

# Relation Algebra

Alasdair Armstrong, Simon Foster, Georg Struth, Tjark Weber

October 13, 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

|          |                                           |           |
|----------|-------------------------------------------|-----------|
| <b>1</b> | <b>Introductory Remarks</b>               | <b>2</b>  |
| <b>2</b> | <b>(More) Boolean Algebra</b>             | <b>2</b>  |
| 2.1      | Laws of Boolean Algebra . . . . .         | 2         |
| 2.2      | Boolean Algebras with Operators . . . . . | 4         |
| <b>3</b> | <b>Relation Algebra</b>                   | <b>7</b>  |
| <b>4</b> | <b>Vectors</b>                            | <b>11</b> |
| <b>5</b> | <b>Tests</b>                              | <b>12</b> |
| 5.1      | Tests . . . . .                           | 13        |
| 5.2      | Test Complements . . . . .                | 14        |
| <b>6</b> | <b>Functions</b>                          | <b>14</b> |
| 6.1      | Functions . . . . .                       | 14        |
| 6.2      | Points and Rectangles . . . . .           | 19        |
| 6.3      | Antidomain . . . . .                      | 19        |
| <b>7</b> | <b>Direct Products</b>                    | <b>20</b> |
| <b>8</b> | <b>Reflexive Transitive Closure</b>       | <b>24</b> |

|                                         |           |
|-----------------------------------------|-----------|
| <b>9 Models of Relation Algebra</b>     | <b>25</b> |
| 9.1 Binary Relations . . . . .          | 25        |
| 9.2 Infinite Boolean Matrices . . . . . | 25        |

## 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]:  $(x + y) \cdot -x = y \cdot -x$   
 $\langle proof \rangle$

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

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

**lemma** *join-iso*:  $x \leq y \implies x + z \leq y + z$   
 $\langle proof \rangle$

**lemma** *join-isol*:  $x \leq y \implies z + x \leq z + y$   
 $\langle proof \rangle$

**lemma** *join-double-iso*:  $x \leq y \implies w + x + z \leq w + y + z$   
 $\langle proof \rangle$

**lemma** *comp-anti*:  $x \leq y \longleftrightarrow -y \leq -x$   
 $\langle proof \rangle$

**lemma** *meet-iso*:  $x \leq y \implies x \cdot z \leq y \cdot z$   
 $\langle proof \rangle$

**lemma** *meet-isor*:  $x \leq y \implies z \cdot x \leq z \cdot y$   
 $\langle proof \rangle$

**lemma** *meet-double-iso*:  $x \leq y \implies w \cdot x \cdot z \leq w \cdot y \cdot z$   
 $\langle proof \rangle$

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

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

**lemma** *dist-alt*:  $\llbracket x + z = y + z; x \cdot z = y \cdot z \rrbracket \implies x = y$   
 $\langle proof \rangle$

Finally we prove the Galois connections for complementation.

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

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

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

**lemma** *galois-2*:  $x \leq y + -z \longleftrightarrow x \cdot z \leq y$   
 $\langle \text{proof} \rangle$

**lemma** *galois-aux3*:  $x + y = 1 \longleftrightarrow -x \leq y$   
 $\langle \text{proof} \rangle$

**lemma** *galois-aux4*:  $-x + y = 1 \longleftrightarrow x \leq y$   
 $\langle \text{proof} \rangle$

## 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 to deal with complete Boolean algebras on 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* ::  $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**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 uniquely defined.

**lemma** *is-conjugation-sym*:  $\text{is-conjugation}\ f\ g \longleftrightarrow \text{is-conjugation}\ g\ f$   
 $\langle \text{proof} \rangle$

**lemma** *is-conjugation-unique*:  $\llbracket \text{is-conjugation}\ f\ g; \text{is-conjugation}\ f\ h \rrbracket \implies g = h$   
 $\langle \text{proof} \rangle$

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

**lemma** *conj-galois-1*:  
**assumes** *is-conjugation*  $f\ g$   
**shows**  $f\ x \leq y \longleftrightarrow x \leq -g\ (-y)$   
 $\langle \text{proof} \rangle$

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

**shows**  $g\ x \leq y \longleftrightarrow x \leq -f\ (-y)$   
 $\langle proof \rangle$

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**  $f\ (x + y) \leq z \longleftrightarrow f\ x + f\ y \leq z$   
 $\langle proof \rangle$

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

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

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

Additivity of adjoints obviously implies their isotonicity.

