

# Kleene Algebra with Tests and Demonic Refinement Algebras

Alasdair Armstrong    Victor B. F. Gomes    Georg Struth

October 10, 2017

## 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

<b>1</b>	<b>Test Dioids</b>	<b>2</b>
1.1	Test Monoids . . . . .	2
1.2	Test Near-Semirings . . . . .	6
1.3	Test Near Semirings with Distributive Tests . . . . .	12
1.4	Test Pre-dioids . . . . .	13
1.5	Test Semirings . . . . .	15
<b>2</b>	<b>Pre-Conway Algebra with Tests</b>	<b>16</b>
<b>3</b>	<b>Kleene Algebra with Tests</b>	<b>17</b>
<b>4</b>	<b>Demonic Refinement Algebra with Tests</b>	<b>18</b>
<b>5</b>	<b>Models for Demonic Refinement Algebra with Tests</b>	<b>20</b>
<b>6</b>	<b>Models for Kleene Algebra with Tests</b>	<b>23</b>
<b>7</b>	<b>Transformation Theorem for while Loops</b>	<b>24</b>
<b>8</b>	<b>Propositional Hoare Logic</b>	<b>26</b>
<b>9</b>	<b>Propositional Hoare Logic</b>	<b>30</b>

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  $\Rightarrow$  'a (n - [90] 91)

class test-monoid = monoid-mult + n-op +
  assumes tm1 [simp]: n n 1 = 1
  and tm2 [simp]: n x  $\cdot$  n n x = n 1
  and 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))

begin

```

```

definition a-zero :: 'a (o) where
  o  $\equiv$  n 1

```

```

abbreviation t x  $\equiv$  n n x

```

```

definition n-add-op :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (infixl  $\oplus$  65) where
  x  $\oplus$  y  $\equiv$  n (n x  $\cdot$  n y)

```

```

lemma n 1  $\cdot$  x = n 1
  <proof>

```

```

lemma x  $\cdot$  n 1 = n 1
  <proof>

```

```

lemma n 1  $\cdot$  x = n 1  $\implies$  n x  $\cdot$  y  $\cdot$  t z = n 1  $\implies$  n x  $\cdot$  y = n x  $\cdot$  y  $\cdot$  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*:  $t x = x \implies t y = y \implies t (x \cdot y) = x \cdot y$   
*<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*:  $n x \sqsubseteq n y \implies n x \oplus n z \sqsubseteq n y \oplus n z$   
*<proof>*

**lemma** *ts-ord-anti*:  $n x \sqsubseteq n y \implies t y \sqsubseteq t x$   
*<proof>*

**lemma** *ts-ord-anti-iff*:  $n x \sqsubseteq n y \iff t y \sqsubseteq t x$   
*<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*:  $n z \sqsubseteq n x \implies n z \sqsubseteq n y \implies n z \sqsubseteq n x \cdot n y$   
*<proof>*

**lemma** *n-mult-glb*:  $n z \sqsubseteq n x \wedge n z \sqsubseteq n y \iff n z \sqsubseteq n x \cdot n y$   
*<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*:  $n x \sqsubseteq n z \implies n y \sqsubseteq n z \implies n x \oplus n y \sqsubseteq n z$   
*<proof>*

**lemma** *n-add-lub*:  $n x \sqsubseteq n z \wedge n y \sqsubseteq n z \iff n x \oplus n y \sqsubseteq n z$   
*<proof>*

**lemma** *n-galois1*:  $n x \sqsubseteq n y \oplus n z \iff n x \cdot t y \sqsubseteq n z$   
*<proof>*

**lemma** *n-galois2*:  $n x \sqsubseteq t y \oplus n z \iff n x \cdot n y \sqsubseteq n z$   
*<proof>*

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

*<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-zero1* = *ab-near-semiring-one-zero1* + *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-zero1*  
*<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*:  $(\forall x. \text{test } x \longrightarrow P\ x) \longleftrightarrow (\forall x. P\ (t\ x))$   
*<proof>*

**lemma** *test-propI*:  $\text{test } x \Longrightarrow P\ x \Longrightarrow P\ (t\ x)$   
*<proof>*

**lemma** *test-propE* [*elim!*]:  $\text{test } x \Longrightarrow P\ (t\ x) \Longrightarrow P\ x$   
*<proof>*

**lemma** *test-comp-closed* [*simp*]:  $\text{test } p \Longrightarrow \text{test } (!p)$   
*<proof>*

**lemma** *test-double-comp-var*:  $\text{test } p \Longrightarrow p = (!p)$   
*<proof>*

**lemma** *test-mult-closed*:  $\text{test } p \Longrightarrow \text{test } q \Longrightarrow \text{test } (p \cdot q)$   
*<proof>*

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

**lemma** *test-add-closed*:  $\text{test } p \Longrightarrow \text{test } q \Longrightarrow \text{test } (p + q)$   
*<proof>*

**lemma** *test-mult-comm-var*:  $\text{test } p \Longrightarrow \text{test } q \Longrightarrow p \cdot q = q \cdot p$   
*<proof>*

**lemma** *t-zero* [*simp*]:  $t\ 0 = 0$   
*<proof>*

**lemma** *test-zero-var*:  $\text{test } 0$   
*<proof>*

**lemma** *test-one-var*:  $\text{test } 1$   
*<proof>*

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

$\langle proof \rangle$

**lemma** *test-preserve2*:  $test\ p \implies test\ q \implies p \cdot q \cdot !p = 0$   
 $\langle proof \rangle$

**lemma** *n-subid'*:  $n\ x \leq 1$   
 $\langle proof \rangle$

**lemma** *test-subid*:  $test\ p \implies p \leq 1$   
 $\langle proof \rangle$

**lemma** *test-mult-idem-var* [*simp*]:  $test\ p \implies p \cdot p = p$   
 $\langle proof \rangle$

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

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

**lemma** *test-add-comp* [*simp*]:  $test\ p \implies p + !p = 1$   
 $\langle proof \rangle$

**lemma** *test-comp-mult1* [*simp*]:  $test\ p \implies !p \cdot p = 0$   
 $\langle proof \rangle$

**lemma** *test-comp-mult2* [*simp*]:  $test\ p \implies p \cdot !p = 0$   
 $\langle proof \rangle$

**lemma** *test-comp*:  $test\ p \implies \exists q. test\ q \wedge p + q = 1 \wedge p \cdot q = 0$   
 $\langle proof \rangle$

