

# Algebras for Iteration, Infinite Executions and Correctness of Sequential Computations

Walter Guttmann

March 19, 2025

## Abstract

We study models of state-based non-deterministic sequential computations and describe them using algebras. We propose algebras that describe iteration for strict and non-strict computations. They unify computation models which differ in the fixpoints used to represent iteration. We propose algebras that describe the infinite executions of a computation. They lead to a unified approximation order and results that connect fixpoints in the approximation and refinement orders. This unifies the semantics of recursion for a range of computation models. We propose algebras that describe preconditions and the effect of while-programs under postconditions. They unify correctness statements in two dimensions: one statement applies in various computation models to various correctness claims.

These theories consolidate results which have appeared in [1–20]. Most are described in [16]. Theorem numbers refer to [16] except in theory *Lattice-Ordered Semirings*, where they refer to [2], and in theories *Capped Omega Algebras*, *N-Algebras*, *N-Omega-Algebras*, *N-Omega Binary Iterings* and *Recursion*, where they refer to [19].

## Contents

<b>1</b>	<b>Base</b>	<b>3</b>
<b>2</b>	<b>Omega Algebras</b>	<b>7</b>
<b>3</b>	<b>Capped Omega Algebras</b>	<b>15</b>
<b>4</b>	<b>General Refinement Algebras</b>	<b>19</b>
<b>5</b>	<b>Lattice-Ordered Semirings</b>	<b>24</b>
<b>6</b>	<b>Boolean Semirings</b>	<b>41</b>

<b>7 Binary Iterings</b>	<b>52</b>
<b>8 Strict Binary Iterings</b>	<b>65</b>
<b>9 Nonstrict Binary Iterings</b>	<b>68</b>
<b>10 Tests</b>	<b>73</b>
<b>11 Test Iterings</b>	<b>77</b>
<b>12 N-Semirings</b>	<b>83</b>
<b>13 Boolean N-Semirings</b>	<b>96</b>
<b>14 Modal N-Semirings</b>	<b>108</b>
<b>15 Approximation</b>	<b>123</b>
<b>16 Strict Recursion</b>	<b>125</b>
<b>17 N-Algebras</b>	<b>131</b>
<b>18 Recursion</b>	<b>140</b>
<b>19 N-Omega-Algebras</b>	<b>146</b>
<b>20 N-Omega Binary Iterings</b>	<b>152</b>
<b>21 N-Relation-Algebras</b>	<b>156</b>
<b>22 Domain</b>	<b>159</b>
<b>23 Domain Iterings</b>	<b>166</b>
<b>24 Domain Recursion</b>	<b>172</b>
<b>25 Extended Designs</b>	<b>178</b>
<b>26 Relative Domain</b>	<b>181</b>
<b>27 Relative Modal Operators</b>	<b>192</b>
<b>28 Complete Tests</b>	<b>203</b>
<b>29 Complete Domain</b>	<b>204</b>
<b>30 Preconditions</b>	<b>205</b>

<b>31 Hoare Calculus</b>	<b>210</b>
<b>32 Hoare Calculus and Modal Operators</b>	<b>225</b>
<b>33 Pre-Post Specifications</b>	<b>232</b>
<b>34 Pre-Post Specifications and Modal Operators</b>	<b>241</b>
<b>35 Monotonic Boolean Transformers</b>	<b>243</b>
<b>36 Instances of Monotonic Boolean Transformers</b>	<b>251</b>

## 1 Base

**theory** *Base*

**imports** *Stone-Relation-Algebras.Semirings*

**begin**

**nitpick-params** [*timeout = 600*]

**class** *while* =  
**fixes** *while* :: 'a ⇒ 'a ⇒ 'a (**infixr** <\*> 59)

**class** *n* =  
**fixes** *n* :: 'a ⇒ 'a

**class** *diamond* =  
**fixes** *diamond* :: 'a ⇒ 'a ⇒ 'a (<| - > -> [50,90] 95)

**class** *box* =  
**fixes** *box* :: 'a ⇒ 'a ⇒ 'a (<| - ] -> [50,90] 95)

**context** *ord*  
**begin**

**definition** *ascending-chain* :: (nat ⇒ 'a) ⇒ bool  
**where** *ascending-chain* *f* ≡ ∀ n . *f* n ≤ *f* (Suc n)

**definition** *descending-chain* :: (nat ⇒ 'a) ⇒ bool  
**where** *descending-chain* *f* ≡ ∀ n . *f* (Suc n) ≤ *f* n

**definition** *directed* :: 'a set ⇒ bool  
**where** *directed* *X* ≡ *X* ≠ {} ∧ (∀ x∈*X* . ∀ y∈*X* . ∃ z∈*X* . x ≤ z ∧ y ≤ z)

**definition** *co-directed* :: 'a set ⇒ bool  
**where** *co-directed* *X* ≡ *X* ≠ {} ∧ (∀ x∈*X* . ∀ y∈*X* . ∃ z∈*X* . z ≤ x ∧ z ≤ y)

**definition** *chain* :: 'a set  $\Rightarrow$  bool  
where *chain*  $X \equiv \forall x \in X . \forall y \in X . x \leq y \vee y \leq x$

**end**

**context** *order*  
**begin**

**lemma** *ascending-chain-k*:  
*ascending-chain*  $f \Longrightarrow f\ m \leq f\ (m + k)$   
(*proof*)

**lemma** *ascending-chain-isotone*:  
*ascending-chain*  $f \Longrightarrow m \leq k \Longrightarrow f\ m \leq f\ k$   
(*proof*)

**lemma** *ascending-chain-comparable*:  
*ascending-chain*  $f \Longrightarrow f\ k \leq f\ m \vee f\ m \leq f\ k$   
(*proof*)

**lemma** *ascending-chain-chain*:  
*ascending-chain*  $f \Longrightarrow \text{chain}\ (\text{range}\ f)$   
(*proof*)

**lemma** *chain-directed*:  
 $X \neq \{\}$   $\Longrightarrow \text{chain}\ X \Longrightarrow \text{directed}\ X$   
(*proof*)

**lemma** *ascending-chain-directed*:  
*ascending-chain*  $f \Longrightarrow \text{directed}\ (\text{range}\ f)$   
(*proof*)

**lemma** *descending-chain-k*:  
*descending-chain*  $f \Longrightarrow f\ (m + k) \leq f\ m$   
(*proof*)

**lemma** *descending-chain-antitone*:  
*descending-chain*  $f \Longrightarrow m \leq k \Longrightarrow f\ k \leq f\ m$   
(*proof*)

**lemma** *descending-chain-comparable*:  
*descending-chain*  $f \Longrightarrow f\ k \leq f\ m \vee f\ m \leq f\ k$   
(*proof*)

**lemma** *descending-chain-chain*:  
*descending-chain*  $f \Longrightarrow \text{chain}\ (\text{range}\ f)$   
(*proof*)

**lemma** *chain-co-directed*:

$X \neq \{\}$   $\implies$  *chain*  $X \implies$  *co-directed*  $X$

*<proof>*

**lemma** *descending-chain-codirected*:

*descending-chain*  $f \implies$  *co-directed* (*range*  $f$ )

*<proof>*

**end**

**context** *semilattice-sup*

**begin**

**lemma** *ascending-chain-left-sup*:

*ascending-chain*  $f \implies$  *ascending-chain*  $(\lambda n . x \sqcup f n)$

*<proof>*

**lemma** *ascending-chain-right-sup*:

*ascending-chain*  $f \implies$  *ascending-chain*  $(\lambda n . f n \sqcup x)$

*<proof>*

**lemma** *descending-chain-left-add*:

*descending-chain*  $f \implies$  *descending-chain*  $(\lambda n . x \sqcup f n)$

*<proof>*

**lemma** *descending-chain-right-add*:

*descending-chain*  $f \implies$  *descending-chain*  $(\lambda n . f n \sqcup x)$

*<proof>*

**primrec** *pSum0* ::  $(\text{nat} \Rightarrow 'a) \Rightarrow \text{nat} \Rightarrow 'a$

**where** *pSum0*  $f$  0 =  $f$  0

| *pSum0*  $f$  (*Suc*  $m$ ) = *pSum0*  $f$   $m \sqcup f$   $m$

**lemma** *pSum0-below*:

$\forall i . f i \leq x \implies$  *pSum0*  $f$   $m \leq x$

*<proof>*

**end**

**context** *non-associative-left-semiring*

**begin**

**lemma** *ascending-chain-left-mult*:

*ascending-chain*  $f \implies$  *ascending-chain*  $(\lambda n . x * f n)$

*<proof>*

**lemma** *ascending-chain-right-mult*:

*ascending-chain*  $f \implies$  *ascending-chain*  $(\lambda n . f n * x)$

*<proof>*

**lemma** *descending-chain-left-mult*:  
 $descending-chain\ f \implies descending-chain\ (\lambda n . x * f\ n)$   
*<proof>*

**lemma** *descending-chain-right-mult*:  
 $descending-chain\ f \implies descending-chain\ (\lambda n . f\ n * x)$   
*<proof>*

**end**

**context** *complete-lattice*  
**begin**

**lemma** *sup-Sup*:  
 $A \neq \{\}$   $\implies sup\ x\ (Sup\ A) = Sup\ ((sup\ x) \text{ ‘ } A)$   
*<proof>*

**lemma** *sup-SUP*:  
 $Y \neq \{\}$   $\implies sup\ x\ (SUP\ y \in Y . f\ y) = (SUP\ y \in Y . sup\ x\ (f\ y))$   
*<proof>*

**lemma** *inf-Inf*:  
 $A \neq \{\}$   $\implies inf\ x\ (Inf\ A) = Inf\ ((inf\ x) \text{ ‘ } A)$   
*<proof>*

**lemma** *inf-INF*:  
 $Y \neq \{\}$   $\implies inf\ x\ (INF\ y \in Y . f\ y) = (INF\ y \in Y . inf\ x\ (f\ y))$   
*<proof>*

**lemma** *SUP-image-id[simp]*:  
 $(SUP\ x \in f \text{ ‘ } A . x) = (SUP\ x \in A . f\ x)$   
*<proof>*

**lemma** *INF-image-id[simp]*:  
 $(INF\ x \in f \text{ ‘ } A . x) = (INF\ x \in A . f\ x)$   
*<proof>*

**end**

**lemma** *image-Collect-2*:  
 $f \text{ ‘ } \{ g\ x \mid x . P\ x \} = \{ f\ (g\ x) \mid x . P\ x \}$   
*<proof>*

The following instantiation and four lemmas are from Jose Divason Malagaray.

**instantiation** *fun* :: (type, type) power  
**begin**

```

definition one-fun :: 'a ⇒ 'a
  where one-fun-def: one-fun ≡ id

definition times-fun :: ('a ⇒ 'a) ⇒ ('a ⇒ 'a) ⇒ ('a ⇒ 'a)
  where times-fun-def: times-fun ≡ comp

instance
  ⟨proof⟩

end

lemma id-power:
  id^m = id
  ⟨proof⟩

lemma power-zero-id:
  f^0 = id
  ⟨proof⟩

lemma power-succ-unfold:
  f^Suc m = f ∘ f^m
  ⟨proof⟩

lemma power-succ-unfold-ext:
  (f^Suc m) x = f ((f^m) x)
  ⟨proof⟩

end

```

## 2 Omega Algebras

```

theory Omega-Algebras

imports Stone-Kleene-Relation-Algebras.Kleene-Algebras

begin

nitpick-params [timeout = 600]

class omega =
  fixes omega :: 'a ⇒ 'a (‹-ω› [100] 100)

class left-omega-algebra = left-kleene-algebra + omega +
  assumes omega-unfold: yω = y * yω
  assumes omega-induct: x ≤ z ⊔ y * x ⟶ x ≤ yω ⊔ y* * z
begin

```

Many lemmas in this class are taken from [Georg Struth's Isabelle theories](#).

**lemma** *star-bot-below-omega*:

$$x^* * \text{bot} \leq x^\omega$$

*<proof>*

**lemma** *star-bot-below-omega-bot*:

$$x^* * \text{bot} \leq x^\omega * \text{bot}$$

*<proof>*

**lemma** *omega-induct-mult*:

$$y \leq x * y \implies y \leq x^\omega$$

*<proof>*

**lemma** *omega-sub-dist*:

$$x^\omega \leq (x \sqcup y)^\omega$$

*<proof>*

**lemma** *omega-isotone*:

$$x \leq y \implies x^\omega \leq y^\omega$$

*<proof>*

**lemma** *omega-induct-equal*:

$$y = z \sqcup x * y \implies y \leq x^\omega \sqcup x^* * z$$

*<proof>*

**lemma** *omega-bot*:

$$\text{bot}^\omega = \text{bot}$$

*<proof>*

**lemma** *omega-one-greatest*:

$$x \leq 1^\omega$$

*<proof>*

**lemma** *star-mult-omega*:

$$x^* * x^\omega = x^\omega$$

*<proof>*

**lemma** *omega-sub-vector*:

$$x^\omega * y \leq x^\omega$$

*<proof>*

**lemma** *omega-simulation*:

$$z * x \leq y * z \implies z * x^\omega \leq y^\omega$$

*<proof>*

**lemma** *omega-omega*:

$$x^{\omega\omega} \leq x^\omega$$

*<proof>*

**lemma** *left-plus-omega*:



$$(x * x^*)^\omega = x^\omega$$

*<proof>*

**lemma** *omega-slide*:

$$x * (y * x)^\omega = (x * y)^\omega$$

*<proof>*

**lemma** *omega-simulation-2*:

$$y * x \leq x * y \implies (x * y)^\omega \leq x^\omega$$

*<proof>*

**lemma** *wagner*:

$$(x \sqcup y)^\omega = x * (x \sqcup y)^\omega \sqcup z \implies (x \sqcup y)^\omega = x^\omega \sqcup x^* * z$$

*<proof>*

**lemma** *right-plus-omega*:

$$(x^* * x)^\omega = x^\omega$$

*<proof>*

**lemma** *omega-sub-dist-1*:

$$(x * y^*)^\omega \leq (x \sqcup y)^\omega$$

*<proof>*

**lemma** *omega-sub-dist-2*:

$$(x^* * y)^\omega \leq (x \sqcup y)^\omega$$

*<proof>*

**lemma** *omega-star*:

$$(x^\omega)^* = 1 \sqcup x^\omega$$

*<proof>*

**lemma** *omega-mult-omega-star*:

$$x^\omega * x^{\omega*} = x^\omega$$

*<proof>*

**lemma** *omega-sum-unfold-1*:

$$(x \sqcup y)^\omega = x^\omega \sqcup x^* * y * (x \sqcup y)^\omega$$

*<proof>*

**lemma** *omega-sum-unfold-2*:

$$(x \sqcup y)^\omega \leq (x^* * y)^\omega \sqcup (x^* * y)^* * x^\omega$$

*<proof>*

**lemma** *omega-sum-unfold-3*:

$$(x^* * y)^* * x^\omega \leq (x \sqcup y)^\omega$$

*<proof>*

**lemma** *omega-decompose*:

$$(x \sqcup y)^\omega = (x^* * y)^\omega \sqcup (x^* * y)^* * x^\omega$$

*<proof>*

**lemma** *omega-loop-fixpoint:*

$$y * (y^\omega \sqcup y^* * z) \sqcup z = y^\omega \sqcup y^* * z$$

*<proof>*

**lemma** *omega-loop-greatest-fixpoint:*

$$y * x \sqcup z = x \implies x \leq y^\omega \sqcup y^* * z$$

*<proof>*

**lemma** *omega-square:*

$$x^\omega = (x * x)^\omega$$

*<proof>*

**lemma** *mult-bot-omega:*

$$(x * \text{bot})^\omega = x * \text{bot}$$

*<proof>*

**lemma** *mult-bot-add-omega:*

$$(x \sqcup y * \text{bot})^\omega = x^\omega \sqcup x^* * y * \text{bot}$$

*<proof>*

**lemma** *omega-mult-star:*

$$x^\omega * x^* = x^\omega$$

*<proof>*

**lemma** *omega-loop-is-greatest-fixpoint:*

$$\text{is-greatest-fixpoint } (\lambda x . y * x \sqcup z) (y^\omega \sqcup y^* * z)$$

*<proof>*

**lemma** *omega-loop-nu:*

$$\nu (\lambda x . y * x \sqcup z) = y^\omega \sqcup y^* * z$$

*<proof>*

**lemma** *omega-loop-bot-is-greatest-fixpoint:*

$$\text{is-greatest-fixpoint } (\lambda x . y * x) (y^\omega)$$

*<proof>*

**lemma** *omega-loop-bot-nu:*

$$\nu (\lambda x . y * x) = y^\omega$$

*<proof>*

**lemma** *affine-has-greatest-fixpoint:*

$$\text{has-greatest-fixpoint } (\lambda x . y * x \sqcup z)$$

*<proof>*

**lemma** *omega-separate-unfold:*

$$(x^* * y)^\omega = y^\omega \sqcup y^* * x * (x^* * y)^\omega$$

*<proof>*

**lemma** *omega-bot-left-slide*:

$(x * y)^* * ((x * y)^\omega * \text{bot} \sqcup 1) * x \leq x * (y * x)^* * ((y * x)^\omega * \text{bot} \sqcup 1)$   
*<proof>*