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

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

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

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

Next we prove cancellation and strictness laws.

**lemma** *cancellation-1*:

**assumes** *is-conjugation*  $f\ g$   
**shows**  $f\ (-g\ x) \leq -x$   
 $\langle proof \rangle$

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

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

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

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

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

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

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

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

**lemma** *conjugate-eq-aux*:  
 $is-conjugation\ f\ g \implies f\ (x \cdot -g\ y) \leq f\ x \cdot -y$   
 $\langle proof \rangle$

**lemma** *conjugate-eq*:  
 $is-conjugation\ f\ g \iff (\forall x\ y. f\ (x \cdot -g\ y) \leq f\ x \cdot -y \wedge g\ (y \cdot -f\ x) \leq g\ y \cdot -x)$   
**(is**  $?l \iff ?r$ **)**  
 $\langle proof \rangle$

**lemma** *conjugation-prop1: is-conjugation*  $f\ g \implies f\ y \cdot z \leq f\ (y \cdot g\ z)$   
 $\langle proof \rangle$

**lemma** *conjugation-prop2: is-conjugation*  $f\ g \implies g\ z \cdot y \leq g\ (z \cdot f\ y)$   
 $\langle proof \rangle$

**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  $\Rightarrow$  'a  $\Rightarrow$  'a (**infixl**  $\langle ; \rangle$  75)  
**and** *converse* :: 'a  $\Rightarrow$  'a ( $\langle (-)^\smile \rangle$  [1000] 999)  
**and** *unit* :: 'a ( $\langle 1'' \rangle$ )  
**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*]:  $(x^\smile)^\smile = x$   
**and** *conv-add* [*simp*]:  $(x + y)^\smile = x^\smile + y^\smile$   
**and** *conv-contrav* [*simp*]:  $(x ; y)^\smile = y^\smile ; x^\smile$   
**and** *comp-res*:  $x^\smile ; -(x ; y) \leq -y$

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*  $\subseteq$  *dioid-one*  $(+)$   $(;)$   $(\leq)$   $(<)$   $1'$   
 $\langle proof \rangle$

**context** *relation-algebra*  
**begin**

First we prove some basic facts about joins and meets.

**lemma** *meet-interchange*:  $(w \cdot x) ; (y \cdot z) \leq w ; y \cdot x ; z$   
 $\langle \text{proof} \rangle$

**lemma** *join-interchange*:  $w ; x + y ; z \leq (w + y) ; (x + z)$   
 $\langle \text{proof} \rangle$

We now prove some simple facts about conversion.

**lemma** *conv-iso*:  $x \leq y \longleftrightarrow x^\smile \leq y^\smile$   
 $\langle \text{proof} \rangle$

**lemma** *conv-zero* [*simp*]:  $0^\smile = 0$   
 $\langle \text{proof} \rangle$

**lemma** *conv-one* [*simp*]:  $1^\smile = 1$   
 $\langle \text{proof} \rangle$

**lemma** *conv-compl-aux*:  $(-x)^\smile = (-x)^\smile + (-x^\smile)$   
 $\langle \text{proof} \rangle$

**lemma** *conv-compl*:  $(-x)^\smile = -(x^\smile)$   
 $\langle \text{proof} \rangle$

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

**lemma** *conv-e* [*simp*]:  $1^\smile = 1'$   
 $\langle \text{proof} \rangle$

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

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

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

**lemma** *conv-self-conjugate-var*: *is-conjugation converse converse*  
 $\langle \text{proof} \rangle$

The following lemmas link the relative product and meet.