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

**lemma** *test-absorb1* [*simp*]:  $test\ p \implies test\ q \implies p + p \cdot q = p$   
 $\langle proof \rangle$

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

**lemma** *test-absorb2*:  $test\ p \implies test\ q \implies p \cdot (p + q) = p$   
 $\langle proof \rangle$

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

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



$\cdot r$   
 $\langle proof \rangle$

**lemma** *de-morgan1'*:  $test\ p \implies test\ q \implies !p + !q = !(p \cdot q)$   
 $\langle proof \rangle$

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

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

**lemma** *de-morgan2*:  $test\ p \implies test\ q \implies !p \cdot !q = !(p + q)$   
 $\langle proof \rangle$

**lemma** *de-morgan3*:  $test\ p \implies test\ q \implies !(!p + !q) = p \cdot q$   
 $\langle proof \rangle$

**lemma** *de-morgan4'*:  $test\ p \implies test\ q \implies !(!p \cdot !q) = p + q$   
 $\langle proof \rangle$

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

**lemma** *test-add-distr*:  $test\ p \implies test\ q \implies test\ r \implies p + q \cdot r = (p + q) \cdot (p + r)$   
 $\langle proof \rangle$

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

**lemma** *test-add-distl-var*:  $test\ p \implies test\ q \implies test\ r \implies p \cdot q + r = (p + r) \cdot (q + r)$   
 $\langle proof \rangle$

**lemma** *n-ord-def-alt*:  $n\ x \leq n\ y \iff n\ x \cdot n\ y = n\ x$   
 $\langle proof \rangle$

**lemma** *test-leq-mult-def-var*:  $test\ p \implies test\ q \implies p \leq q \iff p \cdot q = p$   
 $\langle proof \rangle$

**lemma** *n-anti*:  $n\ x \leq n\ y \implies t\ y \leq t\ x$   
 $\langle proof \rangle$

**lemma** *n-anti-iff*:  $n\ x \leq n\ y \iff t\ y \leq t\ x$   
 $\langle proof \rangle$

**lemma** *test-comp-anti-iff*:  $test\ p \implies test\ q \implies p \leq q \iff !q \leq !p$   
 $\langle proof \rangle$

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

*<proof>*

**lemma** *test-restrictl*:  $test p \implies p \cdot x \leq x$

*<proof>*

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

*<proof>*

**lemma** *test-mult-lb1*:  $test p \implies test q \implies p \cdot q \leq p$

*<proof>*

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

*<proof>*

**lemma** *test-mult-lb2*:  $test p \implies test q \implies p \cdot q \leq q$

*<proof>*

**lemma** *n-mult-glbI'*:  $n z \leq n x \implies n z \leq n y \implies n z \leq n x \cdot n y$

*<proof>*

**lemma** *test-mult-glbI*:  $test p \implies test q \implies test r \implies p \leq q \implies p \leq r \implies p \leq q \cdot r$

*<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*:  $test p \implies test q \implies test r \implies p \leq q \wedge p \leq r \longleftrightarrow p \leq q \cdot r$

*<proof>*

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

*<proof>*

**lemma** *test-galois1*:  $test p \implies test q \implies test r \implies p \leq q + r \longleftrightarrow p \cdot !q \leq r$

*<proof>*

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

*<proof>*

**lemma** *test-galois2*:  $test p \implies test q \implies test r \implies p \leq !q + r \longleftrightarrow p \cdot q \leq r$

*<proof>*

**lemma** *n-huntington2*:  $n (n x + t y) + n (n x + n y) = n n x$

*<proof>*

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

$\langle proof \rangle$

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

**lemma** *test-eq4*:  $test p \implies test q \implies !p \cdot x \cdot !q = x \cdot !q \longleftrightarrow p \cdot x \cdot !q = 0$   
 $\langle proof \rangle$

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

**lemma** *test-kat-1*:  $test p \implies test q \implies p \cdot x \cdot q = x \cdot q \longleftrightarrow !p \cdot x \cdot q = 0$   
 $\langle proof \rangle$

**lemma** *n-kat-21-opp*:  $y \cdot n z \leq n x \cdot y \implies n x \cdot y \cdot n z = y \cdot n z$   
 $\langle proof \rangle$

**lemma** *test-kat-21-opp*:  $test p \implies test q \implies x \cdot q \leq p \cdot x \longrightarrow p \cdot x \cdot q = x \cdot q$   
 $\langle proof \rangle$

**lemma**  $n x \cdot y \cdot n z = y \cdot n z \implies y \cdot n z \leq n x \cdot y$   
 $\langle proof \rangle$

**lemma** *n-distrib-alt'*:  $n x \cdot n z = n y \cdot n z \implies n x + n z = n y + n z \implies n x = n y$   
 $\langle proof \rangle$

**lemma** *test-distrib-alt*:  $test p \implies test q \implies test r \implies p \cdot r = q \cdot r \wedge p + r = q + r \longrightarrow p = q$   
 $\langle proof \rangle$

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

**lemma** *test-eq1*:  $test p \implies y \leq x \longleftrightarrow p \cdot y \leq x \wedge !p \cdot y \leq x$   
 $\langle proof \rangle$

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

**lemma** *test-dist-var1*:  $test p \implies test q \implies test r \implies (p + q) \cdot (!p + r) = !p \cdot q + p \cdot r$   
 $\langle proof \rangle$