**lemma** *omega-bot-add-1*:

$(x \sqcup y)^* * ((x \sqcup y)^\omega * \text{bot} \sqcup 1) = x^* * (x^\omega * \text{bot} \sqcup 1) * (y * x^* * (x^\omega * \text{bot} \sqcup 1))^* * ((y * x^* * (x^\omega * \text{bot} \sqcup 1))^\omega * \text{bot} \sqcup 1)$   
*<proof>*

**lemma** *star-omega-greatest*:

$x^{*\omega} = 1^\omega$   
*<proof>*

**lemma** *omega-vector-greatest*:

$x^\omega * 1^\omega = x^\omega$   
*<proof>*

**lemma** *mult-greatest-omega*:

$(x * 1^\omega)^\omega \leq x * 1^\omega$   
*<proof>*

**lemma** *omega-mult-star-2*:

$x^\omega * y^* = x^\omega$   
*<proof>*

**lemma** *omega-import*:

**assumes**  $p \leq p * p$   
**and**  $p * x \leq x * p$   
**shows**  $p * x^\omega = p * (p * x)^\omega$   
*<proof>*

**proposition** *omega-circ-simulate-right-plus*:  $z * x \leq y * (y^\omega * \text{bot} \sqcup y^*) * z \sqcup w \longrightarrow z * (x^\omega * \text{bot} \sqcup x^*) \leq (y^\omega * \text{bot} \sqcup y^*) * (z \sqcup w * (x^\omega * \text{bot} \sqcup x^*))$  **nitpick**  
[expect=genuine,card=4] *<proof>*

**proposition** *omega-circ-simulate-left-plus*:  $x * z \leq z * (y^\omega * \text{bot} \sqcup y^*) \sqcup w \longrightarrow (x^\omega * \text{bot} \sqcup x^*) * z \leq (z \sqcup (x^\omega * \text{bot} \sqcup x^*) * w) * (y^\omega * \text{bot} \sqcup y^*)$  **nitpick**  
[expect=genuine,card=5] *<proof>*

**end**

## Theorem 50.2

**sublocale** *left-omega-algebra* < *comb0*: *left-conway-semiring* **where**  $\text{circ} = (\lambda x . x^* * (x^\omega * \text{bot} \sqcup 1))$   
*<proof>*

**class** *left-zero-omega-algebra* = *left-zero-kleene-algebra* + *left-omega-algebra*  
**begin**

**lemma** *star-omega-absorb*:  
 $y^* * (y^* * x)^* * y^\omega = (y^* * x)^* * y^\omega$   
 ⟨proof⟩

**lemma** *omega-circ-simulate-right-plus*:  
**assumes**  $z * x \leq y * (y^\omega * \text{bot} \sqcup y^*) * z \sqcup w$   
**shows**  $z * (x^\omega * \text{bot} \sqcup x^*) \leq (y^\omega * \text{bot} \sqcup y^*) * (z \sqcup w * (x^\omega * \text{bot} \sqcup x^*))$   
 ⟨proof⟩

**lemma** *omega-circ-simulate-left-plus*:  
**assumes**  $x * z \leq z * (y^\omega * \text{bot} \sqcup y^*) \sqcup w$   
**shows**  $(x^\omega * \text{bot} \sqcup x^*) * z \leq (z \sqcup (x^\omega * \text{bot} \sqcup x^*) * w) * (y^\omega * \text{bot} \sqcup y^*)$   
 ⟨proof⟩

**lemma** *omega-translate*:  
 $x^* * (x^\omega * \text{bot} \sqcup 1) = x^\omega * \text{bot} \sqcup x^*$   
 ⟨proof⟩

**lemma** *omega-circ-simulate-right*:  
**assumes**  $z * x \leq y * z \sqcup w$   
**shows**  $z * (x^\omega * \text{bot} \sqcup x^*) \leq (y^\omega * \text{bot} \sqcup y^*) * (z \sqcup w * (x^\omega * \text{bot} \sqcup x^*))$   
 ⟨proof⟩

**end**

**sublocale** *left-zero-omega-algebra* < *comb1: left-conway-semiring-1* **where** *circ*  
 =  $(\lambda x . x^* * (x^\omega * \text{bot} \sqcup 1))$   
 ⟨proof⟩

**sublocale** *left-zero-omega-algebra* < *comb0: itering* **where** *circ* =  $(\lambda x . x^* * (x^\omega * \text{bot} \sqcup 1))$   
 ⟨proof⟩

### Theorem 2.2

**sublocale** *left-zero-omega-algebra* < *comb2: itering* **where** *circ* =  $(\lambda x . x^\omega * \text{bot} \sqcup x^*)$   
 ⟨proof⟩

**class** *omega-algebra* = *kleene-algebra* + *left-zero-omega-algebra*

**class** *left-omega-conway-semiring* = *left-omega-algebra* + *left-conway-semiring*  
**begin**

**subclass** *left-kleene-conway-semiring* ⟨proof⟩

**lemma** *circ-below-omega-star*:  
 $x^\circ \leq x^\omega \sqcup x^*$   
 ⟨proof⟩

**lemma** *omega-mult-circ*:

$$x^\omega * x^\circ = x^\omega$$

*<proof>*

**lemma** *circ-mult-omega*:

$$x^\circ * x^\omega = x^\omega$$

*<proof>*

**lemma** *circ-omega-greatest*:

$$x^{\circ\omega} = 1^\omega$$

*<proof>*

**lemma** *omega-circ*:

$$x^{\omega\circ} = 1 \sqcup x^\omega$$

*<proof>*

**end**

**class** *bounded-left-omega-algebra* = *bounded-left-kleene-algebra* +  
*left-omega-algebra*

**begin**

**lemma** *omega-one*:

$$1^\omega = \text{top}$$

*<proof>*

**lemma** *star-omega-top*:

$$x^{*\omega} = \text{top}$$

*<proof>*

**lemma** *omega-vector*:

$$x^\omega * \text{top} = x^\omega$$

*<proof>*

**lemma** *mult-top-omega*:

$$(x * \text{top})^\omega \leq x * \text{top}$$

*<proof>*

**end**

**sublocale** *bounded-left-omega-algebra* < *comb0*: *bounded-left-conway-semiring*

**where** *circ* = ( $\lambda x . x^* * (x^\omega * \text{bot} \sqcup 1)$ ) *<proof>*

**class** *bounded-left-zero-omega-algebra* = *bounded-left-zero-kleene-algebra* +  
*left-zero-omega-algebra*

**begin**

**subclass** *bounded-left-omega-algebra* *<proof>*

**end**

**sublocale** *bounded-left-zero-omega-algebra* < *comb0*: *bounded-itering* **where** *circ* =  $(\lambda x . x^* * (x^\omega * \text{bot} \sqcup 1))$  *<proof>*

**class** *bounded-omega-algebra* = *bounded-kleene-algebra* + *omega-algebra*  
**begin**

**subclass** *bounded-left-zero-omega-algebra* *<proof>*

**end**

**class** *bounded-left-omega-conway-semiring* = *bounded-left-omega-algebra* + *left-omega-conway-semiring*  
**begin**

**subclass** *left-kleene-conway-semiring* *<proof>*

**subclass** *bounded-left-conway-semiring* *<proof>*

**lemma** *circ-omega*:

$x^{\circ\omega} = \text{top}$   
*<proof>*

**end**

**class** *top-left-omega-algebra* = *bounded-left-omega-algebra* +  
**assumes** *top-left-bot*:  $\text{top} * x = \text{top}$   
**begin**

**lemma** *omega-translate-3*:

$x^* * (x^\omega * \text{bot} \sqcup 1) = x^* * (x^\omega \sqcup 1)$   
*<proof>*

**end**

### Theorem 50.2

**sublocale** *top-left-omega-algebra* < *comb4*: *left-conway-semiring* **where** *circ* =  $(\lambda x . x^* * (x^\omega \sqcup 1))$   
*<proof>*

**class** *top-left-bot-omega-algebra* = *bounded-left-zero-omega-algebra* +  
**assumes** *top-left-bot*:  $\text{top} * x = \text{top}$   
**begin**

**lemma** *omega-translate-2*:

$x^\omega * \text{bot} \sqcup x^* = x^\omega \sqcup x^*$   
*<proof>*

**end**

[Theorem 2.3](#)

**sublocale** *top-left-bot-omega-algebra* < *comb3*: *itering* **where** *circ* =  $(\lambda x . x^\omega \sqcup x^*)$   
*<proof>*

**class** *Omega* =  
  **fixes** *Omega* :: 'a  $\Rightarrow$  'a  $\langle \cdot^{-\Omega} \rangle$  [100] 100)

**end**

### 3 Capped Omega Algebras

**theory** *Capped-Omega-Algebras*

**imports** *Omega-Algebras*

**begin**

**class** *capped-omega* =  
  **fixes** *capped-omega* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\langle \cdot^{-\omega} \cdot \rangle$  [100,100] 100)

**class** *capped-omega-algebra* = *bounded-left-zero-kleene-algebra* +  
*bounded-distrib-lattice* + *capped-omega* +  
  **assumes** *capped-omega-unfold*:  $y^\omega_v = y * y^\omega_v \sqcap v$   
  **assumes** *capped-omega-induct*:  $x \leq (y * x \sqcup z) \sqcap v \longrightarrow x \leq y^\omega_v \sqcup y^* * z$

[AACP Theorem 6.1](#)

**notation**

*top*  $\langle \cdot^\top \rangle$

**sublocale** *capped-omega-algebra* < *capped*: *bounded-left-zero-omega-algebra*  
**where** *omega* =  $(\lambda y . y^\omega_\top)$   
*<proof>*

**context** *capped-omega-algebra*

**begin**

[AACP Theorem 6.2](#)

**lemma** *capped-omega-below-omega*:  
 $y^\omega_v \leq y^\omega_\top$   
*<proof>*

[AACP Theorem 6.3](#)

**lemma** *capped-omega-below*:  
 $y^\omega_v \leq v$   
*<proof>*

#### AACP Theorem 6.4

**lemma** *capped-omega-one*:

$$1^\omega_v = v$$

*<proof>*

#### AACP Theorem 6.5

**lemma** *capped-omega-zero*:

$$\text{bot}^\omega_v = \text{bot}$$

*<proof>*

**lemma** *star-below-cap*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies y^* * z \leq v$$

*<proof>*

**lemma** *capped-fix*:

**assumes**  $y \leq u$

**and**  $z \leq v$

**and**  $u * v \leq v$

**shows**  $(y * (y^\omega_v \sqcup y^* * z) \sqcup z) \sqcap v = y^\omega_v \sqcup y^* * z$

*<proof>*

**lemma** *capped-fixpoint*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies \text{is-fixpoint } (\lambda x . (y * x \sqcup z) \sqcap v) (y^\omega_v \sqcup y^* * z)$$

*<proof>*

**lemma** *capped-greatest-fixpoint*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies \text{is-greatest-fixpoint } (\lambda x . (y * x \sqcup z) \sqcap v) (y^\omega_v \sqcup y^* * z)$$

*<proof>*

**lemma** *capped-postfixpoint*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies \text{is-postfixpoint } (\lambda x . (y * x \sqcup z) \sqcap v) (y^\omega_v \sqcup y^* * z)$$

*<proof>*

**lemma** *capped-greatest-postfixpoint*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies \text{is-greatest-postfixpoint } (\lambda x . (y * x \sqcup z) \sqcap v) (y^\omega_v \sqcup y^* * z)$$

*<proof>*

#### AACP Theorem 6.6

**lemma** *capped-nu*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies \nu(\lambda x . (y * x \sqcup z) \sqcap v) = y^\omega_v \sqcup y^* * z$$

*<proof>*

**lemma** *capped-pnu*:

$$y \leq u \implies z \leq v \implies u * v \leq v \implies p\nu(\lambda x . (y * x \sqcup z) \sqcap v) = y^\omega_v \sqcup y^* * z$$

*<proof>*



### AACP Theorem 6.7

**lemma** *unfold-capped-omega*:

$$y \leq u \implies u * v \leq v \implies y * y^\omega_v = y^\omega_v$$

*<proof>*

### AACP Theorem 6.8

**lemma** *star-mult-capped-omega*:

$$\begin{array}{l} \text{assumes } y \leq u \\ \text{and } u * v \leq v \\ \text{shows } y^* * y^\omega_v = y^\omega_v \end{array}$$

*<proof>*

### AACP Theorem 6.9

**lemma** *star-zero-below-capped-omega-zero*:

$$\begin{array}{l} \text{assumes } y \leq u \\ \text{and } u * v \leq v \\ \text{shows } y^* * \text{bot} \leq y^\omega_v * \text{bot} \end{array}$$

*<proof>*

**lemma** *star-zero-below-capped-omega*:

$$y \leq u \implies u * v \leq v \implies y^* * \text{bot} \leq y^\omega_v$$

*<proof>*

**lemma** *capped-omega-induct-meet-zero*:

$$x \leq y * x \sqcap v \implies x \leq y^\omega_v \sqcup y^* * \text{bot}$$

*<proof>*

### AACP Theorem 6.10

**lemma** *capped-omega-induct-meet*:

$$y \leq u \implies u * v \leq v \implies x \leq y * x \sqcap v \implies x \leq y^\omega_v$$

*<proof>*

**lemma** *capped-omega-induct-equal*:

$$x = (y * x \sqcup z) \sqcap v \implies x \leq y^\omega_v \sqcup y^* * z$$

*<proof>*

### AACP Theorem 6.11

**lemma** *capped-meet-nu*:

$$\begin{array}{l} \text{assumes } y \leq u \\ \text{and } u * v \leq v \\ \text{shows } \nu(\lambda x . y * x \sqcap v) = y^\omega_v \end{array}$$

*<proof>*

**lemma** *capped-meet-pnu*:

$$\begin{array}{l} \text{assumes } y \leq u \\ \text{and } u * v \leq v \\ \text{shows } p\nu(\lambda x . y * x \sqcap v) = y^\omega_v \end{array}$$

*<proof>*

### AACP Theorem 6.12

**lemma** *capped-omega-isotone*:

$$y \leq u \implies u * v \leq v \implies t \leq y \implies t^\omega_v \leq y^\omega_v$$

*<proof>*

### AACP Theorem 6.13

**lemma** *capped-omega-simulation*:

**assumes**  $y \leq u$

**and**  $s \leq u$

**and**  $u * v \leq v$

**and**  $s * t \leq y * s$

**shows**  $s * t^\omega_v \leq y^\omega_v$

*<proof>*

**lemma** *capped-omega-slide-sub*:

**assumes**  $s \leq u$

**and**  $y \leq u$

**and**  $u * u \leq u$

**and**  $u * v \leq v$

**shows**  $s * (y * s)^\omega_v \leq (s * y)^\omega_v$

*<proof>*

### AACP Theorem 6.14

**lemma** *capped-omega-slide*:

$$s \leq u \implies y \leq u \implies u * u \leq u \implies u * v \leq v \implies s * (y * s)^\omega_v = (s * y)^\omega_v$$

*<proof>*

**lemma** *capped-omega-sub-dist*:

$$s \leq u \implies y \leq u \implies u * v \leq v \implies s^\omega_v \leq (s \sqcup y)^\omega_v$$

*<proof>*

### AACP Theorem 6.15

**lemma** *capped-omega-simulation-2*:

**assumes**  $s \leq u$

**and**  $y \leq u$

**and**  $u * u \leq u$

**and**  $u * v \leq v$

**and**  $y * s \leq s * y$

**shows**  $(s * y)^\omega_v \leq s^\omega_v$

*<proof>*

### AACP Theorem 6.16

**lemma** *left-plus-capped-omega*:

**assumes**  $y \leq u$

**and**  $u * u \leq u$

**and**  $u * v \leq v$

**shows**  $(y * y^*)^\omega_v = y^\omega_v$

*<proof>*

### AACP Theorem 6.17

**lemma** *capped-omega-sub-vector*:

**assumes**  $z \leq v$

**and**  $y \leq u$

**and**  $u * v \leq v$

**shows**  $y^\omega_u * z \leq y^\omega_v$

*<proof>*

### AACP Theorem 6.18

**lemma** *capped-omega-omega*:

$y \leq u \implies u * v \leq v \implies (y^\omega_u)^\omega_v \leq y^\omega_v$

*<proof>*

**end**

**end**

## 4 General Refinement Algebras

**theory** *General-Refinement-Algebras*

**imports** *Omega-Algebras*

**begin**

**class** *general-refinement-algebra* = *left-kleene-algebra* + *Omega* +

**assumes** *Omega-unfold*:  $y^\Omega \leq 1 \sqcup y * y^\Omega$

**assumes** *Omega-induct*:  $x \leq z \sqcup y * x \longrightarrow x \leq y^\Omega * z$

**begin**

**lemma** *Omega-unfold-equal*:

$y^\Omega = 1 \sqcup y * y^\Omega$

*<proof>*

**lemma** *Omega-sup-1*:

$(x \sqcup y)^\Omega = x^\Omega * (y * x^\Omega)^\Omega$

*<proof>*

**lemma** *Omega-left-slide*:

$(x * y)^\Omega * x \leq x * (y * x)^\Omega$

*<proof>*

**end**

### Theorem 50.3

**sublocale** *general-refinement-algebra* < *Omega*: *left-conway-semiring* **where** *circ*

= *Omega*

*<proof>*

**context** *general-refinement-algebra*  
**begin**

**lemma** *star-below-Omega*:  
 $x^* \leq x^\Omega$   
 $\langle \text{proof} \rangle$

**lemma** *star-mult-Omega*:  
 $x^\Omega = x^* * x^\Omega$   
 $\langle \text{proof} \rangle$

**lemma** *Omega-one-greatest*:  
 $x \leq 1^\Omega$   
 $\langle \text{proof} \rangle$

**lemma** *greatest-left-zero*:  
 $1^\Omega * x = 1^\Omega$   
 $\langle \text{proof} \rangle$

**proposition** *circ-right-unfold*:  $1 \sqcup x^\Omega * x = x^\Omega$  **nitpick**  
 $[\text{expect}=\text{genuine}, \text{card}=8] \langle \text{proof} \rangle$

