin

First we prove some basic facts about joins and meets.

lemma *meet-interchange*: $(w \cdot x) ; (y \cdot z) \leq w ; y \cdot x ; z$
⟨*proof*⟩

lemma *join-interchange*: $w ; x + y ; z \leq (w + y) ; (x + z)$
⟨*proof*⟩

We now prove some simple facts about conversion.

lemma *conv-iso*: $x \leq y \longleftrightarrow x^{\smile} \leq y^{\smile}$
⟨*proof*⟩

lemma *conv-zero* [*simp*]: $0^{\smile} = 0$
⟨*proof*⟩

lemma *conv-one* [*simp*]: $1^{\smile} = 1$
⟨*proof*⟩

lemma *conv-compl-aux*: $(-x)^{\smile} = (-x)^{\smile} + (-x^{\smile})$
⟨*proof*⟩

lemma *conv-compl*: $(-x)^{\smile} = -(x^{\smile})$
⟨*proof*⟩

lemma *comp-res-aux* [*simp*]: $x^{\smile} ; -(x ; y) \cdot y = 0$
⟨*proof*⟩

lemma *conv-e* [*simp*]: $1'^{\smile} = 1'$
⟨*proof*⟩

lemma *conv-times* [*simp*]: $(x \cdot y)^{\smile} = x^{\smile} \cdot y^{\smile}$
⟨*proof*⟩

The next lemmas show that conversion is self-conjugate in the sense of Boolean algebra with operators.

lemma *conv-self-conjugate*: $x^{\smile} \cdot y = 0 \longleftrightarrow x \cdot y^{\smile} = 0$
⟨*proof*⟩

lemma *conv-self-conjugate-var*: *is-conjugation converse converse*
⟨*proof*⟩

The following lemmas link the relative product and meet.

lemma *one-idem-mult* [*simp*]: $1 ; 1 = 1$
⟨*proof*⟩

lemma *mult-subdistl*: $x ; (y \cdot z) \leq x ; y$

⟨*proof*⟩

lemma *mult-subdistr*: $(x \cdot y) ; z \leq x ; z$
⟨*proof*⟩

lemma *mult-subdistr-var*: $(x \cdot y) ; z \leq x ; z \cdot y ; z$
⟨*proof*⟩

The following lemmas deal with variants of the Peirce law, the Schröder laws and the Dedekind law. Some of them are obtained from Boolean algebras with operators by instantiation, using conjugation properties. However, Isabelle does not always pick up this relationship.

lemma *peirce-1*: $x ; y \cdot z^{\smile} = 0 \Longrightarrow y ; z \cdot x^{\smile} = 0$
⟨*proof*⟩

lemma *peirce*: $x ; y \cdot z^{\smile} = 0 \longleftrightarrow y ; z \cdot x^{\smile} = 0$
⟨*proof*⟩

lemma *schroeder-1*: $x ; y \cdot z = 0 \longleftrightarrow y \cdot x^{\smile} ; z = 0$
⟨*proof*⟩

lemma *schroeder-2*: $y ; x \cdot z = 0 \longleftrightarrow y \cdot z ; x^{\smile} = 0$
⟨*proof*⟩

The following two conjugation properties between multiplication with elements and their converses are used for deriving modular laws of relation algebra from those of Boolean algebras with operators.

lemma *schroeder-1-var*: *is-conjugation* (*composition* x) (*composition* $(x^{\smile})$)
⟨*proof*⟩

lemma *schroeder-2-var*: *is-conjugation* $(\lambda x.\ x ; y)$ $(\lambda x.\ x ; y^{\smile})$
⟨*proof*⟩

The following Galois connections define residuals. They link relation algebras with action algebras. This could be further explored and formalised.

lemma *conv-galois-1*: $x ; y \leq z \longleftrightarrow y \leq -(x^{\smile} ; -z)$
⟨*proof*⟩

lemma *conv-galois-2*: $y ; x \leq z \longleftrightarrow y \leq -(-z ; x^{\smile})$
⟨*proof*⟩

Variants of the modular law for relation algebras can now be instantiated from Boolean algebras with operators.

lemma *modular-1-aux'*: $x ; (y \cdot -(x^{\smile} ; z)) \cdot z = 0$
⟨*proof*⟩

lemma *modular-2-aux'*: $(y \cdot -(z ; x^{\smile})) ; x \cdot z = 0$

⟨*proof*⟩

lemma *modular-1'*: $x ; y \cdot z = x ; (y \cdot x^{\smile} ; z) \cdot z$
⟨*proof*⟩

lemma *modular-2'*: $y ; x \cdot z = (y \cdot z ; x^{\smile}) ; x \cdot z$
⟨*proof*⟩

lemma *modular-1-var*: $x ; y \cdot z \leq x ; (y \cdot x^{\smile} ; z)$
⟨*proof*⟩

lemma *modular-2-var*: $x ; y \cdot z \leq (x \cdot z ; y^{\smile}) ; y$
⟨*proof*⟩

lemma *modular-var-2*: $x ; y \leq x ; (y \cdot x^{\smile} ; 1)$
⟨*proof*⟩

lemma *modular-var-3*: $x ; y \leq (x \cdot 1 ; y^{\smile}) ; y$
⟨*proof*⟩

The modular laws are used to prove the Dedekind rule.

