Kleene Algebra with Tests and Demonic Refinement Algebras

Alasdair Armstrong Victor B. F. Gomes Georg Struth

March 17, 2025

Abstract

We formalise Kleene algebra with tests (KAT) and demonic refinement algebra (DRA) with tests in Isabelle/HOL. KAT is relevant for program verification and correctness proofs in the partial correctness setting. DRA targets similar applications in the context of total correctness. Our formalisation contains the two most important models of these algebras: binary relations in the case of KAT and predicate transformers in the case of DRA. In addition, we derive the inference rules for Hoare logic in KAT and its relational model.

Contents

1 Test Dioids **2**
- 1.1 Test Monoids . . . 2
- 1.2 Test Near-Semirings . . . 6
- 1.3 Test Near Semirings with Distributive Tests . . . 12
- 1.4 Test Predioids . . . 13
- 1.5 Test Semirings . . . 15

2 Pre-Conway Algebra with Tests **16**

3 Kleene Algebra with Tests **17**

4 Demonic Refinement Algebra with Tests **18**

5 Models for Demonic Refinement Algebra with Tests **20**

6 Models for Kleene Algebra with Tests **23**

7 Transformation Theorem for while Loops **24**

8 Propositional Hoare Logic **26**

9 Propositional Hoare Logic **30**

10 Two sorted Kleene Algebra with Tests 30

11 Two sorted Demonic Refinement Algebras 33

1 Test Dioids

theory *Test-Dioid*
imports *Kleene-Algebra.Dioid*
begin

Tests are embedded in a weak dioid, a dioid without the right annihilation and left distributivity axioms, using an operator t defined by a complementation operator. This allows us to use tests in weak settings, such as Probabilistic Kleene Algebra and Demonic Refinement Algebra.

1.1 Test Monoids

class *n-op* =
fixes *n-op* :: *'a* ⇒ *'a* (‹*n* -› [*90*] *91*)

class *test-monoid* = *monoid-mult* + *n-op* +
assumes *tm1* [*simp*]: *n n 1* = *1*
and *tm2* [*simp*]: *n x* · *n n x* = *n 1*
and *tm3*: *n x* · *n* (*n n z* · *n n y*) = *n* (*n* (*n x* · *n y*) · *n* (*n x* · *n z*))

begin

definition *a-zero* :: *'a* (‹*o*›) **where**
o ≡ *n 1*

abbreviation *t x* ≡ *n n x*

definition *n-add-op* :: *'a* ⇒ *'a* ⇒ *'a* (**infixl** ‹⊕› *65*) **where**
x ⊕ *y* ≡ *n* (*n x* · *n y*)

lemma *n 1* · *x* = *n 1*
⟨*proof*⟩

lemma *x* · *n 1* = *n 1*
⟨*proof*⟩

lemma *n 1* · *x* = *n 1* ⟹ *n x* · *y* · *t z* = *n 1* ⟹ *n x* · *y* = *n x* · *y* · *n z*
⟨*proof*⟩

lemma *n-t-closed* [*simp*]: *t* (*n x*) = *n x*
⟨*proof*⟩

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

lemma *n-comm-var*: $n\ (n\ x \cdot n\ y) = n\ (n\ y \cdot n\ x)$
⟨*proof*⟩

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

lemma *huntington1* [*simp*]: $n\ (n\ (n\ n\ x \cdot n\ y) \cdot n\ (n\ n\ x \cdot n\ n\ y)) = n\ n\ x$
⟨*proof*⟩

lemma *huntington2* [*simp*]: $n\ (n\ x \oplus n\ n\ y) \oplus n\ (n\ x \oplus n\ y) = n\ n\ x$
⟨*proof*⟩

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

lemma *t-mult-closure*: $t\ x = x \Longrightarrow t\ y = y \Longrightarrow t\ (x \cdot y) = x \cdot y$
⟨*proof*⟩

lemma *n-t-compl* [*simp*]: $n\ x \oplus t\ x = 1$
⟨*proof*⟩

lemma *zero-least1* [*simp*]: $o \oplus n\ x = n\ x$
⟨*proof*⟩

lemma *zero-least2* [*simp*]: $o \cdot n\ x = o$
⟨*proof*⟩

lemma *zero-least3* [*simp*]: $n\ x \cdot o\ = o$
⟨*proof*⟩

lemma *one-greatest1* [*simp*]: $1 \oplus n\ x = 1$
⟨*proof*⟩

lemma *one-greatest2* [*simp*]: $n\ x \oplus 1 = 1$
⟨*proof*⟩

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

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

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

lemma *n-preserve2* [*simp*]: $n\ x \cdot n\ y \cdot t\ x = o$
⟨*proof*⟩

lemma *de-morgan1* [*simp*]: $n\ (n\ x \cdot n\ y) = t\ x \oplus t\ y$
⟨*proof*⟩

lemma *de-morgan4* [*simp*]: $n\ (t\ x \oplus t\ y) = n\ x \cdot n\ y$
⟨*proof*⟩

lemma *n-absorb1* [*simp*]: $n\ x \oplus n\ x \cdot n\ y = n\ x$
⟨*proof*⟩

lemma *n-absorb2* [*simp*]: $n\ x \cdot (n\ x \oplus n\ y) = n\ x$
⟨*proof*⟩

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

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

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

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

definition *ts-ord* :: $'a \Rightarrow 'a \Rightarrow bool$ (**infix** ‹$\sqsubseteq$› *50*) **where**
$x \sqsubseteq y = (n\ x \cdot n\ y = n\ y)$

lemma *ts-ord-alt*: $n\ x \sqsubseteq n\ y \longleftrightarrow n\ x \oplus n\ y = n\ y$
⟨*proof*⟩

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

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

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

lemma *ts-ord-mult-isol*: $n\ x \sqsubseteq n\ y \Longrightarrow n\ z \cdot n\ x \sqsubseteq n\ z \cdot n\ y$
⟨*proof*⟩

lemma *ts-ord-mult-isor*: $n\ x \sqsubseteq n\ y \Longrightarrow n\ x \cdot n\ z \sqsubseteq n\ y \cdot n\ z$
⟨*proof*⟩

lemma *ts-ord-add-isol*: $n\ x \sqsubseteq n\ y \Longrightarrow n\ z \oplus n\ x \sqsubseteq n\ z \oplus n\ y$

⟨*proof*⟩

lemma *ts-ord-add-isor*: $n\ x \sqsubseteq n\ y \Longrightarrow n\ x \oplus n\ z \sqsubseteq n\ y \oplus n\ z$
⟨*proof*⟩

lemma *ts-ord-anti*: $n\ x \sqsubseteq n\ y \Longrightarrow t\ y \sqsubseteq t\ x$
⟨*proof*⟩

lemma *ts-ord-anti-iff*: $n\ x \sqsubseteq n\ y \longleftrightarrow t\ y \sqsubseteq t\ x$
⟨*proof*⟩

lemma *zero-ts-ord*: $o \sqsubseteq n\ x$
⟨*proof*⟩

lemma *n-subid*: $n\ x \sqsubseteq 1$
⟨*proof*⟩

lemma *n-mult-lb1*: $n\ x \cdot n\ y \sqsubseteq n\ x$
⟨*proof*⟩

lemma *n-mult-lb2*: $n\ x \cdot n\ y \sqsubseteq n\ y$
⟨*proof*⟩

lemma *n-mult-glbI*: $n\ z \sqsubseteq n\ x \Longrightarrow\ n\ z \sqsubseteq n\ y \Longrightarrow n\ z \sqsubseteq n\ x \cdot n\ y$
⟨*proof*⟩

lemma *n-mult-glb*: $n\ z \sqsubseteq n\ x \wedge n\ z \sqsubseteq n\ y \longleftrightarrow n\ z \sqsubseteq n\ x \cdot n\ y$
⟨*proof*⟩

lemma *n-add-ub1*: $n\ x \sqsubseteq n\ x \oplus n\ y$
⟨*proof*⟩

lemma *n-add-ub2*: $n\ y \sqsubseteq n\ x \oplus n\ y$
⟨*proof*⟩

lemma *n-add-lubI*: $n\ x \sqsubseteq n\ z \Longrightarrow n\ y \sqsubseteq n\ z \Longrightarrow n\ x \oplus n\ y \sqsubseteq n\ z$
⟨*proof*⟩

lemma *n-add-lub*: $n\ x \sqsubseteq n\ z \wedge n\ y \sqsubseteq n\ z \longleftrightarrow n\ x \oplus n\ y \sqsubseteq n\ z$
⟨*proof*⟩

lemma *n-galois1*: $n\ x \sqsubseteq n\ y \oplus n\ z \longleftrightarrow n\ x \cdot t\ y \sqsubseteq n\ z$
⟨*proof*⟩

lemma *n-galois2*: $n\ x \sqsubseteq t\ y \oplus n\ z \longleftrightarrow n\ x \cdot n\ y \sqsubseteq n\ z$
⟨*proof*⟩

lemma *n-distrib-alt*: $n\ x \cdot n\ z = n\ y \cdot n\ z \Longrightarrow n\ x \oplus n\ z = n\ y \oplus n\ z \Longrightarrow n\ x = n\ y$

⟨*proof*⟩

lemma *n-dist-var1*: $(n\ x \oplus n\ y) \cdot (t\ x \oplus n\ z) = t\ x \cdot n\ y \oplus n\ x \cdot n\ z$
⟨*proof*⟩

lemma *n-dist-var2*: $n\ (n\ x \cdot n\ y \oplus t\ x \cdot n\ z) = n\ x \cdot t\ y \oplus t\ x \cdot t\ z$
⟨*proof*⟩

end

1.2 Test Near-Semirings

class *test-near-semiring-zerol* = *ab-near-semiring-one-zerol* + *n-op* + *plus-ord* +
assumes *test-one* [*simp*]: $n\ n\ 1 = 1$
and *test-mult* [*simp*]: $n\ n\ (n\ x \cdot n\ y) = n\ x \cdot n\ y$
and *test-mult-comp* [*simp*]: $n\ x \cdot n\ n\ x = 0$
and *test-de-morgan* [*simp*]: $n\ (n\ n\ x \cdot n\ n\ y) = n\ x + n\ y$

begin

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

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

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

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

lemma *t-n-closed* [*simp*]: $n\ n\ (n\ x) = n\ x$
⟨*proof*⟩

lemma *t-de-morgan-var1* [*simp*]: $n\ (n\ x \cdot n\ y) = n\ n\ x + n\ n\ y$
⟨*proof*⟩