**proposition** *circ-slide*:  $(x * y)^\Omega * x = x * (y * x)^\Omega$  **nitpick**  
 $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate*:  $z * x \leq y * z \implies z * x^\Omega \leq y^\Omega * z$  **nitpick**  
 $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate-right*:  $z * x \leq y * z \sqcup w \implies z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$  **nitpick**  $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate-right-1*:  $z * x \leq y * z \implies z * x^\Omega \leq y^\Omega * z$  **nitpick**  
 $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate-right-plus*:  $z * x \leq y * y^\Omega * z \sqcup w \implies z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$  **nitpick**  $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate-right-plus-1*:  $z * x \leq y * y^\Omega * z \implies z * x^\Omega \leq y^\Omega * z$  **nitpick**  $[\text{expect}=\text{genuine}, \text{card}=6] \langle \text{proof} \rangle$

**proposition** *circ-simulate-left-1*:  $x * z \leq z * y \implies x^\Omega * z \leq z * y^\Omega \sqcup x^\Omega * \text{bot}$   
 $\langle \text{proof} \rangle$

**proposition** *circ-simulate-left-plus-1*:  $x * z \leq z * y^\Omega \implies x^\Omega * z \leq z * y^\Omega \sqcup x^\Omega * \text{bot}$   
 $\langle \text{proof} \rangle$

**proposition** *circ-simulate-absorb*:  $y * x \leq x \implies y^\Omega * x \leq x \sqcup y^\Omega * \text{bot}$  **nitpick**  
 $[\text{expect}=\text{genuine}, \text{card}=8] \langle \text{proof} \rangle$

**end**

**class** *bounded-general-refinement-algebra* = *general-refinement-algebra* +  
*bounded-left-kleene-algebra*

**begin**

**lemma** *Omega-one*:  
 $1^\Omega = \text{top}$

$\langle proof \rangle$

**lemma** *top-left-zero*:  
 $top * x = top$   
 $\langle proof \rangle$

**end**

**sublocale** *bounded-general-refinement-algebra*  $< \Omega$ :  
*bounded-left-conway-semiring* **where**  $circ = \Omega$   $\langle proof \rangle$

**class** *left-demonic-refinement-algebra* = *general-refinement-algebra* +  
**assumes** *Omega-isolate*:  $y^\Omega \leq y^\Omega * bot \sqcup y^*$   
**begin**

**lemma** *Omega-isolate-equal*:  
 $y^\Omega = y^\Omega * bot \sqcup y^*$   
 $\langle proof \rangle$

**proposition** *Omega-sum-unfold-1*:  $(x \sqcup y)^\Omega = y^\Omega \sqcup y^* * x * (x \sqcup y)^\Omega$   $\langle proof \rangle$   
**proposition** *Omega-sup-3*:  $(x \sqcup y)^\Omega = (x^* * y)^\Omega * x^\Omega$   $\langle proof \rangle$

**end**

**class** *bounded-left-demonic-refinement-algebra* = *left-demonic-refinement-algebra*  
+ *bounded-left-kleene-algebra*  
**begin**

**proposition** *Omega-mult*:  $(x * y)^\Omega = 1 \sqcup x * (y * x)^\Omega * y$   $\langle proof \rangle$   
**proposition** *Omega-sup*:  $(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$   $\langle proof \rangle$   
**proposition** *Omega-simulate*:  $z * x \leq y * z \implies z * x^\Omega \leq y^\Omega * z$  **nitpick**  
 $[expect=genuine,card=6]$   $\langle proof \rangle$   
**proposition** *Omega-separate-2*:  $y * x \leq x * (x \sqcup y) \implies (x \sqcup y)^\Omega = x^\Omega * y^\Omega$   
 $\langle proof \rangle$   
**proposition** *Omega-circ-simulate-right-plus*:  $z * x \leq y * y^\Omega * z \sqcup w \implies z * x^\Omega$   
 $\leq y^\Omega * (z \sqcup w * x^\Omega)$  **nitpick**  $[expect=genuine,card=6]$   $\langle proof \rangle$   
**proposition** *Omega-circ-simulate-left-plus*:  $x * z \leq z * y^\Omega \sqcup w \implies x^\Omega * z \leq (z$   
 $\sqcup x^\Omega * w) * y^\Omega$   $\langle proof \rangle$

**end**

**sublocale** *bounded-left-demonic-refinement-algebra*  $< \Omega$ :  
*bounded-left-conway-semiring* **where**  $circ = \Omega$   $\langle proof \rangle$

**class** *demonic-refinement-algebra* = *left-zero-kleene-algebra* +  
*left-demonic-refinement-algebra*  
**begin**

**lemma** *Omega-mult*:

$$(x * y)^\Omega = 1 \sqcup x * (y * x)^\Omega * y$$

*<proof>*

**lemma** *Omega-sup:*

$$(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$$

*<proof>*

**lemma** *Omega-simulate:*

$$z * x \leq y * z \implies z * x^\Omega \leq y^\Omega * z$$

*<proof>*

**end**

### Theorem 2.4

**sublocale** *demonic-refinement-algebra* < *Omega1: iterating-1* **where** *circ = Omega*

*<proof>*

**sublocale** *demonic-refinement-algebra* < *Omega1: left-zero-conway-semiring-1*  
**where** *circ = Omega* *<proof>*

**context** *demonic-refinement-algebra*  
**begin**

**lemma** *Omega-sum-unfold-1:*

$$(x \sqcup y)^\Omega = y^\Omega \sqcup y^* * x * (x \sqcup y)^\Omega$$

*<proof>*

**lemma** *Omega-sup-3:*

$$(x \sqcup y)^\Omega = (x^* * y)^\Omega * x^\Omega$$

*<proof>*

**lemma** *Omega-separate-2:*

$$y * x \leq x * (x \sqcup y) \implies (x \sqcup y)^\Omega = x^\Omega * y^\Omega$$

*<proof>*

**lemma** *Omega-circ-simulate-right-plus:*

**assumes**  $z * x \leq y * y^\Omega * z \sqcup w$   
**shows**  $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$

*<proof>*

**lemma** *Omega-circ-simulate-left-plus:*

**assumes**  $x * z \leq z * y^\Omega \sqcup w$   
**shows**  $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$

*<proof>*

**lemma** *Omega-circ-simulate-right:*

**assumes**  $z * x \leq y * z \sqcup w$   
**shows**  $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$

*<proof>*

**end**

**sublocale** *demonic-refinement-algebra* < *Omega*: *itering* **where** *circ* = *Omega*  
*<proof>*

**class** *bounded-demonic-refinement-algebra* = *demonic-refinement-algebra* +  
*bounded-left-zero-kleene-algebra*  
**begin**

**lemma** *Omega-one*:  
 $1^\Omega = \text{top}$   
*<proof>*

**lemma** *top-left-zero*:  
 $\text{top} * x = \text{top}$   
*<proof>*

**end**

**sublocale** *bounded-demonic-refinement-algebra* < *Omega*: *bounded-itering* **where**  
*circ* = *Omega* *<proof>*

**class** *general-refinement-algebra-omega* = *left-omega-algebra* + *Omega* +  
**assumes** *omega-left-zero*:  $x^\omega \leq x^\omega * y$   
**assumes** *Omega-def*:  $x^\Omega = x^\omega \sqcup x^*$   
**begin**

**lemma** *omega-left-zero-equal*:  
 $x^\omega * y = x^\omega$   
*<proof>*

**subclass** *left-demonic-refinement-algebra*  
*<proof>*

**end**

**class** *left-demonic-refinement-algebra-omega* = *bounded-left-omega-algebra* +  
*Omega* +  
**assumes** *top-left-zero*:  $\text{top} * x = \text{top}$   
**assumes** *Omega-def*:  $x^\Omega = x^\omega \sqcup x^*$   
**begin**

**subclass** *general-refinement-algebra-omega*  
*<proof>*

**end**

```

class demonic-refinement-algebra-omega = left-demonic-refinement-algebra-omega
+ bounded-left-zero-omega-algebra
begin

lemma Omega-mult:
   $(x * y)^\Omega = 1 \sqcup x * (y * x) * y$ 
  <proof>

lemma Omega-sup:
   $(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$ 
  <proof>

lemma Omega-simulate:
   $z * x \leq y * z \implies z * x^\Omega \leq y^\Omega * z$ 
  <proof>

subclass demonic-refinement-algebra <proof>

end

proposition circ-circ-mult:  $1^\Omega * x^\Omega = x^{\Omega\Omega}$  <proof>
proposition sub-mult-one-circ:  $x * 1^\Omega \leq 1^\Omega * x$  <proof>
proposition circ-circ-mult-1:  $x^\Omega * 1^\Omega = x^{\Omega\Omega}$  <proof>
proposition  $y * x \leq x \implies y^\circ * x \leq 1^\circ * x$  <proof>

proposition circ-simulate-2:  $y * x^\Omega \leq x^\Omega * y^\Omega \iff y^\Omega * x^\Omega \leq x^\Omega * y^\Omega$  <proof>
proposition circ-simulate-3:  $y * x^\Omega \leq x^\Omega \implies y^\Omega * x^\Omega \leq x^\Omega * y^\Omega$  <proof>
proposition circ-separate-mult-1:  $y * x \leq x * y \implies (x * y)^\Omega \leq x^\Omega * y^\Omega$  <proof>
proposition  $x^\circ = (x * x)^\circ * (x \sqcup 1)$  <proof>
proposition  $y^\circ * x^\circ \leq x^\circ * y^\circ \implies (x \sqcup y)^\circ = x^\circ * y^\circ$  <proof>
proposition  $y * x \leq (1 \sqcup x) * y^\circ \implies (x \sqcup y)^\circ = x^\circ * y^\circ$  <proof>

```

**end**

## 5 Lattice-Ordered Semirings

**theory** Lattice-Ordered-Semirings

**imports** Stone-Relation-Algebras.Semirings

**begin**

**nitpick-params** [timeout = 600]

Many results in this theory are taken from a joint paper with Rudolf Berghammer.

M0-algebra



```

class lattice-ordered-pre-left-semiring = pre-left-semiring + bounded-distrib-lattice
begin

subclass bounded-pre-left-semiring
  ⟨proof⟩

lemma top-mult-right-one:
   $x * top = x * top * 1$ 
  ⟨proof⟩

lemma mult-left-sub-dist-inf-left:
   $x * (y \sqcap z) \leq x * y$ 
  ⟨proof⟩

lemma mult-left-sub-dist-inf-right:
   $x * (y \sqcap z) \leq x * z$ 
  ⟨proof⟩

lemma mult-right-sub-dist-inf-left:
   $(x \sqcap y) * z \leq x * z$ 
  ⟨proof⟩

lemma mult-right-sub-dist-inf-right:
   $(x \sqcap y) * z \leq y * z$ 
  ⟨proof⟩

lemma mult-right-sub-dist-inf:
   $(x \sqcap y) * z \leq x * z \sqcap y * z$ 
  ⟨proof⟩

```

Figure 1: fundamental properties

```

definition co-total      :: 'a ⇒ bool where co-total x ≡ x * bot = bot
definition up-closed    :: 'a ⇒ bool where up-closed x ≡ x * 1 = x
definition sup-distributive :: 'a ⇒ bool where sup-distributive x ≡ (∀ y z . x *
(y ∪ z) = x * y ∪ x * z)
definition inf-distributive :: 'a ⇒ bool where inf-distributive x ≡ (∀ y z . x * (y
∩ z) = x * y ∩ x * z)
definition contact      :: 'a ⇒ bool where contact x ≡ x * x ∪ 1 = x
definition kernel       :: 'a ⇒ bool where kernel x ≡ x * x ∩ 1 = x * 1
definition sup-dist-contact :: 'a ⇒ bool where sup-dist-contact x ≡
sup-distributive x ∧ contact x
definition inf-dist-kernel :: 'a ⇒ bool where inf-dist-kernel x ≡ inf-distributive
x ∧ kernel x
definition test         :: 'a ⇒ bool where test x ≡ x * top ∩ 1 = x
definition co-test      :: 'a ⇒ bool where co-test x ≡ x * bot ∪ 1 = x
definition co-vector    :: 'a ⇒ bool where co-vector x ≡ x * bot = x

```

AAMP Theorem 6 / Figure 2: relations between properties

```

lemma reflexive-total:

```

*reflexive*  $x \implies$  *total*  $x$   
(proof)

**lemma** *reflexive-dense*:  
*reflexive*  $x \implies$  *dense-rel*  $x$   
(proof)

**lemma** *reflexive-transitive-up-closed*:  
*reflexive*  $x \implies$  *transitive*  $x \implies$  *up-closed*  $x$   
(proof)

**lemma** *coreflexive-co-total*:  
*coreflexive*  $x \implies$  *co-total*  $x$   
(proof)

**lemma** *coreflexive-transitive*:  
*coreflexive*  $x \implies$  *transitive*  $x$   
(proof)

**lemma** *idempotent-transitive-dense*:  
*idempotent*  $x \iff$  *transitive*  $x \wedge$  *dense-rel*  $x$   
(proof)

**lemma** *contact-reflexive*:  
*contact*  $x \implies$  *reflexive*  $x$   
(proof)

**lemma** *contact-transitive*:  
*contact*  $x \implies$  *transitive*  $x$   
(proof)

**lemma** *contact-dense*:  
*contact*  $x \implies$  *dense-rel*  $x$   
(proof)

**lemma** *contact-idempotent*:  
*contact*  $x \implies$  *idempotent*  $x$   
(proof)

**lemma** *contact-up-closed*:  
*contact*  $x \implies$  *up-closed*  $x$   
(proof)

**lemma** *contact-reflexive-idempotent-up-closed*:  
*contact*  $x \iff$  *reflexive*  $x \wedge$  *idempotent*  $x \wedge$  *up-closed*  $x$   
(proof)

**lemma** *kernel-coreflexive*:  
*kernel*  $x \implies$  *coreflexive*  $x$

$\langle \text{proof} \rangle$

**lemma** *kernel-transitive:*

$\text{kernel } x \implies \text{transitive } x$

$\langle \text{proof} \rangle$

**lemma** *kernel-dense:*

$\text{kernel } x \implies \text{dense-rel } x$

$\langle \text{proof} \rangle$

**lemma** *kernel-idempotent:*

$\text{kernel } x \implies \text{idempotent } x$

$\langle \text{proof} \rangle$

**lemma** *kernel-up-closed:*

$\text{kernel } x \implies \text{up-closed } x$

$\langle \text{proof} \rangle$

**lemma** *kernel-coreflexive-idempotent-up-closed:*

$\text{kernel } x \iff \text{coreflexive } x \wedge \text{idempotent } x \wedge \text{up-closed } x$

$\langle \text{proof} \rangle$

**lemma** *test-coreflexive:*

$\text{test } x \implies \text{coreflexive } x$

$\langle \text{proof} \rangle$

**lemma** *test-up-closed:*

$\text{test } x \implies \text{up-closed } x$

$\langle \text{proof} \rangle$

**lemma** *co-test-reflexive:*

$\text{co-test } x \implies \text{reflexive } x$

$\langle \text{proof} \rangle$

**lemma** *co-test-transitive:*

$\text{co-test } x \implies \text{transitive } x$

$\langle \text{proof} \rangle$

**lemma** *co-test-idempotent:*

$\text{co-test } x \implies \text{idempotent } x$

$\langle \text{proof} \rangle$

**lemma** *co-test-up-closed:*

$\text{co-test } x \implies \text{up-closed } x$

$\langle \text{proof} \rangle$

**lemma** *co-test-contact:*

$\text{co-test } x \implies \text{contact } x$

$\langle \text{proof} \rangle$

**lemma** *vector-transitive*:  
*vector*  $x \implies$  *transitive*  $x$   
*<proof>*

**lemma** *vector-up-closed*:  
*vector*  $x \implies$  *up-closed*  $x$   
*<proof>*

AAMP Theorem 10 / Figure 3: closure properties

**total**

**lemma** *one-total*:  
*total*  $1$   
*<proof>*

**lemma** *top-total*:  
*total*  $top$   
*<proof>*

**lemma** *sup-total*:  
*total*  $x \implies$  *total*  $y \implies$  *total*  $(x \sqcup y)$   
*<proof>*

**co-total**

**lemma** *zero-co-total*:  
*co-total*  $bot$   
*<proof>*

**lemma** *one-co-total*:  
*co-total*  $1$   
*<proof>*

**lemma** *sup-co-total*:  
*co-total*  $x \implies$  *co-total*  $y \implies$  *co-total*  $(x \sqcup y)$   
*<proof>*

**lemma** *inf-co-total*:  
*co-total*  $x \implies$  *co-total*  $y \implies$  *co-total*  $(x \sqcap y)$   
*<proof>*

**lemma** *comp-co-total*:  
*co-total*  $x \implies$  *co-total*  $y \implies$  *co-total*  $(x * y)$   
*<proof>*

**sub-transitive**

**lemma** *zero-transitive*:  
*transitive*  $bot$   
*<proof>*

**lemma** *one-transitive*:

*transitive 1*

*<proof>*

**lemma** *top-transitive*:

*transitive top*

*<proof>*

**lemma** *inf-transitive*:

*transitive x  $\implies$  transitive y  $\implies$  transitive (x  $\sqcap$  y)*

