Kleene Algebras with Domain

Victor B. F. Gomes, Walter Guttmann, Peter Höfner,
Georg Struth and Tjark Weber

September 23, 2023

Abstract

Kleene algebras with domain are Kleene algebras endowed with an operation that maps each element of the algebra to its domain of definition (or its complement) in abstract fashion. They form a simple algebraic basis for Hoare logics, dynamic logics or predicate transformer semantics. We formalise a modular hierarchy of algebras with domain and antidomain (domain complement) operations in Isabelle/HOL that ranges from domain and antidomain semigroups to modal Kleene algebras and divergence Kleene algebras. We link these algebras with models of binary relations and program traces. We include some examples from modal logics, termination and program analysis.

Contents

1 Introductory Remarks

These theory files are intended as a reference formalisation for variants of Kleene algebras with domain. The algebraic hierarchy is developed in a modular way from domain and antidomain semigroups to modal Kleene algebras in which forward and backward box and diamond operators interact via conjugations and Galois connections. Throughout the development we have aimed at readable proofs so that these theories can be seen as a machine-checked introduction to reasoning in this setting. Apart from that, the Isabelle code is only sparsely annotated, and we refer to a series of articles for further information.

Our formalisation follows the approaches of Desharnais, Jipsen and Struth to domain semigroups [3] and Desharnais and Struth to families of domain semirings and Kleene algebras with domain [7, 6]. The link with modal Kleene algebras, Hoare logics and predicate transformers has been elaborated by Möller and Struth [13]; a notion of divergence has been added by Desharnais, Möller and Struth [5]. A previous stage of this formalisation has been documented in a companion article [11].

The target model of these axiomatisations are binary relations, where the domain operation represents the set of those elements that are related to some other element. There is a vast amount of literature on axiomatising the domain of functions, especially in semigroup theory. The deterministic nature of functions, however, leads to different axiom sets. An integration of these approaches is left for future work.

Our Isabelle/HOL formalisation itself is based on a formalisation of variants of Kleene algebras [1]. An adaptation of Kleene algebras with domain to the setting of concurrent dynamic algebra [10] can also be found in the Archive of Formal Proofs [9]. A formalisation of the original two-sorted approach to Kleene algebra with domain [4] is left for future work as well.

2 Domain Semirings

theory *Domain-Semiring*
imports *Kleene-Algebra.Kleene-Algebra*

begin

2.1 Domain Semigroups and Domain Monoids

class *domain-op* =
  fixes *domain-op* :: $'a \Rightarrow 'a$ (d)

First we define the class of domain semigroups. Axioms are taken from [3].

class *domain-semigroup* = *semigroup-mult* + *domain-op* +
  assumes *dsg1* [*simp*]: $d\ x \cdot x = x$
  and *dsg2* [*simp*]: $d\ (x \cdot d\ y) = d\ (x \cdot y)$
  and *dsg3* [*simp*]: $d\ (d\ x \cdot y) = d\ x \cdot d\ y$
  and *dsg4*: $d\ x \cdot d\ y = d\ y \cdot d\ x$

begin

lemma *domain-invol* [*simp*]: $d\ (d\ x) = d\ x$
⟨*proof*⟩

The next lemmas show that domain elements form semilattices.

lemma *dom-el-idem* [*simp*]: $d\ x \cdot d\ x = d\ x$
⟨*proof*⟩

lemma *dom-mult-closed* [*simp*]: $d\ (d\ x \cdot d\ y) = d\ x \cdot d\ y$
  ⟨*proof*⟩

lemma *dom-lc3* [*simp*]: $d\ x \cdot d\ (x \cdot y) = d\ (x \cdot y)$
⟨*proof*⟩

lemma *d-fixpoint*: $(\exists\, y.\ x = d\ y) \longleftrightarrow x = d\ x$
  ⟨*proof*⟩

lemma *d-type*: $\forall\, P.\ (\forall\, x.\ x = d\ x \longrightarrow P\ x) \longleftrightarrow (\forall\, x.\ P\ (d\ x))$
  ⟨*proof*⟩

We define the semilattice ordering on domain semigroups and explore the semilattice of domain elements from the order point of view.

definition *ds-ord* :: $'a \Rightarrow 'a \Rightarrow bool$ (**infix** $\sqsubseteq$ *50*) **where**
$x \sqsubseteq y \longleftrightarrow x = d\ x \cdot y$

lemma *ds-ord-refl*: $x \sqsubseteq x$
⟨*proof*⟩

lemma *ds-ord-trans*: $x \sqsubseteq y \Longrightarrow y \sqsubseteq z \Longrightarrow x \sqsubseteq z$
⟨*proof*⟩

lemma *ds-ord-antisym*: $x \sqsubseteq y \Longrightarrow y \sqsubseteq x \Longrightarrow x = y$
⟨*proof*⟩

This relation is indeed an order.

sublocale *ds*: *ordering* ‹($\sqsubseteq$)› ‹$\lambda x\ y.\ x \sqsubseteq y \wedge x \neq y$›
⟨*proof*⟩

declare *ds.refl* [*simp*]

lemma *ds-ord-eq*: $x \sqsubseteq d\ x \longleftrightarrow x = d\ x$
⟨*proof*⟩

lemma $x \sqsubseteq y \Longrightarrow z \cdot x \sqsubseteq z \cdot y$

⟨*proof*⟩

lemma *ds-ord-iso-right*: $x \sqsubseteq y \Longrightarrow x \cdot z \sqsubseteq y \cdot z$
⟨*proof*⟩

The order on domain elements could as well be defined based on multiplication/meet.

lemma *ds-ord-sl-ord*: $d\ x \sqsubseteq d\ y \longleftrightarrow d\ x \cdot d\ y = d\ x$
⟨*proof*⟩

lemma *ds-ord-1*: $d\ (x \cdot y) \sqsubseteq d\ x$
⟨*proof*⟩

lemma *ds-subid-aux*: $d\ x \cdot y \sqsubseteq y$
⟨*proof*⟩

lemma $y \cdot d\ x \sqsubseteq y$

⟨*proof*⟩

lemma *ds-dom-iso*: $x \sqsubseteq y \Longrightarrow d\ x \sqsubseteq d\ y$
⟨*proof*⟩

lemma *ds-dom-llp*: $x \sqsubseteq d\ y \cdot x \longleftrightarrow d\ x \sqsubseteq d\ y$
⟨*proof*⟩

lemma *ds-dom-llp-strong*: $x = d\ y \cdot x \longleftrightarrow d\ x \sqsubseteq d\ y$
⟨*proof*⟩

definition *refines* :: $'a \Rightarrow 'a \Rightarrow bool$
where $refines\ x\ y \equiv d\ y \sqsubseteq d\ x \wedge (d\ y) \cdot x \sqsubseteq y$

lemma *refines-refl*: *refines x x*
⟨*proof*⟩

lemma *refines-trans*: $refines\ x\ y \Longrightarrow refines\ y\ z \Longrightarrow refines\ x\ z$
⟨*proof*⟩

lemma *refines-antisym*: $refines\ x\ y \Longrightarrow refines\ y\ x \Longrightarrow x = y$
⟨*proof*⟩

sublocale *ref*: *ordering refines* $\lambda x\ y.\ (refines\ x\ y \wedge x \neq y)$
⟨*proof*⟩

end

We expand domain semigroups to domain monoids.

class *domain-monoid* = *monoid-mult* + *domain-semigroup*
begin

lemma *dom-one* [*simp*]: $d\ 1 = 1$
⟨*proof*⟩

lemma *ds-subid-eq*: $x \sqsubseteq 1 \longleftrightarrow x = d\ x$
⟨*proof*⟩

end

2.2 Domain Near-Semirings

The axioms for domain near-semirings are taken from [6].

class *domain-near-semiring* = *ab-near-semiring* + *plus-ord* + *domain-op* +
assumes *dns1* [*simp*]: $d\ x \cdot x = x$
and *dns2* [*simp*]: $d\ (x \cdot d\ y) = d(x \cdot y)$
and *dns3* [*simp*]: $d\ (x + y) = d\ x + d\ y$
and *dns4*: $d\ x \cdot d\ y = d\ y \cdot d\ x$
and *dns5* [*simp*]: $d\ x \cdot (d\ x + d\ y) = d\ x$

begin

Domain near-semirings are automatically dioids; addition is idempotent.

subclass *near-dioid*
⟨*proof*⟩

Next we prepare to show that domain near-semirings are domain semigroups.

lemma *dom-iso*: $x \leq y \Longrightarrow d\ x \leq d\ y$
⟨*proof*⟩

lemma *dom-add-closed* [*simp*]: $d\ (d\ x + d\ y) = d\ x + d\ y$
⟨*proof*⟩

lemma *dom-absorp-2* [*simp*]: $d\ x + d\ x \cdot d\ y = d\ x$
⟨*proof*⟩

lemma *dom-1*: $d\ (x \cdot y) \leq d\ x$
⟨*proof*⟩

lemma *dom-subid-aux2*: $d\ x \cdot y \leq y$
⟨*proof*⟩

lemma *dom-glb*: $d\ x \leq d\ y \Longrightarrow d\ x \leq d\ z \Longrightarrow d\ x \leq d\ y \cdot d\ z$
⟨*proof*⟩

lemma *dom-glb-eq*: $d\ x \leq d\ y \cdot d\ z \longleftrightarrow d\ x \leq d\ y \wedge d\ x \leq d\ z$
⟨*proof*⟩

lemma *dom-ord*: $d\ x \leq d\ y \longleftrightarrow d\ x \cdot d\ y = d\ x$
⟨*proof*⟩

lemma *dom-export* [*simp*]: $d\ (d\ x \cdot y) = d\ x \cdot d\ y$
⟨*proof*⟩

subclass *domain-semigroup*
⟨*proof*⟩

We compare the domain semigroup ordering with that of the dioid.