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

lemma *tm3'*: $n\ x \cdot n\ (n\ n\ z \cdot n\ n\ y) = n\ (n\ (n\ x \cdot n\ y) \cdot n\ (n\ x \cdot n\ z))$
⟨*proof*⟩

subclass *test-monoid*
⟨*proof*⟩

lemma *ord-transl* [*simp*]: $n\ x \leq n\ y \longleftrightarrow n\ x \sqsubseteq n\ y$
⟨*proof*⟩

lemma *add-transl* [*simp*]: $n\ x + n\ y = n\ x \oplus n\ y$
⟨*proof*⟩

lemma *zero-trans*: $0 = o$
⟨*proof*⟩

definition *test* :: $'a \Rightarrow bool$ **where**
$test\ p \equiv t\ p = p$

notation *n-op* (‹!-› [*101*] *100*)

lemma *test-prop*: $(\forall x.\ test\ x \longrightarrow P\ x) \longleftrightarrow (\forall x.\ P\ (t\ x))$
⟨*proof*⟩

lemma *test-propI*: $test\ x \Longrightarrow P\ x \Longrightarrow P\ (t\ x)$
⟨*proof*⟩

lemma *test-propE* [*elim*!]: $test\ x \Longrightarrow P\ (t\ x) \Longrightarrow P\ x$
⟨*proof*⟩

lemma *test-comp-closed* [*simp*]: $test\ p \Longrightarrow test\ (!p)$
⟨*proof*⟩

lemma *test-double-comp-var*: $test\ p \Longrightarrow p = !(!p)$
⟨*proof*⟩

lemma *test-mult-closed*: $test\ p \Longrightarrow test\ q \Longrightarrow test\ (p \cdot q)$
⟨*proof*⟩

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

lemma *test-add-closed*: $test\ p \Longrightarrow test\ q \Longrightarrow test\ (p + q)$
⟨*proof*⟩

lemma *test-mult-comm-var*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot q = q \cdot p$
⟨*proof*⟩

lemma *t-zero* [*simp*]: $t\ 0 = 0$
⟨*proof*⟩

lemma *test-zero-var*: $test\ 0$
⟨*proof*⟩

lemma *test-one-var*: $test\ 1$
⟨*proof*⟩

lemma *test-preserve*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot q \cdot p = q \cdot p$

⟨*proof*⟩

lemma *test-preserve2*: *test p* ⟹ *test q* ⟹ $p \cdot q \cdot !p = 0$
⟨*proof*⟩

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

lemma *test-subid*: *test p* ⟹ $p \leq 1$
⟨*proof*⟩

lemma *test-mult-idem-var* [*simp*]: *test p* ⟹ $p \cdot p = p$
⟨*proof*⟩

lemma *n-add-comp* [*simp*]: $n\ x + t\ x = 1$
⟨*proof*⟩

lemma *n-add-comp-var* [*simp*]: $t\ x + n\ x = 1$
⟨*proof*⟩

lemma *test-add-comp* [*simp*]: *test p* ⟹ $p + !p = 1$
⟨*proof*⟩

lemma *test-comp-mult1* [*simp*]: *test p* ⟹ $!p \cdot p = 0$
⟨*proof*⟩

lemma *test-comp-mult2* [*simp*]: *test p* ⟹ $p \cdot !p = 0$
⟨*proof*⟩

lemma *test-comp*: *test p* ⟹ $\exists\, q.$ *test q* $\wedge\ p + q = 1 \wedge p \cdot q = 0$
⟨*proof*⟩

lemma *n-absorb1'* [*simp*]: $n\ x + n\ x \cdot n\ y = n\ x$
⟨*proof*⟩

lemma *test-absorb1* [*simp*]: *test p* ⟹ *test q* ⟹ $p + p \cdot q = p$
⟨*proof*⟩

lemma *n-absorb2'* [*simp*]: $n\ x \cdot (n\ x + n\ y) = n\ x$
⟨*proof*⟩

lemma *test-absorb2*: *test p* ⟹ *test q* ⟹ $p \cdot (p + q) = p$
⟨*proof*⟩

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

lemma *test-distrib-left*: *test p* ⟹ *test q* ⟹ *test r* ⟹ $p \cdot (q + r) = p \cdot q + p \cdot$

r
⟨*proof*⟩

lemma *de-morgan1′*: *test p* ⟹ *test q* ⟹ $!p + !q = !(p \cdot q)$
⟨*proof*⟩

lemma *n-de-morgan-var2* [*simp*]: $n\ (n\ x + n\ y) = t\ x \cdot t\ y$
⟨*proof*⟩

lemma *n-de-morgan-var3* [*simp*]: $n\ (t\ x + t\ y) = n\ x \cdot n\ y$
⟨*proof*⟩

lemma *de-morgan2*: *test p* ⟹ *test q* ⟹ $!p \cdot !q = !(p + q)$
⟨*proof*⟩

lemma *de-morgan3*: *test p* ⟹ *test q* ⟹ $!(!p + !q) = p \cdot q$
⟨*proof*⟩

lemma *de-morgan4′*: *test p* ⟹ *test q* ⟹ $!(!p \cdot !q) = p + q$
⟨*proof*⟩

lemma *n-add-distr*: $n\ x + (n\ y \cdot n\ z) = (n\ x + n\ y) \cdot (n\ x + n\ z)$
⟨*proof*⟩

lemma *test-add-distr*: *test p* ⟹ *test q* ⟹ *test r* ⟹ $p + q \cdot r = (p + q) \cdot (p + r)$
⟨*proof*⟩

lemma *n-add-distl*: $(n\ x \cdot n\ y) + n\ z = (n\ x + n\ z) \cdot (n\ y + n\ z)$
⟨*proof*⟩

lemma *test-add-distl-var*: *test p* ⟹ *test q* ⟹ *test r* ⟹ $p \cdot q + r = (p + r) \cdot (q + r)$
⟨*proof*⟩

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

lemma *test-leq-mult-def-var*: *test p* ⟹ *test q* ⟹ $p \leq q \longleftrightarrow p \cdot q = p$
⟨*proof*⟩

lemma *n-anti*: $n\ x \leq n\ y \Longrightarrow t\ y \leq t\ x$
⟨*proof*⟩

lemma *n-anti-iff*: $n\ x \leq n\ y \longleftrightarrow t\ y \leq t\ x$
⟨*proof*⟩

lemma *test-comp-anti-iff*: *test p* ⟹ *test q* ⟹ $p \leq q \longleftrightarrow !q \leq !p$
⟨*proof*⟩

lemma *n-restrictl*: $n\ x \cdot y \leq y$
⟨*proof*⟩

lemma *test-restrictl*: *test* $p \Longrightarrow p \cdot x \leq x$
⟨*proof*⟩

lemma *n-mult-lb1'*: $n\ x \cdot n\ y \leq n\ x$
⟨*proof*⟩

lemma *test-mult-lb1*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow p \cdot q \leq p$
⟨*proof*⟩

lemma *n-mult-lb2'*: $n\ x \cdot n\ y \leq n\ y$
⟨*proof*⟩

lemma *test-mult-lb2*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow p \cdot q \leq q$
⟨*proof*⟩

lemma *n-mult-glbI'*: $n\ z \leq n\ x \Longrightarrow\ n\ z \leq n\ y \Longrightarrow n\ z \leq n\ x \cdot n\ y$
⟨*proof*⟩

lemma *test-mult-glbI*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow$ *test* $r \Longrightarrow p \leq q \Longrightarrow p \leq r \Longrightarrow p \leq q \cdot r$
⟨*proof*⟩

lemma *n-mult-glb'*: $n\ z \leq n\ x \wedge n\ z \leq n\ y \longleftrightarrow n\ z \leq n\ x \cdot n\ y$
⟨*proof*⟩

lemma *test-mult-glb*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow$ *test* $r \Longrightarrow p \leq q \wedge p \leq r \longleftrightarrow p \leq q \cdot r$
⟨*proof*⟩

lemma *n-galois1'*: $n\ x \leq n\ y + n\ z \longleftrightarrow n\ x \cdot t\ y \leq n\ z$
⟨*proof*⟩

lemma *test-galois1*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow$ *test* $r \Longrightarrow p \leq q + r \longleftrightarrow p \cdot\ !q \leq r$
⟨*proof*⟩

lemma *n-galois2'*: $n\ x \leq t\ y + n\ z \longleftrightarrow n\ x \cdot n\ y \leq n\ z$
⟨*proof*⟩

lemma *test-galois2*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow$ *test* $r \Longrightarrow p \leq\ !q + r \longleftrightarrow p \cdot q \leq r$
⟨*proof*⟩

lemma *n-huntington2*: $n\ (n\ x + t\ y) + n\ (n\ x + n\ y) = n\ n\ x$
⟨*proof*⟩

lemma *test-huntington2*: *test* $p \Longrightarrow$ *test* $q \Longrightarrow\ !(p + q) + !(p + !q) = !p$

⟨*proof*⟩

lemma *n-kat-1-opp*: $n\ x \cdot y \cdot n\ z = y \cdot n\ z \longleftrightarrow t\ x \cdot y \cdot n\ z = 0$
⟨*proof*⟩

lemma *test-eq4*: $test\ p \Longrightarrow test\ q \Longrightarrow\ !p \cdot x \cdot\ !q = x \cdot\ !q \longleftrightarrow p \cdot x \cdot\ !q = 0$
⟨*proof*⟩

lemma *n-kat-1-var*: $t\ x \cdot y \cdot t\ z = y \cdot t\ z \longleftrightarrow n\ x \cdot y \cdot t\ z = 0$
⟨*proof*⟩

lemma *test-kat-1*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot x \cdot q = x \cdot q \longleftrightarrow\ !p \cdot x \cdot q = 0$
⟨*proof*⟩

lemma *n-kat-21-opp*: $y \cdot n\ z \leq n\ x \cdot y \Longrightarrow n\ x \cdot y \cdot n\ z = y \cdot n\ z$
⟨*proof*⟩

lemma *test-kat-21-opp*: $test\ p \Longrightarrow test\ q \Longrightarrow x \cdot q \leq p \cdot x \longrightarrow p \cdot x \cdot q = x \cdot q$
⟨*proof*⟩

lemma $n\ x \cdot y \cdot n\ z = y \cdot n\ z \Longrightarrow y \cdot n\ z \leq n\ x \cdot y$
⟨*proof*⟩

lemma *n-distrib-alt'*: $n\ x \cdot n\ z = n\ y \cdot n\ z \Longrightarrow n\ x + n\ z = n\ y + n\ z \Longrightarrow n\ x = n\ y$
⟨*proof*⟩