lemma *dedekind*: $x ; y \cdot z \leq (x \cdot z ; y^{\smile}) ; (y \cdot x^{\smile} ; z)$
⟨*proof*⟩

lemma *dedekind-var-1*: $x ; y \leq (x \cdot 1 ; y^{\smile}) ; (y \cdot x^{\smile} ; 1)$
⟨*proof*⟩

end

The Schröder laws allow us, finally, to prove the annihilation laws for zero. We formalise this by proving that relation algebras form dioids with zero.

sublocale *relation-algebra* < *dioid-one-zero* (+) (;) *1' 0* (≤) (<)
⟨*proof*⟩

context *relation-algebra*
begin

Next we prove miscellaneous properties which we found in the books of Maddux and Schmidt and Ströhlein. Most of them do not carry any meaningful names.

lemma *ra-1*: $(x \cdot y ; 1) ; z = x ; z \cdot y ; 1$
⟨*proof*⟩

lemma *ra-2*: $x ; (z \cdot y ; 1) = (x \cdot (y ; 1)^{\smile}) ; z$
⟨*proof*⟩

lemma *one-conv*: $1' \cdot x ; 1 = 1' \cdot x ; x^{\smile}$
⟨*proof*⟩

lemma *maddux-12*: $-(y\ ;\ x)\ ;\ x^{\smile} \leq -y$
⟨*proof*⟩

lemma *maddux-141*: $x\ ;\ y \cdot z = 0 \longleftrightarrow x^{\smile}\ ;\ z \cdot y = 0$
⟨*proof*⟩

lemma *maddux-142*: $x^{\smile}\ ;\ z \cdot y = 0 \longleftrightarrow z\ ;\ y^{\smile} \cdot x = 0$
⟨*proof*⟩

lemmas *maddux-16* = *modular-1-var*

lemmas *maddux-17* = *modular-2-var*

lemma *maddux-20*: $x \leq x\ ;\ 1$
⟨*proof*⟩

lemma *maddux-21*: $x \leq 1\ ;\ x$
⟨*proof*⟩

lemma *maddux-23*: $x\ ;\ y \cdot -(x\ ;\ z) = x\ ;\ (y \cdot -z) \cdot -(x\ ;\ z)$
⟨*proof*⟩

lemma *maddux-24*: $-(x\ ;\ y) + x\ ;\ z = -(x\ ;\ (y \cdot -z)) + x\ ;\ z$
⟨*proof*⟩

lemma *one-compl*: $-(x\ ;\ 1)\ ;\ 1 = -(x\ ;\ 1)$
⟨*proof*⟩

lemma *ss-p18*: $x\ ;\ 1 = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

end

This finishes our development of the basic laws of relation algebras. The next sections are devoted to special elements such as vectors, test or subidentities, and, in particular, functions.

end

4 Vectors

theory *Relation-Algebra-Vectors*
  imports *Relation-Algebra*
begin

Vectors can be used for modelling sets of states. In this section we follow Maddux's book to derive some of their most important properties.

context *relation-algebra*

begin

definition *is-vector* :: $'a \Rightarrow bool$
  where *is-vector* $x \equiv x = x \; ; \; 1$

lemma *vector-compl*: *is-vector* $x \Longrightarrow$ *is-vector* $(-x)$
⟨*proof*⟩

lemma *vector-add*: ⟦*is-vector* x; *is-vector* y⟧ $\Longrightarrow$ *is-vector* $(x + y)$
⟨*proof*⟩

lemma *vector-mult*: ⟦*is-vector* x; *is-vector* y⟧ $\Longrightarrow$ *is-vector* $(x \cdot y)$
⟨*proof*⟩

lemma *vector-comp*: ⟦*is-vector* x; *is-vector* y⟧ $\Longrightarrow$ *is-vector* $(x \; ; \; y)$
⟨*proof*⟩

lemma *vector-1*: *is-vector* $x \Longrightarrow (x \cdot y) \; ; \; z = x \cdot y \; ; \; z$
⟨*proof*⟩

lemma *vector-1-comm*: *is-vector* $y \Longrightarrow (x \cdot y) \; ; \; z = x \; ; \; z \cdot y$
⟨*proof*⟩

lemma *vector-2*: *is-vector* $y \Longrightarrow (x \cdot y^{\smile}) \; ; \; z = x \; ; \; (y \cdot z)$
⟨*proof*⟩

lemma *vector-2-var*: *is-vector* $y \Longrightarrow (x \cdot y^{\smile}) \; ; \; z = (x \cdot y^{\smile}) \; ; \; (y \cdot z)$
⟨*proof*⟩

lemma *vector-idem* [*simp*]: *is-vector* $x \Longrightarrow x \; ; \; x = x$
⟨*proof*⟩

lemma *vector-rectangle* [*simp*]: *is-vector* $x \Longrightarrow x \; ; \; 1 \; ; \; x = x$
⟨*proof*⟩

lemma *vector-3* [*simp*]: *is-vector* $x \Longrightarrow (x \cdot 1') \; ; \; y = x \cdot y$
⟨*proof*⟩

end

end

5 Tests

theory *Relation-Algebra-Tests*
  imports *Relation-Algebra*
begin

5.1 Tests

Tests or subidentities provide another way of modelling sets. Once more we prove the basic properties, most of which stem from Maddux's book.