lemma *d-two-orders*: $d\ x \sqsubseteq d\ y \longleftrightarrow d\ x \leq d\ y$
⟨*proof*⟩

lemma *two-orders*: $x \sqsubseteq y \Longrightarrow x \leq y$
⟨*proof*⟩

lemma $x \leq y \Longrightarrow x \sqsubseteq y$

⟨*proof*⟩

Next we prove additional properties.

lemma *dom-subdist*: $d\ x \leq d\ (x + y)$
⟨*proof*⟩

lemma *dom-distrib*: $d\ x + d\ y \cdot d\ z = (d\ x + d\ y) \cdot (d\ x + d\ z)$
⟨*proof*⟩

lemma *dom-llp1*: $x \leq d\ y \cdot x \Longrightarrow d\ x \leq d\ y$

⟨*proof*⟩

lemma *dom-llp2*: $d\ x \leq d\ y \Longrightarrow x \leq d\ y \cdot x$
⟨*proof*⟩

lemma *dom-llp*: $x \leq d\ y \cdot x \longleftrightarrow d\ x \leq d\ y$
⟨*proof*⟩

end

We expand domain near-semirings by an additive unit, using slightly different axioms.

class *domain-near-semiring-one* = *ab-near-semiring-one* + *plus-ord* + *domain-op* +
assumes *dnso1* [*simp*]: $x + d\ x \cdot x = d\ x \cdot x$
and *dnso2* [*simp*]: $d\ (x \cdot d\ y) = d\ (x \cdot y)$
and *dnso3* [*simp*]: $d\ x + 1 = 1$
and *dnso4* [*simp*]: $d\ (x + y) = d\ x + d\ y$
and *dnso5*: $d\ x \cdot d\ y = d\ y \cdot d\ x$

begin

The previous axioms are derivable.

subclass *domain-near-semiring*
⟨*proof*⟩

subclass *domain-monoid* ⟨*proof*⟩

lemma *dom-subid*: $d\ x \leq 1$
⟨*proof*⟩

end

We add a left unit of multiplication.

class *domain-near-semiring-one-zerol* = *ab-near-semiring-one-zerol* + *domain-near-semiring-one* +
assumes *dnso6* [*simp*]: $d\ 0 = 0$

begin

lemma *domain-very-strict*: $d\ x = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *dom-weakly-local*: $x \cdot y = 0 \longleftrightarrow x \cdot d\ y = 0$
⟨*proof*⟩

end

2.3 Domain Pre-Dioids

Pre-semirings with one and a left zero are automatically dioids. Hence there is no point defining domain pre-semirings separately from domain dioids. The axioms are once again from [6].

class *domain-pre-dioid-one* = *pre-dioid-one* + *domain-op* +
assumes *dpd1* : $x \leq d\ x \cdot x$
and *dpd2* [*simp*]: $d\ (x \cdot d\ y) = d\ (x \cdot y)$
and *dpd3* [*simp*]: $d\ x \leq 1$
and *dpd4* [*simp*]: $d\ (x + y) = d\ x + d\ y$

begin

We prepare to show that every domain pre-dioid with one is a domain near-dioid with one.

lemma *dns1''* [*simp*]: $d\ x \cdot x = x$
⟨*proof*⟩

lemma *d-iso*: $x \leq y \Longrightarrow d\ x \leq d\ y$
⟨*proof*⟩

lemma *domain-1''*: $d\ (x \cdot y) \leq d\ x$
⟨*proof*⟩

lemma *domain-export''* [*simp*]: $d\ (d\ x \cdot y) = d\ x \cdot d\ y$
⟨*proof*⟩

lemma *dom-subid-aux1''*: $d\ x \cdot y \leq y$
⟨*proof*⟩

lemma *dom-subid-aux2''*: $x \cdot d\ y \leq x$
⟨*proof*⟩

lemma *d-comm*: $d\ x \cdot d\ y = d\ y \cdot d\ x$
⟨*proof*⟩

subclass *domain-near-semiring-one*
⟨*proof*⟩

lemma *domain-subid*: $x \leq 1 \Longrightarrow x \leq d\ x$
⟨*proof*⟩

lemma *d-preserves-equation*: $d\ y \cdot x \leq x \cdot d\ z \longleftrightarrow d\ y \cdot x = d\ y \cdot x \cdot d\ z$
⟨*proof*⟩

lemma *d-restrict-iff*: $(x \leq y) \longleftrightarrow (x \leq d\ x \cdot y)$
⟨*proof*⟩

lemma *d-restrict-iff-1*: $(d\ x \cdot y \leq z) \longleftrightarrow (d\ x \cdot y \leq d\ x \cdot z)$

⟨*proof*⟩

end

We add once more a left unit of multiplication.

class *domain-pre-dioid-one-zerol* = *domain-pre-dioid-one* + *pre-dioid-one-zerol* +
assumes *dpd5* [*simp*]: $d\ 0 = 0$

begin

subclass *domain-near-semiring-one-zerol*
⟨*proof*⟩

end

2.4 Domain Semirings

We do not consider domain semirings without units separately at the moment. The axioms are taken from from [7]

class *domain-semiringl* = *semiring-one-zerol* + *plus-ord* + *domain-op* +
assumes *dsr1* [*simp*]: $x + d\ x \cdot x = d\ x \cdot x$
and *dsr2* [*simp*]: $d\ (x \cdot d\ y) = d\ (x \cdot y)$
and *dsr3* [*simp*]: $d\ x + 1 = 1$
and *dsr4* [*simp*]: $d\ 0 = 0$
and *dsr5* [*simp*]: $d\ (x + y) = d\ x + d\ y$

begin

Every domain semiring is automatically a domain pre-dioid with one and left zero.

subclass *dioid-one-zerol*
⟨*proof*⟩

subclass *domain-pre-dioid-one-zerol*
⟨*proof*⟩

end

class *domain-semiring* = *domain-semiringl* + *semiring-one-zero*

2.5 The Algebra of Domain Elements

We show that the domain elements of a domain semiring form a distributive lattice. Unfortunately we cannot prove this within the type class of domain semirings.

typedef (**overloaded**) *'a d-element* = $\{x :: 'a :: domain\text{-}semiring.\ x = d\ x\}$
⟨*proof*⟩

setup-lifting *type-definition-d-element*

instantiation *d-element* :: (*domain-semiring*) *bounded-lattice*

begin

lift-definition *less-eq-d-element* :: $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* $\Rightarrow$ *bool* **is** $(\leq)$ ⟨*proof*⟩

lift-definition *less-d-element* :: $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* $\Rightarrow$ *bool* **is** $(<)$ ⟨*proof*⟩

lift-definition *bot-d-element* :: $'a$ *d-element* **is** *0*
⟨*proof*⟩

lift-definition *top-d-element* :: $'a$ *d-element* **is** *1*
⟨*proof*⟩

lift-definition *inf-d-element* :: $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* **is** $(\cdot)$
⟨*proof*⟩

lift-definition *sup-d-element* :: $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* $\Rightarrow$ $'a$ *d-element* **is** $(+)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

instance *d-element* :: (*domain-semiring*) *distrib-lattice*
⟨*proof*⟩

2.6 Domain Semirings with a Greatest Element

If there is a greatest element in the semiring, then we have another equality.

class *domain-semiring-top* = *domain-semiring* + *order-top*

begin

notation *top* $(\top)$

lemma *kat-equivalence-greatest*: $d\ x \leq d\ y \longleftrightarrow x \leq d\ y \cdot \top$
⟨*proof*⟩

end

2.7 Forward Diamond Operators

context *domain-semiringl*

begin

We define a forward diamond operator over a domain semiring. A more modular consideration is not given at the moment.