lemma *test-distrib-alt*: $test\ p \Longrightarrow test\ q \Longrightarrow test\ r \Longrightarrow p \cdot r = q \cdot r \wedge p + r = q + r \longrightarrow p = q$
⟨*proof*⟩

lemma *n-eq1*: $n\ x \cdot y \leq z \wedge t\ x \cdot y \leq z \longleftrightarrow y \leq z$
⟨*proof*⟩

lemma *test-eq1*: $test\ p \Longrightarrow y \leq x \longleftrightarrow p \cdot y \leq x \wedge\ !p \cdot y \leq x$
⟨*proof*⟩

lemma *n-dist-var1'*: $(n\ x + n\ y) \cdot (t\ x + n\ z) = t\ x \cdot n\ y + n\ x \cdot n\ z$
⟨*proof*⟩

lemma *test-dist-var1*: $test\ p \Longrightarrow test\ q \Longrightarrow test\ r \Longrightarrow (p + q) \cdot (!p + r) =\ !p \cdot q + p \cdot r$
⟨*proof*⟩

lemma *n-dist-var2'*: $n\ (n\ x \cdot n\ y + t\ x \cdot n\ z) = n\ x \cdot t\ y + t\ x \cdot t\ z$
⟨*proof*⟩

lemma *test-dist-var2*: $test\ p \Longrightarrow test\ q \Longrightarrow test\ r \Longrightarrow\ !(p \cdot q +\ !p \cdot r) = (p \cdot\ !q +\ !p \cdot\ !r)$

⟨*proof*⟩

lemma *test-restrictr*: *test p* $\Longrightarrow x \cdot p \leq x$
⟨*proof*⟩

lemma *test-eq2*: *test p* $\Longrightarrow z \leq p \cdot x + !p \cdot y \longleftrightarrow p \cdot z \leq p \cdot x \wedge !p \cdot z \leq !p \cdot y$
⟨*proof*⟩

lemma *test-eq3*: $[\![$*test p*; *test q*$]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q \longleftrightarrow p \cdot x \leq x \cdot q$
⟨*proof*⟩

lemma *test1*: $[\![$*test p*; *test q*; $p \cdot x \cdot !q = 0]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma $[\![$*test p*; *test q*; $x \cdot !q = !p \cdot x \cdot !q]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma *comm-add*: $[\![$*test p*; $p{\cdot}x = x{\cdot}p$; $p{\cdot}y = y{\cdot}p]\!] \Longrightarrow p{\cdot}(x + y) = (x + y){\cdot}p$
⟨*proof*⟩

lemma *comm-add-var*: $[\![$*test p*; *test q*; *test r*; $p{\cdot}x = x{\cdot}p$; $p{\cdot}y = y{\cdot}p]\!] \Longrightarrow p{\cdot}(q{\cdot}x + r{\cdot}y) = (q{\cdot}x + r{\cdot}y){\cdot}p$
⟨*proof*⟩

lemma *test p* $\Longrightarrow p \cdot x = x \cdot p \Longrightarrow p \cdot x = p \cdot x \cdot p \wedge !p \cdot x = !p \cdot x \cdot !p$
⟨*proof*⟩

lemma *test-distrib*: $[\![$*test p*; *test q*$]\!] \Longrightarrow (p + q){\cdot}(q{\cdot}y + !q{\cdot}x) = q{\cdot}y + !q{\cdot}p{\cdot}x$
⟨*proof*⟩

end

1.3 Test Near Semirings with Distributive Tests

We now make the assumption that tests distribute over finite sums of arbitrary elements from the left. This can be justified in models such as multirelations and probabilistic predicate transformers.

class *test-near-semiring-zerol-distrib* = *test-near-semiring-zerol* +
assumes *n-left-distrib*: *n x* $\cdot$ $(y + z)$ = *n x* $\cdot$ *y* + *n x* $\cdot$ *z*

begin

lemma *n-left-distrib-var*: *test p* $\Longrightarrow p \cdot (x + y) = p \cdot x + p \cdot y$
⟨*proof*⟩

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

lemma *test-mult-isol*: *test p* $\Longrightarrow x \leq y \Longrightarrow p \cdot x \leq p \cdot y$

⟨*proof*⟩

lemma *test p* $\Longrightarrow x \cdot p \leq x$
⟨*proof*⟩

lemma $[\![$*test p*; *test q*$]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q \longleftrightarrow p \cdot x \leq x \cdot q$
⟨*proof*⟩

lemma $[\![$*test p*; *test q*; $p \cdot x \cdot\, !q = 0]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma $[\![$*test p*; *test q*; $x \cdot\, !q = !p \cdot x \cdot\, !q]\!] \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

Next, we study tests with commutativity conditions.

lemma *comm-add*: *test p* $\Longrightarrow\ p \cdot x = x \cdot p \Longrightarrow\ p \cdot y = y \cdot p \Longrightarrow p \cdot (x + y) = (x + y) \cdot p$
⟨*proof*⟩

lemma *comm-add-var*: *test p* $\Longrightarrow$ *test q* $\Longrightarrow$ *test r* $\Longrightarrow p \cdot x = x \cdot p \Longrightarrow p \cdot y = y \cdot p \Longrightarrow p \cdot (q \cdot x + r \cdot y) = (q \cdot x + r \cdot y) \cdot p$
⟨*proof*⟩

lemma *test-distrib*: *test p* $\Longrightarrow$ *test q* $\Longrightarrow (p + q) \cdot (q \cdot y + !q \cdot x) = q \cdot y + !q \cdot p \cdot x$
⟨*proof*⟩

end

1.4 Test Predioids

The following class is relevant for probabilistic Kleene algebras.

class *test-pre-dioid-zerol* = *test-near-semiring-zerol-distrib* + *pre-dioid*

begin

lemma *n-restrictr*: $x \cdot n\ y \leq x$
⟨*proof*⟩

lemma *test-restrictr*: *test p* $\Longrightarrow x \cdot p \leq x$
⟨*proof*⟩

lemma *n-kat-2*: $n\ x \cdot y = n\ x \cdot y \cdot n\ z \longleftrightarrow n\ x \cdot y \leq y \cdot n\ z$
⟨*proof*⟩

lemma *test-kat-2*: *test p* $\Longrightarrow$ *test q* $\Longrightarrow p \cdot x = p \cdot x \cdot q \longleftrightarrow p \cdot x \leq x \cdot q$
⟨*proof*⟩

lemma *n-kat-2-opp*: $y \cdot n\ z = n\ x \cdot y \cdot n\ z \longleftrightarrow y \cdot n\ z \leq n\ x \cdot y$

⟨*proof*⟩

lemma *test-kat-2-opp*: *test p* ⟹ *test q* ⟹ $x \cdot q = p \cdot x \cdot q \longleftrightarrow x \cdot q \leq p \cdot x$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $p{\cdot}x{\cdot}!q = 0$⟧ ⟹ $p{\cdot}x = p{\cdot}x{\cdot}q$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $x{\cdot}!q = !p{\cdot}x{\cdot}!q$⟧ ⟹ $p{\cdot}x = p{\cdot}x{\cdot}q$
⟨*proof*⟩

lemma ⟦*test p*; *test q*⟧ ⟹ $x \cdot (p + q) \leq x \cdot p + x \cdot q$
⟨*proof*⟩

end

The following class is relevant for Demonic Refinement Algebras.

class *test-semiring-zerol* = *test-near-semiring-zerol* + *semiring-one-zerol*

begin

subclass *dioid-one-zerol* ⟨*proof*⟩

subclass *test-pre-dioid-zerol*
⟨*proof*⟩

lemma *n-kat-31*: $n\ x \cdot y \cdot t\ z = 0 \Longrightarrow n\ x \cdot y \cdot n\ z = n\ x \cdot y$
⟨*proof*⟩

lemma *test-kat-31*: *test p* ⟹ *test q* ⟹ $p \cdot x \cdot !q = 0 \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma *n-kat-var*: $t\ x \cdot y \cdot t\ z = y \cdot t\ z \Longrightarrow n\ x \cdot y \cdot n\ z = n\ x \cdot y$
⟨*proof*⟩

lemma *test1-var*: *test p* ⟹ *test q* ⟹ $x \cdot !q = !p \cdot x \cdot !q \Longrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $p{\cdot}x{\cdot}!q = 0$⟧ ⟹ $!p{\cdot}x{\cdot}q = 0$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $p{\cdot}x = p{\cdot}x{\cdot}q$⟧ ⟹ $x{\cdot}!q = !p{\cdot}x{\cdot}!q$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $p{\cdot}x = p{\cdot}x{\cdot}q$⟧ ⟹ $p{\cdot}x{\cdot}!q = 0$
⟨*proof*⟩

lemma ⟦*test p*; *test q*; $p{\cdot}x = p{\cdot}x{\cdot}q$⟧ ⟹ $!p{\cdot}x{\cdot}q = 0$
⟨*proof*⟩

lemma $[\![test\ p;\ test\ q;\ x{\cdot}!q = !p{\cdot}x{\cdot}!q]\!] \Longrightarrow !p{\cdot}x{\cdot}q = 0$
⟨*proof*⟩

lemma $[\![test\ p;\ test\ q;\ !p{\cdot}x{\cdot}q = 0]\!] \Longrightarrow p{\cdot}x = p{\cdot}x{\cdot}q$
⟨*proof*⟩

lemma $[\![test\ p;\ test\ q;\ !p{\cdot}x{\cdot}q = 0]\!] \Longrightarrow x{\cdot}!q = !p{\cdot}x{\cdot}!q$
⟨*proof*⟩

lemma $[\![test\ p;\ test\ q;\ !p{\cdot}x{\cdot}q = 0]\!] \Longrightarrow p{\cdot}x{\cdot}!q = 0$
⟨*proof*⟩

lemma $test\ p \Longrightarrow p \cdot x = p \cdot x \cdot p \wedge\ !p \cdot x = !p \cdot x \cdot !p \Longrightarrow p \cdot x = x \cdot p$
⟨*proof*⟩

end

1.5 Test Semirings

The following class is relevant for Kleene Algebra with Tests.