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

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

$\langle proof \rangle$

**lemma** *test-restrictr*:  $test\ p \implies x \cdot p \leq x$   
 $\langle proof \rangle$

**lemma** *test-eq2*:  $test\ p \implies z \leq p \cdot x + !p \cdot y \iff p \cdot z \leq p \cdot x \wedge !p \cdot z \leq !p \cdot y$   
 $\langle proof \rangle$

**lemma** *test-eq3*:  $\llbracket test\ p; test\ q \rrbracket \implies p \cdot x = p \cdot x \cdot q \iff p \cdot x \leq x \cdot q$   
 $\langle proof \rangle$

**lemma** *test1*:  $\llbracket test\ p; test\ q; p \cdot x \cdot !q = 0 \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
 $\langle proof \rangle$

**lemma**  $\llbracket test\ p; test\ q; x \cdot !q = !p \cdot x \cdot !q \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
 $\langle proof \rangle$

**lemma** *comm-add*:  $\llbracket test\ p; p \cdot x = x \cdot p; p \cdot y = y \cdot p \rrbracket \implies p \cdot (x + y) = (x + y) \cdot p$   
 $\langle proof \rangle$

**lemma** *comm-add-var*:  $\llbracket test\ p; test\ q; test\ r; p \cdot x = x \cdot p; p \cdot y = y \cdot p \rrbracket \implies p \cdot (q \cdot x + r \cdot y) = (q \cdot x + r \cdot y) \cdot p$   
 $\langle proof \rangle$

**lemma** *test*  $p \implies p \cdot x = x \cdot p \implies p \cdot x = p \cdot x \cdot p \wedge !p \cdot x = !p \cdot x \cdot !p$   
 $\langle proof \rangle$

**lemma** *test-distrib*:  $\llbracket test\ p; test\ q \rrbracket \implies (p + q) \cdot (q \cdot y + !q \cdot x) = q \cdot y + !q \cdot p \cdot x$   
 $\langle proof \rangle$

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

**begin**

**lemma** *n-left-distrib-var*:  $test\ p \implies p \cdot (x + y) = p \cdot x + p \cdot y$   
 $\langle proof \rangle$

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

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

*<proof>*

**lemma** *test p*  $\implies x \cdot p \leq x$   
*<proof>*

**lemma**  $\llbracket \text{test } p; \text{test } q \rrbracket \implies p \cdot x = p \cdot x \cdot q \iff p \cdot x \leq x \cdot q$   
*<proof>*

**lemma**  $\llbracket \text{test } p; \text{test } q; p \cdot x \cdot !q = 0 \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

**lemma**  $\llbracket \text{test } p; \text{test } q; x \cdot !q = !p \cdot x \cdot !q \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

Next, we study tests with commutativity conditions.

**lemma** *comm-add*: *test p*  $\implies p \cdot x = x \cdot p \implies p \cdot y = y \cdot p \implies p \cdot (x + y)$   
 $= (x + y) \cdot p$   
*<proof>*

**lemma** *comm-add-var*: *test p*  $\implies \text{test } q \implies \text{test } r \implies p \cdot x = x \cdot p \implies p \cdot y =$   
 $y \cdot p \implies p \cdot (q \cdot x + r \cdot y) = (q \cdot x + r \cdot y) \cdot p$   
*<proof>*

**lemma** *test-distrib*: *test p*  $\implies \text{test } q \implies (p + q) \cdot (q \cdot y + !q \cdot x) = q \cdot y + !q \cdot$   
 $p \cdot x$   
*<proof>*

**end**

## 1.4 Test Preidoids

The following class is relevant for probabilistic Kleene algebras.

**class** *test-pre-diod-zero1* = *test-near-semiring-zero1-distrib* + *pre-diod*

**begin**

**lemma** *n-restrict*:  $x \cdot n y \leq x$   
*<proof>*

**lemma** *test-restrict*: *test p*  $\implies x \cdot p \leq x$   
*<proof>*

**lemma** *n-kat-2*:  $n x \cdot y = n x \cdot y \cdot n z \iff n x \cdot y \leq y \cdot n z$   
*<proof>*

**lemma** *test-kat-2*: *test p*  $\implies \text{test } q \implies p \cdot x = p \cdot x \cdot q \iff p \cdot x \leq x \cdot q$   
*<proof>*

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

*<proof>*

**lemma** *test-kat-2-opp*:  $test\ p \implies test\ q \implies x \cdot q = p \cdot x \cdot q \iff x \cdot q \leq p \cdot x$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; p \cdot x \cdot !q = 0 \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; x \cdot !q = !p \cdot x \cdot !q \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q \rrbracket \implies x \cdot (p + q) \leq x \cdot p + x \cdot q$   
*<proof>*

**end**

The following class is relevant for Demonic Refinement Algebras.

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

**begin**

**subclass** *dioid-one-zero* *<proof>*

**subclass** *test-pre-dioid-zero*  
*<proof>*

**lemma** *n-kat-31*:  $n\ x \cdot y \cdot t\ z = 0 \implies n\ x \cdot y \cdot n\ z = n\ x \cdot y$   
*<proof>*

**lemma** *test-kat-31*:  $test\ p \implies test\ q \implies p \cdot x \cdot !q = 0 \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

**lemma** *n-kat-var*:  $t\ x \cdot y \cdot t\ z = y \cdot t\ z \implies n\ x \cdot y \cdot n\ z = n\ x \cdot y$   
*<proof>*

**lemma** *test1-var*:  $test\ p \implies test\ q \implies x \cdot !q = !p \cdot x \cdot !q \implies p \cdot x = p \cdot x \cdot q$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; p \cdot x \cdot !q = 0 \rrbracket \implies !p \cdot x \cdot q = 0$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; p \cdot x = p \cdot x \cdot q \rrbracket \implies x \cdot !q = !p \cdot x \cdot !q$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; p \cdot x = p \cdot x \cdot q \rrbracket \implies p \cdot x \cdot !q = 0$   
*<proof>*