definition $fd :: 'a \Rightarrow 'a \Rightarrow 'a$ ((|-⟩ -) [61,81] 82) **where**
$|x\rangle\ y = d\ (x \cdot y)$

lemma *fdia-d-simp* [*simp*]: $|x\rangle\ d\ y = |x\rangle\ y$
⟨*proof*⟩

lemma *fdia-dom* [*simp*]: $|x\rangle\ 1 = d\ x$
⟨*proof*⟩

lemma *fdia-add1*: $|x\rangle\ (y + z) = |x\rangle\ y + |x\rangle\ z$
⟨*proof*⟩

lemma *fdia-add2*: $|x + y\rangle\ z = |x\rangle\ z + |y\rangle\ z$
⟨*proof*⟩

lemma *fdia-mult*: $|x \cdot y\rangle\ z = |x\rangle\ |y\rangle\ z$
⟨*proof*⟩

lemma *fdia-one* [*simp*]: $|1\rangle\ x = d\ x$
⟨*proof*⟩

lemma *fdemodalisation1*: $d\ z \cdot |x\rangle\ y = 0 \longleftrightarrow d\ z \cdot x \cdot d\ y = 0$
⟨*proof*⟩

lemma *fdemodalisation2*: $|x\rangle\ y \leq d\ z \longleftrightarrow x \cdot d\ y \leq d\ z \cdot x$
⟨*proof*⟩

lemma *fd-iso1*: $d\ x \leq d\ y \Longrightarrow |z\rangle\ x \leq |z\rangle\ y$
⟨*proof*⟩

lemma *fd-iso2*: $x \leq y \Longrightarrow |x\rangle\ z \leq |y\rangle\ z$
⟨*proof*⟩

lemma *fd-zero-var* [*simp*]: $|0\rangle\ x = 0$
⟨*proof*⟩

lemma *fd-subdist-1*: $|x\rangle\ y \leq |x\rangle\ (y + z)$
⟨*proof*⟩

lemma *fd-subdist-2*: $|x\rangle\ (d\ y \cdot d\ z) \leq |x\rangle\ y$
⟨*proof*⟩

lemma *fd-subdist*: $|x\rangle\ (d\ y \cdot d\ z) \leq |x\rangle\ y \cdot |x\rangle\ z$
⟨*proof*⟩

lemma *fdia-export-1*: $d\ y \cdot |x\rangle\ z = |d\ y \cdot x\rangle\ z$
⟨*proof*⟩

end

context *domain-semiring*

begin

lemma *fdia-zero* [*simp*]: $|x\rangle\ 0 = 0$
⟨*proof*⟩

end

2.8 Domain Kleene Algebras

We add the Kleene star to our considerations. Special domain axioms are not needed.

class *domain-left-kleene-algebra* = *left-kleene-algebra-zerol* + *domain-semiringl*

begin

lemma *dom-star* [*simp*]: $d\ (x^\star) = 1$
⟨*proof*⟩

lemma *fdia-star-unfold* [*simp*]: $|1\rangle\ y + |x\rangle\ |x^\star\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *fdia-star-unfoldr* [*simp*]: $|1\rangle\ y + |x^\star\rangle\ |x\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *fdia-star-unfold-var* [*simp*]: $d\ y + |x\rangle\ |x^\star\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *fdia-star-unfoldr-var* [*simp*]: $d\ y + |x^\star\rangle\ |x\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *fdia-star-induct-var*: $|x\rangle\ y \leq d\ y \Longrightarrow |x^\star\rangle\ y \leq d\ y$
⟨*proof*⟩

lemma *fdia-star-induct*: $d\ z + |x\rangle\ y \leq d\ y \Longrightarrow |x^\star\rangle\ z \leq d\ y$
⟨*proof*⟩

lemma *fdia-star-induct-eq*: $d\ z + |x\rangle\ y = d\ y \Longrightarrow |x^\star\rangle\ z \leq d\ y$
⟨*proof*⟩

end

class *domain-kleene-algebra* = *kleene-algebra* + *domain-semiring*

begin

subclass *domain-left-kleene-algebra* ⟨*proof*⟩

end

end

3 Antidomain Semirings

theory *Antidomain-Semiring*
imports *Domain-Semiring*
begin

3.1 Antidomain Monoids

We axiomatise antidomain monoids, using the axioms of [3].

class *antidomain-op* =
  fixes *antidomain-op* :: *'a* ⇒ *'a* (*ad*)

class *antidomain-left-monoid* = *monoid-mult* + *antidomain-op* +
  assumes *am1* [*simp*]: *ad x* · *x* = *ad 1*
  and *am2*: *ad x* · *ad y* = *ad y* · *ad x*
  and *am3* [*simp*]: *ad* (*ad x*) · *x* = *x*
  and *am4* [*simp*]: *ad* (*x* · *y*) · *ad* (*x* · *ad y*) = *ad x*
  and *am5* [*simp*]: *ad* (*x* · *y*) · *x* · *ad y* = *ad* (*x* · *y*) · *x*

begin

no-notation *domain-op* (*d*)
no-notation *zero-class.zero* (*0*)

We define a zero element and operations of domain and addition.

definition *a-zero* :: *'a* (*0*) **where**
  0 = *ad 1*

definition *am-d* :: *'a* ⇒ *'a* (*d*) **where**
  d x = *ad* (*ad x*)

definition *am-add-op* :: *'a* ⇒ *'a* ⇒ *'a* (**infixl** ⊕ *65*) **where**
  x ⊕ *y* ≡ *ad* (*ad x* · *ad y*)

lemma *a-d-zero* [*simp*]: *ad x* · *d x* = *0*
  ⟨*proof*⟩

lemma *a-d-one* [*simp*]: $d\ x \oplus ad\ x = 1$
⟨*proof*⟩

lemma *n-annil* [*simp*]: $0 \cdot x = 0$
⟨*proof*⟩

lemma *a-mult-idem* [*simp*]: $ad\ x \cdot ad\ x = ad\ x$
⟨*proof*⟩

lemma *a-add-idem* [*simp*]: $ad\ x \oplus ad\ x = ad\ x$
⟨*proof*⟩

The next three axioms suffice to show that the domain elements form a Boolean algebra.

lemma *a-add-comm*: $x \oplus y = y \oplus x$
⟨*proof*⟩

lemma *a-add-assoc*: $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
⟨*proof*⟩

lemma *huntington* [*simp*]: $ad\ (x \oplus y) \oplus ad\ (x \oplus ad\ y) = ad\ x$
⟨*proof*⟩

lemma *a-absorb1* [*simp*]: $(ad\ x \oplus ad\ y) \cdot ad\ x = ad\ x$
⟨*proof*⟩

lemma *a-absorb2* [*simp*]: $ad\ x \oplus ad\ x \cdot ad\ y = ad\ x$
⟨*proof*⟩

The distributivity laws remain to be proved; our proofs follow those of Maddux [12].

lemma *prod-split* [*simp*]: $ad\ x \cdot ad\ y \oplus ad\ x \cdot d\ y = ad\ x$
⟨*proof*⟩

lemma *sum-split* [*simp*]: $(ad\ x \oplus ad\ y) \cdot (ad\ x \oplus d\ y) = ad\ x$
⟨*proof*⟩

lemma *a-comp-simp* [*simp*]: $(ad\ x \oplus ad\ y) \cdot d\ x = ad\ y \cdot d\ x$
⟨*proof*⟩

lemma *a-distrib1*: $ad\ x \cdot (ad\ y \oplus ad\ z) = ad\ x \cdot ad\ y \oplus ad\ x \cdot ad\ z$
⟨*proof*⟩

lemma *a-distrib2*: $ad\ x \oplus ad\ y \cdot ad\ z = (ad\ x \oplus ad\ y) \cdot (ad\ x \oplus ad\ z)$
⟨*proof*⟩

lemma *aa-loc* [*simp*]: $d\ (x \cdot d\ y) = d\ (x \cdot y)$
⟨*proof*⟩

lemma *a-loc* [*simp*]: $ad\ (x \cdot d\ y) = ad\ (x \cdot y)$
⟨*proof*⟩

lemma *d-a-export* [*simp*]: $d\ (ad\ x \cdot y) = ad\ x \cdot d\ y$
⟨*proof*⟩

Every antidomain monoid is a domain monoid.

sublocale *dm*: *domain-monoid am-d* $(\cdot)$ *1*
⟨*proof*⟩

lemma *ds-ord-iso1*: $x \sqsubseteq y \Longrightarrow z \cdot x \sqsubseteq z \cdot y$

⟨*proof*⟩

lemma *a-very-costrict*: $ad\ x = 1 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *a-weak-loc*: $x \cdot y = 0 \longleftrightarrow x \cdot d\ y = 0$
⟨*proof*⟩

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

lemma *a-d-mult-closure* [*simp*]: $d\ (ad\ x \cdot ad\ y) = ad\ x \cdot ad\ y$
⟨*proof*⟩

lemma *kat-3'*: $d\ x \cdot y \cdot ad\ z = 0 \Longrightarrow d\ x \cdot y = d\ x \cdot y \cdot d\ z$
⟨*proof*⟩

lemma *s4* [*simp*]: $ad\ x \cdot ad\ (ad\ x \cdot y) = ad\ x \cdot ad\ y$
⟨*proof*⟩

end

class *antidomain-monoid* = *antidomain-left-monoid* +
assumes *am6* [*simp*]: $x \cdot ad\ 1 = ad\ 1$

begin

lemma *kat-3-equiv*: $d\ x \cdot y \cdot ad\ z = 0 \longleftrightarrow d\ x \cdot y = d\ x \cdot y \cdot d\ z$
⟨*proof*⟩

no-notation *a-zero* (*0*)
no-notation *am-d* (*d*)

end

3.2 Antidomain Near-Semirings

We define antidomain near-semirings. We do not consider units separately. The axioms are taken from [6].