class *test-semiring* = *test-semiring-zerol* + *semiring-one-zero*

begin

lemma *n-kat-1*: $n\ x \cdot y \cdot t\ z = 0 \longleftrightarrow n\ x \cdot y \cdot n\ z = n\ x \cdot y$
⟨*proof*⟩

lemma *test-kat-1′*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot x \cdot !q = 0 \longleftrightarrow p \cdot x = p \cdot x \cdot q$
⟨*proof*⟩

lemma *n-kat-3*: $n\ x \cdot y \cdot t\ z = 0 \longleftrightarrow n\ x \cdot y \leq y \cdot n\ z$
⟨*proof*⟩

lemma *test-kat-3*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot x \cdot !q = 0 \longleftrightarrow p \cdot x \leq x \cdot q$
⟨*proof*⟩

lemma *n-kat-prop*: $n\ x \cdot y \cdot n\ z = n\ x \cdot y \longleftrightarrow t\ x \cdot y \cdot t\ z = y \cdot t\ z$
⟨*proof*⟩

lemma *test-kat-prop*: $test\ p \Longrightarrow test\ q \Longrightarrow p \cdot x = p \cdot x \cdot q \longleftrightarrow x \cdot !q = !p \cdot x \cdot !q$
⟨*proof*⟩

lemma *n-kat-3-opp*: $t\ x \cdot y \cdot n\ z = 0 \longleftrightarrow y \cdot n\ z \leq n\ x \cdot y$
⟨*proof*⟩

lemma *kat-1-var*: $n\ x \cdot y \cdot n\ z = y \cdot n\ z \longleftrightarrow y \cdot n\ z \leq n\ x \cdot y$
⟨*proof*⟩

lemma $[\![test\ p;\ test\ q]\!] \Longrightarrow (p{\cdot}x{\cdot}!q = 0) \Longrightarrow (!p{\cdot}x{\cdot}q = 0)$
⟨*proof*⟩

lemma $n\ x \cdot y + t\ x \cdot z = n\ x \cdot y \vee n\ x \cdot y + t\ x \cdot z = t\ x \cdot z$

⟨*proof*⟩

end

end

2 Pre-Conway Algebra with Tests

theory *Conway-Tests*
imports *Kleene-Algebra.Conway Test-Dioid*

begin

class *near-conway-zerol-tests* = *near-conway-zerol* + *test-near-semiring-zerol-distrib*

begin

lemma *n-preserve1-var*: $n\ x \cdot y \leq n\ x \cdot y \cdot n\ x \Longrightarrow n\ x \cdot (n\ x \cdot y + t\ x \cdot z)^\dagger \leq (n\ x \cdot y)^\dagger \cdot n\ x$
⟨*proof*⟩

lemma *test-preserve1-var*: $test\ p \Longrightarrow p \cdot x \leq p \cdot x \cdot p \Longrightarrow p \cdot (p \cdot x + !p \cdot y)^\dagger \leq (p \cdot x)^\dagger \cdot p$
⟨*proof*⟩

end

class *test-pre-conway* = *pre-conway* + *test-pre-dioid-zerol*

begin

subclass *near-conway-zerol-tests*
⟨*proof*⟩

lemma *test-preserve*: $test\ p \Longrightarrow\ p \cdot x \leq p \cdot x \cdot p\ \Longrightarrow p \cdot x^\dagger = (p \cdot x)^\dagger \cdot p$
⟨*proof*⟩

lemma *test-preserve1*: $test\ p \Longrightarrow p \cdot x \leq p \cdot x \cdot p \Longrightarrow p \cdot (p \cdot x + !p \cdot y)^\dagger = (p \cdot x)^\dagger \cdot p$
⟨*proof*⟩

lemma *test-preserve2*: $test\ p \Longrightarrow p \cdot x \cdot p = p \cdot x \Longrightarrow p \cdot (p \cdot x + !p \cdot y)^\dagger \leq x^\dagger$
⟨*proof*⟩

end

end

3 Kleene Algebra with Tests

theory *KAT*
imports *Kleene-Algebra.Kleene-Algebra Conway-Tests*
begin

First, we study left Kleene algebras with tests which also have only a left zero. These structures can be expanded to demonic refinement algebras.

class *left-kat-zerol* = *left-kleene-algebra-zerol* + *test-semiring-zerol*
begin

lemma *star-n-export1*: $(n\ x \cdot y)^\star \cdot n\ x \leq n\ x \cdot y^\star$
⟨*proof*⟩

lemma *star-test-export1*: *test* $p \Longrightarrow (p \cdot x)^\star \cdot p \leq p \cdot x^\star$
⟨*proof*⟩

lemma *star-n-export2*: $(n\ x \cdot y)^\star \cdot n\ x \leq y^\star \cdot n\ x$
⟨*proof*⟩

lemma *star-test-export2*: *test* $p \Longrightarrow (p \cdot x)^\star \cdot p \leq x^\star \cdot p$
⟨*proof*⟩

lemma *star-n-export-left*: $x \cdot n\ y \leq n\ y \cdot x \Longrightarrow x^\star \cdot n\ y = n\ y \cdot (x \cdot n\ y)^\star$
⟨*proof*⟩

lemma *star-test-export-left*: *test* $p \Longrightarrow x \cdot p \leq p \cdot x \longrightarrow x^\star \cdot p = p \cdot (x \cdot p)^\star$
⟨*proof*⟩

lemma *star-n-export-right*: $x \cdot n\ y \leq n\ y \cdot x \Longrightarrow x^\star \cdot n\ y = (n\ y \cdot x)^\star \cdot n\ y$
⟨*proof*⟩

lemma *star-test-export-right*: *test* $p \Longrightarrow x \cdot p \leq p \cdot x \longrightarrow x^\star \cdot p = (p \cdot x)^\star \cdot p$
⟨*proof*⟩

lemma *star-n-folk*: $n\ z \cdot x = x \cdot n\ z \Longrightarrow n\ z \cdot y = y \cdot n\ z \Longrightarrow (n\ z \cdot x + t\ z \cdot y)^\star \cdot n\ z = n\ z \cdot (n\ z \cdot x)^\star$
⟨*proof*⟩

lemma *star-test-folk*: *test* $p \Longrightarrow p \cdot x = x \cdot p \longrightarrow p \cdot y = y \cdot p \longrightarrow (p \cdot x + !p \cdot y)^\star \cdot p = p \cdot (p \cdot x)^\star$
⟨*proof*⟩

end

class *kat-zerol* = *kleene-algebra-zerol* + *test-semiring-zerol*
begin

sublocale *conway*: *near-conway-zerol-tests star* ⟨*proof*⟩

lemma *n-star-sim-right*: $n\ y \cdot x = x \cdot n\ y \Longrightarrow n\ y \cdot x^\star = (n\ y \cdot x)^\star \cdot n\ y$
⟨*proof*⟩

lemma *star-sim-right*: *test* $p \Longrightarrow p \cdot x = x \cdot p \longrightarrow p \cdot x^\star = (p \cdot x)^\star \cdot p$
⟨*proof*⟩

lemma *n-star-sim-left*: $n\ y \cdot x = x \cdot n\ y \Longrightarrow n\ y \cdot x^\star = n\ y \cdot (x \cdot n\ y)^\star$
⟨*proof*⟩

lemma *star-sim-left*: *test* $p \Longrightarrow p \cdot x = x \cdot p \longrightarrow p \cdot x^\star = p \cdot (x \cdot p)^\star$
⟨*proof*⟩

lemma *n-comm-star*: $n\ z \cdot x = x \cdot n\ z \Longrightarrow\ n\ z \cdot y = y \cdot n\ z \Longrightarrow n\ z \cdot x \cdot (n\ z \cdot y)^\star = n\ z \cdot x \cdot y^\star$
⟨*proof*⟩

lemma *comm-star*: *test* $p \Longrightarrow p \cdot x = x \cdot p \longrightarrow\ p \cdot y = y \cdot p \longrightarrow p \cdot x \cdot (p \cdot y)^\star = p \cdot x \cdot y^\star$
⟨*proof*⟩

lemma *n-star-sim-right-var*: $n\ y \cdot x = x \cdot n\ y \Longrightarrow x^\star \cdot n\ y = n\ y \cdot (x \cdot n\ y)^\star$
⟨*proof*⟩

lemma *star-sim-right-var*: *test* $p \Longrightarrow p \cdot x = x \cdot p \longrightarrow x^\star \cdot p = p \cdot (x \cdot p)^\star$
⟨*proof*⟩

end

Finally, we define Kleene algebra with tests.

class *kat* = *kleene-algebra* + *test-semiring*

begin

sublocale *conway*: *test-pre-conway star* ⟨*proof*⟩

end

end

4 Demonic Refinement Algebra with Tests

theory *DRAT*
imports *KAT Kleene-Algebra.DRA*

begin

In this section, we define demonic refinement algebras with tests and prove the most important theorems from the literature. In this context, tests are also known as guards.

class *drat* = *dra* + *test-semiring-zerol*
begin

subclass *kat-zerol* ⟨*proof*⟩

An assertion is a mapping from a guard to a subset similar to tests, but it aborts if the predicate does not hold.

definition *assertion* :: *'a* ⇒ *'a* (⟨-^{o}⟩ [*101*] *100*) **where**
test p ⟹ p^o = !*p*·⊤ + *1*

lemma *asg*: ⟦*test p*; *test q*⟧ ⟹ *q* ≤ *1* ∧ *1* ≤ p^o
⟨*proof*⟩

lemma *assertion-isol*: *test p* ⟹ *y* ≤ p^o·*x* ⟷ *p*·*y* ≤ *x*
⟨*proof*⟩

lemma *assertion-isor*: *test p* ⟹ *y* ≤ *x*·*p* ⟷ *y*·p^o ≤ *x*
⟨*proof*⟩

lemma *assertion-iso*: ⟦*test p*; *test q*⟧ ⟹ *x*·q^o ≤ p^o·*x* ⟷ *p*·*x* ≤ *x*·*q*
⟨*proof*⟩

lemma *total-correctness*: ⟦*test p*; *test q*⟧ ⟹ *p*·*x*·!*q* = *0* ⟷ *x*·!*q* ≤ !*p*·⊤
⟨*proof*⟩

lemma *test-iteration-sim*: ⟦*test p*; *p*·*x* ≤ *x*·*p*⟧ ⟹ *p*·x^∞ ≤ x^∞·*p*
⟨*proof*⟩

lemma *test-iteration-annir*: *test p* ⟹ !*p*·$(p{\cdot}x)^\infty$ = !*p*
⟨*proof*⟩

Next we give an example of a program transformation from von Wright [5].

lemma *loop-refinement*: ⟦*test p*; *test q*⟧ ⟹ $(p{\cdot}x)^\infty$·!*p* ≤ $(p{\cdot}q{\cdot}x)^\infty$·!(*p*·*q*)·$(p{\cdot}x)^\infty$·!*p*
⟨*proof*⟩