*<proof>*

**dense**

**lemma** *zero-dense*:

*dense-rel bot*

*<proof>*

**lemma** *one-dense*:

*dense-rel 1*

*<proof>*

**lemma** *top-dense*:

*dense-rel top*

*<proof>*

**lemma** *sup-dense*:

**assumes** *dense-rel x*

**and** *dense-rel y*

**shows** *dense-rel (x  $\sqcup$  y)*

*<proof>*

**reflexive**

**lemma** *one-reflexive*:

*reflexive 1*

*<proof>*

**lemma** *top-reflexive*:

*reflexive top*

*<proof>*

**lemma** *sup-reflexive*:

*reflexive x  $\implies$  reflexive y  $\implies$  reflexive (x  $\sqcup$  y)*

*<proof>*

**lemma** *inf-reflexive*:

*reflexive x  $\implies$  reflexive y  $\implies$  reflexive (x  $\sqcap$  y)*

*<proof>*

**lemma** *comp-reflexive*:

*reflexive*  $x \implies \text{reflexive } y \implies \text{reflexive } (x * y)$   
<proof>

### co-reflexive

**lemma** *zero-coreflexive*:  
*coreflexive bot*  
<proof>

**lemma** *one-coreflexive*:  
*coreflexive 1*  
<proof>

**lemma** *sup-coreflexive*:  
*coreflexive*  $x \implies \text{coreflexive } y \implies \text{coreflexive } (x \sqcup y)$   
<proof>

**lemma** *inf-coreflexive*:  
*coreflexive*  $x \implies \text{coreflexive } y \implies \text{coreflexive } (x \sqcap y)$   
<proof>

**lemma** *comp-coreflexive*:  
*coreflexive*  $x \implies \text{coreflexive } y \implies \text{coreflexive } (x * y)$   
<proof>

### idempotent

**lemma** *zero-idempotent*:  
*idempotent bot*  
<proof>

**lemma** *one-idempotent*:  
*idempotent 1*  
<proof>

**lemma** *top-idempotent*:  
*idempotent top*  
<proof>

### up-closed

**lemma** *zero-up-closed*:  
*up-closed bot*  
<proof>

**lemma** *one-up-closed*:  
*up-closed 1*  
<proof>

**lemma** *top-up-closed*:  
*up-closed top*  
<proof>

**lemma** *sup-up-closed*:

*up-closed*  $x \implies$  *up-closed*  $y \implies$  *up-closed*  $(x \sqcup y)$   
*<proof>*

**lemma** *inf-up-closed*:

*up-closed*  $x \implies$  *up-closed*  $y \implies$  *up-closed*  $(x \sqcap y)$   
*<proof>*

**lemma** *comp-up-closed*:

*up-closed*  $x \implies$  *up-closed*  $y \implies$  *up-closed*  $(x * y)$   
*<proof>*

**add-distributive**

**lemma** *zero-sup-distributive*:

*sup-distributive bot*  
*<proof>*

**lemma** *one-sup-distributive*:

*sup-distributive 1*  
*<proof>*

**lemma** *sup-sup-distributive*:

*sup-distributive*  $x \implies$  *sup-distributive*  $y \implies$  *sup-distributive*  $(x \sqcup y)$   
*<proof>*

**inf-distributive**

**lemma** *zero-inf-distributive*:

*inf-distributive bot*  
*<proof>*

**lemma** *one-inf-distributive*:

*inf-distributive 1*  
*<proof>*

**contact**

**lemma** *one-contact*:

*contact 1*  
*<proof>*

**lemma** *top-contact*:

*contact top*  
*<proof>*

**lemma** *inf-contact*:

*contact*  $x \implies$  *contact*  $y \implies$  *contact*  $(x \sqcap y)$   
*<proof>*

**kernel**

**lemma** *zero-kernel*:

*kernel bot*

*<proof>*

**lemma** *one-kernel*:

*kernel 1*

*<proof>*

**lemma** *sup-kernel*:

*kernel x  $\implies$  kernel y  $\implies$  kernel (x  $\sqcup$  y)*

*<proof>*

**add-distributive contact**

**lemma** *one-sup-dist-contact*:

*sup-dist-contact 1*

*<proof>*

**inf-distributive kernel**

**lemma** *zero-inf-dist-kernel*:

*inf-dist-kernel bot*

*<proof>*

**lemma** *one-inf-dist-kernel*:

*inf-dist-kernel 1*

*<proof>*

**test**

**lemma** *zero-test*:

*test bot*

*<proof>*

**lemma** *one-test*:

*test 1*

*<proof>*

**lemma** *sup-test*:

*test x  $\implies$  test y  $\implies$  test (x  $\sqcup$  y)*

*<proof>*

**lemma** *inf-test*:

*test x  $\implies$  test y  $\implies$  test (x  $\sqcap$  y)*

*<proof>*

**co-test**

**lemma** *one-co-test*:

*co-test 1*

*<proof>*

**lemma** *sup-co-test*:



```

    co-test x ==> co-test y ==> co-test (x ⊔ y)
    ⟨proof⟩

    vector

lemma zero-vector:
  vector bot
  ⟨proof⟩

lemma top-vector:
  vector top
  ⟨proof⟩

lemma sup-vector:
  vector x ==> vector y ==> vector (x ⊔ y)
  ⟨proof⟩

lemma inf-vector:
  vector x ==> vector y ==> vector (x ⊓ y)
  ⟨proof⟩

lemma comp-vector:
  vector y ==> vector (x * y)
  ⟨proof⟩

end

class lattice-ordered-pre-left-semiring-1 = non-associative-left-semiring +
  bounded-distrib-lattice +
  assumes mult-associative-one: x * (y * z) = (x * (y * 1)) * z
  assumes mult-right-dist-inf-one: (x * 1 ⊓ y * 1) * z = x * z ⊓ y * z
begin

subclass pre-left-semiring
  ⟨proof⟩

subclass lattice-ordered-pre-left-semiring ⟨proof⟩

lemma mult-zero-associative:
  x * bot * y = x * bot
  ⟨proof⟩

lemma mult-zero-sup-one-dist:
  (x * bot ⊔ 1) * z = x * bot ⊔ z
  ⟨proof⟩

lemma mult-zero-sup-dist:
  (x * bot ⊔ y) * z = x * bot ⊔ y * z
  ⟨proof⟩

```

**lemma** *vector-zero-inf-one-comp*:  
 $(x * \text{bot} \sqcap 1) * y = x * \text{bot} \sqcap y$   
 ⟨proof⟩

AAMP Theorem 6 / Figure 2: relations between properties

**lemma** *co-test-inf-distributive*:  
 $\text{co-test } x \implies \text{inf-distributive } x$   
 ⟨proof⟩

**lemma** *co-test-sup-distributive*:  
 $\text{co-test } x \implies \text{sup-distributive } x$   
 ⟨proof⟩

**lemma** *co-test-sup-dist-contact*:  
 $\text{co-test } x \implies \text{sup-dist-contact } x$   
 ⟨proof⟩

AAMP Theorem 10 / Figure 3: closure properties

**co-test**

**lemma** *inf-co-test*:  
 $\text{co-test } x \implies \text{co-test } y \implies \text{co-test } (x \sqcap y)$   
 ⟨proof⟩

**lemma** *comp-co-test*:  
 $\text{co-test } x \implies \text{co-test } y \implies \text{co-test } (x * y)$   
 ⟨proof⟩

**end**

**class** *lattice-ordered-pre-left-semiring-2* = *lattice-ordered-pre-left-semiring* +  
**assumes** *mult-sub-associative-one*:  $x * (y * z) \leq (x * (y * 1)) * z$   
**assumes** *mult-right-dist-inf-one-sub*:  $x * z \sqcap y * z \leq (x * 1 \sqcap y * 1) * z$   
**begin**

**subclass** *lattice-ordered-pre-left-semiring-1*  
 ⟨proof⟩

**end**

**class** *multirelation-algebra-1* = *lattice-ordered-pre-left-semiring* +  
**assumes** *mult-left-top*:  $\text{top} * x = \text{top}$   
**begin**

AAMP Theorem 10 / Figure 3: closure properties

**lemma** *top-sup-distributive*:  
 $\text{sup-distributive } \text{top}$   
 ⟨proof⟩

**lemma** *top-inf-distributive*:

*inf-distributive top*  
*<proof>*

**lemma** *top-sup-dist-contact:*  
*sup-dist-contact top*  
*<proof>*

**lemma** *top-co-test:*  
*co-test top*  
*<proof>*

**end**

### M1-algebra

**class** *multirelation-algebra-2 = multirelation-algebra-1 +*  
*lattice-ordered-pre-left-semiring-2*  
**begin**

**lemma** *mult-top-associative:*  
 $x * top * y = x * top$   
*<proof>*

**lemma** *vector-inf-one-comp:*  
 $(x * top \sqcap 1) * y = x * top \sqcap y$   
*<proof>*

**lemma** *vector-left-annihilator:*  
 $vector\ x \implies x * y = x$   
*<proof>*

### properties

**lemma** *test-comp-inf:*  
 $test\ x \implies test\ y \implies x * y = x \sqcap y$   
*<proof>*

### AAMP Theorem 6 / Figure 2: relations between properties

**lemma** *test-sup-distributive:*  
 $test\ x \implies sup-distributive\ x$   
*<proof>*

**lemma** *test-inf-distributive:*  
 $test\ x \implies inf-distributive\ x$   
*<proof>*

**lemma** *test-inf-dist-kernel:*  
 $test\ x \implies inf-dist-kernel\ x$   
*<proof>*

**lemma** *vector-idempotent:*

*vector*  $x \implies$  *idempotent*  $x$   
*<proof>*

**lemma** *vector-sup-distributive*:  
*vector*  $x \implies$  *sup-distributive*  $x$   
*<proof>*

**lemma** *vector-inf-distributive*:  
*vector*  $x \implies$  *inf-distributive*  $x$   
*<proof>*

**lemma** *vector-co-vector*:  
*vector*  $x \longleftrightarrow$  *co-vector*  $x$   
*<proof>*

AAMP Theorem 10 / Figure 3: closure properties

*test*

**lemma** *comp-test*:  
*test*  $x \implies$  *test*  $y \implies$  *test*  $(x * y)$   
*<proof>*

**end**

**class** *dual* =  
  **fixes** *dual* :: 'a  $\Rightarrow$  'a ( $\leftarrow^d$ ) [100] 100)

**class** *multirelation-algebra-3* = *lattice-ordered-pre-left-semiring* + *dual* +  
  **assumes** *dual-involutive*:  $x^{dd} = x$   
  **assumes** *dual-dist-sup*:  $(x \sqcup y)^d = x^d \sqcap y^d$   
  **assumes** *dual-one*:  $1^d = 1$

**begin**

**lemma** *dual-dist-inf*:  
 $(x \sqcap y)^d = x^d \sqcup y^d$   
*<proof>*

**lemma** *dual-antitone*:  
 $x \leq y \implies y^d \leq x^d$   
*<proof>*

**lemma** *dual-zero*:  
 $bot^d = top$   
*<proof>*

**lemma** *dual-top*:  
 $top^d = bot$   
*<proof>*

AAMP Theorem 10 / Figure 3: closure properties

**lemma** *reflexive-coreflexive-dual*:  
*reflexive*  $x \longleftrightarrow$  *coreflexive*  $(x^d)$   
 ⟨*proof*⟩

**end**

**class** *multirelation-algebra-4* = *multirelation-algebra-3* +  
**assumes** *dual-sub-dist-comp*:  $(x * y)^d \leq x^d * y^d$   
**begin**

**subclass** *multirelation-algebra-1*  
 ⟨*proof*⟩

**lemma** *dual-sub-dist-comp-one*:  
 $(x * y)^d \leq (x * 1)^d * y^d$   
 ⟨*proof*⟩

AAMP Theorem 10 / Figure 3: closure properties

**lemma** *co-total-total-dual*:  
*co-total*  $x \implies$  *total*  $(x^d)$   
 ⟨*proof*⟩

**lemma** *transitive-dense-dual*:  
*transitive*  $x \implies$  *dense-rel*  $(x^d)$   
 ⟨*proof*⟩

**end**

M2-algebra

**class** *multirelation-algebra-5* = *multirelation-algebra-3* +  
**assumes** *dual-dist-comp-one*:  $(x * y)^d = (x * 1)^d * y^d$   
**begin**

**subclass** *multirelation-algebra-4*  
 ⟨*proof*⟩

**lemma** *strong-up-closed*:  
 $x * 1 \leq x \implies x^d * y^d \leq (x * y)^d$   
 ⟨*proof*⟩

**lemma** *strong-up-closed-2*:  
*up-closed*  $x \implies (x * y)^d = x^d * y^d$   
 ⟨*proof*⟩

**subclass** *lattice-ordered-pre-left-semiring-2*  
 ⟨*proof*⟩

AAMP Theorem 8

**subclass** *multirelation-algebra-2* ⟨*proof*⟩

AAMP Theorem 10 / Figure 3: closure properties

up-closed

**lemma** *dual-up-closed*:  
 $up\text{-closed } x \longleftrightarrow up\text{-closed } (x^d)$   
*<proof>*

contact

**lemma** *contact-kernel-dual*:  
 $contact\ x \longleftrightarrow kernel\ (x^d)$   
*<proof>*

add-distributive contact

**lemma** *sup-dist-contact-inf-dist-kernel-dual*:  
 $sup\text{-dist-contact } x \longleftrightarrow inf\text{-dist-kernel } (x^d)$   
*<proof>*

test

**lemma** *test-co-test-dual*:  
 $test\ x \longleftrightarrow co\text{-test } (x^d)$   
*<proof>*

vector

**lemma** *vector-dual*:  
 $vector\ x \longleftrightarrow vector\ (x^d)$   
*<proof>*

**end**

**class** *multirelation-algebra-6* = *multirelation-algebra-4* +  
  **assumes** *dual-sub-dist-comp-one*:  $(x * 1)^d * y^d \leq (x * y)^d$   
**begin**

**subclass** *multirelation-algebra-5*  
  *<proof>*

**proposition** *dense-rel x ∧ coreflexive x → up-closed x nitpick*  
[*expect=genuine,card=5*] *<proof>*

**proposition**  $x * top \sqcap y * z \leq (x * top \sqcap y) * z$  **nitpick**  
[*expect=genuine,card=8*] *<proof>*

**end**

M3-algebra

**class** *up-closed-multirelation-algebra* = *multirelation-algebra-3* +  
  **assumes** *dual-dist-comp*:  $(x * y)^d = x^d * y^d$   
**begin**

**lemma** *mult-right-dist-inf*:

$(x \sqcap y) * z = x * z \sqcap y * z$   
<proof>

**AAMP Theorem 9**

**subclass** *idempotent-left-semiring*  
<proof>

**subclass** *multirelation-algebra-6*  
<proof>

**lemma** *vector-inf-comp*:  
 $(x * \text{top} \sqcap y) * z = x * \text{top} \sqcap y * z$   
<proof>

**lemma** *vector-zero-inf-comp*:  
 $(x * \text{bot} \sqcap y) * z = x * \text{bot} \sqcap y * z$   
<proof>

**AAMP Theorem 10 / Figure 3: closure properties**

**total**

**lemma** *inf-total*:  
 $\text{total } x \implies \text{total } y \implies \text{total } (x \sqcap y)$   
<proof>

**lemma** *comp-total*:  
 $\text{total } x \implies \text{total } y \implies \text{total } (x * y)$   
<proof>

**lemma** *total-co-total-dual*:  
 $\text{total } x \longleftrightarrow \text{co-total } (x^d)$   
<proof>

**dense**

**lemma** *transitive-iff-dense-dual*:  
 $\text{transitive } x \longleftrightarrow \text{dense-rel } (x^d)$   
<proof>

**idempotent**

**lemma** *idempotent-dual*:  
 $\text{idempotent } x \longleftrightarrow \text{idempotent } (x^d)$   
<proof>

**add-distributive**

**lemma** *comp-sup-distributive*:  
 $\text{sup-distributive } x \implies \text{sup-distributive } y \implies \text{sup-distributive } (x * y)$   
<proof>

**lemma** *sup-inf-distributive-dual*:

*sup-distributive*  $x \iff \text{inf-distributive } (x^d)$   
*<proof>*

### inf-distributive

**lemma** *inf-inf-distributive:*

*inf-distributive*  $x \implies \text{inf-distributive } y \implies \text{inf-distributive } (x \sqcap y)$   
*<proof>*

**lemma** *comp-inf-distributive:*

*inf-distributive*  $x \implies \text{inf-distributive } y \implies \text{inf-distributive } (x * y)$   
*<proof>*