notation *zero-class.zero* (*0*)

class *antidomain-near-semiring* = *ab-near-semiring-one-zerol* + *antidomain-op* + *plus-ord* +
assumes *ans1* [*simp*]: $ad\ x \cdot x = 0$
and *ans2* [*simp*]: $ad\ (x \cdot y) + ad\ (x \cdot ad\ (ad\ y)) = ad\ (x \cdot ad\ (ad\ y))$
and *ans3* [*simp*]: $ad\ (ad\ x) + ad\ x = 1$
and *ans4* [*simp*]: $ad\ (x + y) = ad\ x \cdot ad\ y$

begin

definition *ans-d* :: $'a \Rightarrow 'a$ (*d*) **where**
$d\ x = ad\ (ad\ x)$

lemma *a-a-one* [*simp*]: $d\ 1 = 1$
⟨*proof*⟩

lemma *a-very-costrict'*: $ad\ x = 1 \longleftrightarrow x = 0$
⟨*proof*⟩

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

Every antidomain near-semiring is automatically a dioid, and therefore ordered.

subclass *near-dioid-one-zerol*
⟨*proof*⟩

lemma *d1-a* [*simp*]: $d\ x \cdot x = x$
⟨*proof*⟩

lemma *a-comm*: $ad\ x \cdot ad\ y = ad\ y \cdot ad\ x$
⟨*proof*⟩

lemma *a-subid*: $ad\ x \leq 1$
⟨*proof*⟩

lemma *a-subid-aux1*: $ad\ x \cdot y \leq y$
⟨*proof*⟩

lemma *a-subdist*: $ad\ (x + y) \leq ad\ x$
⟨*proof*⟩

lemma *a-antitone*: $x \leq y \Longrightarrow ad\ y \leq ad\ x$
⟨*proof*⟩

lemma *a-mul-d* [*simp*]: $ad\ x \cdot d\ x = 0$
⟨*proof*⟩

lemma *a-gla1*: $ad\ x \cdot y = 0 \Longrightarrow ad\ x \leq ad\ y$
⟨*proof*⟩

lemma *a-gla2*: $ad\ x \leq ad\ y \Longrightarrow ad\ x \cdot y = 0$
⟨*proof*⟩

lemma *a2-eq* [*simp*]: $ad\ (x \cdot d\ y) = ad\ (x \cdot y)$
⟨*proof*⟩

lemma *a-export'* [*simp*]: $ad\ (ad\ x \cdot y) = d\ x + ad\ y$
⟨*proof*⟩

Every antidomain near-semiring is a domain near-semiring.

sublocale *dnsz*: *domain-near-semiring-one-zerol* (+) (·) *1 0 ans-d* (≤) (<)
⟨*proof*⟩

lemma *a-idem* [*simp*]: $ad\ x \cdot ad\ x = ad\ x$
⟨*proof*⟩

lemma *a-3-var* [*simp*]: $ad\ x \cdot ad\ y \cdot (x + y) = 0$
⟨*proof*⟩

lemma *a-3* [*simp*]: $ad\ x \cdot ad\ y \cdot d\ (x + y) = 0$
⟨*proof*⟩

lemma *a-closure'* [*simp*]: $d\ (ad\ x) = ad\ x$
⟨*proof*⟩

The following counterexamples show that some of the antidomain monoid axioms do not need to hold.

lemma $x \cdot ad\ 1 = ad\ 1$

⟨*proof*⟩

lemma $ad\ (x \cdot y) \cdot ad\ (x \cdot ad\ y) = ad\ x$

⟨*proof*⟩

lemma $ad\ (x \cdot y) \cdot ad\ (x \cdot ad\ y) = ad\ x$

⟨*proof*⟩

lemma *phl-seq-inv*: $d\ v \cdot x \cdot y \cdot ad\ w = 0 \Longrightarrow \exists z.\ d\ v \cdot x \cdot d\ z = 0 \land ad\ z \cdot y \cdot ad\ w = 0$
⟨*proof*⟩

lemma *a-fixpoint*: $ad\ x = x \Longrightarrow (\forall y.\ y = 0)$
⟨*proof*⟩

no-notation *ans-d* (d)

end

3.3 Antidomain Pre-Dioids

Antidomain pre-diods are based on a different set of axioms, which are again taken from [6].

class *antidomain-pre-dioid* = *pre-dioid-one-zerol* + *antidomain-op* +
assumes *apd1* [*simp*]: $ad\ x \cdot x = 0$
and *apd2* [*simp*]: $ad\ (x \cdot y) \leq ad\ (x \cdot ad\ (ad\ y))$
and *apd3* [*simp*]: $ad\ (ad\ x) + ad\ x = 1$

begin

definition *apd-d* :: $'a \Rightarrow\ 'a$ (d) **where**
$d\ x = ad\ (ad\ x)$

lemma *a-very-costrict''*: $ad\ x = 1 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *a-subid'*: $ad\ x \leq 1$
⟨*proof*⟩

lemma *d1-a'* [*simp*]: $d\ x \cdot x = x$
⟨*proof*⟩

lemma *a-subid-aux1'*: $ad\ x \cdot y \leq y$
⟨*proof*⟩

lemma *a-mul-d'* [*simp*]: $ad\ x \cdot d\ x = 0$
⟨*proof*⟩

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

lemma *meet-ord-def*: $ad\ x \leq ad\ y \longleftrightarrow ad\ x \cdot ad\ y = ad\ x$
⟨*proof*⟩

lemma *d-weak-loc*: $x \cdot y = 0 \longleftrightarrow x \cdot d\ y = 0$
⟨*proof*⟩

lemma *gla-1*: $ad\ x \cdot y = 0 \Longrightarrow ad\ x \leq ad\ y$
⟨*proof*⟩

lemma *a2-eq′* [*simp*]: $ad\ (x \cdot d\ y) = ad\ (x \cdot y)$
⟨*proof*⟩

lemma *a-supdist-var*: $ad\ (x + y) \leq ad\ x$
⟨*proof*⟩

lemma *a-antitone′*: $x \leq y \Longrightarrow ad\ y \leq ad\ x$
⟨*proof*⟩

lemma *a-comm-var*: $ad\ x \cdot ad\ y \leq ad\ y \cdot ad\ x$
⟨*proof*⟩

lemma *a-comm′*: $ad\ x \cdot ad\ y = ad\ y \cdot ad\ x$
⟨*proof*⟩

lemma *a-closed* [*simp*]: $d\ (ad\ x \cdot ad\ y) = ad\ x \cdot ad\ y$
⟨*proof*⟩

lemma *a-export″* [*simp*]: $ad\ (ad\ x \cdot y) = d\ x + ad\ y$
⟨*proof*⟩

lemma *d1-sum-var*: $x + y \leq (d\ x + d\ y) \cdot (x + y)$
⟨*proof*⟩

lemma *a4′*: $ad\ (x + y) = ad\ x \cdot ad\ y$
⟨*proof*⟩

Antidomain pre-dioids are domain pre-dioids and antidomain near-semirings, but still not antidomain monoids.

sublocale *dpdz*: *domain-pre-dioid-one-zerol* $(+)\ (\cdot)\ 1\ 0\ (\leq)\ (<)\ \lambda x.\ ad\ (ad\ x)$
⟨*proof*⟩

subclass *antidomain-near-semiring*
⟨*proof*⟩

lemma *a-supdist*: $ad\ (x + y) \leq ad\ x + ad\ y$
⟨*proof*⟩

lemma *a-gla*: $ad\ x \cdot y = 0 \longleftrightarrow ad\ x \leq ad\ y$
⟨*proof*⟩

lemma *a-subid-aux2*: $x \cdot ad\ y \leq x$
⟨*proof*⟩

lemma *a42-var*: $d\ x \cdot d\ y \leq ad\ (ad\ x + ad\ y)$
⟨*proof*⟩

lemma *d1-weak* [*simp*]: $(d\ x + d\ y) \cdot x = x$
⟨*proof*⟩

lemma $x \cdot ad\ 1 = ad\ 1$

⟨*proof*⟩

lemma $ad\ x \cdot (y + z) = ad\ x \cdot y + ad\ x \cdot z$

⟨*proof*⟩

lemma $ad\ (x \cdot y) \cdot ad\ (x \cdot ad\ y) = ad\ x$

⟨*proof*⟩

lemma $ad\ (x \cdot y) \cdot ad\ (x \cdot ad\ y) = ad\ x$

⟨*proof*⟩

no-notation *apd-d* (*d*)

end

3.4 Antidomain Semirings

Antidomain semirings are direct expansions of antidomain pre-dioids, but do not require idempotency of addition. Hence we give a slightly different axiomatisation, following [7].

class *antidomain-semiringl* = *semiring-one-zerol* + *plus-ord* + *antidomain-op* +
  assumes *as1* [*simp*]: $ad\ x \cdot x = 0$
  and *as2* [*simp*]: $ad\ (x \cdot y) + ad\ (x \cdot ad\ (ad\ y)) = ad\ (x \cdot ad\ (ad\ y))$
  and *as3* [*simp*]: $ad\ (ad\ x) + ad\ x = 1$

begin

definition *ads-d* :: $'a \Rightarrow 'a$ (*d*) **where**
  $d\ x = ad\ (ad\ x)$

lemma *one-idem'*: $1 + 1 = 1$
  ⟨*proof*⟩

Every antidomain semiring is a dioid and an antidomain pre-dioid.

subclass *dioid*
  ⟨*proof*⟩

subclass *antidomain-pre-dioid*
  ⟨*proof*⟩

lemma *am5-lem* [*simp*]: $ad\ (x \cdot y) \cdot ad\ (x \cdot ad\ y) = ad\ x$
⟨*proof*⟩

lemma *am6-lem* [*simp*]: $ad\ (x \cdot y) \cdot x \cdot ad\ y = ad\ (x \cdot y) \cdot x$
⟨*proof*⟩