context *relation-algebra*
begin

definition *is-test* :: $'a \Rightarrow bool$
  where *is-test* $x \equiv x \leq 1'$

lemma *test-conv*: *is-test* $x \Longrightarrow$ *is-test* $(x^{\smile})$
⟨*proof*⟩

lemma *test-conv-var*: *is-test* $x \Longrightarrow x^{\smile} \leq 1'$
⟨*proof*⟩

lemma *test-eq-conv* [*simp*]: *is-test* $x \Longrightarrow x^{\smile} = x$
⟨*proof*⟩

lemma *test-sum*: ⟦*is-test* x; *is-test* y⟧ $\Longrightarrow$ *is-test* $(x + y)$
⟨*proof*⟩

lemma *test-prod*: ⟦*is-test* x; *is-test* y⟧ $\Longrightarrow$ *is-test* $(x \cdot y)$
⟨*proof*⟩

lemma *test-comp*: ⟦*is-test* x; *is-test* y⟧ $\Longrightarrow$ *is-test* $(x ; y)$
⟨*proof*⟩

lemma *test-comp-eq-mult*:
  assumes *is-test* x
    and *is-test* y
  shows $x ; y = x \cdot y$
⟨*proof*⟩

lemma *test-1* [*simp*]: *is-test* $x \Longrightarrow x ; 1 \cdot y = x ; y$
⟨*proof*⟩

lemma *maddux-32* [*simp*]: *is-test* $x \Longrightarrow -(x ; 1) \cdot 1' = -x \cdot 1'$
⟨*proof*⟩

lemma *test-distr-1* :
  assumes *is-test* x
    and *is-test* y
  shows $x ; z \cdot y ; z = (x \cdot y) ; z$
⟨*proof*⟩

lemma *maddux-35*: *is-test* $x \Longrightarrow x ; y \cdot -z = x ; y \cdot -(x ; z)$
⟨*proof*⟩

5.2 Test Complements

Text complements are complements of elements that are "pushed below" the multiplicative unit.

definition *tc* :: *'a* ⇒ *'a*
  where $tc\ x = 1' \cdot -x$

lemma *test-compl-1* [*simp*]: $is\text{-}test\ x \Longrightarrow x + tc\ x = 1'$
  ⟨*proof*⟩

lemma *test-compl-2* [*simp*]: $is\text{-}test\ x \Longrightarrow x \cdot tc\ x = 0$
  ⟨*proof*⟩

lemma *test-test-compl*: $is\text{-}test\ x \Longrightarrow is\text{-}test\ (tc\ x)$
  ⟨*proof*⟩

lemma *test-compl-de-morgan-1*: $tc\ (x + y) = tc\ x \cdot tc\ y$
  ⟨*proof*⟩

lemma *test-compl-de-morgan-2*: $tc\ (x \cdot y) = tc\ x + tc\ y$
  ⟨*proof*⟩

lemma *test-compl-three* [*simp*]: $tc\ (tc\ (tc\ x)) = tc\ x$
  ⟨*proof*⟩

lemma *test-compl-double* [*simp*]: $is\text{-}test\ x \Longrightarrow tc\ (tc\ x) = x$
  ⟨*proof*⟩

end

end

6 Functions

theory *Relation-Algebra-Functions*
  imports *Relation-Algebra-Vectors Relation-Algebra-Tests*
begin

6.1 Functions

This section collects the most important properties of functions. Most of them can be found in the books by Maddux and by Schmidt and Ströhlein. The main material is on partial and total functions, injections, surjections, bijections.

context *relation-algebra*
begin

definition *is-p-fun* :: *'a* ⇒ *bool*
where *is-p-fun* $x \equiv x^{\smile} ; x \leq 1'$

definition *is-total* :: *'a* ⇒ *bool*
where *is-total* $x \equiv 1' \leq x ; x^{\smile}$

definition *is-map* :: *'a* ⇒ *bool*
where *is-map* $x \equiv$ *is-p-fun* $x \wedge$ *is-total* x

definition *is-inj* :: *'a* ⇒ *bool*
where *is-inj* $x \equiv x ; x^{\smile} \leq 1'$

definition *is-sur* :: *'a* ⇒ *bool*
where *is-sur* $x \equiv 1' \leq x^{\smile} ; x$