**proposition** *co-total*  $x \wedge \text{transitive } x \wedge \text{up-closed } x \longrightarrow \text{coreflexive } x$  **nitpick**

[*expect=genuine,card=5*] *<proof>*

**proposition** *total*  $x \wedge \text{dense-rel } x \wedge \text{up-closed } x \longrightarrow \text{reflexive } x$  **nitpick**

[*expect=genuine,card=5*] *<proof>*

**proposition**  $x * \text{top} \sqcap x^d * \text{bot} = \text{bot}$  **nitpick** [*expect=genuine,card=6*] *<proof>*

**end**

**class** *multirelation-algebra-7* = *multirelation-algebra-4* +

**assumes** *vector-inf-comp*:  $(x * \text{top} \sqcap y) * z = x * \text{top} \sqcap y * z$

**begin**

**lemma** *vector-zero-inf-comp:*

$(x * \text{bot} \sqcap y) * z = x * \text{bot} \sqcap y * z$   
*<proof>*

**lemma** *test-sup-distributive:*

*test*  $x \implies \text{sup-distributive } x$   
*<proof>*

**lemma** *test-inf-distributive:*

*test*  $x \implies \text{inf-distributive } x$   
*<proof>*

**lemma** *test-inf-dist-kernel:*

*test*  $x \implies \text{inf-dist-kernel } x$   
*<proof>*

**lemma** *co-test-inf-distributive:*

**assumes** *co-test*  $x$

**shows** *inf-distributive*  $x$

*<proof>*

**lemma** *co-test-sup-distributive:*

**assumes** *co-test*  $x$

**shows** *sup-distributive*  $x$

*<proof>*



**lemma** *co-test-sup-dist-contact*:  
*co-test*  $x \implies \text{sup-dist-contact } x$   
 ⟨*proof*⟩

**end**

**end**

## 6 Boolean Semirings

**theory** *Boolean-Semirings*

**imports** *Stone-Algebras.P-Algebras Lattice-Ordered-Semirings*

**begin**

**class** *complemented-distributive-lattice* = *bounded-distrib-lattice* + *uminus* +  
**assumes** *inf-complement*:  $x \sqcap (-x) = \text{bot}$   
**assumes** *sup-complement*:  $x \sqcup (-x) = \text{top}$

**begin**

**sublocale** *boolean-algebra* **where** *minus* =  $\lambda x y . x \sqcap (-y)$  **and** *inf* = *inf* **and**  
*sup* = *sup* **and** *bot* = *bot* **and** *top* = *top*  
 ⟨*proof*⟩

**end**

[M0-algebra](#)

**context** *lattice-ordered-pre-left-semiring*

**begin**

[Section 7](#)

**lemma** *vector-1*:

*vector*  $x \longleftrightarrow x * \text{top} \leq x$   
 ⟨*proof*⟩

**definition** *zero-vector* ::  $'a \Rightarrow \text{bool}$  **where** *zero-vector*  $x \equiv x \leq x * \text{bot}$

**definition** *one-vector* ::  $'a \Rightarrow \text{bool}$  **where** *one-vector*  $x \equiv x * \text{bot} \leq x$

**lemma** *zero-vector-left-zero*:

**assumes** *zero-vector*  $x$   
**shows**  $x * y = x * \text{bot}$

⟨*proof*⟩

**lemma** *zero-vector-1*:

*zero-vector*  $x \longleftrightarrow (\forall y . x * y = x * \text{bot})$   
 ⟨*proof*⟩

**lemma** *zero-vector-2*:  
*zero-vector*  $x \longleftrightarrow (\forall y . x * y \leq x * \text{bot})$   
(*proof*)

**lemma** *zero-vector-3*:  
*zero-vector*  $x \longleftrightarrow x * 1 = x * \text{bot}$   
(*proof*)

**lemma** *zero-vector-4*:  
*zero-vector*  $x \longleftrightarrow x * 1 \leq x * \text{bot}$   
(*proof*)

**lemma** *zero-vector-5*:  
*zero-vector*  $x \longleftrightarrow x * \text{top} = x * \text{bot}$   
(*proof*)

**lemma** *zero-vector-6*:  
*zero-vector*  $x \longleftrightarrow x * \text{top} \leq x * \text{bot}$   
(*proof*)

**lemma** *zero-vector-7*:  
*zero-vector*  $x \longleftrightarrow (\forall y . x * \text{top} = x * y)$   
(*proof*)

**lemma** *zero-vector-8*:  
*zero-vector*  $x \longleftrightarrow (\forall y . x * \text{top} \leq x * y)$   
(*proof*)

**lemma** *zero-vector-9*:  
*zero-vector*  $x \longleftrightarrow (\forall y . x * 1 = x * y)$   
(*proof*)

**lemma** *zero-vector-0*:  
*zero-vector*  $x \longleftrightarrow (\forall y z . x * y = x * z)$   
(*proof*)

**Theorem 6 / Figure 2: relations between properties**

**lemma** *co-vector-zero-vector-one-vector*:  
*co-vector*  $x \longleftrightarrow \text{zero-vector } x \wedge \text{one-vector } x$   
(*proof*)

**lemma** *up-closed-one-vector*:  
*up-closed*  $x \implies \text{one-vector } x$   
(*proof*)

**lemma** *zero-vector-dense*:  
*zero-vector*  $x \implies \text{dense-rel } x$   
(*proof*)

**lemma** *zero-vector-sup-distributive:*  
 $zero\text{-}vector\ x \implies sup\text{-}distributive\ x$   
(proof)

**lemma** *zero-vector-inf-distributive:*  
 $zero\text{-}vector\ x \implies inf\text{-}distributive\ x$   
(proof)

**lemma** *up-closed-zero-vector-vector:*  
 $up\text{-}closed\ x \implies zero\text{-}vector\ x \implies vector\ x$   
(proof)

**lemma** *zero-vector-one-vector-vector:*  
 $zero\text{-}vector\ x \implies one\text{-}vector\ x \implies vector\ x$   
(proof)

**lemma** *co-vector-vector:*  
 $co\text{-}vector\ x \implies vector\ x$   
(proof)

Theorem 10 / Figure 3: closure properties

zero-vector

**lemma** *zero-zero-vector:*  
 $zero\text{-}vector\ bot$   
(proof)

**lemma** *sup-zero-vector:*  
 $zero\text{-}vector\ x \implies zero\text{-}vector\ y \implies zero\text{-}vector\ (x \sqcup y)$   
(proof)

**lemma** *comp-zero-vector:*  
 $zero\text{-}vector\ x \implies zero\text{-}vector\ y \implies zero\text{-}vector\ (x * y)$   
(proof)

one-vector

**lemma** *zero-one-vector:*  
 $one\text{-}vector\ bot$   
(proof)

**lemma** *one-one-vector:*  
 $one\text{-}vector\ 1$   
(proof)

**lemma** *top-one-vector:*  
 $one\text{-}vector\ top$   
(proof)

**lemma** *sup-one-vector:*  
 $one\text{-}vector\ x \implies one\text{-}vector\ y \implies one\text{-}vector\ (x \sqcup y)$

*<proof>*

**lemma** *inf-one-vector*:

*one-vector*  $x \implies$  *one-vector*  $y \implies$  *one-vector*  $(x \sqcap y)$

*<proof>*

**lemma** *comp-one-vector*:

*one-vector*  $x \implies$  *one-vector*  $y \implies$  *one-vector*  $(x * y)$

*<proof>*

**end**

**context** *multirelation-algebra-1*

**begin**

[Theorem 10 / Figure 3: closure properties](#)

[zero-vector](#)

**lemma** *top-zero-vector*:

*zero-vector* *top*

*<proof>*

**end**

[M1-algebra](#)

**context** *multirelation-algebra-2*

**begin**

[Section 7](#)

**lemma** *zero-vector-10*:

*zero-vector*  $x \iff x * \text{top} = x * 1$

*<proof>*

**lemma** *zero-vector-11*:

*zero-vector*  $x \iff x * \text{top} \leq x * 1$

*<proof>*

[Theorem 6 / Figure 2: relations between properties](#)

**lemma** *vector-zero-vector*:

*vector*  $x \implies$  *zero-vector*  $x$

*<proof>*

**lemma** *vector-up-closed-zero-vector*:

*vector*  $x \iff$  *up-closed*  $x \wedge$  *zero-vector*  $x$

*<proof>*

**lemma** *vector-zero-vector-one-vector*:

*vector*  $x \iff$  *zero-vector*  $x \wedge$  *one-vector*  $x$

*<proof>*

**proposition**  $(x * \text{bot} \sqcap y) * 1 = x * \text{bot} \sqcap y * 1$  **nitpick**  
 $[expect=genuine,card=7]$   $\langle proof \rangle$

**end**

[M3-algebra](#)

**context** *up-closed-multirelation-algebra*  
**begin**

**lemma** *up-closed:*

*up-closed x*  
 $\langle proof \rangle$

**lemma** *dedekind-1-left:*

$x * 1 \sqcap y \leq (x \sqcap y * 1) * 1$   
 $\langle proof \rangle$

[Theorem 10 / Figure 3: closure properties](#)

[zero-vector](#)

**lemma** *zero-vector-dual:*

*zero-vector x*  $\longleftrightarrow$  *zero-vector*  $(x^d)$   
 $\langle proof \rangle$

**end**

[complemented M0-algebra](#)

**class** *lattice-ordered-pre-left-semiring-b* = *lattice-ordered-pre-left-semiring* +  
*complemented-distributive-lattice*  
**begin**

**definition** *down-closed* :: 'a  $\Rightarrow$  bool **where** *down-closed x*  $\equiv -x * 1 \leq -x$

[Theorem 10 / Figure 3: closure properties](#)

[down-closed](#)

**lemma** *zero-down-closed:*

*down-closed bot*  
 $\langle proof \rangle$

**lemma** *top-down-closed:*

*down-closed top*  
 $\langle proof \rangle$

**lemma** *complement-down-closed-up-closed:*

*down-closed x*  $\longleftrightarrow$  *up-closed*  $(-x)$   
 $\langle proof \rangle$

**lemma** *sup-down-closed:*

*down-closed x*  $\Longrightarrow$  *down-closed y*  $\Longrightarrow$  *down-closed*  $(x \sqcup y)$

*<proof>*

**lemma** *inf-down-closed*:

*down-closed*  $x \implies$  *down-closed*  $y \implies$  *down-closed*  $(x \sqcap y)$

*<proof>*

**end**

**class** *multirelation-algebra-1b* = *multirelation-algebra-1* +  
*complemented-distributive-lattice*

**begin**

**subclass** *lattice-ordered-pre-left-semiring-b* *<proof>*

[Theorem 7.1](#)

**lemma** *complement-mult-zero-sub*:

$-(x * \text{bot}) \leq -x * \text{bot}$

*<proof>*

[Theorem 7.2](#)

**lemma** *transitive-zero-vector-complement*:

*transitive*  $x \implies$  *zero-vector*  $(-x)$

*<proof>*

**lemma** *transitive-dense-complement*:

*transitive*  $x \implies$  *dense-rel*  $(-x)$

*<proof>*

**lemma** *transitive-sup-distributive-complement*:

*transitive*  $x \implies$  *sup-distributive*  $(-x)$

*<proof>*

**lemma** *transitive-inf-distributive-complement*:

*transitive*  $x \implies$  *inf-distributive*  $(-x)$

*<proof>*

**lemma** *up-closed-zero-vector-complement*:

*up-closed*  $x \implies$  *zero-vector*  $(-x)$

*<proof>*

**lemma** *up-closed-dense-complement*:

*up-closed*  $x \implies$  *dense-rel*  $(-x)$

*<proof>*

**lemma** *up-closed-sup-distributive-complement*:

*up-closed*  $x \implies$  *sup-distributive*  $(-x)$

*<proof>*

**lemma** *up-closed-inf-distributive-complement*:

*up-closed*  $x \implies$  *inf-distributive*  $(-x)$   
*<proof>*

Theorem 10 / Figure 3: closure properties

closure under complement

**lemma** *co-total-total*:  
*co-total*  $x \implies$  *total*  $(-x)$   
*<proof>*

**lemma** *complement-one-vector-zero-vector*:  
*one-vector*  $x \implies$  *zero-vector*  $(-x)$   
*<proof>*

Theorem 6 / Figure 2: relations between properties

**lemma** *down-closed-zero-vector*:  
*down-closed*  $x \implies$  *zero-vector*  $x$   
*<proof>*

**lemma** *down-closed-one-vector-vector*:  
*down-closed*  $x \implies$  *one-vector*  $x \implies$  *vector*  $x$   
*<proof>*

**proposition** *complement-vector*: *vector*  $x \longrightarrow$  *vector*  $(-x)$  **nitpick**  
[*expect=genuine,card=8*] *<proof>*

**end**

**class** *multirelation-algebra-1c* = *multirelation-algebra-1b* +  
**assumes** *dedekind-top-left*:  $x * top \sqcap y \leq (x \sqcap y * top) * top$   
**assumes** *comp-zero-inf*:  $(x * bot \sqcap y) * bot \leq (x \sqcap y) * bot$   
**begin**

Theorem 7.3

**lemma** *schroeder-top-sub*:  
 $-(x * top) * top \leq -x$   
*<proof>*

Theorem 7.4

**lemma** *schroeder-top*:  
 $x * top \leq y \iff -y * top \leq -x$   
*<proof>*

Theorem 7.5

**lemma** *schroeder-top-eq*:  
 $-(x * top) * top = -(x * top)$   
*<proof>*

**lemma** *schroeder-one-eq*:

$$-(x * top) * 1 = -(x * top)$$

*<proof>*

### Theorem 7.6

**lemma** *vector-inf-comp:*

$$x * top \sqcap y * z = (x * top \sqcap y) * z$$

*<proof>*

**lemma** *dedekind-top-left-var:*

$$x * top \sqcap y \leq (x \sqcap y * top) * top$$

*<proof>*

### Theorem 7.7

**lemma** *vector-zero-inf-comp:*

$$(x * bot \sqcap y) * z = x * bot \sqcap y * z$$

*<proof>*

**lemma** *vector-zero-inf-comp-2:*

$$(x * bot \sqcap y) * z = (x * bot \sqcap y * 1) * z$$

*<proof>*

### Theorem 7.8

**lemma** *comp-zero-inf-2:*

$$x * bot \sqcap y * bot = (x \sqcap y) * bot$$

*<proof>*

**lemma** *comp-zero-inf-3:*

$$x * bot \sqcap y * bot = (x * bot \sqcap y) * bot$$

*<proof>*

**lemma** *comp-zero-inf-4:*

$$x * bot \sqcap y * bot = (x * bot \sqcap y * bot) * bot$$

*<proof>*

**lemma** *comp-zero-inf-5:*

$$x * bot \sqcap y * bot = (x * 1 \sqcap y * 1) * bot$$

*<proof>*

**lemma** *comp-zero-inf-6:*

$$x * bot \sqcap y * bot = (x * 1 \sqcap y * bot) * bot$$

*<proof>*

**lemma** *comp-zero-inf-7:*

$$x * bot \sqcap y * bot = (x * 1 \sqcap y) * bot$$

*<proof>*

### Theorem 10 / Figure 3: closure properties

#### zero-vector



**lemma** *inf-zero-vector*:

*zero-vector*  $x \implies$  *zero-vector*  $y \implies$  *zero-vector*  $(x \sqcap y)$

*<proof>*

[down-closed](#)

**lemma** *comp-down-closed*:

*down-closed*  $x \implies$  *down-closed*  $y \implies$  *down-closed*  $(x * y)$

*<proof>*

[closure under complement](#)

**lemma** *complement-vector*:

*vector*  $x \longleftrightarrow$  *vector*  $(-x)$

*<proof>*

**lemma** *complement-zero-vector-one-vector*:

*zero-vector*  $x \implies$  *one-vector*  $(-x)$

*<proof>*

**lemma** *complement-zero-vector-one-vector-iff*:

*zero-vector*  $x \longleftrightarrow$  *one-vector*  $(-x)$

*<proof>*

**lemma** *complement-one-vector-zero-vector-iff*:

*one-vector*  $x \longleftrightarrow$  *zero-vector*  $(-x)$

*<proof>*

[Theorem 6 / Figure 2: relations between properties](#)

**lemma** *vector-down-closed*:

*vector*  $x \implies$  *down-closed*  $x$

*<proof>*

**lemma** *co-vector-down-closed*:

*co-vector*  $x \implies$  *down-closed*  $x$

*<proof>*

**lemma** *vector-down-closed-one-vector*:

*vector*  $x \longleftrightarrow$  *down-closed*  $x \wedge$  *one-vector*  $x$

*<proof>*

**lemma** *vector-up-closed-down-closed*:

*vector*  $x \longleftrightarrow$  *up-closed*  $x \wedge$  *down-closed*  $x$

*<proof>*

[Section 7](#)