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

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

subclass *antidomain-left-monoid*
⟨*proof*⟩

Every antidomain left semiring is a domain left semiring.

no-notation *domain-semiringl-class.fd* ((|-⟩ -) [*61*,*81*] *82*)

definition *fdia* :: $'a \Rightarrow 'a \Rightarrow 'a$ ((|-⟩ -) [*61*,*81*] *82*) **where**
$|x\rangle\ y = ad\ (ad\ (x \cdot y))$

sublocale *ds*: *domain-semiringl* $(+)$ $(\cdot)$ *1* *0* $\lambda x.\ ad\ (ad\ x)$ $(\le)$ $(<)$
rewrites *ds.fd* $x\ y \equiv$ *fdia* $x\ y$
⟨*proof*⟩

lemma *fd-eq-fdia* [*simp*]: *domain-semiringl.fd* $(\cdot)$ $d\ x\ y \equiv$ *fdia* $x\ y$
⟨*proof*⟩

end

class *antidomain-semiring* = *antidomain-semiringl* + *semiring-one-zero*

begin

Every antidomain semiring is an antidomain monoid.

subclass *antidomain-monoid*
⟨*proof*⟩

lemma *a-zero* = *0*
⟨*proof*⟩

sublocale *ds*: *domain-semiring* $(+)$ $(\cdot)$ *1* *0* $\lambda x.\ ad\ (ad\ x)$ $(\le)$ $(<)$
rewrites *ds.fd* $x\ y \equiv$ *fdia* $x\ y$
⟨*proof*⟩

end

3.5 The Boolean Algebra of Domain Elements

typedef (**overloaded**) $'a$ *a2-element* = $\{x :: 'a ::$ *antidomain-semiring*. $x = d\ x\}$
⟨*proof*⟩

setup-lifting *type-definition-a2-element*

instantiation *a2-element* :: (*antidomain-semiring*) *boolean-algebra*

begin

lift-definition *less-eq-a2-element* :: *'a a2-element* ⇒ *'a a2-element* ⇒ *bool* **is** (≤)
⟨*proof*⟩

lift-definition *less-a2-element* :: *'a a2-element* ⇒ *'a a2-element* ⇒ *bool* **is** (<)
⟨*proof*⟩

lift-definition *bot-a2-element* :: *'a a2-element* **is** *0*
⟨*proof*⟩

lift-definition *top-a2-element* :: *'a a2-element* **is** *1*
⟨*proof*⟩

lift-definition *inf-a2-element* :: *'a a2-element* ⇒ *'a a2-element* ⇒ *'a a2-element* **is** (·)
⟨*proof*⟩

lift-definition *sup-a2-element* :: *'a a2-element* ⇒ *'a a2-element* ⇒ *'a a2-element* **is** (+)
⟨*proof*⟩

lift-definition *minus-a2-element* :: *'a a2-element* ⇒ *'a a2-element* ⇒ *'a a2-element* **is** *λx y. x · ad y*
⟨*proof*⟩

lift-definition *uminus-a2-element* :: *'a a2-element* ⇒ *'a a2-element* **is** *antidomain-op*
⟨*proof*⟩

instance
⟨*proof*⟩

end

3.6 Further Properties

context *antidomain-semiringl*

begin

lemma *a-2-var*: *ad x · d y = 0* ⟷ *ad x* ≤ *ad y*
⟨*proof*⟩

The following two lemmas give the Galois connection of Heyting algebras.

lemma *da-shunt1*: $x \le d\ y + z \Longrightarrow x \cdot ad\ y \le z$
⟨*proof*⟩

lemma *da-shunt2*: $x \le ad\ y + z \Longrightarrow x \cdot d\ y \le z$
⟨*proof*⟩

lemma *d-a-galois1*: $d\ x \cdot ad\ y \le d\ z \longleftrightarrow d\ x \le d\ z + d\ y$
⟨*proof*⟩

lemma *d-a-galois2*: $d\ x \cdot d\ y \le d\ z \longleftrightarrow d\ x \le d\ z + ad\ y$
⟨*proof*⟩

lemma *d-cancellation-1*: $d\ x \le d\ y + d\ x \cdot ad\ y$
⟨*proof*⟩

lemma *d-cancellation-2*: $(d\ z + d\ y) \cdot ad\ y \le d\ z$
⟨*proof*⟩

lemma *a-de-morgan*: $ad\ (ad\ x \cdot ad\ y) = d\ (x + y)$
⟨*proof*⟩

lemma *a-de-morgan-var-3*: $ad\ (d\ x + d\ y) = ad\ x \cdot ad\ y$
⟨*proof*⟩

lemma *a-de-morgan-var-4*: $ad\ (d\ x \cdot d\ y) = ad\ x + ad\ y$
⟨*proof*⟩

lemma *a-4*: $ad\ x \le ad\ (x \cdot y)$
⟨*proof*⟩

lemma *a-6*: $ad\ (d\ x \cdot y) = ad\ x + ad\ y$
⟨*proof*⟩

lemma *a-7*: $d\ x \cdot ad\ (d\ y + d\ z) = d\ x \cdot ad\ y \cdot ad\ z$
⟨*proof*⟩

lemma *a-d-add-closure* [*simp*]: $d\ (ad\ x + ad\ y) = ad\ x + ad\ y$
⟨*proof*⟩

lemma *d-6* [*simp*]: $d\ x + ad\ x \cdot d\ y = d\ x + d\ y$
⟨*proof*⟩

lemma *d-7* [*simp*]: $ad\ x + d\ x \cdot ad\ y = ad\ x + ad\ y$
⟨*proof*⟩

lemma *a-mult-add*: $ad\ x \cdot (y + x) = ad\ x \cdot y$
⟨*proof*⟩

lemma *kat-2*: $y \cdot ad\ z \le ad\ x \cdot y \Longrightarrow d\ x \cdot y \cdot ad\ z = 0$

⟨*proof*⟩

lemma *kat-3*: $d\ x \cdot y \cdot ad\ z = 0 \Longrightarrow d\ x \cdot y = d\ x \cdot y \cdot d\ z$
⟨*proof*⟩

lemma *kat-4*: $d\ x \cdot y = d\ x \cdot y \cdot d\ z \Longrightarrow d\ x \cdot y \leq y \cdot d\ z$
⟨*proof*⟩

lemma *kat-2-equiv*: $y \cdot ad\ z \leq ad\ x \cdot y \longleftrightarrow d\ x \cdot y \cdot ad\ z = 0$
⟨*proof*⟩

lemma *kat-4-equiv*: $d\ x \cdot y = d\ x \cdot y \cdot d\ z \longleftrightarrow d\ x \cdot y \leq y \cdot d\ z$
⟨*proof*⟩

lemma *kat-3-equiv-opp*: $ad\ z \cdot y \cdot d\ x = 0 \longleftrightarrow y \cdot d\ x = d\ z \cdot y \cdot d\ x$
⟨*proof*⟩

lemma *kat-4-equiv-opp*: $y \cdot d\ x = d\ z \cdot y \cdot d\ x \longleftrightarrow y \cdot d\ x \leq d\ z \cdot y$
⟨*proof*⟩

3.7 Forward Box and Diamond Operators

lemma *fdemodalisation22*: $|x\rangle\ y \leq d\ z \longleftrightarrow ad\ z \cdot x \cdot d\ y = 0$
⟨*proof*⟩

lemma *dia-diff-var*: $|x\rangle\ y \leq |x\rangle\ (d\ y \cdot ad\ z) + |x\rangle\ z$
⟨*proof*⟩

lemma *dia-diff*: $|x\rangle\ y \cdot ad\ (\ |x\rangle\ z\ ) \leq |x\rangle\ (d\ y \cdot ad\ z)$
⟨*proof*⟩

lemma *fdia-export-2*: $ad\ y \cdot |x\rangle\ z = |ad\ y \cdot x\rangle\ z$
⟨*proof*⟩

lemma *fdia-split*: $|x\rangle\ y = d\ z \cdot |x\rangle\ y + ad\ z \cdot |x\rangle\ y$
⟨*proof*⟩

definition *fbox* :: $'a \Rightarrow 'a \Rightarrow 'a$ ((|-] -) [*61*,*81*] *82*) **where**
$|x]\ y = ad\ (x \cdot ad\ y)$

The next lemmas establish the De Morgan duality between boxes and diamonds.