Finally, we prove different versions of Back's atomicity refinement theorem for action systems.

lemma *atom-step1*: *r*·*b* ≤ *b*·*r* ⟹ $(a + b + r)^\infty$ = b^∞·r^∞·$(a{\cdot}b^\infty{\cdot}r^\infty)^\infty$
⟨*proof*⟩

lemma *atom-step2*:
assumes *s* = *s*·*q* *q*·*b* = *0* *r*·*q* ≤ *q*·*r* *q*·*l* ≤ *l*·*q* r^∞ = $r^\star$ *test q*
shows *s*·l^∞·b^∞·r^∞·*q*·$(a{\cdot}b^\infty{\cdot}r^\infty{\cdot}q)^\infty$ ≤ *s*·l^∞·r^∞·$(a{\cdot}b^\infty{\cdot}q{\cdot}r^\infty)^\infty$

⟨*proof*⟩

lemma *atom-step3*:
assumes $r{\cdot}l \leq l{\cdot}r$ $a{\cdot}l \leq l{\cdot}a$ $b{\cdot}l \leq l{\cdot}b$ $q{\cdot}l \leq l{\cdot}q$ $b^\infty = b^\star$
shows $l^\infty{\cdot}r^\infty{\cdot}(a{\cdot}b^\infty{\cdot}q{\cdot}r^\infty)^\infty = (a{\cdot}b^\infty{\cdot}q + l + r)^\infty$
⟨*proof*⟩

This is Back's atomicity refinement theorem, as specified by von Wright [5].

theorem *atom-ref-back*:
assumes $s = s{\cdot}q$ $a = q{\cdot}a$ $q{\cdot}b = 0$
$r{\cdot}b \leq b{\cdot}r$ $r{\cdot}l \leq l{\cdot}r$ $r{\cdot}q \leq q{\cdot}r$
$a{\cdot}l \leq l{\cdot}a$ $b{\cdot}l \leq l{\cdot}b$ $q{\cdot}l \leq l{\cdot}q$
$r^\infty = r^\star$ $b^\infty = b^\star$ *test* q
shows $s{\cdot}(a + b + r + l)^\infty{\cdot}q \leq s{\cdot}(a{\cdot}b^\infty{\cdot}q + r + l)^\infty$
⟨*proof*⟩

This variant is due to Höfner, Struth and Sutcliffe [2].

theorem *atom-ref-back-struth*:
assumes $s \leq s{\cdot}q$ $a \leq q{\cdot}a$ $q{\cdot}b = 0$
$r{\cdot}b \leq b{\cdot}r$ $r{\cdot}q \leq q{\cdot}r$
$(a + r + b){\cdot}l \leq l{\cdot}(a + r + b)$ $q{\cdot}l \leq l{\cdot}q$
$r^\infty = r^\star$ $q \leq 1$
shows $s{\cdot}(a + b + r + l)^\infty{\cdot}q \leq s{\cdot}(a{\cdot}b^\infty{\cdot}q + r + l)^\infty$
⟨*proof*⟩

Finally, we prove Cohen's [1] variation of the atomicity refinement theorem.

lemma *atom-ref-cohen*:
assumes $r{\cdot}p{\cdot}y \leq y{\cdot}r$ $y{\cdot}p{\cdot}l \leq l{\cdot}y$ $r{\cdot}p{\cdot}l \leq l{\cdot}r$
$p{\cdot}r{\cdot}!p = 0$ $p{\cdot}l{\cdot}!p = 0$ $!p{\cdot}l{\cdot}p = 0$
$y{\cdot}0 = 0$ $r{\cdot}0 = 0$ *test* p
shows $(y + r + l)^\infty = (p{\cdot}l)^\infty{\cdot}(y + !p{\cdot}l + r{\cdot}!p)^\infty{\cdot}(r{\cdot}p)^\infty$
⟨*proof*⟩
end

end

5 Models for Demonic Refinement Algebra with Tests

theory *DRA-Models*
imports *DRAT*
begin

We formalise the predicate transformer model of demonic refinement algebra. Predicate transformers are formalised as strict and additive functions over a field of sets, or alternatively as costrict and multiplicative functions.

In the future, this should be merged with Preoteasa's more abstract formalisation [4].

no-notation
plus (**infixl** ‹+› *65*) **and**
less-eq (‹'(≤')›) **and**
less-eq (‹(‹*notation*=‹*infix* ≤››-/ ≤ -)› [*51*, *51*] *50*)

notation *comp* (**infixl** ‹·› *55*)

type-synonym *'a bfun* = *'a set* ⇒ *'a set*

Definitions of signature:

definition *top* :: *'a bfun* **where** *top* ≡ λ*x*. *UNIV*
definition *bot* :: *'a bfun* **where** *bot* ≡ λ*x*. {}
definition *adjoint* :: *'a bfun* ⇒ *'a bfun* **where** *adjoint f* ≡ (λ*p*. − *f* (−*p*))

definition *fun-inter* :: *'a bfun* ⇒ *'a bfun* ⇒ *'a bfun* (**infix** ‹⊓› *51*) **where**
f ⊓ *g* ≡ λ*p*. *f p* ∩ *g p*

definition *fun-union* :: *'a bfun* ⇒ *'a bfun* ⇒ *'a bfun* (**infix** ‹+› *52*) **where**
f + *g* ≡ λ*p*. *f p* ∪ *g p*

definition *fun-order* :: *'a bfun* ⇒ *'a bfun* ⇒ *bool* (**infix** ‹≤› *50*) **where**
f ≤ *g* ≡ ∀ *p*. *f p* ⊆ *g p*

definition *fun-strict-order* :: *'a bfun* ⇒ *'a bfun* ⇒ *bool* (**infix** ‹<.› *50*) **where**
f <. *g* ≡ *f* ≤ *g* ∧ *f* ≠ *g*

definition *N* :: *'a bfun* ⇒ *'a bfun* **where**
N f ≡ ((*adjoint f o bot*) ⊓ *id*)

lemma *top-max*: *f* ≤ *top*
⟨*proof*⟩

lemma *bot-min*: *bot* ≤ *f*
⟨*proof*⟩

lemma *oder-def*: *f* ⊓ *g* = *f* ⟹ *f* ≤ *g*
⟨*proof*⟩

lemma *order-def-var*: *f* ≤ *g* ⟹ *f* ⊓ *g* = *f*
⟨*proof*⟩

lemma *adjoint-idem* [*simp*]: *adjoint* (*adjoint f*) = *f*
⟨*proof*⟩

lemma *adjoint-prop1*[*simp*]: (*f o top*) ⊓ (*adjoint f o bot*) = *bot*
⟨*proof*⟩

lemma *adjoint-prop2*[*simp*]: $(f\ o\ top) + (adjoint\ f\ o\ bot) = top$
⟨*proof*⟩

lemma *adjoint-mult*: $adjoint\ (f\ o\ g) = adjoint\ f\ o\ adjoint\ g$
⟨*proof*⟩

lemma *adjoint-top*[*simp*]: $adjoint\ top = bot$
⟨*proof*⟩

lemma *N-comp1*: $(N\ (N\ f)) + N\ f = id$
⟨*proof*⟩

lemma *N-comp2*: $(N\ (N\ f))\ o\ N\ f = bot$
⟨*proof*⟩

lemma *N-comp3*: $N\ f\ o\ (N\ (N\ f)) = bot$
⟨*proof*⟩

lemma *N-de-morgan*: $N\ (N\ f)\ o\ N\ (N\ g) = N\ (N\ f) \sqcap N\ (N\ g)$
⟨*proof*⟩

lemma *conj-pred-aux* [*simp*]: $(\lambda p.\ x\ p \cup y\ p) = y \Longrightarrow \forall p.\ x\ p \subseteq y\ p$
⟨*proof*⟩

Next, we define a type for conjuctive or multiplicative predicate transformers.

typedef $'a\ bool\text{-}op = \{f{::}'a\ bfun.\ (\forall g\ h.\ mono\ f \wedge f \circ g + f \circ h = f \circ (g + h) \wedge bot\ o\ f = bot)\}$
⟨*proof*⟩

setup-lifting *type-definition-bool-op*

Conjuctive predicate transformers form a dioid with tests without right annihilator.

instantiation *bool-op* :: (*type*) *dioid-one-zerol*
begin
lift-definition *less-eq-bool-op* :: $'a\ bool\text{-}op \Rightarrow 'a\ bool\text{-}op \Rightarrow bool$ **is** *fun-order*
⟨*proof*⟩

lift-definition *less-bool-op* :: $'a\ bool\text{-}op \Rightarrow 'a\ bool\text{-}op \Rightarrow bool$ **is** $(<.)$ ⟨*proof*⟩

lift-definition *zero-bool-op* :: $'a\ bool\text{-}op$ **is** *bot*
⟨*proof*⟩

lift-definition *one-bool-op* :: $'a\ bool\text{-}op$ **is** *id*
⟨*proof*⟩

lift-definition *times-bool-op* :: $'a\ bool\text{-}op \Rightarrow 'a\ bool\text{-}op \Rightarrow 'a\ bool\text{-}op$ **is** (o)
⟨*proof*⟩

lift-definition *plus-bool-op* :: *'a bool-op* ⇒ *'a bool-op* ⇒ *'a bool-op* **is** (+)
⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation *bool-op* :: (*type*) *test-semiring-zerol*
begin
lift-definition *n-op-bool-op* :: *'a bool-op* ⇒ *'a bool-op* **is** *N*
⟨*proof*⟩

instance
⟨*proof*⟩

end

definition *fun-star* :: *'a bfun* ⇒ *'a bfun* **where**
fun-star f = *lfp* (λ*r*. *f o r* + *id*)

definition *fun-iteration* :: *'a bfun* ⇒ *'a bfun* **where**
fun-iteration f = *gfp* (λ*g*. *f o g* + *id*)

Verifying the iteration laws is left for future work. This could be obtained by integrating Preoteasa's approach [4].

end

6 Models for Kleene Algebra with Tests

theory *KAT-Models*
imports *Kleene-Algebra.Kleene-Algebra-Models KAT*
begin

We show that binary relations under the obvious definitions form a Kleene algebra with tests.