**lemma** *vector-b1*:

*vector*  $x \longleftrightarrow -x * top = -x$

*<proof>*

**lemma** *vector-b2*:

*vector*  $x \longleftrightarrow -x * bot = -x$   
*<proof>*

**lemma** *covector-b1*:

*co-vector*  $x \longleftrightarrow -x * top = -x$   
*<proof>*

**lemma** *covector-b2*:

*co-vector*  $x \longleftrightarrow -x * bot = -x$   
*<proof>*

**lemma** *vector-co-vector-iff*:

*vector*  $x \longleftrightarrow$  *co-vector*  $x$   
*<proof>*

**lemma** *zero-vector-b*:

*zero-vector*  $x \longleftrightarrow -x * bot \leq -x$   
*<proof>*

**lemma** *one-vector-b1*:

*one-vector*  $x \longleftrightarrow -x \leq -x * bot$   
*<proof>*

**lemma** *one-vector-b0*:

*one-vector*  $x \longleftrightarrow (\forall y z . -x * y = -x * z)$   
*<proof>*

**proposition** *schroeder-one*:  $x * -1 \leq y \longleftrightarrow -y * -1 \leq -x$  **nitpick**  
[*expect=genuine,card=8*] *<proof>*

**end**

**class** *multirelation-algebra-2b* = *multirelation-algebra-2* +  
*complemented-distributive-lattice*  
**begin**

**subclass** *multirelation-algebra-1b* *<proof>*

**proposition**  $-x * bot \leq -(x * bot)$  **nitpick** [*expect=genuine,card=8*] *<proof>*

**end**

**complemented M1-algebra**

**class** *multirelation-algebra-2c* = *multirelation-algebra-2b* +  
*multirelation-algebra-1c*

**class** *multirelation-algebra-3b* = *multirelation-algebra-3* +  
*complemented-distributive-lattice*  
**begin**

```

subclass lattice-ordered-pre-left-semiring-b ⟨proof⟩

lemma dual-complement-commute:
   $-(x^d) = (-x)^d$ 
  ⟨proof⟩

end

  complemented M2-algebra

class multirelation-algebra-5b = multirelation-algebra-5 +
complemented-distributive-lattice
begin

subclass multirelation-algebra-2b ⟨proof⟩

subclass multirelation-algebra-3b ⟨proof⟩

lemma dual-down-closed:
  down-closed  $x \longleftrightarrow$  down-closed  $(x^d)$ 
  ⟨proof⟩

end

class multirelation-algebra-5c = multirelation-algebra-5b +
multirelation-algebra-1c
begin

lemma complement-mult-zero-below:
   $-x * bot \leq -(x * bot)$ 
  ⟨proof⟩

proposition  $x * 1 \sqcap y * 1 \leq (x \sqcap y) * 1$  nitpick [expect=genuine,card=4]
  ⟨proof⟩
proposition  $x * 1 \sqcap (y * 1) \leq (x * 1 \sqcap y) * 1$  nitpick
  [expect=genuine,card=4] ⟨proof⟩

end

class up-closed-multirelation-algebra-b = up-closed-multirelation-algebra +
complemented-distributive-lattice
begin

subclass multirelation-algebra-5c
  ⟨proof⟩

lemma complement-zero-vector:
  zero-vector  $x \longleftrightarrow$  zero-vector  $(-x)$ 
  ⟨proof⟩

```

**lemma** *down-closed*:

*down-closed*  $x$

$\langle$ *proof* $\rangle$

**lemma** *vector*:

*vector*  $x$

$\langle$ *proof* $\rangle$

**end**

**end**

## 7 Binary Iterings

**theory** *Binary-Iterings*

**imports** *Base*

**begin**

**class** *binary-itering* = *idempotent-left-zero-semiring* + *while* +

**assumes** *while-productstar*:  $(x * y) * z = z \sqcup x * ((y * x) * (y * z))$

**assumes** *while-sumstar*:  $(x \sqcup y) * z = (x * y) * (x * z)$

**assumes** *while-left-dist-sup*:  $x * (y \sqcup z) = (x * y) \sqcup (x * z)$

**assumes** *while-sub-associative*:  $(x * y) * z \leq x * (y * z)$

**assumes** *while-simulate-left-plus*:  $x * z \leq z * (y * 1) \sqcup w \longrightarrow x * (z * v) \leq z * (y * v) \sqcup (x * (w * (y * v)))$

**assumes** *while-simulate-right-plus*:  $z * x \leq y * (y * z) \sqcup w \longrightarrow z * (x * v) \leq y * (z * v) \sqcup w * (x * v)$

**begin**

[Theorem 9.1](#)

**lemma** *while-zero*:

$bot * x = x$

$\langle$ *proof* $\rangle$

[Theorem 9.4](#)

**lemma** *while-mult-increasing*:

$x * y \leq x * y$

$\langle$ *proof* $\rangle$

[Theorem 9.2](#)

**lemma** *while-one-increasing*:

$x \leq x * 1$

$\langle$ *proof* $\rangle$

[Theorem 9.3](#)

**lemma** *while-increasing*:

$$y \leq x \star y$$

*<proof>*

**Theorem 9.42**

**lemma** *while-right-isotone*:

$$y \leq z \implies x \star y \leq x \star z$$

*<proof>*

**Theorem 9.41**

**lemma** *while-left-isotone*:

$$x \leq y \implies x \star z \leq y \star z$$

*<proof>*

**lemma** *while-isotone*:

$$w \leq x \implies y \leq z \implies w \star y \leq x \star z$$

*<proof>*

**Theorem 9.17**

**lemma** *while-left-unfold*:

$$x \star y = y \sqcup x \star (x \star y)$$

*<proof>*

**lemma** *while-simulate-left-plus-1*:

$$x \star z \leq z \star (y \star 1) \implies x \star (z \star w) \leq z \star (y \star w) \sqcup (x \star bot)$$

*<proof>*

**Theorem 11.1**

**lemma** *while-simulate-absorb*:

$$y \star x \leq x \implies y \star x \leq x \sqcup (y \star bot)$$

*<proof>*

**Theorem 9.10**

**lemma** *while-transitive*:

$$x \star (x \star y) = x \star y$$

*<proof>*

**Theorem 9.25**

**lemma** *while-slide*:

$$(x \star y) \star (x \star z) = x \star ((y \star x) \star z)$$

*<proof>*

**Theorem 9.21**

**lemma** *while-zero-2*:

$$(x \star bot) \star y = x \star bot \sqcup y$$

*<proof>*

**Theorem 9.5**

**lemma** *while-mult-star-exchange*:

$$x \star (x \star y) = x \star (x \star y)$$

*<proof>*

**Theorem 9.18**

**lemma** *while-right-unfold*:

$$x \star y = y \sqcup (x \star (x \star y))$$

*<proof>*

**Theorem 9.7**

**lemma** *while-one-mult-below*:

$$(x \star 1) \star y \leq x \star y$$

*<proof>*

**lemma** *while-plus-one*:

$$x \star y = y \sqcup (x \star y)$$

*<proof>*

**Theorem 9.19**

**lemma** *while-rtc-2*:

$$y \sqcup x \star y \sqcup (x \star (x \star y)) = x \star y$$

*<proof>*

**Theorem 9.6**

**lemma** *while-left-plus-below*:

$$x \star (x \star y) \leq x \star y$$

*<proof>*

**lemma** *while-right-plus-below*:

$$x \star (x \star y) \leq x \star y$$

*<proof>*

**lemma** *while-right-plus-below-2*:

$$(x \star x) \star y \leq x \star y$$

*<proof>*

**Theorem 9.47**

**lemma** *while-mult-transitive*:

$$x \leq z \star y \implies y \leq z \star w \implies x \leq z \star w$$

*<proof>*

**Theorem 9.48**

**lemma** *while-mult-upper-bound*:

$$x \leq z \star 1 \implies y \leq z \star w \implies x \star y \leq z \star w$$

*<proof>*

**lemma** *while-one-mult-while-below*:

$$(y \star 1) \star (y \star v) \leq y \star v$$

*<proof>*

**Theorem 9.34**

**lemma** *while-sub-dist*:

$$x \star z \leq (x \sqcup y) \star z$$

*<proof>*

**lemma** *while-sub-dist-1*:

$$x \star z \leq (x \sqcup y) \star z$$

*<proof>*

**lemma** *while-sub-dist-2*:

$$x \star y \star z \leq (x \sqcup y) \star z$$

*<proof>*

[Theorem 9.36](#)

**lemma** *while-sub-dist-3*:

$$x \star (y \star z) \leq (x \sqcup y) \star z$$

*<proof>*

[Theorem 9.44](#)

**lemma** *while-absorb-2*:

$$x \leq y \implies y \star (x \star z) = y \star z$$

*<proof>*

**lemma** *while-simulate-right-plus-1*:

$$z \star x \leq y \star (y \star z) \implies z \star (x \star w) \leq y \star (z \star w)$$

*<proof>*

[Theorem 9.39](#)

**lemma** *while-sumstar-1-below*:

$$x \star ((y \star (x \star 1)) \star z) \leq ((x \star 1) \star y) \star (x \star z)$$

*<proof>*

**lemma** *while-sumstar-2-below*:

$$((x \star 1) \star y) \star (x \star z) \leq (x \star y) \star (x \star z)$$

*<proof>*

[Theorem 9.38](#)

**lemma** *while-sup-1-below*:

$$x \star ((y \star (x \star 1)) \star z) \leq (x \sqcup y) \star z$$

*<proof>*

[Theorem 9.16](#)

**lemma** *while-while-while*:

$$((x \star 1) \star 1) \star y = (x \star 1) \star y$$

*<proof>*

**lemma** *while-one*:

$$(1 \star 1) \star y = 1 \star y$$

*<proof>*

[Theorem 9.22](#)

**lemma** *while-sup-below*:

$$x \sqcup y \leq x \star (y \star 1)$$

*<proof>*

[Theorem 9.32](#)

**lemma** *while-sup-2*:

$$(x \sqcup y) \star z \leq (x \star (y \star 1)) \star z$$

*<proof>*

[Theorem 9.45](#)

**lemma** *while-sup-one-left-unfold*:

$$1 \leq x \implies x \star (x \star y) = x \star y$$

*<proof>*

**lemma** *while-sup-one-right-unfold*:

$$1 \leq x \implies x \star (x \star y) = x \star y$$

*<proof>*

[Theorem 9.30](#)

**lemma** *while-decompose-7*:

$$(x \sqcup y) \star z = x \star (y \star ((x \sqcup y) \star z))$$

*<proof>*

[Theorem 9.31](#)

**lemma** *while-decompose-8*:

$$(x \sqcup y) \star z = (x \sqcup y) \star (x \star (y \star z))$$

*<proof>*

[Theorem 9.27](#)

**lemma** *while-decompose-9*:

$$(x \star (y \star 1)) \star z = x \star (y \star ((x \star (y \star 1)) \star z))$$

*<proof>*

**lemma** *while-decompose-10*:

$$(x \star (y \star 1)) \star z = (x \star (y \star 1)) \star (x \star (y \star z))$$

*<proof>*

**lemma** *while-back-loop-fixpoint*:

$$z \star (y \star (y \star x)) \sqcup z \star x = z \star (y \star x)$$

*<proof>*

**lemma** *while-back-loop-prefixpoint*:

$$z \star (y \star 1) \star y \sqcup z \leq z \star (y \star 1)$$

*<proof>*

[Theorem 9](#)

**lemma** *while-loop-is-fixpoint*:



*is-fixpoint*  $(\lambda x . y * x \sqcup z) (y * z)$   
*<proof>*

### Theorem 9

**lemma** *while-back-loop-is-prefixpoint*:  
*is-prefixpoint*  $(\lambda x . x * y \sqcup z) (z * (y * 1))$   
*<proof>*

### Theorem 9.20

**lemma** *while-while-sup*:  
 $(1 \sqcup x) * y = (x * 1) * y$   
*<proof>*

**lemma** *while-while-mult-sub*:  
 $x * (1 * y) \leq (x * 1) * y$   
*<proof>*

### Theorem 9.11

**lemma** *while-right-plus*:  
 $(x * x) * y = x * y$   
*<proof>*

### Theorem 9.12

**lemma** *while-left-plus*:  
 $(x * (x * 1)) * y = x * y$   
*<proof>*

### Theorem 9.9

**lemma** *while-below-while-one*:  
 $x * x \leq x * 1$   
*<proof>*

**lemma** *while-below-while-one-mult*:  
 $x * (x * x) \leq x * (x * 1)$   
*<proof>*

### Theorem 9.23

**lemma** *while-sup-sub-sup-one*:  
 $x * (x \sqcup y) \leq x * (1 \sqcup y)$   
*<proof>*

**lemma** *while-sup-sub-sup-one-mult*:  
 $x * (x * (x \sqcup y)) \leq x * (x * (1 \sqcup y))$   
*<proof>*

**lemma** *while-elimination*:  
 $x * y = \text{bot} \implies x * (y * z) = x * z$   
*<proof>*

### Theorem 9.8

**lemma** *while-square*:

$$(x * x) * y \leq x * y$$

*<proof>*

### Theorem 9.35

**lemma** *while-mult-sub-sup*:

$$(x * y) * z \leq (x \sqcup y) * z$$

*<proof>*

### Theorem 9.43

**lemma** *while-absorb-1*:

$$x \leq y \implies x * (y * z) = y * z$$

*<proof>*

**lemma** *while-absorb-3*:

$$x \leq y \implies x * (y * z) = y * (x * z)$$

*<proof>*

### Theorem 9.24

**lemma** *while-square-2*:

$$(x * x) * ((x \sqcup 1) * y) \leq x * y$$

*<proof>*

**lemma** *while-separate-unfold-below*:

$$(y * (x * 1)) * z \leq (y * z) \sqcup (y * (y * x * (x * ((y * (x * 1)) * z))))$$

*<proof>*

### Theorem 9.33

**lemma** *while-mult-zero-sup*:

$$(x \sqcup y * \text{bot}) * z = x * ((y * \text{bot}) * z)$$

*<proof>*

**lemma** *while-sup-mult-zero*:

$$(x \sqcup y * \text{bot}) * y = x * y$$

*<proof>*

**lemma** *while-mult-zero-sup-2*:

$$(x \sqcup y * \text{bot}) * z = (x * z) \sqcup (x * (y * \text{bot}))$$

*<proof>*

**lemma** *while-sup-zero-star*:

$$(x \sqcup y * \text{bot}) * z = x * (y * \text{bot} \sqcup z)$$

*<proof>*

**lemma** *while-unfold-sum*:

$$(x \sqcup y) * z = (x * z) \sqcup (x * (y * ((x \sqcup y) * z)))$$

*<proof>*

**lemma** *while-simulate-left*:

$$x * z \leq z * y \sqcup w \implies x \star (z * v) \leq z * (y \star v) \sqcup (x \star (w * (y \star v)))$$

*<proof>*

**lemma** *while-simulate-right*:

**assumes**  $z * x \leq y * z \sqcup w$   
**shows**  $z * (x \star v) \leq y \star (z * v \sqcup w * (x \star v))$

*<proof>*

**lemma** *while-simulate*:

$$z * x \leq y * z \implies z * (x \star v) \leq y \star (z * v)$$

*<proof>*

[Theorem 9.14](#)

**lemma** *while-while-mult*:

$$1 \star (x \star y) = (x \star 1) \star y$$

*<proof>*

**lemma** *while-simulate-left-1*:

$$x * z \leq z * y \implies x \star (z * v) \leq z * (y \star v) \sqcup (x \star \text{bot})$$

*<proof>*

[Theorem 9.46](#)

**lemma** *while-associative-1*:

**assumes**  $1 \leq z$   
**shows**  $x \star (y * z) = (x \star y) * z$

*<proof>*

[Theorem 9.29](#)

**lemma** *while-associative-while-1*:

$$x \star (y * (z \star 1)) = (x \star y) * (z \star 1)$$

*<proof>*

[Theorem 9.13](#)

**lemma** *while-one-while*:

$$(x \star 1) * (y \star 1) = x \star (y \star 1)$$

*<proof>*

**lemma** *while-decompose-5-below*:

$$(x \star (y \star 1)) \star z \leq (y \star (x \star 1)) \star z$$

*<proof>*

[Theorem 9.26](#)