lemma *fdia-fbox-de-morgan-2*: $ad\ (\ |x\rangle\ y) = |x]\ ad\ y$
⟨*proof*⟩

lemma *fbox-simp*: $|x]\ y = |x]\ d\ y$
⟨*proof*⟩

lemma *fbox-dom* [*simp*]: $|x]\ 0 = ad\ x$

⟨*proof*⟩

lemma *fbox-add1*: $|x]\ (d\ y \cdot d\ z) = |x]\ y \cdot |x]\ z$
⟨*proof*⟩

lemma *fbox-add2*: $|x + y]\ z = |x]\ z \cdot |y]\ z$
⟨*proof*⟩

lemma *fbox-mult*: $|x \cdot y]\ z = |x]\ |y]\ z$
⟨*proof*⟩

lemma *fbox-zero* [*simp*]: $|0]\ x = 1$
⟨*proof*⟩

lemma *fbox-one* [*simp*]: $|1]\ x = d\ x$
⟨*proof*⟩

lemma *fbox-iso*: $d\ x \leq d\ y \Longrightarrow |z]\ x \leq |z]\ y$
⟨*proof*⟩

lemma *fbox-antitone-var*: $x \leq y \Longrightarrow |y]\ z \leq |x]\ z$
⟨*proof*⟩

lemma *fbox-subdist-1*: $|x]\ (d\ y \cdot d\ z) \leq |x]\ y$
⟨*proof*⟩

lemma *fbox-subdist-2*: $|x]\ y \leq |x]\ (d\ y + d\ z)$
⟨*proof*⟩

lemma *fbox-subdist*: $|x]\ d\ y + |x]\ d\ z \leq |x]\ (d\ y + d\ z)$
⟨*proof*⟩

lemma *fbox-diff-var*: $|x]\ (d\ y + ad\ z) \cdot |x]\ z \leq |x]\ y$
⟨*proof*⟩

lemma *fbox-diff*: $|x]\ (d\ y + ad\ z) \leq |x]\ y + ad\ (\ |x]\ z\ )$
⟨*proof*⟩

end

context *antidomain-semiring*

begin

lemma *kat-1*: $d\ x \cdot y \leq y \cdot d\ z \Longrightarrow y \cdot ad\ z \leq ad\ x \cdot y$
⟨*proof*⟩

lemma *kat-1-equiv*: $d\ x \cdot y \leq y \cdot d\ z \longleftrightarrow y \cdot ad\ z \leq ad\ x \cdot y$
⟨*proof*⟩

lemma *kat-3-equiv'*: $d\ x \cdot y \cdot ad\ z = 0 \longleftrightarrow d\ x \cdot y = d\ x \cdot y \cdot d\ z$
⟨*proof*⟩

lemma *kat-1-equiv-opp*: $y \cdot d\ x \leq d\ z \cdot y \longleftrightarrow ad\ z \cdot y \leq y \cdot ad\ x$
⟨*proof*⟩

lemma *kat-2-equiv-opp*: $ad\ z \cdot y \leq y \cdot ad\ x \longleftrightarrow ad\ z \cdot y \cdot d\ x = 0$
⟨*proof*⟩

lemma *fbox-one-1* [*simp*]: $|x]\ 1 = 1$
⟨*proof*⟩

lemma *fbox-demodalisation3*: $d\ y \leq |x]\ d\ z \longleftrightarrow d\ y \cdot x \leq x \cdot d\ z$
⟨*proof*⟩

end

3.8 Antidomain Kleene Algebras

class *antidomain-left-kleene-algebra* = *antidomain-semiringl* + *left-kleene-algebra-zerol*

begin

sublocale *dka*: *domain-left-kleene-algebra* (+) (·) *1 0 d* (≤) (<) *star*
rewrites *domain-semiringl.fd* (·) $d\ x\ y \equiv |x\rangle\ y$
⟨*proof*⟩

lemma *a-star* [*simp*]: $ad\ (x^\star) = 0$
⟨*proof*⟩

lemma *fbox-star-unfold* [*simp*]: $|1]\ z \cdot |x]\ |x^\star]\ z = |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-star-unfold-var* [*simp*]: $d\ z \cdot |x]\ |x^\star]\ z = |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-star-unfoldr* [*simp*]: $|1]\ z \cdot |x^\star]\ |x]\ z = |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-star-unfoldr-var* [*simp*]: $d\ z \cdot |x^\star]\ |x]\ z = |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-star-induct-var*: $d\ y \leq |x]\ y \Longrightarrow d\ y \leq |x^\star]\ y$
⟨*proof*⟩

lemma *fbox-star-induct*: $d\ y \leq d\ z \cdot |x]\ y \Longrightarrow d\ y \leq |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-star-induct-eq*: $d\ z \cdot |x]\ y = d\ y \Longrightarrow d\ y \leq |x^\star]\ z$
⟨*proof*⟩

lemma *fbox-export-1*: $ad\ y + |x]\ y = |d\ y \cdot x]\ y$
⟨*proof*⟩

lemma *fbox-export-2*: $d\ y + |x]\ y = |ad\ y \cdot x]\ y$
⟨*proof*⟩

end

class *antidomain-kleene-algebra* = *antidomain-semiring* + *kleene-algebra*

begin

subclass *antidomain-left-kleene-algebra* ⟨*proof*⟩

lemma $d\ p \leq |(d\ t \cdot x)^\star \cdot ad\ t]\ (d\ q \cdot ad\ t) \Longrightarrow d\ p \leq |d\ t \cdot x]\ d\ q$

⟨*proof*⟩

end

end

4 Range and Antirange Semirings

theory *Range-Semiring*
imports *Antidomain-Semiring*
begin

4.1 Range Semirings

We set up the duality between domain and antidomain semirings on the one hand and range and antirange semirings on the other hand.

class *range-op* =
fixes *range-op* :: $'a \Rightarrow 'a\ (r)$

class *range-semiring* = *semiring-one-zero* + *plus-ord* + *range-op* +
assumes *rsr1* [*simp*]: $x + (x \cdot r\ x) = x \cdot r\ x$
and *rsr2* [*simp*]: $r\ (r\ x \cdot y) = r(x \cdot y)$
and *rsr3* [*simp*]: $r\ x + 1 = 1$
and *rsr4* [*simp*]: $r\ 0 = 0$
and *rsr5* [*simp*]: $r\ (x + y) = r\ x + r\ y$

begin

definition *bd* :: $'a \Rightarrow 'a \Rightarrow 'a$ (⟨-| - [*61*,*81*] *82*) **where**

$\langle x|\ y = r\ (y \cdot x)$

no-notation *range-op* (r)

end

sublocale *range-semiring* ⊆ *rdual*: *domain-semiring* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0 range-op* $(\leq)$ $(<)$
  rewrites *rdual.fd* $x\ y = \langle x|\ y$
⟨*proof*⟩

sublocale *domain-semiring* ⊆ *ddual*: *range-semiring* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0 domain-op* $(\leq)$ $(<)$
  rewrites *ddual.bd* $x\ y =$ *domain-semiringl-class.fd* $x\ y$
⟨*proof*⟩

4.2 Antirange Semirings

class *antirange-op* =
  fixes *antirange-op* :: $'a \Rightarrow 'a$ (*ar* - [*999*] *1000*)

class *antirange-semiring* = *semiring-one-zero* + *plus-ord* + *antirange-op* +
  assumes *ars1* [*simp*]: $x \cdot ar\ x = 0$
  and *ars2* [*simp*]: $ar\ (x \cdot y) + ar\ (ar\ ar\ x \cdot y) = ar\ (ar\ ar\ x \cdot y)$
  and *ars3* [*simp*]: $ar\ ar\ x + ar\ x = 1$

begin

no-notation *bd* (⟨-| - [*61*,*81*] *82*)

definition *ars-r* :: $'a \Rightarrow 'a$ (r) **where**
  $r\ x = ar\ (ar\ x)$

definition *bdia* :: $'a \Rightarrow 'a \Rightarrow 'a$ (⟨-| - [*61*,*81*] *82*) **where**
  $\langle x|\ y = ar\ ar\ (y \cdot x)$

definition *bbox* :: $'a \Rightarrow 'a \Rightarrow 'a$ ([-| - [*61*,*81*] *82*) **where**
  $[x|\ y = ar\ (ar\ y \cdot x)$

end

sublocale *antirange-semiring* ⊆ *ardual*: *antidomain-semiring antirange-op* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0* $(\leq)$ $(<)$
  rewrites *ardual.ads-d* $x = r\ x$
  and *ardual.fdia* $x\ y = \langle x|\ y$
  and *ardual.fbox* $x\ y = [x|\ y$
⟨*proof*⟩

context *antirange-semiring*

begin

sublocale *rs*: *range-semiring* $(+)$ $(\cdot)$ *1 0* $\lambda x.$ *ar ar x* $(\leq)$ $(<)$
⟨*proof*⟩

end

sublocale *antidomain-semiring* ⊆ *addual*: *antirange-semiring* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0* *antidomain-op* $(\leq)$ $(<)$
rewrites *addual.ars-r* $x = d\ x$
and *addual.bdia* $x\ y = |x\rangle\ y$
and *addual.bbox* $x\ y = |x]\ y$
⟨*proof*⟩

4.3 Antirange Kleene Algebras

class *antirange-kleene-algebra* = *antirange-semiring* + *kleene-algebra*

sublocale *antirange-kleene-algebra* ⊆ *dual*: *antidomain-kleene-algebra antirange-op* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0* $(\leq)$ $(<)$ *star*
⟨*proof*⟩

sublocale *antidomain-kleene-algebra* ⊆ *dual*: *antirange-kleene-algebra* $(+)$ $\lambda x\ y.\ y \cdot x$ *1 0* $(\leq)$ $(<)$ *star antidomain-op*
⟨*proof*⟩

Hence all range theorems have been derived by duality in a generic way.

end

5 Modal Kleene Algebras

This section studies laws that relate antidomain and antirange semirings and Kleene algebra, notably Galois connections and conjugations as those studied in [13, 7].