interpretation *rel-dioid-tests*: *test-semiring* (∪) (*O*) *Id* {} (⊆) (⊂) λ*x*. *Id* ∩ (− *x*)
⟨*proof*⟩

interpretation *rel-kat*: *kat* (∪) (*O*) *Id* {} (⊆) (⊂) *rtrancl* λ*x*. *Id* ∩ (− *x*)
⟨*proof*⟩

typedef *'a relation* = *UNIV*::*'a rel set* ⟨*proof*⟩

setup-lifting *type-definition-relation*

instantiation *relation* :: (*type*) *kat*
begin

lift-definition *n-op-relation* :: *'a relation* ⇒ *'a relation* **is** λ*x*. *Id* ∩ (− *x*) ⟨*proof*⟩
lift-definition *zero-relation* :: *'a relation* **is** {} ⟨*proof*⟩
lift-definition *star-relation* :: *'a relation* ⇒ *'a relation* **is** *rtrancl* ⟨*proof*⟩
lift-definition *less-eq-relation* :: *'a relation* ⇒ *'a relation* ⇒ *bool* **is** (⊆) ⟨*proof*⟩
lift-definition *less-relation* :: *'a relation* ⇒ *'a relation* ⇒ *bool* **is** (⊂) ⟨*proof*⟩
lift-definition *one-relation* :: *'a relation* **is** *Id* ⟨*proof*⟩
lift-definition *times-relation* :: *'a relation* ⇒ *'a relation* ⇒ *'a relation* **is** (*O*) ⟨*proof*⟩
lift-definition *plus-relation* :: *'a relation* ⇒ *'a relation* ⇒ *'a relation* **is** (∪) ⟨*proof*⟩

instance
⟨*proof*⟩

end

end

7 Transformation Theorem for while Loops

theory *FolkTheorem*
imports *Conway-Tests KAT DRAT*
begin

We prove Kozen's transformation theorem for while loops [3] in a weak setting that unifies previous proofs in Kleene algebra with tests, demonic refinement algebras and a variant of probabilistic Kleene algebra.

context *test-pre-conway*
begin

abbreviation *pres* :: *'a* ⇒ *'a* ⇒ *bool* **where**
pres x y ≡ *y* · *x* = *y* · *x* · *y*

lemma *pres-comp*: *pres y z* ⟹ *pres x z* ⟹ *pres* (*x* · *y*) *z*
⟨*proof*⟩

lemma *test-pres1*: *test p* ⟹ *test q* ⟹ *pres p q*
⟨*proof*⟩

lemma *test-pres2*: *test p* ⟹ *test q* ⟹ *pres x q* ⟹ *pres* (*p* · *x*) *q*
⟨*proof*⟩

lemma *cond-helper1*:
assumes *test p* **and** *pres x p*
shows *p* · (*p* · *x* + !*p* · *y*)† · (*p* · *z* + !*p* · *w*) = *p* · *x*† · *z*

⟨*proof*⟩

lemma *cond-helper2*:
assumes *test p* **and** *pres y* (!*p*)
shows !$p \cdot (p \cdot x + !p \cdot y)^\dagger \cdot (p \cdot z + !p \cdot w) = !p \cdot y^\dagger \cdot w$
⟨*proof*⟩

lemma *cond-distr-var*:
assumes *test p* **and** *test q* **and** *test r*
shows $(q \cdot p + r \cdot !p) \cdot (p \cdot x + !p \cdot y) = q \cdot p \cdot x + r \cdot !p \cdot y$
⟨*proof*⟩

lemma *cond-distr*:
assumes *test p* **and** *test q* **and** *test r*
shows $(p \cdot q + !p \cdot r) \cdot (p \cdot x + !p \cdot y) = p \cdot q \cdot x + !p \cdot r \cdot y$
⟨*proof*⟩

theorem *conditional*:
assumes *test p* **and** *test q* **and** *test r1* **and** *test r2*
and *pres x1 q* **and** *pres y1 q* **and** *pres x2* (!*q*) **and** *pres y2* (!*q*)
shows $(p \cdot q + !p \cdot !q) \cdot (q \cdot x1 + !q \cdot x2) \cdot ((q \cdot r1 + !q \cdot r2) \cdot (q \cdot y1 + !q \cdot y2))^\dagger \cdot !(q \cdot r1 + !q \cdot r2) =$
$(p \cdot q + !p \cdot !q) \cdot (p \cdot x1 \cdot (r1 \cdot y1)^\dagger \cdot !r1 + !p \cdot x2 \cdot (r2 \cdot y2)^\dagger \cdot !r2)$
⟨*proof*⟩

theorem *nested-loops*:
assumes *test p* **and** *test q*
shows $p \cdot x \cdot ((p + q) \cdot (q \cdot y + !q \cdot x))^\dagger \cdot !(p + q) + !p = (p \cdot x \cdot (q \cdot y)^\dagger \cdot !q)^\dagger \cdot !p$
⟨*proof*⟩

lemma *postcomputation*:
assumes *test p* **and** *pres y* (!*p*)
shows $!p \cdot y + p \cdot (p \cdot x \cdot (!p \cdot y + p))^\dagger \cdot !p = (p \cdot x)^\dagger \cdot !p \cdot y$
⟨*proof*⟩

lemma *composition-helper*:
assumes *test p* **and** *test q*
and *pres x p*
shows $p \cdot (q \cdot x)^\dagger \cdot !q \cdot p = p \cdot (q \cdot x)^\dagger \cdot !q$
⟨*proof*⟩

theorem *composition*:
assumes *test p* **and** *test q*
and *pres y p* **and** *pres y* (!*p*)
shows $(p \cdot x)^\dagger \cdot !p \cdot (q \cdot y)^\dagger \cdot !q = !p \cdot (q \cdot y)^\dagger \cdot !q + p \cdot (p \cdot x \cdot (!p \cdot (q \cdot y)^\dagger \cdot !q + p))^\dagger \cdot !p$
⟨*proof*⟩
end

Kleene algebras with tests form pre-Conway algebras, therefore the transformation theorem is valid for KAT as well.

sublocale *kat* $\subseteq$ *pre-conway star* ⟨*proof*⟩

Demonic refinement algebras form pre-Conway algebras, therefore the transformation theorem is valid for DRA as well.

sublocale *drat* $\subseteq$ *pre-conway strong-iteration*
⟨*proof*⟩

We do not currently consider an expansion of probabilistic Kleene algebra.

end

8 Propositional Hoare Logic

theory *PHL-KAT*
imports *KAT Kleene-Algebra.PHL-KA*
begin

We define a class of pre-dioids with notions of assertions, tests and iteration. The above rules of PHL are derivable in that class.

class *at-pre-dioid* = *pre-dioid-one* +
fixes *alpha* :: $'a \Rightarrow 'a$ ($\langle\alpha\rangle$)
and *tau* :: $'a \Rightarrow 'a$ ($\langle\tau\rangle$)
assumes *at-pres*: $\alpha\ x \cdot \tau\ y \leq \tau\ y \cdot \alpha\ x \cdot \tau\ y$
and *as-left-supdistl*: $\alpha\ x \cdot (y + z) \leq \alpha\ x \cdot y + \alpha\ x \cdot z$

begin

Only the conditional and the iteration rule need to be considered (in addition to the export laws. In this context, they no longer depend on external assumptions. The other ones do not depend on any assumptions anyway.

lemma *at-phl-cond*:
assumes $\alpha\ u \cdot \tau\ v \cdot x \leq x \cdot z$ **and** $\alpha\ u \cdot \tau\ w \cdot y \leq y \cdot z$
shows $\alpha\ u \cdot (\tau\ v \cdot x + \tau\ w \cdot y) \leq (\tau\ v \cdot x + \tau\ w \cdot y) \cdot z$
⟨*proof*⟩

lemma *ht-at-phl-cond*:
assumes $\{\!|\alpha\ u \cdot \tau\ v|\!\}\ x\ \{\!|z|\!\}$ **and** $\{\!|\alpha\ u \cdot \tau\ w|\!\}\ y\ \{\!|z|\!\}$
shows $\{\!|\alpha\ u|\!\}\ (\tau\ v \cdot x + \tau\ w \cdot y)\ \{\!|z|\!\}$
⟨*proof*⟩

lemma *ht-at-phl-export1*:
assumes $\{\!|\alpha\ x \cdot \tau\ y|\!\}\ z\ \{\!|w|\!\}$
shows $\{\!|\alpha\ x|\!\}\ \tau\ y \cdot z\ \{\!|w|\!\}$
⟨*proof*⟩

lemma *ht-at-phl-export2*:
assumes $\{\!|x|\!\}\ y\ \{\!|\alpha\ z|\!\}$
shows $\{\!|x|\!\}\ y \cdot \tau\ w\ \{\!|\alpha\ z \cdot \tau\ w|\!\}$
⟨*proof*⟩

end

class *at-it-pre-dioid* = *at-pre-dioid* + *it-pre-dioid*
begin

lemma *at-phl-while*:
assumes $\alpha\ p \cdot \tau\ s \cdot x \leq x \cdot \alpha\ p$
shows $\alpha\ p \cdot (it\ (\tau\ s \cdot x) \cdot \tau\ w) \leq it\ (\tau\ s \cdot x) \cdot \tau\ w \cdot (\alpha\ p \cdot \tau\ w)$
⟨*proof*⟩

lemma *ht-at-phl-while*:
assumes $\{\!|\alpha\ p \cdot \tau\ s|\!\}\ x\ \{\!|\alpha\ p|\!\}$
shows $\{\!|\alpha\ p|\!\}\ it\ (\tau\ s \cdot x) \cdot \tau\ w\ \{\!|\alpha\ p \cdot \tau\ w|\!\}$
⟨*proof*⟩

end

The following statements show that pre-Conway algebras, Kleene algebras with tests and demonic refinement algebras form pre-dioids with test and assertions. This automatically generates propositional Hoare logics for these structures.