**lemma** *while-decompose-5*:

$$(x \star (y \star 1)) \star z = (y \star (x \star 1)) \star z$$

*<proof>*

**lemma** *while-decompose-4*:

$$(x \star (y \star 1)) \star z = x \star ((y \star (x \star 1)) \star z)$$

$\langle proof \rangle$

### Theorem 11.7

**lemma** *while-simulate-2*:

$$y * (x * 1) \leq x * (y * 1) \iff y * (x * 1) \leq x * (y * 1)$$

$\langle proof \rangle$

**lemma** *while-simulate-1*:

$$y * x \leq x * y \implies y * (x * 1) \leq x * (y * 1)$$

$\langle proof \rangle$

**lemma** *while-simulate-3*:

$$y * (x * 1) \leq x * 1 \implies y * (x * 1) \leq x * (y * 1)$$

$\langle proof \rangle$

### Theorem 9.28

**lemma** *while-extra-while*:

$$(y * (x * 1)) * z = (y * (y * (x * 1))) * z$$

$\langle proof \rangle$

### Theorem 11.6

**lemma** *while-separate-4*:

**assumes**  $y * x \leq x * (x \sqcup y)$   
**shows**  $(x \sqcup y) * z = x * (y * z)$

$\langle proof \rangle$

**lemma** *while-separate-5*:

$$y * x \leq x * (x * (x \sqcup y)) \implies (x \sqcup y) * z = x * (y * z)$$

$\langle proof \rangle$

**lemma** *while-separate-6*:

$$y * x \leq x * (x \sqcup y) \implies (x \sqcup y) * z = x * (y * z)$$

$\langle proof \rangle$

### Theorem 11.4

**lemma** *while-separate-1*:

$$y * x \leq x * y \implies (x \sqcup y) * z = x * (y * z)$$

$\langle proof \rangle$

### Theorem 11.2

**lemma** *while-separate-mult-1*:

$$y * x \leq x * y \implies (x * y) * z \leq x * (y * z)$$

$\langle proof \rangle$

### Theorem 11.5

**lemma** *separation*:

**assumes**  $y * x \leq x * (y * 1)$   
**shows**  $(x \sqcup y) * z = x * (y * z)$

$\langle proof \rangle$

### Theorem 11.5

**lemma** *while-separate-left*:

$$y * x \leq x * (y * 1) \implies y * (x * z) \leq x * (y * z)$$

*<proof>*

### Theorem 11.6

**lemma** *while-simulate-4*:

$$y * x \leq x * (x * (1 \sqcup y)) \implies y * (x * z) \leq x * (y * z)$$

*<proof>*

**lemma** *while-simulate-5*:

$$y * x \leq x * (x * (x \sqcup y)) \implies y * (x * z) \leq x * (y * z)$$

*<proof>*

**lemma** *while-simulate-6*:

$$y * x \leq x * (x \sqcup y) \implies y * (x * z) \leq x * (y * z)$$

*<proof>*

### Theorem 11.3

**lemma** *while-simulate-7*:

$$y * x \leq x * y \implies y * (x * z) \leq x * (y * z)$$

*<proof>*

**lemma** *while-while-mult-1*:

$$x * (1 * y) = 1 * (x * y)$$

*<proof>*

### Theorem 9.15

**lemma** *while-while-mult-2*:

$$x * (1 * y) = (x * 1) * y$$

*<proof>*

### Theorem 11.8

**lemma** *while-import*:

**assumes**  $p \leq p * p \wedge p \leq 1 \wedge p * x \leq x * p$

**shows**  $p * (x * y) = p * ((p * x) * y)$

*<proof>*

### Theorem 11.8

**lemma** *while-preserve*:

**assumes**  $p \leq p * p$

**and**  $p \leq 1$

**and**  $p * x \leq x * p$

**shows**  $p * (x * y) = p * (x * (p * y))$

*<proof>*

**lemma** *while-plus-below-while*:

$$(x * 1) * x \leq x * 1$$

*<proof>*

### Theorem 9.40

**lemma** *while-01*:

$$(w * (x * 1)) * (y * z) \leq (x * w) * ((x * y) * z)$$

*<proof>*

### Theorem 9.37

**lemma** *while-while-sub-associative*:

$$x * (y * z) \leq ((x * y) * z) \sqcup (x * z)$$

*<proof>*

**lemma** *while-induct*:

$$x * z \leq z \wedge y \leq z \wedge x * 1 \leq z \implies x * y \leq z$$

*<proof>*

**proposition** *while-sumstar-4-below*:  $(x * y) * ((x * 1) * z) \leq x * ((y * (x * 1)) * z)$  *<proof>*

**proposition** *while-sumstar-2*:  $(x \sqcup y) * z = x * ((y * (x * 1)) * z)$  *<proof>*

**proposition** *while-sumstar-3*:  $(x \sqcup y) * z = ((x * 1) * y) * (x * z)$  *<proof>*

**proposition** *while-decompose-6*:  $x * ((y * (x * 1)) * z) = y * ((x * (y * 1)) * z)$  *<proof>*

**proposition** *while-finite-associative*:  $x * \text{bot} = \text{bot} \implies (x * y) * z = x * (y * z)$  *<proof>*

**proposition** *atomicity-refinement*:  $s = s * q \implies x = q * x \implies q * b = \text{bot} \implies r * b \leq b * r \implies r * l \leq l * r \implies x * l \leq l * x \implies b * l \leq l * b \implies q * l \leq l * q \implies r * q \leq q * (r * 1) \implies q \leq 1 \implies s * ((x \sqcup b \sqcup r \sqcup l) * (q * z)) \leq s * ((x * (b * q) \sqcup r \sqcup l) * z)$  *<proof>*

**proposition** *while-separate-right-plus*:  $y * x \leq x * (x * (1 \sqcup y)) \sqcup 1 \implies y * (x * z) \leq x * (y * z)$  *<proof>*

**proposition** *while-square-1*:  $x * 1 = (x * x) * (x \sqcup 1)$  *<proof>*

**proposition** *while-absorb-below-one*:  $y * x \leq x \implies y * x \leq 1 * x$  *<proof>*

**proposition**  $y * (x * 1) \leq x * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**proposition**  $y * x \leq (1 \sqcup x) * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**end**

**class** *bounded-binary-itering* = *bounded-idempotent-left-zero-semiring* + *binary-itering*

**begin**

### Theorem 9

**lemma** *while-right-top*:

$$x * \text{top} = \text{top}$$

*<proof>*

### Theorem 9

**lemma** *while-left-top*:

$$\text{top} * (x * 1) = \text{top}$$

*<proof>*

**end**

**class** *extended-binary-itering* = *binary-itering* +  
  **assumes** *while-denest-0*:  $w * (x * (y * z)) \leq (w * (x * y)) * (w * (x * y) * z)$   
**begin**

[Theorem 10.2](#)

**lemma** *while-denest-1*:  
   $w * (x * (y * z)) \leq (w * (x * y)) * z$   
   $\langle proof \rangle$

**lemma** *while-mult-sub-while-while*:  
   $x * (y * z) \leq (x * y) * z$   
   $\langle proof \rangle$

**lemma** *while-zero-zero*:  
   $(x * bot) * bot = x * bot$   
   $\langle proof \rangle$

[Theorem 10.11](#)

**lemma** *while-mult-zero-zero*:  
   $(x * (y * bot)) * bot = x * (y * bot)$   
   $\langle proof \rangle$

[Theorem 10.3](#)

**lemma** *while-denest-2*:  
   $w * ((x * (y * w)) * z) = w * (((x * y) * w) * z)$   
   $\langle proof \rangle$

[Theorem 10.12](#)

**lemma** *while-denest-3*:  
   $(x * w) * (x * bot) = (x * w) * bot$   
   $\langle proof \rangle$

[Theorem 10.15](#)

**lemma** *while-02*:  
   $x * ((x * w) * ((x * y) * z)) = (x * w) * ((x * y) * z)$   
   $\langle proof \rangle$

**lemma** *while-sumstar-3-below*:  
   $(x * y) * (x * z) \leq (x * y) * ((x * 1) * z)$   
   $\langle proof \rangle$

**lemma** *while-sumstar-4-below*:  
   $(x * y) * ((x * 1) * z) \leq x * ((y * (x * 1)) * z)$   
   $\langle proof \rangle$

[Theorem 10.10](#)

**lemma** *while-sumstar-1*:

$$(x \sqcup y) \star z = (x \star y) \star ((x \star 1) \star z)$$

*<proof>*

[Theorem 10.8](#)

**lemma** *while-sumstar-2*:

$$(x \sqcup y) \star z = x \star ((y \star (x \star 1)) \star z)$$

*<proof>*

[Theorem 10.9](#)

**lemma** *while-sumstar-3*:

$$(x \sqcup y) \star z = ((x \star 1) \star y) \star (x \star z)$$

*<proof>*

[Theorem 10.6](#)

**lemma** *while-decompose-6*:

$$x \star ((y \star (x \star 1)) \star z) = y \star ((x \star (y \star 1)) \star z)$$

*<proof>*

[Theorem 10.4](#)

**lemma** *while-denest-4*:

$$(x \star w) \star (x \star (y \star z)) = (x \star w) \star ((x \star y) \star z)$$

*<proof>*

[Theorem 10.13](#)

**lemma** *while-denest-5*:

$$w \star ((x \star (y \star w)) \star (x \star (y \star z))) = w \star (((x \star y) \star w) \star ((x \star y) \star z))$$

*<proof>*

[Theorem 10.5](#)

**lemma** *while-denest-6*:

$$(w \star (x \star y)) \star z = z \sqcup w \star ((x \sqcup y \star w) \star (y \star z))$$

*<proof>*

[Theorem 10.1](#)

**lemma** *while-sum-below-one*:

$$y \star ((x \sqcup y) \star z) \leq (y \star (x \star 1)) \star z$$

*<proof>*

[Theorem 10.14](#)

**lemma** *while-separate-unfold*:

$$(y \star (x \star 1)) \star z = (y \star z) \sqcup (y \star (y \star x \star (x \star ((y \star (x \star 1)) \star z))))$$

*<proof>*

[Theorem 10.7](#)

**lemma** *while-finite-associative*:

$$x \star \text{bot} = \text{bot} \implies (x \star y) \star z = x \star (y \star z)$$

*<proof>*



### Theorem 12

**lemma** *atomicity-refinement*:

**assumes**  $s = s * q$

**and**  $x = q * x$

**and**  $q * b = \text{bot}$

**and**  $r * b \leq b * r$

**and**  $r * l \leq l * r$

**and**  $x * l \leq l * x$

**and**  $b * l \leq l * b$

**and**  $q * l \leq l * q$

**and**  $r * q \leq q * (r * 1) \wedge q \leq 1$

**shows**  $s * ((x \sqcup b \sqcup r \sqcup l) * (q * z)) \leq s * ((x * (b * q) \sqcup r \sqcup l) * z)$

*<proof>*

**end**

**class** *bounded-extended-binary-itering* = *bounded-binary-itering* +  
*extended-binary-itering*

**end**

## 8 Strict Binary Iterings

**theory** *Binary-Iterings-Strict*

**imports** *Stone-Kleene-Relation-Algebras.Iterings Binary-Iterings*

**begin**

**class** *strict-itering* = *itering* + *while* +

**assumes** *while-def*:  $x * y = x^\circ * y$

**begin**

### Theorem 8.1

**subclass** *extended-binary-itering*

*<proof>*

### Theorem 13.1

**lemma** *while-associative*:

$(x * y) * z = x * (y * z)$

*<proof>*

### Theorem 13.3

**lemma** *while-one-mult*:

$(x * 1) * x = x * x$

*<proof>*

**lemma** *while-back-loop-is-fixpoint*:

*is-fixpoint*  $(\lambda x . x * y \sqcup z) (z * (y * 1))$   
*<proof>*

#### Theorem 13.4

**lemma** *while-sumstar-var*:  
 $(x \sqcup y) * z = ((x * 1) * y) * ((x * 1) * z)$   
*<proof>*

#### Theorem 13.2

**lemma** *while-mult-1-assoc*:  
 $(x * 1) * y = x * y$   
*<proof>*

**proposition**  $y * (x * 1) \leq x * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**proposition**  $y * x \leq (1 \sqcup x) * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**proposition** *while-square-1*:  $x * 1 = (x * x) * (x \sqcup 1)$  *<proof>*

**proposition** *while-absorb-below-one*:  $y * x \leq x \implies y * x \leq 1 * x$  *<proof>*

**end**

**class** *bounded-strict-itering* = *bounded-itering* + *strict-itering*  
**begin**

**subclass** *bounded-extended-binary-itering* *<proof>*

#### Theorem 13.6

**lemma** *while-top-2*:  
 $top * z = top * z$   
*<proof>*

#### Theorem 13.5

**lemma** *while-mult-top-2*:  
 $(x * top) * z = z \sqcup x * top * z$   
*<proof>*

#### Theorem 13 counterexamples

**proposition** *while-one-top*:  $1 * x = top$  **nitpick** [*expect=genuine,card=2*]  
*<proof>*

**proposition** *while-top*:  $top * x = top$  **nitpick** [*expect=genuine,card=2*] *<proof>*

**proposition** *while-sub-mult-one*:  $x * (1 * y) \leq 1 * x$  *<proof>*

**proposition** *while-unfold-below-1*:  $x = y * x \implies x \leq y * 1$  *<proof>*

**proposition** *while-unfold-below*:  $x = z \sqcup y * x \implies x \leq y * z$  **nitpick**  
[*expect=genuine,card=2*] *<proof>*

**proposition** *while-unfold-below*:  $x \leq z \sqcup y * x \implies x \leq y * z$  **nitpick**  
[*expect=genuine,card=2*] *<proof>*

**proposition** *while-mult-top*:  $(x * top) * z = z \sqcup x * top$  **nitpick**  
[*expect=genuine,card=2*] *<proof>*

**proposition** *tarski-mult-top-idempotent*:  $x * top = x * top * x * top$  *<proof>*

**proposition** *while-loop-is-greatest-postfixpoint*: *is-greatest-postfixpoint*  $(\lambda x . y * x \sqcup z) (y * z)$  **nitpick** [*expect=genuine,card=2*]  $\langle$ *proof* $\rangle$

**proposition** *while-loop-is-greatest-fixpoint*: *is-greatest-fixpoint*  $(\lambda x . y * x \sqcup z) (y * z)$  **nitpick** [*expect=genuine,card=2*]  $\langle$ *proof* $\rangle$

**proposition** *while-sub-while-zero*:  $x * z \leq (x * y) * z$   $\langle$ *proof* $\rangle$

**proposition** *while-while-sub-associative*:  $x * (y * z) \leq (x * y) * z$   $\langle$ *proof* $\rangle$

**proposition** *tarski*:  $x \leq x * top * x * top$   $\langle$ *proof* $\rangle$

**proposition** *tarski-top-omega-below*:  $x * top \leq (x * top) * bot$  **nitpick** [*expect=genuine,card=2*]  $\langle$ *proof* $\rangle$

**proposition** *tarski-top-omega*:  $x * top = (x * top) * bot$  **nitpick** [*expect=genuine,card=2*]  $\langle$ *proof* $\rangle$

**proposition** *tarski-below-top-omega*:  $x \leq (x * top) * bot$  **nitpick** [*expect=genuine,card=2*]  $\langle$ *proof* $\rangle$

**proposition** *tarski*:  $x = bot \vee top * x * top = top$   $\langle$ *proof* $\rangle$

**proposition**  $1 = (x * bot) * 1$   $\langle$ *proof* $\rangle$

**proposition**  $1 \sqcup x * bot = x * 1$   $\langle$ *proof* $\rangle$

**proposition**  $x = x * (x * 1)$   $\langle$ *proof* $\rangle$

**proposition**  $x * (x * 1) = x * 1$   $\langle$ *proof* $\rangle$

**proposition**  $x * 1 = x * (1 * 1)$   $\langle$ *proof* $\rangle$

**proposition**  $(x \sqcup y) * 1 = (x * (y * 1)) * 1$   $\langle$ *proof* $\rangle$

**proposition**  $z \sqcup y * x = x \implies y * z \leq x$   $\langle$ *proof* $\rangle$

**proposition**  $y * x = x \implies y * x \leq x$   $\langle$ *proof* $\rangle$

**proposition**  $z \sqcup x * y = x \implies z * (y * 1) \leq x$   $\langle$ *proof* $\rangle$

**proposition**  $x * y = x \implies x * (y * 1) \leq x$   $\langle$ *proof* $\rangle$

**proposition**  $x * z = z * y \implies x * z \leq z * (y * 1)$   $\langle$ *proof* $\rangle$

end

**class** *binary-itering-unary* = *extended-binary-itering* + *circ* +  
**assumes** *circ-def*:  $x^\circ = x * 1$   
**begin**

[Theorem 50.7](#)

**subclass** *left-conway-semiring*  
 $\langle$ *proof* $\rangle$

end

**class** *strict-binary-itering* = *binary-itering* + *circ* +  
**assumes** *while-associative*:  $(x * y) * z = x * (y * z)$   
**assumes** *circ-def*:  $x^\circ = x * 1$   
**begin**

[Theorem 2.8](#)

**subclass** *itering*  
 $\langle$ *proof* $\rangle$

[Theorem 8.5](#)

**subclass** *extended-binary-itering*

*<proof>*

**end**

**end**

## 9 Nonstrict Binary Iterings

**theory** *Binary-Iterings-Nonstrict*

**imports** *Omega-Algebras Binary-Iterings*

**begin**

**class** *nonstrict-itering* = *bounded-left-zero-omega-algebra* + *while* +

**assumes** *while-def*:  $x \star y = x^\omega \sqcup x^* * y$

**begin**

[Theorem 8.2](#)

**subclass** *bounded-binary-itering*

*<proof>*

[Theorem 13.8](#)

**lemma** *while-top*:

$top \star x = top$

*<proof>*

[Theorem 13.7](#)

**lemma** *while-one-top*:

$1 \star x = top$

*<proof>*

**lemma** *while-finite-associative*:

$x^\omega = bot \implies (x \star y) * z = x \star (y * z)$

*<proof>*

**lemma** *star-below-while*:

$x^* * y \leq x \star y$

*<proof>*

[Theorem 13.9](#)

**lemma** *while-sub-mult-one*:

$x * (1 \star y) \leq 1 \star x$

*<proof>*

**lemma** *while-while-one*:

$y \star (x \star 1) = y^\omega \sqcup y^* * x^\omega \sqcup y^* * x^*$

*<proof>*

**lemma** *while-simulate-4-plus*:  
**assumes**  $y * x \leq x * (x \star (1 \sqcup y))$   
**shows**  $y * x * x^* \leq x * (x \star (1 \sqcup y))$   
 $\langle proof \rangle$

**lemma** *while-simulate-4-omega*:  
**assumes**  $y * x \leq x * (x \star (1 \sqcup y))$   
**shows**  $y * x^\omega \leq x^\omega$   
 $\langle proof \rangle$