We distinguish between partial and total bijections. As usual we call the latter just bijections.

definition *is-p-bij* :: *'a* ⇒ *bool*
where *is-p-bij* $x \equiv$ *is-p-fun* $x \wedge$ *is-inj* $x \wedge$ *is-sur* x

definition *is-bij* :: *'a* ⇒ *bool*
where *is-bij* $x \equiv$ *is-map* $x \wedge$ *is-inj* $x \wedge$ *is-sur* x

Our first set of lemmas relates the various concepts.

lemma *inj-p-fun*: *is-inj* $x \longleftrightarrow$ *is-p-fun* $(x^{\smile})$
⟨*proof*⟩

lemma *p-fun-inj*: *is-p-fun* $x \longleftrightarrow$ *is-inj* $(x^{\smile})$
⟨*proof*⟩

lemma *sur-total*: *is-sur* $x \longleftrightarrow$ *is-total* $(x^{\smile})$
⟨*proof*⟩

lemma *total-sur*: *is-total* $x \longleftrightarrow$ *is-sur* $(x^{\smile})$
⟨*proof*⟩

lemma *bij-conv*: *is-bij* $x \longleftrightarrow$ *is-bij* $(x^{\smile})$
⟨*proof*⟩

Next we show that tests are partial injections.

lemma *test-is-inj-fun*: *is-test* $x \Longrightarrow$ (*is-p-fun* $x \wedge$ *is-inj* x)
⟨*proof*⟩

Next we show composition properties.

lemma *p-fun-comp*:
assumes *is-p-fun* x **and** *is-p-fun* y
shows *is-p-fun* $(x ; y)$
⟨*proof*⟩

lemma *p-fun-mult-var*: $x^{\smile}$; $x \leq 1'$ $\Longrightarrow$ $(x \cdot y)^{\smile}$; $(x \cdot y) \leq 1'$
⟨*proof*⟩

lemma *inj-compose*:
assumes *is-inj* x **and** *is-inj* y
shows *is-inj* $(x ; y)$
⟨*proof*⟩

lemma *inj-mult-var*: $x^{\smile}$; $x \leq 1'$ $\Longrightarrow$ $(x \cdot y)^{\smile}$; $(x \cdot y) \leq 1'$
⟨*proof*⟩

lemma *total-comp*:
assumes *is-total* x **and** *is-total* y
shows *is-total* $(x ; y)$
⟨*proof*⟩

lemma *total-add-var*: $1' \leq x^{\smile}$; x $\Longrightarrow$ $1' \leq (x + y)^{\smile}$; $(x + y)$
⟨*proof*⟩

lemma *sur-comp*:
assumes *is-sur* x **and** *is-sur* y
shows *is-sur* $(x ; y)$
⟨*proof*⟩

lemma *sur-sum-var*: $1' \leq x^{\smile}$; x $\Longrightarrow$ $1' \leq (x + y)^{\smile}$; $(x + y)$
⟨*proof*⟩

lemma *map-comp*:
assumes *is-map* x **and** *is-map* y
shows *is-map* $(x ; y)$
⟨*proof*⟩

lemma *bij-comp*:
assumes *is-bij* x **and** *is-bij* y
shows *is-bij* $(x ; y)$
⟨*proof*⟩

We now show that (partial) functions, unlike relations, distribute over meets from the left.

lemma *p-fun-distl*: *is-p-fun* x $\Longrightarrow$ $x ; (y \cdot z) = x ; y \cdot x ; z$
⟨*proof*⟩

lemma *map-distl*: *is-map* x $\Longrightarrow$ $x ; (y \cdot z) = x ; y \cdot x ; z$
⟨*proof*⟩

Next we prove simple properties of functions which arise in equivalent definitions of those concepts.

lemma *p-fun-zero*: *is-p-fun* x $\Longrightarrow$ $x ; y \cdot x ; -y = 0$

⟨*proof*⟩

lemma *total-one*: *is-total* $x \Longrightarrow x \; ; \; 1 = 1$
⟨*proof*⟩

lemma *total-1*: *is-total* $x \Longrightarrow (\forall y. \; y \; ; \; x = 0 \longrightarrow y = 0)$
⟨*proof*⟩

lemma *surj-one*: *is-sur* $x \Longrightarrow 1 \; ; \; x = 1$
⟨*proof*⟩

lemma *surj-1*: *is-sur* $x \Longrightarrow (\forall y. \; x \; ; \; y = 0 \longrightarrow y = 0)$
⟨*proof*⟩

lemma *bij-is-maprop*:
assumes *is-bij* x **and** *is-map* x
shows $x^\smile \; ; \; x = 1' \wedge x \; ; \; x^\smile = 1'$
⟨*proof*⟩

We now provide alternative definitions for functions. These can be found in Schmidt and Ströhlein's book.