**lemma**  $\llbracket test\ p; test\ q; p \cdot x = p \cdot x \cdot q \rrbracket \implies !p \cdot x \cdot q = 0$   
*<proof>*

**lemma**  $\llbracket \text{test } p; \text{test } q; x \cdot !q = !p \cdot x \cdot !q \rrbracket \implies !p \cdot x \cdot q = 0$   
 $\langle \text{proof} \rangle$

**lemma**  $\llbracket \text{test } p; \text{test } q; !p \cdot x \cdot q = 0 \rrbracket \implies p \cdot x = p \cdot x \cdot q$   
 $\langle \text{proof} \rangle$

**lemma**  $\llbracket \text{test } p; \text{test } q; !p \cdot x \cdot q = 0 \rrbracket \implies x \cdot !q = !p \cdot x \cdot !q$   
 $\langle \text{proof} \rangle$

**lemma**  $\llbracket \text{test } p; \text{test } q; !p \cdot x \cdot q = 0 \rrbracket \implies p \cdot x \cdot !q = 0$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{test } p \implies p \cdot x = p \cdot x \cdot p \wedge !p \cdot x = !p \cdot x \cdot !p \implies p \cdot x = x \cdot p$   
 $\langle \text{proof} \rangle$

**end**

## 1.5 Test Semirings

The following class is relevant for Kleene Algebra with Tests.

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

**begin**

**lemma** *n-kat-1*:  $n x \cdot y \cdot t z = 0 \iff n x \cdot y \cdot n z = n x \cdot y$   
 $\langle \text{proof} \rangle$

**lemma** *test-kat-1'*:  $\text{test } p \implies \text{test } q \implies p \cdot x \cdot !q = 0 \iff p \cdot x = p \cdot x \cdot q$   
 $\langle \text{proof} \rangle$

**lemma** *n-kat-3*:  $n x \cdot y \cdot t z = 0 \iff n x \cdot y \leq y \cdot n z$   
 $\langle \text{proof} \rangle$

**lemma** *test-kat-3*:  $\text{test } p \implies \text{test } q \implies p \cdot x \cdot !q = 0 \iff p \cdot x \leq x \cdot q$   
 $\langle \text{proof} \rangle$

**lemma** *n-kat-prop*:  $n x \cdot y \cdot n z = n x \cdot y \iff t x \cdot y \cdot t z = y \cdot t z$   
 $\langle \text{proof} \rangle$

**lemma** *test-kat-prop*:  $\text{test } p \implies \text{test } q \implies p \cdot x = p \cdot x \cdot q \iff x \cdot !q = !p \cdot x \cdot !q$   
 $\langle \text{proof} \rangle$

**lemma** *n-kat-3-opp*:  $t x \cdot y \cdot n z = 0 \iff y \cdot n z \leq n x \cdot y$   
 $\langle \text{proof} \rangle$

**lemma** *kat-1-var*:  $n x \cdot y \cdot n z = y \cdot n z \iff y \cdot n z \leq n x \cdot y$   
 $\langle \text{proof} \rangle$

**lemma**  $\llbracket \text{test } p; \text{test } q \rrbracket \implies (p \cdot x \cdot !q = 0) \implies (!p \cdot x \cdot q = 0)$   
 $\langle \text{proof} \rangle$

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

$\langle \text{proof} \rangle$

**end**

**end**

## 2 Pre-Conway Algebra with Tests

**theory** *Conway-Tests*

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

**begin**

**class** *near-conway-zero1-tests* = *near-conway-zero1* + *test-near-semiring-zero1-distrib*

**begin**

**lemma** *n-preserve1-var*:  $n x \cdot y \leq n x \cdot y \cdot n x \implies n x \cdot (n x \cdot y + t x \cdot z)^\dagger \leq (n x \cdot y)^\dagger \cdot n x$   
 $\langle \text{proof} \rangle$

**lemma** *test-preserve1-var*:  $\text{test } p \implies p \cdot x \leq p \cdot x \cdot p \implies p \cdot (p \cdot x + !p \cdot y)^\dagger \leq (p \cdot x)^\dagger \cdot p$   
 $\langle \text{proof} \rangle$

**end**

**class** *test-pre-conway* = *pre-conway* + *test-pre-dioid-zero1*

**begin**

**subclass** *near-conway-zero1-tests*  
 $\langle \text{proof} \rangle$

**lemma** *test-preserve*:  $\text{test } p \implies p \cdot x \leq p \cdot x \cdot p \implies p \cdot x^\dagger = (p \cdot x)^\dagger \cdot p$   
 $\langle \text{proof} \rangle$

**lemma** *test-preserve1*:  $\text{test } p \implies p \cdot x \leq p \cdot x \cdot p \implies p \cdot (p \cdot x + !p \cdot y)^\dagger = (p \cdot x)^\dagger \cdot p$   
 $\langle \text{proof} \rangle$

**lemma** *test-preserve2*:  $\text{test } p \implies p \cdot x \cdot p = p \cdot x \implies p \cdot (p \cdot x + !p \cdot y)^\dagger \leq x^\dagger$   
 $\langle \text{proof} \rangle$



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-zero1* = *left-kleene-algebra-zero1* + *test-semiring-zero1*

begin

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

lemma *star-test-export1*:  $\text{test } p \implies (p \cdot x)^* \cdot p \leq p \cdot x^*$   
*<proof>*