sublocale *test-pre-dioid-zerol* < *phl*: *at-pre-dioid* **where** *alpha* = *t* **and** *tau* = *t*
⟨*proof*⟩

sublocale *test-pre-conway* < *phl*: *at-it-pre-dioid* **where** *alpha* = *t* **and** *tau* = *t* **and** *it* = *dagger* ⟨*proof*⟩

sublocale *kat-zerol* < *phl*: *at-it-pre-dioid* **where** *alpha* = *t* **and** *tau* = *t* **and** *it* = *star* ⟨*proof*⟩

context *test-pre-dioid-zerol* **begin**

abbreviation *if-then-else* :: *'a* ⇒ *'a* ⇒ *'a* ⇒ *'a* (‹*if - then - else - fi*› [*64*,*64*,*64*] *63*) **where**
if p then x else y fi ≡ $(p \cdot x + !p \cdot y)$

lemma *phl-n-cond*:
$\{\!|n\ v \cdot n\ w|\!\}\ x\ \{\!|z|\!\} \Longrightarrow \{\!|n\ v \cdot t\ w|\!\}\ y\ \{\!|z|\!\} \Longrightarrow \{\!|n\ v|\!\}\ n\ w \cdot x + t\ w \cdot y\ \{\!|z|\!\}$
⟨*proof*⟩

lemma *phl-test-cond*:
assumes *test p* **and** *test b*
and $\{\!|p \cdot b|\!\}\ x\ \{\!|q|\!\}$ **and** $\{\!|p \cdot !b|\!\}\ y\ \{\!|q|\!\}$

shows $\{\!|p|\!\}\ b \cdot x + !b \cdot y\ \{\!|q|\!\}$
⟨*proof*⟩

lemma *phl-cond-syntax*:
assumes *test p* **and** *test b*
and $\{\!|p \cdot b|\!\}\ x\ \{\!|q|\!\}$ **and** $\{\!|p \cdot !b|\!\}\ y\ \{\!|q|\!\}$
shows $\{\!|p|\!\}$ *if b then x else y fi* $\{\!|q|\!\}$
⟨*proof*⟩

lemma *phl-cond-syntax-iff*:
assumes *test p* **and** *test b*
shows $\{\!|p \cdot b|\!\}\ x\ \{\!|q|\!\} \wedge \{\!|p \cdot !b|\!\}\ y\ \{\!|q|\!\} \longleftrightarrow \{\!|p|\!\}$ *if b then x else y fi* $\{\!|q|\!\}$
⟨*proof*⟩

end

context *test-pre-conway*
begin

lemma *phl-n-while*:
assumes $\{\!|n\ x \cdot\ \ n\ y|\!\}\ z\ \{\!|n\ x|\!\}$
shows $\{\!|n\ x|\!\}\ (n\ y \cdot z)^{\dagger} \cdot t\ y\ \{\!|n\ x \cdot t\ y|\!\}$
⟨*proof*⟩

end

context *kat-zerol*
begin

abbreviation *while* :: *'a* ⇒ *'a* ⇒ *'a* (‹*while - do - od*› [*64*,*64*] *63*) **where**
while b do x od ≡ $(b \cdot x)^{\star} \cdot !b$

lemma *phl-n-while*:
assumes $\{\!|n\ x \cdot\ \ n\ y|\!\}\ z\ \{\!|n\ x|\!\}$
shows $\{\!|n\ x|\!\}\ (n\ y \cdot z)^{\star} \cdot t\ y\ \{\!|n\ x \cdot t\ y|\!\}$
⟨*proof*⟩

lemma *phl-test-while*:
assumes *test p* **and** *test b*
and $\{\!|p \cdot b|\!\}\ x\ \{\!|p|\!\}$
shows $\{\!|p|\!\}\ (b \cdot x)^{\star} \cdot !b\ \{\!|p \cdot !b|\!\}$
⟨*proof*⟩

lemma *phl-while-syntax*:
assumes *test p* **and** *test b* **and** $\{\!|p \cdot b|\!\}\ x\ \{\!|p|\!\}$
shows $\{\!|p|\!\}$ *while b do x od* $\{\!|p \cdot !b|\!\}$
⟨*proof*⟩

end

lemma (**in** *kat*) *test p* $\Longrightarrow$ $p \cdot x^\star \leq x^\star \cdot p \Longrightarrow p \cdot x \leq x \cdot p$
⟨*proof*⟩

lemma (**in** *kat*) *test p* $\Longrightarrow$ *test b* $\Longrightarrow$ $p \cdot (b \cdot x)^\star \cdot\, !b \leq (b \cdot x)^\star \cdot\, !b \cdot p \cdot\, !b \Longrightarrow p \cdot b \cdot x \leq x \cdot p$
⟨*proof*⟩

lemma (**in** *kat*) *test p* $\Longrightarrow$ *test q* $\Longrightarrow$ $p \cdot x \cdot y \leq x \cdot y \cdot q \Longrightarrow (\exists\, r.\ test\ r \wedge p \cdot x \leq x \cdot r \wedge r \cdot y \leq y \cdot q)$
⟨*proof*⟩

lemma (**in** *kat*) *test p* $\Longrightarrow$ *test q* $\Longrightarrow$ $p \cdot x \cdot y \cdot\, !q = 0 \Longrightarrow (\exists\, r.\ test\ r \wedge p \cdot x \cdot\, !r = 0 \wedge r \cdot y \cdot\, !q = 0)$
⟨*proof*⟩

The following facts should be moved. They show that the rules of Hoare logic based on Tarlecki triples are invertible.

abbreviation (**in** *near-dioid*) *tt* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ ($\langle(\!|\text{-}|\!)\text{-}(\!|\text{-}|\!)\rangle$) **where**
$(\!|x|\!)\ y\ (\!|z|\!) \equiv x \cdot y \leq z$

lemma (**in** *near-dioid-one*) *tt-skip*: $(\!|p|\!)\ 1\ (\!|p|\!)$
⟨*proof*⟩

lemma (**in** *near-dioid*) *tt-cons1*: $(\exists\, q'.\ (\!|p|\!)\ x\ (\!|q'|\!) \wedge q' \leq q) \longleftrightarrow (\!|p|\!)\ x\ (\!|q|\!)$
⟨*proof*⟩

lemma (**in** *near-dioid*) *tt-cons2*: $(\exists\, p'.\ (\!|p'|\!)\ x\ (\!|q|\!) \wedge p \leq p') \longleftrightarrow (\!|p|\!)\ x\ (\!|q|\!)$
⟨*proof*⟩

lemma (**in** *near-dioid*) *tt-seq*: $(\exists\, r.\ (\!|p|\!)\ x\ (\!|r|\!) \wedge (\!|r|\!)\ y\ (\!|q|\!)) \longleftrightarrow (\!|p|\!)\ x \cdot y\ (\!|q|\!)$
⟨*proof*⟩

lemma (**in** *dioid*) *tt-cond*: $(\!|p \cdot v|\!)\ x\ (\!|q|\!) \wedge (\!|p \cdot w|\!)\ y\ (\!|q|\!) \longleftrightarrow (\!|p|\!)\ (v \cdot x + w \cdot y)(\!|q|\!)$
⟨*proof*⟩

lemma (**in** *kleene-algebra*) *tt-while*: $w \leq 1 \Longrightarrow (\!|p \cdot v|\!)\ x\ (\!|p|\!) \Longrightarrow (\!|p|\!)\ (v \cdot x)^\star \cdot w\ (\!|p|\!)$
⟨*proof*⟩

The converse implication can be refuted. The situation is similar to the ht case.

lemma (**in** *kat*) *tt-while*: *test v* $\Longrightarrow$ $(\!|p|\!)\ (v \cdot x)^\star \cdot\, !v\ (\!|p|\!) \Longrightarrow (\!|p \cdot v|\!)\ x\ (\!|p|\!)$
⟨*proof*⟩

lemma (**in** *kat*) *tt-while*: *test v* $\Longrightarrow$ $(\!|p|\!)\ (v \cdot x)^\star\ (\!|p|\!) \Longrightarrow (\!|p \cdot v|\!)\ x\ (\!|p|\!)$
⟨*proof*⟩

Perhaps this holds with possibly infinite loops in DRA...

wlp in KAT

lemma (**in** *kat*) *test y* $\Longrightarrow$ ($\exists$ *z*. *test z* $\wedge$ *z* $\cdot$ *x* $\cdot$!*y* = *0*)
⟨*proof*⟩

end

9 Propositional Hoare Logic

theory *PHL-DRAT*
imports *DRAT Kleene-Algebra.PHL-DRA PHL-KAT*
begin

sublocale *drat* < *phl*: *at-it-pre-dioid* **where** *alpha* = *t* **and** *tau* = *t* **and** *it* = *strong-iteration* ⟨*proof*⟩

context *drat*
begin

no-notation *while* (‹*while* - *do* - *od*› [*64*,*64*] *63*)

abbreviation *while* :: *'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a* (‹*while* - *do* - *od*› [*64*,*64*] *63*) **where**
while b do x od $\equiv$ (*b* $\cdot$ *x*)$^\infty$ $\cdot$!*b*

lemma *phl-n-while*:
assumes {|*n x* $\cdot$ *n y*|} *z* {|*n x*|}
shows {|*n x*|} (*n y* $\cdot$ *z*)$^\infty$ $\cdot$ *t y* {|*n x* $\cdot$ *t y*|}
⟨*proof*⟩

lemma *phl-test-while*:
assumes *test p* **and** *test b*
and {|*p* $\cdot$ *b*|} *x* {|*p*|}
shows {|*p*|} (*b* $\cdot$ *x*)$^\infty$ $\cdot$!*b* {|*p* $\cdot$!*b*|}
⟨*proof*⟩

lemma *phl-while-syntax*:
assumes *test p* **and** *test b* **and** {|*p* $\cdot$ *b*|} *x* {|*p*|}
shows {|*p*|} *while b do x od* {|*p* $\cdot$!*b*|}
⟨*proof*⟩

end

end

10 Two sorted Kleene Algebra with Tests

theory *KAT2*
imports *Kleene-Algebra.Kleene-Algebra*
begin

As an alternative to the one-sorted implementation of tests, we provide a two-sorted, more conventional one. In this setting, Isabelle's Boolean algebra theory can be used. This alternative can be developed further along the lines of the one-sorted implementation.

syntax *-kat* :: *'a* ⇒ *'a* (‹ ‘-‘›)