lemma *p-fun-def-var*: *is-p-fun* $x \longleftrightarrow x \; ; \; -(1') \leq -x$
⟨*proof*⟩

lemma *total-def-var-1*: *is-total* $x \longleftrightarrow x \; ; \; 1 = 1$
⟨*proof*⟩

lemma *total-def-var-2*: *is-total* $x \longleftrightarrow -x \leq x \; ; \; -(1')$
⟨*proof*⟩

lemma *sur-def-var1*: *is-sur* $x \longleftrightarrow 1 \; ; \; x = 1$
⟨*proof*⟩

lemma *sur-def-var2*: *is-sur* $x \longleftrightarrow -x \leq -(1') \; ; \; x$
⟨*proof*⟩

lemma *inj-def-var1*: *is-inj* $x \longleftrightarrow -(1') \; ; \; x \leq -x$
⟨*proof*⟩

lemma *is-maprop*: *is-map* $x \longleftrightarrow x \; ; \; -(1') = -x$
⟨*proof*⟩

Finally we prove miscellaneous properties of functions.

lemma *ss-422iii*: *is-p-fun* $y \Longrightarrow (x \cdot z \; ; \; y^\smile) \; ; \; y = x \; ; \; y \cdot z$

⟨*proof*⟩

lemma *p-fun-compl*: *is-p-fun* $x \Longrightarrow x \; ; \; -y \leq -(x; \; y)$
⟨*proof*⟩

lemma *ss-422v*: *is-p-fun* $x \Longrightarrow x\ ;\ -y = x\ ;\ 1\ \cdot\ -(x\ ;\ y)$
⟨*proof*⟩

The next property is a Galois connection.

lemma *ss43iii*: *is-map* $x \longleftrightarrow (\forall\, y.\ x\ ;\ -y = -(x\ ;\ y))$
⟨*proof*⟩

Next we prove a lemma from Schmidt and Ströhlein's book and some of its consequences. We show the proof in detail since the textbook proof uses Tarski's rule which we omit.

lemma *ss423*: *is-map* $x \Longrightarrow y\ ;\ x \le z \longleftrightarrow y \le z\ ;\ x^{\smile}$
⟨*proof*⟩

lemma *ss424i*: *is-total* $x \longleftrightarrow (\forall\, y.\ -(x\ ;\ y) \le x\ ;\ -y)$
⟨*proof*⟩

lemma *ss434ii*: *is-p-fun* $x \longleftrightarrow (\forall\, y.\ x\ ;\ -y \le -(x\ ;\ y))$
⟨*proof*⟩

lemma *is-maprop1*: *is-map* $x \Longrightarrow (y \le x\ ;\ z\ ;\ x^{\smile} \longleftrightarrow y\ ;\ x \le x\ ;\ z)$
⟨*proof*⟩

lemma *is-maprop2*: *is-map* $x \Longrightarrow (y\ ;\ x \le x\ ;\ z \longleftrightarrow x^{\smile}\ ;\ y;\ x \le z)$
⟨*proof*⟩

lemma *is-maprop3*: *is-map* $x \Longrightarrow (x^{\smile}\ ;\ y;\ x \le z \longleftrightarrow x^{\smile}\ ;\ y \le z\ ;\ x^{\smile})$
⟨*proof*⟩

lemma *p-fun-sur-id* [*simp*]:
  assumes *is-p-fun* x **and** *is-sur* x
  shows $x^{\smile}\ ;\ x = 1'$
⟨*proof*⟩

lemma *total-inj-id* [*simp*]:
  assumes *is-total* x **and** *is-inj* x
  shows $x\ ;\ x^{\smile} = 1'$
⟨*proof*⟩

lemma *bij-inv-1* [*simp*]: *is-bij* $x \Longrightarrow x\ ;\ x^{\smile} = 1'$
⟨*proof*⟩

lemma *bij-inv-2* [*simp*]: *is-bij* $x \Longrightarrow x^{\smile}\ ;\ x = 1'$
⟨*proof*⟩

lemma *bij-inv-comm*: *is-bij* $x \Longrightarrow x\ ;\ x^{\smile} = x^{\smile}\ ;\ x$
⟨*proof*⟩

lemma *is-bijrop*: *is-bij* $x \Longrightarrow (y = x\ ;\ z \longleftrightarrow z = x^{\smile}\ ;\ y)$

⟨*proof*⟩

lemma *inj-map-monomorph*: $[\![is\text{-}inj\ x;\ is\text{-}map\ x]\!] \Longrightarrow (\forall\, y\ z.\ y\ ;\ x = z\ ;\ x \longrightarrow y = z)$
⟨*proof*⟩

lemma *sur-map-epimorph*: $[\![is\text{-}sur\ x;\ is\text{-}map\ x]\!] \Longrightarrow (\forall\, y\ z.\ x\ ;\ y = x\ ;\ z \longrightarrow y = z)$
⟨*proof*⟩

6.2 Points and Rectangles

Finally here is a section on points and rectangles. This is only a beginning.