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

lemma *star-test-export2*:  $\text{test } p \implies (p \cdot x)^* \cdot p \leq x^* \cdot p$   
*<proof>*

lemma *star-n-export-left*:  $x \cdot n \cdot y \leq n \cdot y \cdot x \implies x^* \cdot n \cdot y = n \cdot y \cdot (x \cdot n \cdot y)^*$   
*<proof>*

lemma *star-test-export-left*:  $\text{test } p \implies x \cdot p \leq p \cdot x \implies x^* \cdot p = p \cdot (x \cdot p)^*$   
*<proof>*

lemma *star-n-export-right*:  $x \cdot n \cdot y \leq n \cdot y \cdot x \implies x^* \cdot n \cdot y = (n \cdot y \cdot x)^* \cdot n \cdot y$   
*<proof>*

lemma *star-test-export-right*:  $\text{test } p \implies x \cdot p \leq p \cdot x \implies x^* \cdot p = (p \cdot x)^* \cdot p$   
*<proof>*

lemma *star-n-folk*:  $n \cdot z \cdot x = x \cdot n \cdot z \implies n \cdot z \cdot y = y \cdot n \cdot z \implies (n \cdot z \cdot x + t \cdot z \cdot y)^* \cdot n \cdot z = n \cdot z \cdot (n \cdot z \cdot x)^*$   
*<proof>*

lemma *star-test-folk*:  $\text{test } p \implies p \cdot x = x \cdot p \implies p \cdot y = y \cdot p \implies (p \cdot x + !p \cdot y)^* \cdot p = p \cdot (p \cdot x)^*$   
*<proof>*

end

```

class kat-zero1 = kleene-algebra-zero1 + test-semiring-zero1
begin

sublocale conway: near-conway-zero1-tests star <proof>

lemma n-star-sim-right:  $n y \cdot x = x \cdot n y \implies n y \cdot x^* = (n y \cdot x)^* \cdot n y$ 
  <proof>

lemma star-sim-right:  $test\ p \implies p \cdot x = x \cdot p \longrightarrow p \cdot x^* = (p \cdot x)^* \cdot p$ 
  <proof>

lemma n-star-sim-left:  $n y \cdot x = x \cdot n y \implies n y \cdot x^* = n y \cdot (x \cdot n y)^*$ 
  <proof>

lemma star-sim-left:  $test\ p \implies p \cdot x = x \cdot p \longrightarrow p \cdot x^* = p \cdot (x \cdot p)^*$ 
  <proof>

lemma n-comm-star:  $n z \cdot x = x \cdot n z \implies n z \cdot y = y \cdot n z \implies n z \cdot x \cdot (n z \cdot y)^* = n z \cdot x \cdot y^*$ 
  <proof>

lemma comm-star:  $test\ p \implies p \cdot x = x \cdot p \longrightarrow p \cdot y = y \cdot p \longrightarrow p \cdot x \cdot (p \cdot y)^* = p \cdot x \cdot y^*$ 
  <proof>

lemma n-star-sim-right-var:  $n y \cdot x = x \cdot n y \implies x^* \cdot n y = n y \cdot (x \cdot n y)^*$ 
  <proof>

lemma star-sim-right-var:  $test\ p \implies p \cdot x = x \cdot p \longrightarrow x^* \cdot p = p \cdot (x \cdot p)^*$ 
  <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-zero1*  
**begin**

**subclass** *kat-zero1* *<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 (-<sup>o</sup> [101] 100) **where**  
*test* *p* ⇒ *p*<sup>o</sup> = !*p*·⊤ + 1

**lemma** *asg*:  $\llbracket \text{test } p; \text{test } q \rrbracket \Longrightarrow q \leq 1 \wedge 1 \leq p^o$   
*<proof>*

**lemma** *assertion-isol*:  $\text{test } p \Longrightarrow y \leq p^o \cdot x \longleftrightarrow p \cdot y \leq x$   
*<proof>*

**lemma** *assertion-isor*:  $\text{test } p \Longrightarrow y \leq x \cdot p \longleftrightarrow y \cdot p^o \leq x$   
*<proof>*

**lemma** *assertion-iso*:  $\llbracket \text{test } p; \text{test } q \rrbracket \Longrightarrow x \cdot q^o \leq p^o \cdot x \longleftrightarrow p \cdot x \leq x \cdot q$   
*<proof>*

**lemma** *total-correctness*:  $\llbracket \text{test } p; \text{test } q \rrbracket \Longrightarrow p \cdot x \cdot !q = 0 \longleftrightarrow x \cdot !q \leq !p \cdot \top$   
*<proof>*

**lemma** *test-iteration-sim*:  $\llbracket \text{test } p; p \cdot x \leq x \cdot p \rrbracket \Longrightarrow p \cdot x^\infty \leq x^\infty \cdot p$   
*<proof>*

**lemma** *test-iteration-annir*:  $\text{test } p \Longrightarrow !p \cdot (p \cdot x)^\infty = !p$   
*<proof>*

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

**lemma** *loop-refinement*:  $\llbracket \text{test } p; \text{test } q \rrbracket \Longrightarrow (p \cdot x)^\infty \cdot !p \leq (p \cdot q \cdot x)^\infty \cdot !(p \cdot q) \cdot (p \cdot x)^\infty \cdot !p$   
*<proof>*

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

**lemma** *atom-step1*:  $r \cdot b \leq b \cdot r \Longrightarrow (a + b + r)^\infty = b^\infty \cdot r^\infty \cdot (a \cdot b^\infty \cdot r^\infty)^\infty$   
*<proof>*

**lemma** *atom-step2*:

**assumes**  $s = s \cdot q \cdot q \cdot b = 0 \quad r \cdot q \leq q \cdot r \quad q \cdot l \leq l \cdot q \quad r^\infty = r^* \quad \text{test } q$   
**shows**  $s \cdot l^\infty \cdot b^\infty \cdot r^\infty \cdot q \cdot (a \cdot b^\infty \cdot r^\infty \cdot q)^\infty \leq s \cdot l^\infty \cdot r^\infty \cdot (a \cdot b^\infty \cdot q \cdot r^\infty)^\infty$

*<proof>*

**lemma** *atom-step3*:

**assumes**  $r \cdot l \leq l \cdot r$   $a \cdot l \leq l \cdot a$   $b \cdot l \leq l \cdot b$   $q \cdot l \leq l \cdot q$   $b^\infty = b^*$

**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**  $s = s \cdot q$   $a = q \cdot a$   $q \cdot b = 0$

$r \cdot b \leq b \cdot r$   $r \cdot l \leq l \cdot r$   $r \cdot q \leq q \cdot r$

$a \cdot l \leq l \cdot a$   $b \cdot l \leq l \cdot b$   $q \cdot l \leq l \cdot q$

$r^\infty = r^*$   $b^\infty = b^*$  *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**  $s \leq s \cdot q$   $a \leq q \cdot a$   $q \cdot b = 0$

$r \cdot b \leq b \cdot r$   $r \cdot q \leq q \cdot r$

$(a + r + b) \cdot l \leq l \cdot (a + r + b)$   $q \cdot l \leq l \cdot q$

$r^\infty = r^*$   $q \leq 1$

**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* ((-/ ≤ -) [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*:  $\text{adjoint } (f \circ g) = \text{adjoint } f \circ \text{adjoint } g$   
*<proof>*

**lemma** *adjoint-top[simp]*:  $\text{adjoint } \text{top} = \text{bot}$   
*<proof>*

**lemma** *N-comp1*:  $(N \ (N \ f)) + N \ f = \text{id}$   
*<proof>*

**lemma** *N-comp2*:  $(N \ (N \ f)) \circ N \ f = \text{bot}$   
*<proof>*

**lemma** *N-comp3*:  $N \ f \circ (N \ (N \ f)) = \text{bot}$   
*<proof>*

**lemma** *N-de-morgan*:  $N \ (N \ f) \circ N \ (N \ g) = N \ (N \ f) \sqcap N \ (N \ g)$   
*<proof>*

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

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

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

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

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

**instantiation** *bool-op* :: (type) *dioid-one-zero*

**begin**

**lift-definition** *less-eq-bool-op* :: *'a bool-op*  $\Rightarrow$  *'a bool-op*  $\Rightarrow$  *bool* **is** *fun-order*  
*<proof>*

**lift-definition** *less-bool-op* :: *'a bool-op*  $\Rightarrow$  *'a bool-op*  $\Rightarrow$  *bool* **is** *op <*. *<proof>*

**lift-definition** *zero-bool-op* :: *'a bool-op* **is** *bot*  
*<proof>*

**lift-definition** *one-bool-op* :: *'a bool-op* **is** *id*  
*<proof>*

**lift-definition** *times-bool-op* :: *'a bool-op*  $\Rightarrow$  *'a bool-op*  $\Rightarrow$  *'a bool-op* **is** *op o*  
*<proof>*