theory *Modal-Kleene-Algebra*
imports *Range-Semiring*
begin

class *modal-semiring* = *antidomain-semiring* + *antirange-semiring* +
assumes *domrange* [*simp*]: $d\ (r\ x) = r\ x$
and *rangedom* [*simp*]: $r\ (d\ x) = d\ x$

begin

These axioms force that the domain algebra and the range algebra coincide.

lemma *domrangefix*: $d\ x = x \longleftrightarrow r\ x = x$

⟨*proof*⟩

lemma *box-diamond-galois-1*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $\langle x|\ p \leq q \longleftrightarrow p \leq |x]\ q$
⟨*proof*⟩

lemma *box-diamond-galois-2*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $|x\rangle\ p \leq q \longleftrightarrow p \leq [x|\ q$
⟨*proof*⟩

lemma *diamond-conjugation-var-1*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $|x\rangle\ p \leq ad\ q \longleftrightarrow \langle x|\ q \leq ad\ p$
⟨*proof*⟩

lemma *diamond-conjungation-aux*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $\langle x|\ p \leq ad\ q \longleftrightarrow q \cdot \langle x|\ p = 0$
⟨*proof*⟩

lemma *diamond-conjugation*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $p \cdot |x\rangle\ q = 0 \longleftrightarrow q \cdot \langle x|\ p = 0$
⟨*proof*⟩

lemma *box-conjugation-var-1*:
assumes $d\ p = p$ **and** $d\ q = q$
shows $ad\ p \leq [x|\ q \longleftrightarrow ad\ q \leq |x]\ p$
⟨*proof*⟩

lemma *box-diamond-cancellation-1*: $d\ p = p \Longrightarrow p \leq |x]\ \langle x|\ p$
⟨*proof*⟩

lemma *box-diamond-cancellation-2*: $d\ p = p \Longrightarrow p \leq [x|\ |x\rangle\ p$
⟨*proof*⟩

lemma *box-diamond-cancellation-3*: $d\ p = p \Longrightarrow |x\rangle\ [x|\ p \leq p$
⟨*proof*⟩

lemma *box-diamond-cancellation-4*: $d\ p = p \Longrightarrow \langle x|\ |x]\ p \leq p$
⟨*proof*⟩

end

class *modal-kleene-algebra* = *modal-semiring* + *kleene-algebra*
begin

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

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

end

end

6 Models of Modal Kleene Algebras

theory *Modal-Kleene-Algebra-Models*
imports *Kleene-Algebra.Kleene-Algebra-Models*
Modal-Kleene-Algebra

begin

This section develops the relation model. We also briefly develop the trace model for antidomain Kleene algebras, but not for antirange or full modal Kleene algebras. The reason is that traces are implemented as lists; we therefore expect tedious inductive proofs in the presence of range. The language model is not particularly interesting.

definition *rel-ad* :: *'a rel* ⇒ *'a rel* **where**
rel-ad R = {(*x*,*x*) | *x*. ¬ (∃ *y*. (*x*,*y*) ∈ *R*)}

interpretation *rel-antidomain-kleene-algebra*: *antidomain-kleene-algebra rel-ad* (∪) (*O*) *Id* {} (⊆) (⊂) *rtrancl*
⟨*proof*⟩

definition *trace-a* :: (*'p*, *'a*) *trace set* ⇒ (*'p*, *'a*) *trace set* **where**
trace-a X = {(*p*,[]) | *p*. ¬ (∃ *x*. *x* ∈ *X* ∧ *p* = *first x*)}

interpretation *trace-antidomain-kleene-algebra*: *antidomain-kleene-algebra trace-a* (∪) *t-prod t-one t-zero* (⊆) (⊂) *t-star*
⟨*proof*⟩

The trace model should be extended to cover modal Kleene algebras in the future.

definition *rel-ar* :: *'a rel* ⇒ *'a rel* **where**
rel-ar R = {(*y*,*y*) | *y*. ¬ (∃ *x*. (*x*,*y*) ∈ *R*)}

interpretation *rel-antirange-kleene-algebra*: *antirange-semiring* (∪) (*O*) *Id* {} *rel-ar* (⊆) (⊂)
⟨*proof*⟩

interpretation *rel-modal-kleene-algebra*: *modal-kleene-algebra* (∪) (*O*) *Id* {} (⊆) (⊂) *rtrancl rel-ad rel-ar*
⟨*proof*⟩

end

7 Applications of Modal Kleene Algebras

theory *Modal-Kleene-Algebra-Applications*
imports *Antidomain-Semiring*
begin

This file collects some applications of the theories developed so far. These are described in [11].

context *antidomain-kleene-algebra*
begin

7.1 A Reachability Result

This example is taken from [4].

lemma *opti-iterate-var* [*simp*]: $|(ad\ y \cdot x)^\star\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *opti-iterate* [*simp*]: $d\ y + |(x \cdot ad\ y)^\star\rangle\ |x\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *opti-iterate-var-2* [*simp*]: $d\ y + |ad\ y \cdot x\rangle\ |x^\star\rangle\ y = |x^\star\rangle\ y$
⟨*proof*⟩

7.2 Derivation of Segerberg's Formula

This example is taken from [5].

definition *Alpha* :: $'a \Rightarrow 'a \Rightarrow 'a$ (A)
where $A\ x\ y = d\ (x \cdot y) \cdot ad\ y$

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

lemma *A-fdia*: $A\ x\ y = |x\rangle\ y \cdot ad\ y$
⟨*proof*⟩

lemma *A-fdia-var*: $A\ x\ y = |x\rangle\ d\ y \cdot ad\ y$
⟨*proof*⟩

lemma *a-A*: $ad\ (A\ x\ (ad\ y)) = |x]\ y + ad\ y$
⟨*proof*⟩

lemma *fsegerberg* [*simp*]: $d\ y + |x^\star\rangle\ A\ x\ y = |x^\star\rangle\ y$
⟨*proof*⟩

lemma *fbox-segerberg* [*simp*]: $d\ y \cdot |x^\star]\ (\ |x]\ y + ad\ y\ ) = |x^\star]\ y$
⟨*proof*⟩

7.3 Wellfoundedness and Loeb's Formula

This example is taken from [7].

definition *Omega* :: $'a \Rightarrow 'a \Rightarrow 'a$ (Ω)
where $\Omega\ x\ y = d\ y \cdot ad\ (x \cdot y)$

If y is a set, then $\Omega(x, y)$ describes those elements in y from which no further x transitions are possible.

lemma *omega-fdia*: $\Omega\ x\ y = d\ y \cdot ad\ (\ |x\rangle\ y\ )$
⟨*proof*⟩

lemma *omega-fbox*: $\Omega\ x\ y = d\ y \cdot |x]\ (ad\ y)$
⟨*proof*⟩

lemma *omega-absorb1* [*simp*]: $\Omega\ x\ y \cdot ad\ (\ |x\rangle\ y\ ) = \Omega\ x\ y$
⟨*proof*⟩

lemma *omega-absorb2* [*simp*]: $\Omega\ x\ y \cdot ad\ (x \cdot y) = \Omega\ x\ y$
⟨*proof*⟩

lemma *omega-le-1*: $\Omega\ x\ y \leq d\ y$
⟨*proof*⟩

lemma *omega-subid*: $\Omega\ x\ (d\ y) \leq d\ y$
⟨*proof*⟩

lemma *omega-le-2*: $\Omega\ x\ y \leq ad\ (\ |x\rangle\ y\ )$
⟨*proof*⟩

lemma *omega-dom* [*simp*]: $d\ (\Omega\ x\ y) = \Omega\ x\ y$
⟨*proof*⟩

lemma *a-omega*: $ad\ (\Omega\ x\ y) = ad\ y + |x\rangle\ y$
⟨*proof*⟩

lemma *omega-fdia-3* [*simp*]: $d\ y \cdot ad\ (\Omega\ x\ y) = d\ y \cdot |x\rangle\ y$
⟨*proof*⟩

lemma *omega-zero-equiv-1*: $\Omega\ x\ y = 0 \longleftrightarrow d\ y \leq |x\rangle\ y$
⟨*proof*⟩

definition *Loebian* :: $'a \Rightarrow bool$
where $Loebian\ x = (\forall\, y.\ |x\rangle\ y \leq |x\rangle\ \Omega\ x\ y)$

definition *PreLoebian* :: $'a \Rightarrow bool$
where $PreLoebian\ x = (\forall\, y.\ d\ y \leq |x^\star\rangle\ \Omega\ x\ y)$

definition *Noetherian* :: $'a \Rightarrow bool$
where $Noetherian\ x = (\forall\, y.\ \Omega\ x\ y = 0 \longrightarrow d\ y = 0)$

lemma *noetherian-alt*: *Noetherian x* $\longleftrightarrow$ $(\forall\, y.\ d\ y \leq |x\rangle\ y \longrightarrow d\ y = 0)$
⟨*proof*⟩

lemma *Noetherian-iff-PreLoebian*: *Noetherian x* $\longleftrightarrow$ *PreLoebian x*
⟨*proof*⟩

lemma *Loebian-imp-Noetherian*: *Loebian x* $\Longrightarrow$ *Noetherian x*
⟨*proof*⟩

lemma *d-transitive*: $(\forall\, y.\ |x\rangle\ |x\rangle\ y \leq |x\rangle\ y) \Longrightarrow (\forall\, y.\ |x\rangle\ y = |x^\star\rangle\ |x\rangle\ y)$
⟨*proof*⟩

lemma *d-transitive-var*: $(\forall\, y.\ |x\rangle\ |x\rangle\ y \leq |x\rangle\ y) \Longrightarrow (\forall\, y.\ |x\rangle\ y = |x\rangle\ |x^\star\rangle\ y)$
⟨*proof*⟩

lemma *d-transitive-PreLoebian-imp-Loebian*: $(\forall\, y.\ |x\rangle\ |x\rangle\ y \leq |x\rangle\ y) \Longrightarrow$ *PreLoebian x* $\Longrightarrow$ *Loebian x*
⟨*proof*⟩

lemma *d-transitive-Noetherian-iff-Loebian*: $\forall\, y.\ |x\rangle\ |x\rangle\ y \leq |x\rangle\ y \Longrightarrow$ *Noetherian x* $\longleftrightarrow$ *Loebian x*
⟨*proof*⟩

lemma *Loeb-iff-box-Loeb*: *Loebian x* $\longleftrightarrow$ $(\forall\, y.\ |x]\ (ad\ (\ |x]\ y\ ) + d\ y) \leq |x]\ y)$
⟨*proof*⟩

end

7.4 Divergence Kleene Algebras and Separation of Termination

The notion of divergence has been added to modal Kleene algebras in [5]. More facts about divergence could be added in the future. Some could be adapted from omega algebras.