**lemma** *one-idem-mult* [*simp*]:  $1 ; 1 = 1$   
 $\langle \text{proof} \rangle$

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

$\langle proof \rangle$

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

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

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*:  $x ; y \cdot z^\smile = 0 \implies y ; z \cdot x^\smile = 0$   
 $\langle proof \rangle$

**lemma** *peirce*:  $x ; y \cdot z^\smile = 0 \longleftrightarrow y ; z \cdot x^\smile = 0$   
 $\langle proof \rangle$

**lemma** *schroeder-1*:  $x ; y \cdot z = 0 \longleftrightarrow y \cdot x^\smile ; z = 0$   
 $\langle proof \rangle$

**lemma** *schroeder-2*:  $y ; x \cdot z = 0 \longleftrightarrow y \cdot z ; x^\smile = 0$   
 $\langle proof \rangle$

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  $(x^\smile)$ )  
 $\langle proof \rangle$

**lemma** *schroeder-2-var*: *is-conjugation*  $(\lambda x. x ; y)$   $(\lambda x. x ; y^\smile)$   
 $\langle proof \rangle$

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

**lemma** *conv-galois-1*:  $x ; y \leq z \longleftrightarrow y \leq -(x^\smile ; -z)$   
 $\langle proof \rangle$

**lemma** *conv-galois-2*:  $y ; x \leq z \longleftrightarrow y \leq -(-z ; x^\smile)$   
 $\langle proof \rangle$

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

**lemma** *modular-1-aux'*:  $x ; (y \cdot -(x^\smile ; z)) \cdot z = 0$   
 $\langle proof \rangle$

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

$\langle proof \rangle$

**lemma** *modular-1'*:  $x ; y \cdot z = x ; (y \cdot x^\smile ; z) \cdot z$   
 $\langle proof \rangle$

**lemma** *modular-2'*:  $y ; x \cdot z = (y \cdot z ; x^\smile) ; x \cdot z$   
 $\langle proof \rangle$

**lemma** *modular-1-var*:  $x ; y \cdot z \leq x ; (y \cdot x^\smile ; z)$   
 $\langle proof \rangle$

**lemma** *modular-2-var*:  $x ; y \cdot z \leq (x \cdot z ; y^\smile) ; y$   
 $\langle proof \rangle$

**lemma** *modular-var-2*:  $x ; y \leq x ; (y \cdot x^\smile ; 1)$   
 $\langle proof \rangle$

**lemma** *modular-var-3*:  $x ; y \leq (x \cdot 1 ; y^\smile) ; y$   
 $\langle proof \rangle$

The modular laws are used to prove the Dedekind rule.

**lemma** *dedekind*:  $x ; y \cdot z \leq (x \cdot z ; y^\smile) ; (y \cdot x^\smile ; z)$   
 $\langle proof \rangle$

**lemma** *dedekind-var-1*:  $x ; y \leq (x \cdot 1 ; y^\smile) ; (y \cdot x^\smile ; 1)$   
 $\langle proof \rangle$

**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*  $(+)$   $(;)$   $1'$   $0$   $(\leq)$   $(<)$   
 $\langle proof \rangle$

**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*:  $(x \cdot y ; 1) ; z = x ; z \cdot y ; 1$   
 $\langle proof \rangle$

**lemma** *ra-2*:  $x ; (z \cdot y ; 1) = (x \cdot (y ; 1)^\smile) ; z$   
 $\langle proof \rangle$

**lemma** *one-conv*:  $1' \cdot x ; 1 = 1' \cdot x ; x^\smile$   
 $\langle proof \rangle$

**lemma** *maddux-12*:  $-(y ; x) ; x^\smile \leq -y$   
 $\langle proof \rangle$

**lemma** *maddux-141*:  $x ; y \cdot z = 0 \longleftrightarrow x^\smile ; z \cdot y = 0$   
 $\langle proof \rangle$

**lemma** *maddux-142*:  $x^\smile ; z \cdot y = 0 \longleftrightarrow z ; y^\smile \cdot x = 0$   
 $\langle proof \rangle$