```

lift-definition plus-bool-op :: 'a bool-op  $\Rightarrow$  'a bool-op  $\Rightarrow$  'a bool-op is op +
  <proof>

instance
  <proof>

end

instantiation bool-op :: (type) test-semiring-zero
begin
  lift-definition n-op-bool-op :: 'a bool-op  $\Rightarrow$  'a bool-op is N
    <proof>

  instance
    <proof>

end

definition fun-star :: 'a bfun  $\Rightarrow$  'a bfun where
  fun-star f = lfp ( $\lambda r. f \circ r + id$ )

definition fun-iteration :: 'a bfun  $\Rightarrow$  'a bfun where
  fun-iteration f = gfp ( $\lambda g. f \circ g + id$ )

  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 op  $\cup$  op O Id {} op  $\subseteq$  op  $\subset$   $\lambda x. Id$ 
   $\cap (- x)$ 
    <proof>

  interpretation rel-kat: kat op  $\cup$  op O Id {} op  $\subseteq$  op  $\subset$  rtrancl  $\lambda x. Id \cap (- x)$ 
    <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  $\Rightarrow$  'a relation **is**  $\lambda x. Id \cap (- x)$   $\langle proof \rangle$   
**lift-definition** *zero-relation* :: 'a relation **is**  $\{\}$   $\langle proof \rangle$   
**lift-definition** *star-relation* :: 'a relation  $\Rightarrow$  'a relation **is** *rtrancl*  $\langle proof \rangle$   
**lift-definition** *less-eq-relation* :: 'a relation  $\Rightarrow$  'a relation  $\Rightarrow$  bool **is** *op*  $\subseteq$   $\langle proof \rangle$   
**lift-definition** *less-relation* :: 'a relation  $\Rightarrow$  'a relation  $\Rightarrow$  bool **is** *op*  $\subset$   $\langle proof \rangle$   
**lift-definition** *one-relation* :: 'a relation **is** *Id*  $\langle proof \rangle$   
**lift-definition** *times-relation* :: 'a relation  $\Rightarrow$  'a relation  $\Rightarrow$  'a relation **is** *op* *O*  $\langle proof \rangle$   
**lift-definition** *plus-relation* :: 'a relation  $\Rightarrow$  'a relation  $\Rightarrow$  'a relation **is** *op*  $\cup$   $\langle proof \rangle$

**instance**  
 $\langle proof \rangle$

**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  $\Rightarrow$  'a  $\Rightarrow$  bool **where**

*pres* *x y*  $\equiv$  *y*  $\cdot$  *x* = *y*  $\cdot$  *x*  $\cdot$  *y*

**lemma** *pres-comp*: *pres* *y z*  $\Longrightarrow$  *pres* *x z*  $\Longrightarrow$  *pres* (*x*  $\cdot$  *y*) *z*

$\langle proof \rangle$

**lemma** *test-pres1*: *test* *p*  $\Longrightarrow$  *test* *q*  $\Longrightarrow$  *pres* *p* *q*

$\langle proof \rangle$

**lemma** *test-pres2*: *test* *p*  $\Longrightarrow$  *test* *q*  $\Longrightarrow$  *pres* *x* *q*  $\Longrightarrow$  *pres* (*p*  $\cdot$  *x*) *q*

$\langle proof \rangle$

**lemma** *cond-helper1*:

**assumes** *test* *p* **and** *pres* *x* *p*

**shows** *p*  $\cdot$  (*p*  $\cdot$  *x* + !*p*  $\cdot$  *y*)<sup>†</sup>  $\cdot$  (*p*  $\cdot$  *z* + !*p*  $\cdot$  *w*) = *p*  $\cdot$  *x*<sup>†</sup>  $\cdot$  *z*

$\langle proof \rangle$



**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 trans-

formation theorem is valid for KAT as well.