⟨*ML*⟩

locale *dioid-tests* =
  fixes *test* :: *'a*::*boolean-algebra* ⇒ *'b*::*dioid-one-zerol*
  and *not* :: *'b*::*dioid-one-zerol* ⇒ *'b*::*dioid-one-zerol* (‹−›)
  assumes *test-sup* [*simp*,*kat-hom*]: *test* (*sup p q*) = ‘*p* + *q*‘
  and *test-inf* [*simp*,*kat-hom*]: *test* (*inf p q*) = ‘*p* · *q*‘
  and *test-top* [*simp*,*kat-hom*]: *test top* = *1*
  and *test-bot* [*simp*,*kat-hom*]: *test bot* = *0*
  and *test-not* [*simp*,*kat-hom*]: *test* (− *p*) = ‘−*p*‘
  and *test-iso-eq* [*kat-hom*]: *p* ≤ *q* ⟷ ‘*p* ≤ *q*‘
begin

notation *test* (‹ι›)

lemma *test-eq* [*kat-hom*]: *p* = *q* ⟷ ‘*p* = *q*‘
  ⟨*proof*⟩

⟨*ML*⟩

lemma *test-iso*: *p* ≤ *q* ⟹ ‘*p* ≤ *q*‘
  ⟨*proof*⟩

lemma *test-meet-comm*: ‘*p* · *q* = *q* · *p*‘
  ⟨*proof*⟩

lemmas *test-one-top*[*simp*] = *test-iso*[*OF top-greatest*, *simplified*]

lemma [*simp*]: ‘−*p* + *p* = *1*‘
  ⟨*proof*⟩

lemma [*simp*]: ‘*p* + (−*p*) = *1*‘
  ⟨*proof*⟩

lemma [*simp*]: ‘(−*p*) · *p* = *0*‘
  ⟨*proof*⟩

lemma [*simp*]: ‘*p* · (−*p*) = *0*‘
  ⟨*proof*⟩

end

locale *kat* =
  fixes *test* :: *'a*::*boolean-algebra* ⇒ *'b*::*kleene-algebra*
  and *not* :: *'b*::*kleene-algebra* ⇒ *'b*::*kleene-algebra* (‹!›)
  assumes *is-dioid-tests*: *dioid-tests test not*

sublocale *kat* ⊆ *dioid-tests* ⟨*proof*⟩

context *kat*
begin

notation *test* (‹ι›)

lemma *test-eq* [*kat-hom*]: *p* = *q* ⟷ *‘p* = *q‘*
  ⟨*proof*⟩

⟨*ML*⟩

lemma *test-iso*: *p* ≤ *q* ⟹ *‘p* ≤ *q‘*
  ⟨*proof*⟩

lemma *test-meet-comm*: *‘p* · *q* = *q* · *p‘*
  ⟨*proof*⟩

lemmas *test-one-top*[*simp*] = *test-iso*[*OF top-greatest*, *simplified*]

lemma *test-star* [*simp*]: *‘p*⋆ = *1‘*
  ⟨*proof*⟩

lemmas [*kat-hom*] = *test-star*[*symmetric*]

lemma [*simp*]: *‘*¬*p* + *p* = *1‘*
  ⟨*proof*⟩

lemma [*simp*]: *‘p* + ¬*p* = *1‘*
  ⟨*proof*⟩

lemma [*simp*]: *‘*¬*p* · *p* = *0‘*
  ⟨*proof*⟩

lemma [*simp*]: *‘p* · ¬*p* = *0‘*
  ⟨*proof*⟩

definition *hoare-triple* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool* (‹{|-|} - {|-|}›) **where**
  {|*p*|} *c* {|*q*|} ≡ *p*·*c* ≤ *c*·*q*

declare *hoare-triple-def*[*iff*]

lemma *hoare-triple-def-var*: ‘$p{\cdot}c \leq c{\cdot}q \longleftrightarrow p{\cdot}c{\cdot}!q = 0$‘
⟨*proof*⟩

lemmas [*intro*!] = *star-sim2*[*rule-format*]

lemma *hoare-weakening*: $p \leq p' \Longrightarrow q' \leq q \Longrightarrow$ ‘$\{\!|p'|\!\}\ c\ \{\!|q'|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ c\ \{\!|q|\!\}$‘
⟨*proof*⟩

lemma *hoare-star*: ‘$\{\!|p|\!\}\ c\ \{\!|p|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ c^\star\ \{\!|p|\!\}$‘
⟨*proof*⟩

lemmas [*vcg*] = *hoare-weakening*[*OF order-refl - hoare-star*]

lemma *hoare-test* [*vcg*]: ‘$p \cdot t \leq q$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ t\ \{\!|q|\!\}$‘
⟨*proof*⟩

lemma *hoare-mult* [*vcg*]: ‘$\{\!|p|\!\}\ x\ \{\!|r|\!\}$‘ $\Longrightarrow$ ‘$\{\!|r|\!\}\ y\ \{\!|q|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ x{\cdot}y\ \{\!|q|\!\}$‘
⟨*proof*⟩

lemma [*simp*]: ‘$!p \cdot !p = !p$‘
⟨*proof*⟩

lemma *hoare-plus* [*vcg*]: ‘$\{\!|p|\!\}\ x\ \{\!|q|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ y\ \{\!|q|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}\ x + y\ \{\!|q|\!\}$‘
⟨*proof*⟩

definition *While* :: $'b \Rightarrow 'b \Rightarrow 'b$ (‹*While - Do - End*› [*50*,*50*] *51*) **where**
While t Do c End = $(t{\cdot}c)^\star{\cdot}!t$

lemma *hoare-while*: ‘$\{\!|p \cdot t|\!\}\ c\ \{\!|p|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}$ *While t Do c End* $\{\!|!t \cdot p|\!\}$‘
⟨*proof*⟩

lemma [*vcg*]: ‘$\{\!|p \cdot t|\!\}\ c\ \{\!|p|\!\}$‘ $\Longrightarrow$ ‘$!t \cdot p \leq q$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}$ *While t Do c End* $\{\!|q|\!\}$‘
⟨*proof*⟩

definition *If* :: $'b \Rightarrow 'b \Rightarrow 'b \Rightarrow 'b$ (‹*If - Then - Else -*› [*50*,*50*,*50*] *51*) **where**
If p Then c1 Else c2 ≡ $p{\cdot}c1 + !p{\cdot}c2$

lemma *hoare-if* [*vcg*]: ‘$\{\!|p \cdot t|\!\}$ *c1* $\{\!|q|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p \cdot !t|\!\}$ *c2* $\{\!|q|\!\}$‘ $\Longrightarrow$ ‘$\{\!|p|\!\}$ *If t Then c1 Else c2* $\{\!|q|\!\}$‘
⟨*proof*⟩

end

end

11 Two sorted Demonic Refinement Algebras

theory *DRAT2*

imports *Kleene-Algebra.DRA*
begin

As an alternative to the one-sorted implementation of demonic refinement algebra with tests, we provide a two-sorted, more conventional one. This alternative can be developed further along the lines of the one-sorted implementation.

syntax *-dra* :: *'a* ⇒ *'a* (‹ ‘-‘›)

⟨*ML*⟩

locale *drat* =
  fixes *test* :: *'a*::*boolean-algebra* ⇒ *'b*::*dra*
  and *not* :: *'b*::*dra* ⇒ *'b*::*dra* (‹!›)
  assumes *test-sup* [*simp*,*kat-hom*]: *test* (*sup p q*) = ‘*p* + *q*‘
  and *test-inf* [*simp*,*kat-hom*]: *test* (*inf p q*) = ‘*p* · *q*‘
  and *test-top* [*simp*,*kat-hom*]: *test top* = *1*
  and *test-bot* [*simp*,*kat-hom*]: *test bot* = *0*
  and *test-not* [*simp*,*kat-hom*]: *test* (− *p*) = ‘!*p*‘
  and *test-iso-eq* [*kat-hom*]: *p* ≤ *q* ⟷ ‘*p* ≤ *q*‘

begin

notation *test* (‹*ι*›)

lemma *test-eq* [*kat-hom*]: *p* = *q* ⟷ ‘*p* = *q*‘
  ⟨*proof*⟩

⟨*ML*⟩

lemma *test-iso*: *p* ≤ *q* ⟹ ‘*p* ≤ *q*‘
  ⟨*proof*⟩

lemma *test-meet-comm*: ‘*p* · *q* = *q* · *p*‘
  ⟨*proof*⟩

lemmas *test-one-top*[*simp*] = *test-iso*[*OF top-greatest*, *simplified*]

lemma *test-star* [*simp*]: ‘*p*⋆ = *1*‘
  ⟨*proof*⟩

lemmas [*kat-hom*] = *test-star*[*symmetric*]

lemma *test-comp-add1* [*simp*]: ‘!*p* + *p* = *1*‘
  ⟨*proof*⟩

lemma *test-comp-add2* [*simp*]: ‘*p* + !*p* = *1*‘
  ⟨*proof*⟩

lemma *test-comp-mult1* [*simp*]: ‘!$p \cdot p = 0$‘
⟨*proof*⟩

lemma *test-comp-mult2* [*simp*]: ‘$p \cdot !p = 0$‘
⟨*proof*⟩

lemma *test-eq1*: ‘$y \leq x$‘ ⟷ ‘$p \cdot y \leq x$‘ ∧ ‘!$p \cdot y \leq x$‘
⟨*proof*⟩

lemma ‘$p \cdot x = p \cdot x \cdot q$‘ ⟹ ‘$p \cdot x \cdot !q = 0$‘
nitpick ⟨*proof*⟩

lemma *test1*: ‘$p \cdot x \cdot !q = 0$‘ ⟹ ‘$p \cdot x = p \cdot x \cdot q$‘
⟨*proof*⟩

lemma *test2*: ‘$p \cdot q \cdot p = p \cdot q$‘
⟨*proof*⟩

lemma *test3*: ‘$p \cdot q \cdot !p = 0$‘
⟨*proof*⟩

lemma *test4*: ‘!$p \cdot q \cdot p = 0$‘
⟨*proof*⟩

lemma *total-correctness*: ‘$p \cdot x \cdot !q = 0$‘ ⟷ ‘$x \cdot !q \leq !p \cdot \top$‘
⟨*proof*⟩

lemma *test-iteration-sim*: ‘$p \cdot x \leq x \cdot p$‘ ⟹ ‘$p \cdot x^\infty \leq x^\infty \cdot p$‘
⟨*proof*⟩

lemma *test-iteration-annir*: ‘!$p \cdot (p \cdot x)^\infty = !p$‘
⟨*proof*⟩

end

end

References

[1] E. Cohen. Separation and reduction. In R. C. Backhouse and J. N. Oliveira, editors, *MPC*, volume 1837 of *LNCS*, pages 45–59. Springer, 2000.

[2] P. Höfner, G. Struth, and G. Sutcliffe. Automated verification of refinement laws. *Ann. Mathematics and Artificial Intelligence*, 55(1-2):35–62, 2009.

[3] D. Kozen. Kleene algebra with tests. *ACM TOPLAS*, 19(3):427–443, 1997.

[4] V. Preoteasa. Algebra of monotonic boolean transformers. In A. S. Simão and C. Morgan, editors, *SBMF*, volume 7021 of *LNCS*, pages 140–155. Springer, 2011.

[5] J. von Wright. From Kleene algebra to refinement algebra. In E. A. Boiten and B. Möller, editors, *MPC*, volume 2386 of *LNCS*, pages 233–262. Springer, 2002.