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

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

**lemma** *maddux-20*:  $x \leq x ; 1$   
 $\langle proof \rangle$

**lemma** *maddux-21*:  $x \leq 1 ; x$   
 $\langle proof \rangle$

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

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

**lemma** *one-compl*:  $-(x ; 1) ; 1 = -(x ; 1)$   
 $\langle proof \rangle$

**lemma** *ss-p18*:  $x ; 1 = 0 \longleftrightarrow x = 0$   
 $\langle proof \rangle$

**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  $x \Longrightarrow is\_vector\ (-x)$ 
   $\langle proof \rangle$ 

lemma vector-add:  $\llbracket is\_vector\ x ; is\_vector\ y \rrbracket \Longrightarrow is\_vector\ (x + y)$ 
   $\langle proof \rangle$ 

lemma vector-mult:  $\llbracket is\_vector\ x ; is\_vector\ y \rrbracket \Longrightarrow is\_vector\ (x \cdot y)$ 
   $\langle proof \rangle$ 

lemma vector-comp:  $\llbracket is\_vector\ x ; is\_vector\ y \rrbracket \Longrightarrow is\_vector\ (x ; y)$ 
   $\langle proof \rangle$ 

lemma vector-1: is-vector  $x \Longrightarrow (x \cdot y) ; z = x \cdot y ; z$ 
   $\langle proof \rangle$ 

lemma vector-1-comm: is-vector  $y \Longrightarrow (x \cdot y) ; z = x ; z \cdot y$ 
   $\langle proof \rangle$ 

lemma vector-2: is-vector  $y \Longrightarrow (x \cdot y^\smile) ; z = x ; (y \cdot z)$ 
   $\langle proof \rangle$ 

lemma vector-2-var: is-vector  $y \Longrightarrow (x \cdot y^\smile) ; z = (x \cdot y^\smile) ; (y \cdot z)$ 
   $\langle proof \rangle$ 

lemma vector-idem [simp]: is-vector  $x \Longrightarrow x ; x = x$ 
   $\langle proof \rangle$ 

lemma vector-rectangle [simp]: is-vector  $x \Longrightarrow x ; 1 ; x = x$ 
   $\langle proof \rangle$ 

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

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*  $x \Longrightarrow \text{is-test } (x^\smile)$   
 $\langle \text{proof} \rangle$

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

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

**lemma** *test-sum*:  $\llbracket \text{is-test } x; \text{is-test } y \rrbracket \Longrightarrow \text{is-test } (x + y)$   
 $\langle \text{proof} \rangle$

**lemma** *test-prod*:  $\llbracket \text{is-test } x; \text{is-test } y \rrbracket \Longrightarrow \text{is-test } (x \cdot y)$   
 $\langle \text{proof} \rangle$

**lemma** *test-comp*:  $\llbracket \text{is-test } x; \text{is-test } y \rrbracket \Longrightarrow \text{is-test } (x ; y)$   
 $\langle \text{proof} \rangle$

**lemma** *test-comp-eq-mult*:  
**assumes** *is-test*  $x$   
**and** *is-test*  $y$   
**shows**  $x ; y = x \cdot y$   
 $\langle \text{proof} \rangle$

**lemma** *test-1* [*simp*]: *is-test*  $x \Longrightarrow x ; 1 \cdot y = x ; y$   
 $\langle \text{proof} \rangle$

**lemma** *maddux-32* [*simp*]: *is-test*  $x \Longrightarrow -(x ; 1) \cdot 1' = -x \cdot 1'$   
 $\langle \text{proof} \rangle$

**lemma** *test-distr-1* :  
**assumes** *is-test*  $x$   
**and** *is-test*  $y$   
**shows**  $x ; z \cdot y ; z = (x \cdot y) ; z$   
 $\langle \text{proof} \rangle$

**lemma** *maddux-35*: *is-test*  $x \Longrightarrow x ; y \cdot -z = x ; y \cdot -(x ; z)$   
 $\langle \text{proof} \rangle$

## 5.2 Test Complements

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

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

**lemma** *test-compl-1* [simp]:  $is-test\ x \Longrightarrow x + tc\ x = 1'$   
*<proof>*

**lemma** *test-compl-2* [simp]:  $is-test\ x \Longrightarrow x \cdot tc\ x = 0$   
*<proof>*

**lemma** *test-test-compl*:  $is-test\ x \Longrightarrow is-test\ (tc\ x)$   
*<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]:  $is-test\ x \Longrightarrow tc\ (tc\ x) = x$   
*<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**  $is-p-fun :: 'a \Rightarrow bool$   
**where**  $is-p-fun\ x \equiv x^\smile ; x \leq 1'$