class *nabla-op* =
fixes *nabla* :: $'a \Rightarrow 'a$ (∇- [*999*] *1000*)

class *fdivergence-kleene-algebra* = *antidomain-kleene-algebra* + *nabla-op* +
assumes *nabla-closure* [*simp*]: $d\ \nabla\ x = \nabla\ x$
and *nabla-unfold*: $\nabla\ x \leq |x\rangle\ \nabla\ x$
and *nabla-coinduction*: $d\ y \leq |x\rangle\ y + d\ z \Longrightarrow d\ y \leq \nabla\ x + |x^\star\rangle\ z$

begin

lemma *nabla-coinduction-var*: $d\ y \leq |x\rangle\ y \Longrightarrow d\ y \leq \nabla\ x$
⟨*proof*⟩

lemma *nabla-unfold-eq* [*simp*]: $|x\rangle\ \nabla\ x = \nabla\ x$
⟨*proof*⟩

lemma *nabla-subdist*: $\nabla\ x \leq \nabla\ (x + y)$
⟨*proof*⟩

lemma *nabla-iso*: $x \leq y \Longrightarrow \nabla\ x \leq \nabla\ y$
⟨*proof*⟩

lemma *nabla-omega*: $\Omega\ x\ (d\ y) = 0 \Longrightarrow d\ y \leq \nabla\ x$
⟨*proof*⟩

lemma *nabla-noether*: $\nabla\ x = 0 \Longrightarrow$ *Noetherian* x
⟨*proof*⟩

lemma *nabla-preloeb*: $\nabla\ x = 0 \Longrightarrow$ *PreLoebian* x
⟨*proof*⟩

lemma *star-nabla-1* [*simp*]: $|x^\star\rangle\ \nabla\ x = \nabla\ x$
⟨*proof*⟩

lemma *nabla-sum-expand* [*simp*]: $|x\rangle\ \nabla\ (x + y) + |y\rangle\ \nabla\ (x + y) = \nabla\ (x + y)$
⟨*proof*⟩

lemma *wagner-3*:
assumes $d\ z + |x\rangle\ \nabla\ (x + y) = \nabla\ (x + y)$
shows $\nabla\ (x + y) = \nabla\ x + |x^\star\rangle\ z$
⟨*proof*⟩

lemma *nabla-sum-unfold* [*simp*]: $\nabla\ x + |x^\star\rangle\ |y\rangle\ \nabla\ (x + y) = \nabla\ (x + y)$
⟨*proof*⟩

lemma *nabla-separation*: $y \cdot x \leq x \cdot (x + y)^\star \Longrightarrow (\nabla\ (x + y) = \nabla\ x + |x^\star\rangle\ \nabla\ y)$
⟨*proof*⟩

The next lemma is a separation of termination theorem by Bachmair and Dershowitz [2].

lemma *bachmair-dershowitz*: $y \cdot x \leq x \cdot (x + y)^\star \Longrightarrow \nabla\ x + \nabla\ y = 0 \longleftrightarrow \nabla\ (x + y) = 0$
⟨*proof*⟩

The next lemma is a more complex separation of termination theorem by Doornbos, Backhouse and van der Woude [8].

lemma *separation-of-termination*:
assumes $y \cdot x \leq x \cdot (x + y)^\star + y$
shows $\nabla\ x + \nabla\ y = 0 \longleftrightarrow \nabla\ (x + y) = 0$
⟨*proof*⟩

The final examples can be found in [11].

definition $T :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ (- $\rightsquigarrow$ - $\rightsquigarrow$ - [61,61,61] 60)
where $p{\rightsquigarrow}x{\rightsquigarrow}q \equiv ad\ p + |x]\ d\ q$

lemma *T-d* [*simp*]: $d\ (p{\rightsquigarrow}x{\rightsquigarrow}q) = p{\rightsquigarrow}x{\rightsquigarrow}q$
⟨*proof*⟩

lemma *T-p*: $d\ p \cdot (p{\rightsquigarrow}x{\rightsquigarrow}q) = d\ p \cdot |x]\ d\ q$
⟨*proof*⟩

lemma *T-a* [*simp*]: $ad\ p \cdot (p{\rightsquigarrow}x{\rightsquigarrow}q) = ad\ p$
⟨*proof*⟩

lemma *T-seq*: $(p{\rightsquigarrow}x{\rightsquigarrow}q) \cdot |x](q{\rightsquigarrow}y{\rightsquigarrow}s) \leq p{\rightsquigarrow}x \cdot y{\rightsquigarrow}s$
⟨*proof*⟩

lemma *T-square*: $(p{\rightsquigarrow}x{\rightsquigarrow}q) \cdot |x](q{\rightsquigarrow}x{\rightsquigarrow}p) \leq p{\rightsquigarrow}x{\cdot}x{\rightsquigarrow}p$
⟨*proof*⟩

lemma *T-segerberg* [*simp*]: $d\ p \cdot |x^\star](p{\rightsquigarrow}x{\rightsquigarrow}p) = |x^\star]\ d\ p$
⟨*proof*⟩

lemma *lookahead* [*simp*]: $|x^\star](d\ p \cdot |x]\ d\ p) = |x^\star]\ d\ p$
⟨*proof*⟩

lemma *alternation*: $d\ p \cdot |x^\star]((p{\rightsquigarrow}x{\rightsquigarrow}q) \cdot (q{\rightsquigarrow}x{\rightsquigarrow}p)) = |(x{\cdot}x)^\star](d\ p \cdot (q{\rightsquigarrow}x{\rightsquigarrow}p))$
$\cdot\ |x{\cdot}(x{\cdot}x)^\star](d\ q \cdot (p{\rightsquigarrow}x{\rightsquigarrow}q))$
⟨*proof*⟩

lemma $|(x{\cdot}x)^\star]\ d\ p \cdot |x{\cdot}(x{\cdot}x)^\star]\ ad\ p = d\ p \cdot |x^\star]((p{\rightsquigarrow}x{\rightsquigarrow}ad\ p) \cdot (ad\ p{\rightsquigarrow}x{\rightsquigarrow}p))$
⟨*proof*⟩

lemma $|x^\star]\ d\ p = d\ p \cdot |x^\star](p{\rightsquigarrow}x{\rightsquigarrow}p)$
⟨*proof*⟩

end

end

References

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

[2] L. Bachmair and N. Dershowitz. Commutation, transformation, and termination. In J. H. Siekmann, editor, *Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 5–20. Springer, 1986.

[3] J. Desharnais, P. Jipsen, and G. Struth. Domain and antidomain semigroups. In R. Berghammer, A. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2009.

[4] J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM TOCL*, 7(4):798–833, 2006.

[5] J. Desharnais, B. Möller, and G. Struth. Algebraic notions of termination. *Logical Methods in Computer Science*, 7(1), 2011.

[6] J. Desharnais and G. Struth. Domain axioms for a family of near-semirings. In J. Meseguer and G. Rosu, editors, *AMAST 2008*, volume 5140 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2008.

[7] J. Desharnais and G. Struth. Internal axioms for domain semirings. *Science of Computer Programming*, 76(3):181–203, 2011.

[8] H. Doornbos, R. C. Backhouse, and J. van der Woude. A calculational approach to mathematical induction. *Theoretical Computer Science*, 179(1–2):103–135, 1997.

[9] H. Furusawa and G. Struth. Binary multirelations. *Archive of Formal Proofs*, 2015.

[10] H. Furusawa and G. Struth. Concurrent dynamic algebra. *ACM TOCL*, 16(4):30, 2015.

[11] W. Guttmann, G. Struth, and T. Weber. Automating algebraic methods in Isabelle. In S. Qin and Z. Qiu, editors, *ICFEM 2011*, volume 6991 of *Lecture Notes in Computer Science*, pages 617–632. Springer, 2011.

[12] R. D. Maddux. *Relation Algebras*. Elsevier, 2006.

[13] B. Möller and G. Struth. Algebras of modal operators and partial correctness. *Theoretical Computer Science*, 351(2):221–239, 2006.