definition *is-point* :: $'a \Rightarrow bool$
where $is\text{-}point\ x \equiv is\text{-}vector\ x \wedge is\text{-}inj\ x \wedge x \neq 0$

definition *is-rectangle* :: $'a \Rightarrow bool$
where $is\text{-}rectangle\ x \equiv x\ ;\ 1\ ;\ x \leq x$

lemma *rectangle-eq* [*simp*]: $is\text{-}rectangle\ x \longleftrightarrow x\ ;\ 1\ ;\ x = x$
⟨*proof*⟩

6.3 Antidomain

This section needs to be linked with domain semirings. We essentially prove the antidomain semiring axioms. Then we have the abstract properties at our disposition.

definition *antidom* :: $'a \Rightarrow 'a$ (‹*a*›)
where $a\ x = 1' \cdot (-(x\ ;\ 1))$

definition *dom* :: $'a \Rightarrow 'a$ (‹*d*›)
where $d\ x = a\ (a\ x)$

lemma *antidom-test-comp* [*simp*]: $a\ x = tc\ (x\ ;\ 1)$
⟨*proof*⟩

lemma *dom-def-aux*: $d\ x = 1' \cdot x\ ;\ 1$
⟨*proof*⟩

lemma *dom-def-aux-var*: $d\ x = 1' \cdot x\ ;\ x^{\smile}$
⟨*proof*⟩

lemma *antidom-dom* [*simp*]: $a\ (d\ x) = a\ x$
⟨*proof*⟩

lemma *dom-antidom* [*simp*]: $d\ (a\ x) = a\ x$
⟨*proof*⟩

lemma *dom-verystrict*: $d\ x = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *a-1* [*simp*]: $a\ x\ ;\ x = 0$
⟨*proof*⟩

lemma *a-2*: $a\ (x\ ;\ y) = a\ (x\ ;\ d\ y)$
⟨*proof*⟩

lemma *a-3* [*simp*]: $a\ x + d\ x = 1'$
⟨*proof*⟩

lemma *test-domain*: $x = d\ x \longleftrightarrow x \leq 1'$
⟨*proof*⟩

At this point we have all the necessary ingredients to prove that relation algebras form Boolean domain semirings. However, we omit a formal proof since we haven't formalized the latter.

lemma *dom-one*: $x\ ;\ 1 = d\ x\ ;\ 1$
⟨*proof*⟩

lemma *test-dom*: *is-test* $(d\ x)$
⟨*proof*⟩

lemma *p-fun-dom*: *is-p-fun* $(d\ x)$
⟨*proof*⟩

lemma *inj-dom*: *is-inj* $(d\ x)$
⟨*proof*⟩

lemma *total-alt-def*: *is-total* $x \longleftrightarrow (d\ x) = 1'$
⟨*proof*⟩

end

end

7 Direct Products

theory *Relation-Algebra-Direct-Products*
imports *Relation-Algebra-Functions*
begin

This section uses the definition of direct products from Schmidt and Ströhlein's book to prove the well known universal property.

context *relation-algebra*
begin

definition *is-direct-product* :: $'a \Rightarrow 'a \Rightarrow bool$
where *is-direct-product* $x\ y \equiv x^{\smile}\ ;\ x = 1' \wedge y^{\smile}\ ;\ y = 1' \wedge x\ ;\ x^{\smile} \cdot y\ ;\ y^{\smile} = 1' \wedge x^{\smile}\ ;\ y = 1$

We collect some basic properties.

lemma *dp-p-fun1*: *is-direct-product* $x\ y \Longrightarrow$ *is-p-fun* x
⟨*proof*⟩

lemma *dp-sur1*: *is-direct-product* $x\ y \Longrightarrow$ *is-sur* x
⟨*proof*⟩

lemma *dp-total1*: *is-direct-product* $x\ y \Longrightarrow$ *is-total* x
⟨*proof*⟩

lemma *dp-map1*: *is-direct-product* $x\ y \Longrightarrow$ *is-map* x
⟨*proof*⟩

lemma *dp-p-fun2*: *is-direct-product* $x\ y \Longrightarrow$ *is-p-fun* y
⟨*proof*⟩

lemma *dp-sur2*: *is-direct-product* $x\ y \Longrightarrow$ *is-sur* y
⟨*proof*⟩

lemma *dp-total2*: *is-direct-product* $x\ y \Longrightarrow$ *is-total* y
⟨*proof*⟩

lemma *dp-map2*: *is-direct-product* $x\ y \Longrightarrow$ *is-map* y
⟨*proof*⟩

Next we prove four auxiliary lemmas.

lemma *dp-aux1* [*simp*]:
assumes *is-p-fun* z
and *is-total* w
and $x^{\smile}\ ;\ z = 1$
shows $(w\ ;\ x^{\smile} \cdot y\ ;\ z^{\smile})\ ;\ z = y$
⟨*proof*⟩

lemma *dp-aux2* [*simp*]:
assumes *is-p-fun* z
and *is-total* w
and $z^{\smile}\ ;\ x = 1$
shows $(w\ ;\ x^{\smile} \cdot y\ ;\ z^{\smile})\ ;\ z = y$
⟨*proof*⟩

lemma *dp-aux3* [*simp*]:
assumes *is-p-fun* z
and *is-total* w
and $x^{\smile}\ ;\ z = 1$
shows $(y\ ;\ z^{\smile} \cdot w\ ;\ x^{\smile})\ ;\ z = y$