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

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

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

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

**definition**  $is-bij :: 'a \Rightarrow bool$   
**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**  $inj-p-fun: is-inj\ x \longleftrightarrow is-p-fun\ (x^\smile)$   
 $\langle proof \rangle$

**lemma**  $p-fun-inj: is-p-fun\ x \longleftrightarrow is-inj\ (x^\smile)$   
 $\langle proof \rangle$

**lemma**  $sur-total: is-sur\ x \longleftrightarrow is-total\ (x^\smile)$   
 $\langle proof \rangle$

**lemma**  $total-sur: is-total\ x \longleftrightarrow is-sur\ (x^\smile)$   
 $\langle proof \rangle$

**lemma**  $bij-conv: is-bij\ x \longleftrightarrow is-bij\ (x^\smile)$   
 $\langle proof \rangle$

Next we show that tests are partial injections.

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

Next we show composition properties.

**lemma**  $p-fun-comp:$   
**assumes**  $is-p-fun\ x$  **and**  $is-p-fun\ y$   
**shows**  $is-p-fun\ (x ; y)$   
 $\langle proof \rangle$

**lemma** *p-fun-mult-var*:  $x^\smile ; x \leq 1' \implies (x \cdot y)^\smile ; (x \cdot y) \leq 1'$   
 $\langle \text{proof} \rangle$

**lemma** *inj-compose*:  
 assumes *is-inj*  $x$  and *is-inj*  $y$   
 shows *is-inj*  $(x ; y)$   
 $\langle \text{proof} \rangle$

**lemma** *inj-mult-var*:  $x^\smile ; x \leq 1' \implies (x \cdot y)^\smile ; (x \cdot y) \leq 1'$   
 $\langle \text{proof} \rangle$

**lemma** *total-comp*:  
 assumes *is-total*  $x$  and *is-total*  $y$   
 shows *is-total*  $(x ; y)$   
 $\langle \text{proof} \rangle$

**lemma** *total-add-var*:  $1' \leq x^\smile ; x \implies 1' \leq (x + y)^\smile ; (x + y)$   
 $\langle \text{proof} \rangle$

**lemma** *sur-comp*:  
 assumes *is-sur*  $x$  and *is-sur*  $y$   
 shows *is-sur*  $(x ; y)$   
 $\langle \text{proof} \rangle$

**lemma** *sur-sum-var*:  $1' \leq x^\smile ; x \implies 1' \leq (x + y)^\smile ; (x + y)$   
 $\langle \text{proof} \rangle$

**lemma** *map-comp*:  
 assumes *is-map*  $x$  and *is-map*  $y$   
 shows *is-map*  $(x ; y)$   
 $\langle \text{proof} \rangle$

**lemma** *bij-comp*:  
 assumes *is-bij*  $x$  and *is-bij*  $y$   
 shows *is-bij*  $(x ; y)$   
 $\langle \text{proof} \rangle$

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

**lemma** *p-fun-distl*:  $\text{is-p-fun } x \implies x ; (y \cdot z) = x ; y \cdot x ; z$   
 $\langle \text{proof} \rangle$

**lemma** *map-distl*:  $\text{is-map } x \implies x ; (y \cdot z) = x ; y \cdot x ; z$   
 $\langle \text{proof} \rangle$

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

**lemma** *p-fun-zero*:  $\text{is-p-fun } x \implies x ; y \cdot x ; -y = 0$

$\langle proof \rangle$

**lemma** *total-one*:  $is-total\ x \implies x ; 1 = 1$

$\langle proof \rangle$

**lemma** *total-1*:  $is-total\ x \implies (\forall y. y ; x = 0 \longrightarrow y = 0)$

$\langle proof \rangle$

**lemma** *surj-one*:  $is-sur\ x \implies 1 ; x = 1$

$\langle proof \rangle$

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

$\langle proof \rangle$

**lemma** *bij-is-maprop*:

assumes *is-bij*  $x$  and *is-map*  $x$

shows  $x^\smile ; x = 1' \wedge x ; x^\smile = 1'$

$\langle proof \rangle$

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

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

Finally we prove miscellaneous properties of functions.