**sublocale** *kat*  $\subseteq$  *pre-conway star*  $\langle$ *proof* $\rangle$

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

**sublocale** *drat*  $\subseteq$  *pre-conway strong-iteration*  
 $\langle$ *proof* $\rangle$

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 ( $\alpha$ )  
**and** *tau* :: 'a  $\Rightarrow$  'a ( $\tau$ )  
**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**  $\alpha u \cdot (\tau v \cdot x + \tau w \cdot y) \leq (\tau v \cdot x + \tau w \cdot y) \cdot z$   
 $\langle$ *proof* $\rangle$

**lemma** *ht-at-phl-cond*:

**assumes**  $\{\alpha u \cdot \tau v\} x \{z\}$  **and**  $\{\alpha u \cdot \tau w\} y \{z\}$   
**shows**  $\{\alpha u\} (\tau v \cdot x + \tau w \cdot y) \{z\}$   
 $\langle$ *proof* $\rangle$

**lemma** *ht-at-phl-export1*:

**assumes**  $\{\alpha x \cdot \tau y\} z \{w\}$   
**shows**  $\{\alpha x\} \tau y \cdot z \{w\}$   
 $\langle$ *proof* $\rangle$

**lemma** *ht-at-phl-export2*:  
**assumes**  $\{x\} y \{\alpha z\}$   
**shows**  $\{x\} y \cdot \tau w \{\alpha z \cdot \tau w\}$   
 $\langle proof \rangle$

**end**

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

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

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

**end**

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

**sublocale** *test-pre-diod-zero1* < *phl*: *at-pre-diod* **where** *alpha* = *t* **and** *tau* = *t*  
 $\langle proof \rangle$

**sublocale** *test-pre-conway* < *phl*: *at-it-pre-diod* **where** *alpha* = *t* **and** *tau* = *t*  
**and** *it* = *dagger*  $\langle proof \rangle$

**sublocale** *kat-zero1* < *phl*: *at-it-pre-diod* **where** *alpha* = *t* **and** *tau* = *t* **and** *it*  
= *star*  $\langle proof \rangle$

**context** *test-pre-diod-zero1* **begin**

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

**lemma** *phl-n-cond*:  
 $\{n v \cdot n w\} x \{z\} \Longrightarrow \{n v \cdot t w\} y \{z\} \Longrightarrow \{n v\} n w \cdot x + t w \cdot y \{z\}$   
 $\langle proof \rangle$

**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-zero1*

**begin**

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

**lemma** *phl-n-while:*

**assumes**  $\{n x \cdot n y\} z \{n x\}$   
**shows**  $\{n x\} (n y \cdot z)^* \cdot t y \{n x \cdot t y\}$   
*<proof>*

**lemma** *phl-test-while:*

**assumes** *test p and test b*  
**and**  $\{p \cdot b\} x \{p\}$   
**shows**  $\{p\} (b \cdot x)^* \cdot !b \{p \cdot !b\}$   
*<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*)  $test\ p \implies p \cdot x^* \leq x^* \cdot p \implies p \cdot x \leq x \cdot p$   
 ⟨proof⟩

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

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

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

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

**abbreviation** (in *near-diod*)  $tt :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool\ ((|-)-(|-))$  **where**  
 $(|x|)\ y\ (|z|) \equiv x \cdot y \leq z$

**lemma** (in *near-diod-one*)  $tt\text{-skip}: (|p|)\ 1\ (|p|)$   
 ⟨proof⟩

**lemma** (in *near-diod*)  $tt\text{-cons1}: (\exists q'. (|p|)\ x\ (|q'|) \wedge q' \leq q) \longleftrightarrow (|p|)\ x\ (|q|)$   
 ⟨proof⟩

**lemma** (in *near-diod*)  $tt\text{-cons2}: (\exists p'. (|p'|)\ x\ (|q|) \wedge p \leq p') \longleftrightarrow (|p|)\ x\ (|q|)$   
 ⟨proof⟩

**lemma** (in *near-diod*)  $tt\text{-seq}: (\exists r. (|p|)\ x\ (|r|) \wedge (|r|)\ y\ (|q|)) \longleftrightarrow (|p|)\ x \cdot y\ (|q|)$   
 ⟨proof⟩

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

**lemma** (in *kleene-algebra*)  $tt\text{-while}: w \leq 1 \implies (|p \cdot v|)\ x\ (|p|) \implies (|p|)\ (v \cdot x)^* \cdot w\ (|p|)$   
 ⟨proof⟩

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

**lemma** (in *kat*)  $tt\text{-while}: test\ v \implies (|p|)\ (v \cdot x)^* \cdot !v\ (|p|) \implies (|p \cdot v|)\ x\ (|p|)$   
 ⟨proof⟩

**lemma** (in *kat*)  $tt\text{-while}: test\ v \implies (|p|)\ (v \cdot x)^*\ (|p|) \implies (|p \cdot v|)\ x\ (|p|)$   
 ⟨proof⟩

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

wlp in KAT

**lemma** (in *kat*)  $test\ y \implies (\exists\ z.\ test\ z \wedge z \cdot x \cdot !y = 0)$   
*<proof>*

**end**

## 9 Propositional Hoare Logic

**theory** *PHL-DRAT*

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

**begin**

**sublocale** *drat* < *phl*: *at-it-pre-diod* **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  $\Rightarrow$  'a ('-')

$\langle ML \rangle$

**locale** *diod-tests* =

**fixes** *test* :: 'a::boolean-algebra  $\Rightarrow$  'b::diod-one-zero  
**and** *not* :: 'b::diod-one-zero  $\Rightarrow$  'b::diod-one-zero (-)  
**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*  $\leq$  *q*  $\longleftrightarrow$  '*p*  $\leq$  *q*'