⟨*proof*⟩

lemma *dp-aux4* [*simp*]:
  assumes *is-p-fun z*
    and *is-total w*
    and $z^{\smile}$; $x = 1$
  shows (y ; $z^{\smile}$ · w ; $x^{\smile}$) ; $z = y$
⟨*proof*⟩

Next we define a function which is an isomorphism on projections.

definition *Phi* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ (‹Φ›)
  where $\Phi \equiv (\lambda w\ x\ y\ z.\ w$; $y^{\smile}$ · x ; $z^{\smile})$

lemma *Phi-conv*: $(\Phi\ w\ x\ y\ z)^{\smile} = y$; $w^{\smile}$ · z ; $x^{\smile}$
⟨*proof*⟩

We prove that Φ is an isomorphism with respect to the projections.

lemma *mono-dp-1*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows $\Phi\ w\ x\ y\ z$; $y = w$
⟨*proof*⟩

lemma *mono-dp-2*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows $\Phi\ w\ x\ y\ z$; $z = x$
⟨*proof*⟩

We now show that Φ is an injective function.

lemma *Phi-map*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-map* $(\Phi\ w\ x\ y\ z)$
⟨*proof*⟩

lemma *Phi-inj*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-inj* $(\Phi\ w\ x\ y\ z)$
⟨*proof*⟩

Next we show that the converse of Φ is an injective function.

lemma *Phi-conv-map*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-map* $((\Phi\ w\ x\ y\ z)^{\smile})$
⟨*proof*⟩

lemma *Phi-conv-inj*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-inj* $((\Phi\ w\ x\ y\ z)^{\smile})$
$\langle proof \rangle$

lemma *Phi-sur*:
  assumes *is-direct-product w x*
  and *is-direct-product y z*
  shows *is-sur* $(\Phi\ w\ x\ y\ z)$
$\langle proof \rangle$

lemma *Phi-conv-sur*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-sur* $((\Phi\ w\ x\ y\ z)^{\smile})$
$\langle proof \rangle$

lemma *Phi-bij*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-bij* $(\Phi\ w\ x\ y\ z)$
$\langle proof \rangle$

lemma *Phi-conv-bij*:
  assumes *is-direct-product w x*
    and *is-direct-product y z*
  shows *is-bij* $((\Phi\ w\ x\ y\ z)^{\smile})$
$\langle proof \rangle$

Next we construct, for given functions f and g, a function F which makes the standard product diagram commute, and we verify these commutation properties.

definition $F :: {}'a \Rightarrow {}'a \Rightarrow {}'a \Rightarrow {}'a \Rightarrow {}'a$
  where $F \equiv (\lambda f\ x\ g\ y.\ f\ ;\ x^{\smile} \cdot g\ ;\ y^{\smile})$

lemma *f-proj*:
  assumes *is-direct-product x y*
  and *is-map g*
  shows $F\ f\ x\ g\ y\ ;\ x = f$
$\langle proof \rangle$

lemma *g-proj*:
  assumes *is-direct-product x y*
  and *is-map f*
  shows $F\ f\ x\ g\ y\ ;\ y = g$
$\langle proof \rangle$

Finally we show uniqueness of F, hence universality of the construction.

lemma
  assumes *is-direct-product* $x\ y$
  and *is-map* f
  and *is-map* g
  and *is-map* G
  and $f = G\ ;\ x$
  and $g = G\ ;\ y$
  shows $G = F\ f\ x\ g\ y$
⟨*proof*⟩

end

end

8 Reflexive Transitive Closure

theory *Relation-Algebra-RTC*
  imports *Relation-Algebra*
begin

We impose the Kleene algebra axioms for the star on relation algebras. This gives us a reflexive transitive closure operation.

class *relation-algebra-rtc* = *relation-algebra* + *star-op* +
  assumes *rtc-unfoldl*: $1' + x\ ;\ x^\star \leq x^\star$
    and *rtc-inductl*: $z + x\ ;\ y \leq y \longrightarrow x^\star\ ;\ z \leq y$
    and *rtc-inductr*: $z + y\ ;\ x \leq y \longrightarrow z\ ;\ x^\star \leq y$

sublocale *relation-algebra-rtc* ⊆ *kleene-algebra* (+) (;) *1′ 0* (≤) (<) *star*
⟨*proof*⟩

context *relation-algebra-rtc*
begin

First, we prove that the obvious interaction between the star and converse is captured by the axioms.

lemma *star-conv*: $(x^\star)^\smile = (x^\smile)^\star$
⟨*proof*⟩

Next we provide an example to show how facts from Kleene algebra are picked up in relation algebra.

lemma *rel-church-rosser*: $(x^\smile)^\star\ ;\ x^\star \leq x^\star\ ;\ (x^\smile)^\star \Longrightarrow (x + x^\smile)^\star = x^\star\ ;\ (x^\smile)^\star$
⟨*proof*⟩

end

end

9 Models of Relation Algebra

theory *Relation-Algebra-Models*
imports *Relation-Algebra Kleene-Algebra.Inf-Matrix*
begin

We formalise two models. First we show the obvious: binary relations form a relation algebra. Then we show that infinite matrices (which we formalised originally for Kleene algebras) form models of relation algebra if we restrict their element type to *bool*.