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

$\langle proof \rangle$

**lemma** *p-fun-compl*:  $is-p-fun\ x \implies x ; -y \leq -(x ; y)$

$\langle proof \rangle$

**lemma** *ss-422v*:  $is-p-fun\ x \implies x ; -y = x ; 1 \cdot -(x ; y)$   
 $\langle proof \rangle$

The next property is a Galois connection.

**lemma** *ss43iii*:  $is-map\ x \longleftrightarrow (\forall y. x ; -y = -(x ; y))$   
 $\langle proof \rangle$

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*:  $is-map\ x \implies y ; x \leq z \longleftrightarrow y \leq z ; x^\smile$   
 $\langle proof \rangle$

**lemma** *ss424i*:  $is-total\ x \longleftrightarrow (\forall y. -(x ; y) \leq x ; -y)$   
 $\langle proof \rangle$

**lemma** *ss434ii*:  $is-p-fun\ x \longleftrightarrow (\forall y. x ; -y \leq -(x ; y))$   
 $\langle proof \rangle$

**lemma** *is-mapprop1*:  $is-map\ x \implies (y \leq x ; z ; x^\smile \longleftrightarrow y ; x \leq x ; z)$   
 $\langle proof \rangle$

**lemma** *is-mapprop2*:  $is-map\ x \implies (y ; x \leq x ; z \longleftrightarrow x^\smile ; y ; x \leq z)$   
 $\langle proof \rangle$

**lemma** *is-mapprop3*:  $is-map\ x \implies (x^\smile ; y ; x \leq z \longleftrightarrow x^\smile ; y \leq z ; x^\smile)$   
 $\langle proof \rangle$

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

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

**lemma** *bij-inv-1* [*simp*]:  $is-bij\ x \implies x ; x^\smile = 1'$   
 $\langle proof \rangle$

**lemma** *bij-inv-2* [*simp*]:  $is-bij\ x \implies x^\smile ; x = 1'$   
 $\langle proof \rangle$

**lemma** *bij-inv-comm*:  $is-bij\ x \implies x ; x^\smile = x^\smile ; x$   
 $\langle proof \rangle$

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

$\langle proof \rangle$

**lemma** *inj-map-monomorph*:  $\llbracket is-inj\ x; is-map\ x \rrbracket \implies (\forall\ y\ z. y ; x = z ; x \longrightarrow y = z)$   
 $\langle proof \rangle$

**lemma** *sur-map-epimorph*:  $\llbracket is-sur\ x; is-map\ x \rrbracket \implies (\forall\ y\ z. x ; y = x ; z \longrightarrow y = z)$   
 $\langle proof \rangle$

## 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-point*  $x \equiv is-vector\ x \wedge is-inj\ x \wedge x \neq 0$

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

**lemma** *rectangle-eq* [*simp*]: *is-rectangle*  $x \longleftrightarrow x ; 1 ; x = x$   
 $\langle proof \rangle$

## 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* ::  $'a \Rightarrow 'a\ (\langle a \rangle)$   
**where**  $a\ x = 1' \cdot -(x ; 1)$

**definition** *dom* ::  $'a \Rightarrow 'a\ (\langle d \rangle)$   
**where**  $d\ x = a\ (a\ x)$

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

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

**lemma** *dom-def-aux-var*:  $d\ x = 1' \cdot x ; x^\smile$   
 $\langle proof \rangle$

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

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

**lemma** *dom-verystrict*:  $d\ x = 0 \longleftrightarrow x = 0$   
 $\langle proof \rangle$

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

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

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

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

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*:  $x ; 1 = d\ x ; 1$   
 $\langle proof \rangle$