**begin**

**notation** *test* ( $\iota$ )

**lemma** *test-eq* [*kat-hom*]: *p* = *q*  $\longleftrightarrow$  '*p* = *q*'  
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *test-iso*: *p*  $\leq$  *q*  $\implies$  '*p*  $\leq$  *q*'  
 $\langle proof \rangle$

**lemma** *test-meet-comm*: '*p* · *q* = *q* · *p*'  
 $\langle proof \rangle$

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

**lemma** [*simp*]: '-*p* + *p* = 1'  
 $\langle proof \rangle$

**lemma** [*simp*]: '*p* + (-*p*) = 1'  
 $\langle proof \rangle$

**lemma** [*simp*]: '(-*p*) · *p* = 0'  
 $\langle proof \rangle$

**lemma** [*simp*]: '*p* · (-*p*) = 0'  
 $\langle proof \rangle$

**end**

```

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

sublocale kat ⊆ diooid-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*: ‘ $p \cdot c \leq c \cdot q \iff p \cdot c \cdot !q = 0$ ’  
 ⟨*proof*⟩

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

**lemma** *hoare-weakening*:  $p \leq p' \implies q' \leq q \implies \llbracket p' \rrbracket c \llbracket q' \rrbracket \implies \llbracket p \rrbracket c \llbracket q \rrbracket$   
 ⟨*proof*⟩

**lemma** *hoare-star*: ‘ $\llbracket p \rrbracket c \llbracket p \rrbracket \implies \llbracket p \rrbracket c^* \llbracket p \rrbracket$ ’  
 ⟨*proof*⟩

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

**lemma** *hoare-test* [*vcg*]: ‘ $p \cdot t \leq q \implies \llbracket p \rrbracket t \llbracket q \rrbracket$ ’  
 ⟨*proof*⟩

**lemma** *hoare-mult* [*vcg*]: ‘ $\llbracket p \rrbracket x \llbracket r \rrbracket \implies \llbracket r \rrbracket y \llbracket q \rrbracket \implies \llbracket p \rrbracket x \cdot y \llbracket q \rrbracket$ ’  
 ⟨*proof*⟩

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

**lemma** *hoare-plus* [*vcg*]: ‘ $\llbracket p \rrbracket x \llbracket q \rrbracket \implies \llbracket p \rrbracket y \llbracket q \rrbracket \implies \llbracket p \rrbracket x + y \llbracket q \rrbracket$ ’  
 ⟨*proof*⟩

**definition** *While* :: ‘ $b \implies 'b \implies 'b$  (*While - Do - End* [50,50] 51) **where**  
*While t Do c End* =  $(t \cdot c)^* \cdot !t$

**lemma** *hoare-while*: ‘ $\llbracket p \cdot t \rrbracket c \llbracket p \rrbracket \implies \llbracket p \rrbracket \text{While } t \text{ Do } c \text{ End } \llbracket !t \cdot p \rrbracket$ ’  
 ⟨*proof*⟩

**lemma** [*vcg*]: ‘ $\llbracket p \cdot t \rrbracket c \llbracket p \rrbracket \implies !t \cdot p \leq q \implies \llbracket p \rrbracket \text{While } t \text{ Do } c \text{ End } \llbracket q \rrbracket$ ’  
 ⟨*proof*⟩

**definition** *If* :: ‘ $b \implies 'b \implies 'b \implies 'b$  (*If - Then - Else -* [50,50,50] 51) **where**  
*If p Then c1 Else c2* =  $p \cdot c1 + !p \cdot c2$

**lemma** *hoare-if* [*vcg*]: ‘ $\llbracket p \cdot t \rrbracket c1 \llbracket q \rrbracket \implies \llbracket p \cdot !t \rrbracket c2 \llbracket q \rrbracket \implies \llbracket p \rrbracket \text{If } t \text{ Then } c1 \text{ Else } c2 \llbracket q \rrbracket$ ’  
 ⟨*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  $\Rightarrow$  'a ('-')

$\langle ML \rangle$

**locale** *drat* =

**fixes** *test* :: 'a::boolean-algebra  $\Rightarrow$  'b::dra  
**and** *not* :: 'b::dra  $\Rightarrow$  '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* ( $\neg$  *p*) = '!*p*'  
**and** *test-iso-eq* [*kat-hom*]: *p*  $\leq$  *q*  $\longleftrightarrow$  '*p*  $\leq$  *q*'

**begin**

**notation** *test* ( $\iota$ )

**lemma** *test-eq* [*kat-hom*]: *p* = *q*  $\longleftrightarrow$  '*p* = *q*'  
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *test-iso*: *p*  $\leq$  *q*  $\implies$  '*p*  $\leq$  *q*'  
 $\langle proof \rangle$

**lemma** *test-meet-comm*: '*p* · *q* = *q* · *p*'  
 $\langle proof \rangle$

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

**lemma** *test-star* [*simp*]: '*p*<sup>\*</sup> = 1'  
 $\langle proof \rangle$

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

**lemma** *test-comp-add1* [*simp*]: '!*p* + *p* = 1'  
 $\langle proof \rangle$

**lemma** *test-comp-add2* [*simp*]: '*p* + !*p* = 1'  
 $\langle proof \rangle$

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

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

**lemma** *test-eq1*: ‘ $y \leq x \iff p \cdot y \leq x \wedge !p \cdot y \leq x$ ’  
 ⟨*proof*⟩

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

**lemma** *test1*: ‘ $p \cdot x \cdot !q = 0 \implies p \cdot x = p \cdot x \cdot q$ ’  
 ⟨*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*: ‘ $p \cdot x \cdot !q = 0 \iff x \cdot !q \leq !p \cdot \top$ ’  
 ⟨*proof*⟩

**lemma** *test-iteration-sim*: ‘ $p \cdot x \leq x \cdot p \implies p \cdot x^\infty \leq x^\infty \cdot p$ ’  
 ⟨*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. Preteasa. 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.