9.1 Binary Relations

Since Isabelle's libraries for binary relations are very well developed, the proof for this model is entirely trivial.

interpretation *rel-relation-algebra*: *relation-algebra* (−) *uminus* (∩) (⊆) (⊂) (∪) {} *UNIV* (*O*) *Relation.converse Id*
⟨*proof*⟩

9.2 Infinite Boolean Matrices

Next we consider infinite Boolean matrices. We define the maximal Boolean matrix (all of its entries are *True*), the converse or transpose of a matrix, the intersection of two Boolean matrices and the complement of a Boolean matrix.

definition *mat-top* :: ('*a*, '*b*, *bool*) *matrix* (‹τ›)
where $\tau\ i\ j \equiv True$

definition *mat-transpose* :: ('*a*, '*b*, '*c*) *matrix* ⇒ ('*b*, '*a*, '*c*) *matrix* (‹-†› [*101*] *100*)
where $f^\dagger \equiv (\lambda i\ j.\ f\ j\ i)$

definition *mat-inter* :: ('*a*, '*b*, *bool*) *matrix* ⇒ ('*a*, '*b*, *bool*) *matrix* ⇒ ('*a*, '*b*, *bool*) *matrix* (**infixl** ‹⊓› *70*)
where $f \sqcap g \equiv (\lambda i\ j.\ f\ i\ j \cdot g\ i\ j)$

definition *mat-complement* :: ('*a*, '*b*, *bool*) *matrix* ⇒ ('*a*, '*b*, *bool*) *matrix* (‹-^{c}› [*101*] *100*)
where $f^c = (\lambda i\ j.\ -\ f\ i\ j)$

Next we show that the Booleans form a dioid. We state this as an *instantiation* result. The Kleene algebra files contain an *interpretation* proof, which is not sufficient for our purposes.

instantiation *bool* :: *dioid-one-zero*
begin

definition *zero-bool-def*:
zero-bool ≡ *False*

definition *one-bool-def*:
one-bool ≡ *True*

definition *times-bool-def*:
times-bool ≡ (∧)

definition *plus-bool-def*:
plus-bool ≡ (∨)

instance
⟨*proof*⟩

end

We now show that infinite Boolean matrices form a Boolean algebra.

lemma *le-funI2*: $(\bigwedge i\ j.\ f\ i\ j \le g\ i\ j) \Longrightarrow f \le g$
⟨*proof*⟩

interpretation *matrix-ba*: *boolean-algebra* $\lambda f\ g.\ f \sqcap g^c$ *mat-complement* $(\sqcap)$ $(\le)$ $(<)$ *mat-add mat-zero mat-top*
⟨*proof*⟩

We continue working towards the main result of this section, that infinite Boolean matrices form a relation algebra.

lemma *mat-mult-var*: $(f \otimes g) = (\lambda i\ j.\ \sum \{(f\ i\ k) * (g\ k\ j) \mid k.\ k \in UNIV\})$
⟨*proof*⟩

The following fact is related to proving the last relation algebra axiom in the matrix model. It is more complicated than necessary since finite infima are not well developed in Isabelle. Instead we translate properties of finite infima into properties of finite suprema by using Boolean algebra. For finite suprema we have developed special-purpose theorems in the Kleene algebra files.

lemma *mat-res-pointwise*:
fixes *i j* :: *'a::finite*
and *x* :: *('a, 'a, bool) matrix*
shows $(x^\dagger \otimes (x \otimes y)^c)\ i\ j \le (y^c)\ i\ j$
⟨*proof*⟩

Finally the main result of this section.

interpretation *matrix-ra*: *relation-algebra* $\lambda f\ g.\ f \sqcap g^c$ *mat-complement* $(\sqcap)$ $(\le)$ $(<)$ $(\oplus)$ $\lambda i\ j.\ False\ \tau$ $(\otimes)$ *mat-transpose* ε
⟨*proof*⟩

end

References

[1] A. Armstrong, G. Struth, and T. Weber. Programming and automating mathematics in the Tarski-Kleene hierarchy. 2014. To appear in Journal of Logic and Algebraic Programming.

[2] R. Maddux. *Relation Algebras*, volume 150 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, 2006.

[3] G. Schmidt and T. Ströhlein. *Relationen und Graphen*. Springer, 1987.

[4] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6:73–89, 1941.