**lemma** *test-dom*: *is-test* ( $d\ x$ )  
 $\langle proof \rangle$

**lemma** *p-fun-dom*: *is-p-fun* ( $d\ x$ )  
 $\langle proof \rangle$

**lemma** *inj-dom*: *is-inj* ( $d\ x$ )  
 $\langle proof \rangle$

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

**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*  $x$   
 $\langle$ *proof* $\rangle$

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

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

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

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

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

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

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

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**  $(w ; x^\smile \cdot y ; z^\smile) ; z = y$   
 $\langle$ *proof* $\rangle$

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

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

$\langle proof \rangle$

**lemma** *dp-aux4* [*simp*]:  
**assumes** *is-p-fun* *z*  
**and** *is-total* *w*  
**and**  $z^\smile ; x = 1$   
**shows**  $(y ; z^\smile \cdot w ; x^\smile) ; z = y$   
 $\langle proof \rangle$

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

**definition** *Phi* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$  ( $\langle \Phi \rangle$ )  
**where**  $\Phi \equiv (\lambda w x y z. w ; y^\smile \cdot x ; z^\smile)$

**lemma** *Phi-conv*:  $(\Phi w x y z)^\smile = y ; w^\smile \cdot z ; x^\smile$   
 $\langle proof \rangle$

We prove that  $\Phi$  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**  $\Phi w x y z ; y = w$   
 $\langle proof \rangle$

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

We now show that  $\Phi$  is an injective function.

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

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

Next we show that the converse of  $\Phi$  is an injective function.

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

**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**  $G = F f x g y$

$\langle proof \rangle$

**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*:  $1' + x ; x^* \leq x^*$

**and** *rtc-inductl*:  $z + x ; y \leq y \longrightarrow x^* ; z \leq y$

**and** *rtc-inductr*:  $z + y ; x \leq y \longrightarrow z ; x^* \leq y$

**sublocale** *relation-algebra-rtc*  $\subseteq$  *kleene-algebra*  $(+)$   $(;)$   $1' 0 (\leq) (<) star$

$\langle proof \rangle$

**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*:  $(x^*)^\smile = (x^\smile)^*$

$\langle proof \rangle$

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

**lemma** *rel-church-rosser*:  $(x^\smile)^* ; x^* \leq x^* ; (x^\smile)^* \implies (x + x^\smile)^* = x^* ; (x^\smile)^*$

$\langle proof \rangle$

**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 τ i j ≡ True
```

```
definition mat-transpose :: ('a, 'b, 'c) matrix ⇒ ('b, 'a, 'c) matrix (⟨-†⟩ [101] 100)
where f† ≡ (λi j. f j i)
```

```
definition mat-inter :: ('a, 'b, bool) matrix ⇒ ('a, 'b, bool) matrix ⇒ ('a, 'b, bool)
matrix (infixl ⟨∩⟩ 70)
where f ∩ g ≡ (λi j. f i j · g i j)
```

```
definition mat-complement :: ('a, 'b, bool) matrix ⇒ ('a, 'b, bool) matrix (⟨-c⟩
[101] 100)
where fc = (λi 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*  $\equiv$  *True*

**definition** *times-bool-def*:

*times-bool*  $\equiv$   $(\wedge)$

**definition** *plus-bool-def*:

*plus-bool*  $\equiv$   $(\vee)$

**instance**

*<proof>*

**end**

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

**lemma** *le-funI2*:  $(\bigwedge i j. f\ i\ j \leq g\ i\ j) \implies f \leq g$

*<proof>*

**interpretation** *matrix-ba*: *boolean-algebra*  $\lambda f\ g. f \sqcap g^c$  *mat-complement*  $(\sqcap)$   $(\leq)$

$(<)$  *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**  $(x^\dagger \otimes (x \otimes y)^c)\ i\ j \leq (y^c)\ i\ j$

*<proof>*

Finally the main result of this section.

**interpretation** *matrix-ra*: *relation-algebra*  $\lambda f\ g. f \sqcap g^c$  *mat-complement*  $(\sqcap)$   $(\leq)$

$(<)$   $(\oplus)$   $\lambda i j. \text{False}$   $\tau$   $(\otimes)$  *mat-transpose*  $\varepsilon$

*<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.