**Theorem 13.11**

**lemma** *while-unfold-below*:  
 $x = z \sqcup y * x \implies x \leq y \star z$   
 $\langle proof \rangle$

**Theorem 13.12**

**lemma** *while-unfold-below-sub*:  
 $x \leq z \sqcup y * x \implies x \leq y \star z$   
 $\langle proof \rangle$

**Theorem 13.10**

**lemma** *while-unfold-below-1*:  
 $x = y * x \implies x \leq y \star 1$   
 $\langle proof \rangle$

**lemma** *while-square-1*:  
 $x \star 1 = (x * x) \star (x \sqcup 1)$   
 $\langle proof \rangle$

**lemma** *while-absorb-below-one*:  
 $y * x \leq x \implies y \star x \leq 1 \star x$   
 $\langle proof \rangle$

**lemma** *while-loop-is-greatest-postfixpoint*:  
*is-greatest-postfixpoint*  $(\lambda x . y * x \sqcup z)$   $(y \star z)$   
 $\langle proof \rangle$

**lemma** *while-loop-is-greatest-fixpoint*:  
*is-greatest-fixpoint*  $(\lambda x . y * x \sqcup z)$   $(y \star z)$   
 $\langle proof \rangle$

**proposition** *while-sumstar-4-below*:  $(x \star y) \star ((x \star 1) * z) \leq x \star ((y * (x \star 1)) * z)$  **nitpick** [*expect=genuine,card=6*]  $\langle proof \rangle$

**proposition** *while-sumstar-2*:  $(x \sqcup y) \star z = x \star ((y * (x \star 1)) \star z)$  **nitpick** [*expect=genuine,card=6*]  $\langle proof \rangle$

**proposition** *while-sumstar-3*:  $(x \sqcup y) \star z = ((x \star 1) * y) \star (x \star z)$   $\langle proof \rangle$

**proposition** *while-decompose-6*:  $x \star ((y * (x \star 1)) \star z) = y \star ((x * (y \star 1)) \star z)$

**nitpick** [*expect=genuine,card=6*]  $\langle proof \rangle$

**proposition** *while-finite-associative*:  $x \star bot = bot \implies (x \star y) * z = x \star (y * z)$   
 $\langle proof \rangle$

**proposition** *atomicity-refinement*:  $s = s * q \implies x = q * x \implies q * b = bot \implies r * b \leq b * r \implies r * l \leq l * r \implies x * l \leq l * x \implies b * l \leq l * b \implies q * l \leq l * q \implies r * q \leq q * (r * 1) \implies q \leq 1 \implies s * ((x \sqcup b \sqcup r \sqcup l) * (q * z)) \leq s * ((x * (b * q) \sqcup r \sqcup l) * z)$  *<proof>*

**proposition** *while-separate-right-plus*:  $y * x \leq x * (x * (1 \sqcup y)) \sqcup 1 \implies y * (x * z) \leq x * (y * z)$  *<proof>*

**proposition**  $y * (x * 1) \leq x * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**proposition**  $y * x \leq (1 \sqcup x) * (y * 1) \implies (x \sqcup y) * 1 = x * (y * 1)$  *<proof>*

**proposition** *while-mult-sub-while-while*:  $x * (y * z) \leq (x * y) * z$  *<proof>*

**proposition** *while-zero-zero*:  $(x * bot) * bot = x * bot$  *<proof>*

**proposition** *while-denest-3*:  $(x * w) * (x * bot) = (x * w) * bot$  *<proof>*

**proposition** *while-02*:  $x * ((x * w) * ((x * y) * z)) = (x * w) * ((x * y) * z)$  *<proof>*

**proposition** *while-sumstar-3-below*:  $(x * y) * (x * z) \leq (x * y) * ((x * 1) * z)$  *<proof>*

**proposition** *while-sumstar-1*:  $(x \sqcup y) * z = (x * y) * ((x * 1) * z)$  *<proof>*

**proposition** *while-denest-4*:  $(x * w) * (x * (y * z)) = (x * w) * ((x * y) * z)$  *<proof>*

end

**class** *nonstrict-itering-zero* = *nonstrict-itering* +  
**assumes** *mult-right-zero*:  $x * bot = bot$   
**begin**

**lemma** *while-finite-associative-2*:

$x * bot = bot \implies (x * y) * z = x * (y * z)$   
*<proof>*

### Theorem 13 counterexamples

**proposition** *while-mult-top*:  $(x * top) * z = z \sqcup x * top$  **nitpick**  
*[expect=genuine,card=3]* *<proof>*

**proposition** *tarski-mult-top-idempotent*:  $x * top = x * top * x * top$  **nitpick**  
*[expect=genuine,card=3]* *<proof>*

**proposition** *while-denest-0*:  $w * (x * (y * z)) \leq (w * (x * y)) * (w * (x * y) * z)$  **nitpick** *[expect=genuine,card=3]* *<proof>*

**proposition** *while-denest-1*:  $w * (x * (y * z)) \leq (w * (x * y)) * z$  **nitpick**  
*[expect=genuine,card=3]* *<proof>*

**proposition** *while-mult-zero-zero*:  $(x * (y * bot)) * bot = x * (y * bot)$  **nitpick**  
*[expect=genuine,card=3]* *<proof>*

**proposition** *while-denest-2*:  $w * ((x * (y * w)) * z) = w * (((x * y) * w) * z)$  **nitpick** *[expect=genuine,card=3]* *<proof>*

**proposition** *while-denest-5*:  $w * ((x * (y * w)) * (x * (y * z))) = w * (((x * y) * w) * w) * ((x * y) * z)$  **nitpick** *[expect=genuine,card=3]* *<proof>*

**proposition** *while-denest-6*:  $(w * (x * y)) * z = z \sqcup w * ((x \sqcup y * w) * (y * z))$  **nitpick** *[expect=genuine,card=3]* *<proof>*

**proposition** *while-sum-below-one*:  $y * ((x \sqcup y) * z) \leq (y * (x * 1)) * z$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *while-separate-unfold*:  $(y * (x * 1)) * z = (y * z) \sqcup (y * (y * x * (x * ((y * (x * 1)) * z))))$  **nitpick**  $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *while-sub-while-zero*:  $x * z \leq (x * y) * z$  **nitpick**  
 $[expect=genuine,card=4]$   $\langle proof \rangle$

**proposition** *while-while-sub-associative*:  $x * (y * z) \leq (x * y) * z$  **nitpick**  
 $[expect=genuine,card=4]$   $\langle proof \rangle$

**proposition** *tarski*:  $x \leq x * top * x * top$  **nitpick**  $[expect=genuine,card=3]$   
 $\langle proof \rangle$

**proposition** *tarski-top-omega-below*:  $x * top \leq (x * top)^\omega$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *tarski-top-omega*:  $x * top = (x * top)^\omega$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *tarski-below-top-omega*:  $x \leq (x * top)^\omega$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *tarski-mult-omega-omega*:  $(x * y^\omega)^\omega = x * y^\omega$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *tarski-mult-omega-omega*:  $(\forall z . z^{\omega\omega} = z^\omega) \implies (x * y^\omega)^\omega = x * y^\omega$   
**nitpick**  $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *tarski*:  $x = bot \vee top * x * top = top$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**end**

**class** *nonstrict-itering-tarski* = *nonstrict-itering* +  
**assumes** *tarski*:  $x \leq x * top * x * top$   
**begin**

**Theorem 13.14**

**lemma** *tarski-mult-top-idempotent*:  
 $x * top = x * top * x * top$   
 $\langle proof \rangle$

**lemma** *tarski-top-omega-below*:  
 $x * top \leq (x * top)^\omega$   
 $\langle proof \rangle$

**lemma** *tarski-top-omega*:  
 $x * top = (x * top)^\omega$   
 $\langle proof \rangle$

**lemma** *tarski-below-top-omega*:  
 $x \leq (x * top)^\omega$   
 $\langle proof \rangle$

**lemma** *tarski-mult-omega-omega*:  
 $(x * y^\omega)^\omega = x * y^\omega$

$\langle proof \rangle$

**lemma** *tarski-omega-idempotent*:

$$x^{\omega\omega} = x^{\omega}$$

$\langle proof \rangle$

**lemma** *while-denest-2a*:

$$w * ((x * (y * w)) * z) = w * (((x * y) * w) * z)$$

$\langle proof \rangle$

**lemma** *while-denest-3*:

$$(x * w) * x^{\omega} = (x * w)^{\omega}$$

$\langle proof \rangle$

**lemma** *while-denest-4a*:

$$(x * w) * (x * (y * z)) = (x * w) * ((x * y) * z)$$

$\langle proof \rangle$

### Theorem 8.3

**subclass** *bounded-extended-binary-itering*

$\langle proof \rangle$

### Theorem 13.13

**lemma** *while-mult-top*:

$$(x * top) * z = z \sqcup x * top$$

$\langle proof \rangle$

**lemma** *tarski-top-omega-below-2*:

$$x * top \leq (x * top) * bot$$

$\langle proof \rangle$

**lemma** *tarski-top-omega-2*:

$$x * top = (x * top) * bot$$

$\langle proof \rangle$

**lemma** *tarski-below-top-omega-2*:

$$x \leq (x * top) * bot$$

$\langle proof \rangle$

**proposition**  $1 = (x * bot) * 1$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**end**

**class** *nonstrict-itering-tarski-zero* = *nonstrict-itering-tarski* +  
*nonstrict-itering-zero*

**begin**

**lemma** *while-bot-1*:

$$1 = (x * bot) * 1$$



*<proof>*

### Theorem 13 counterexamples

**proposition** *while-associative*:  $(x \star y) \star z = x \star (y \star z)$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition**  $(x \star 1) \star y = x \star y$  **nitpick** [*expect=genuine,card=2*] *<proof>*

**proposition** *while-one-mult*:  $(x \star 1) \star x = x \star x$  **nitpick**

[*expect=genuine,card=4*] *<proof>*

**proposition**  $(x \sqcup y) \star z = ((x \star 1) \star y) \star ((x \star 1) \star z)$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *while-mult-top-2*:  $(x \star \text{top}) \star z = z \sqcup x \star \text{top} \star z$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *while-top-2*:  $\text{top} \star z = \text{top} \star z$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *tarski*:  $x = \text{bot} \vee \text{top} \star x \star \text{top} = \text{top}$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition** *while-back-loop-is-fixpoint*: *is-fixpoint*  $(\lambda x . x \star y \sqcup z) (z \star (y \star 1))$

**nitpick** [*expect=genuine,card=4*] *<proof>*

**proposition**  $1 \sqcup x \star \text{bot} = x \star 1$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**proposition**  $x = x \star (x \star 1)$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**proposition**  $x \star (x \star 1) = x \star 1$  **nitpick** [*expect=genuine,card=2*] *<proof>*

**proposition**  $x \star 1 = x \star (1 \star 1)$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**proposition**  $(x \sqcup y) \star 1 = (x \star (y \star 1)) \star 1$  **nitpick** [*expect=genuine,card=3*]

*<proof>*

**proposition**  $z \sqcup y \star x = x \implies y \star z \leq x$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition**  $y \star x = x \implies y \star x \leq x$  **nitpick** [*expect=genuine,card=2*] *<proof>*

**proposition**  $z \sqcup x \star y = x \implies z \star (y \star 1) \leq x$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $x \star y = x \implies x \star (y \star 1) \leq x$  **nitpick** [*expect=genuine,card=3*]

*<proof>*

**proposition**  $x \star z = z \star y \implies x \star z \leq z \star (y \star 1)$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *tarski*:  $x = \text{bot} \vee \text{top} \star x \star \text{top} = \text{top}$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition** *tarski-case*: **assumes**  $t1: x = \text{bot} \implies P x$  **and**  $t2: \text{top} \star x \star \text{top} =$

$\text{top} \implies P x$  **shows**  $P x$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**end**

**end**

## 10 Tests

**theory** *Tests*

**imports** *Subset-Boolean-Algebras.Subset-Boolean-Algebras Base*

**begin**

**context** *subset-boolean-algebra-extended*  
**begin**

**sublocale** *sba-dual: subset-boolean-algebra-extended* **where** *uminus = uminus*  
**and** *sup = inf* **and** *minus =  $\lambda x y . \neg(-x \sqcap y)$*  **and** *inf = sup* **and** *bot = top*  
**and** *less-eq = greater-eq* **and** *less = greater* **and** *top = bot*  
*<proof>*

**lemma** *strict-leq-def:*  
 $-x < -y \iff -x \leq -y \wedge \neg(-y \leq -x)$   
*<proof>*

**lemma** *one-def:*  
 $top = -bot$   
*<proof>*

**end**

**class** *tests = times + uminus + one + ord + sup + bot +*  
**assumes** *sub-assoc:  $-x * (-y * -z) = (-x * -y) * -z$*   
**assumes** *sub-comm:  $-x * -y = -y * -x$*   
**assumes** *sub-compl:  $-x = -(-x * -y) * -(-x * -y)$*   
**assumes** *sub-mult-closed:  $-x * -y = -(-x * -y)$*   
**assumes** *the-bot-def:  $bot = (THE x . (\forall y . x = -y * -y))$*   
**assumes** *one-def:  $1 = - bot$*   
**assumes** *sup-def:  $-x \sqcup -y = -(-x * -y)$*   
**assumes** *leq-def:  $-x \leq -y \iff -x * -y = -x$*   
**assumes** *strict-leq-def:  $-x < -y \iff -x \leq -y \wedge \neg(-y \leq -x)$*   
**begin**

**sublocale** *tests-dual: subset-boolean-algebra-extended* **where** *uminus = uminus*  
**and** *sup = times* **and** *minus =  $\lambda x y . \neg(-x * y)$*  **and** *inf = sup* **and** *bot = 1*  
**and** *less-eq = greater-eq* **and** *less = greater* **and** *top = bot*  
*<proof>*

**sublocale** *sba: subset-boolean-algebra-extended* **where** *uminus = uminus* **and**  
*sup = sup* **and** *minus =  $\lambda x y . \neg(-x \sqcup y)$*  **and** *inf = times* **and** *bot = bot* **and**  
*less-eq = less-eq* **and** *less = less* **and** *top = 1* *<proof>*

[sets and sequences of tests](#)

**definition** *test-set :: 'a set  $\Rightarrow$  bool*  
**where** *test-set A  $\equiv \forall x \in A . x = --x$*

**lemma** *mult-left-dist-test-set:*  
 $test\text{-set } A \implies test\text{-set } \{ -p * x \mid x . x \in A \}$   
*<proof>*

**lemma** *mult-right-dist-test-set*:

$test\text{-}set\ A \implies test\text{-}set\ \{ x * -p \mid x . x \in A \}$   
*<proof>*

**lemma** *sup-left-dist-test-set*:

$test\text{-}set\ A \implies test\text{-}set\ \{ -p \sqcup x \mid x . x \in A \}$   
*<proof>*

**lemma** *sup-right-dist-test-set*:

$test\text{-}set\ A \implies test\text{-}set\ \{ x \sqcup -p \mid x . x \in A \}$   
*<proof>*

**lemma** *test-set-closed*:

$A \subseteq B \implies test\text{-}set\ B \implies test\text{-}set\ A$   
*<proof>*

**definition** *test-seq* ::  $(nat \Rightarrow 'a) \Rightarrow bool$

**where**  $test\text{-}seq\ t \equiv \forall n . t\ n = \neg\neg t\ n$

**lemma** *test-seq-test-set*:

$test\text{-}seq\ t \implies test\text{-}set\ \{ t\ n \mid n::nat . True \}$   
*<proof>*

**definition** *nat-test* ::  $(nat \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$

**where**  $nat\text{-}test\ t\ s \equiv (\forall n . t\ n = \neg\neg t\ n) \wedge s = \neg\neg s \wedge (\forall n . t\ n \leq s) \wedge (\forall x\ y . (\forall n . t\ n * -x \leq -y) \longrightarrow s * -x \leq -y)$

**lemma** *nat-test-seq*:

$nat\text{-}test\ t\ s \implies test\text{-}seq\ t$   
*<proof>*

**primrec** *pSum* ::  $(nat \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a$

**where**  $pSum\ f\ 0 = bot$   
 $\mid pSum\ f\ (Suc\ m) = pSum\ f\ m \sqcup f\ m$

**lemma** *pSum-test*:

$test\text{-}seq\ t \implies pSum\ t\ m = \neg\neg(pSum\ t\ m)$   
*<proof>*

**lemma** *pSum-test-nat*:

$nat\text{-}test\ t\ s \implies pSum\ t\ m = \neg\neg(pSum\ t\ m)$   
*<proof>*

**lemma** *pSum-upper*:

$test\text{-}seq\ t \implies i < m \implies t\ i \leq pSum\ t\ m$   
*<proof>*

**lemma** *pSum-below*:

$test\text{-}seq\ t \implies (\forall m < k . t\ m * -p \leq -q) \implies pSum\ t\ k * -p \leq -q$   
 ⟨proof⟩

**lemma** *pSum-below-nat:*

$nat\text{-}test\ t\ s \implies (\forall m < k . t\ m * -p \leq -q) \implies pSum\ t\ k * -p \leq -q$   
 ⟨proof⟩

**lemma** *pSum-below-sum:*

$nat\text{-}test\ t\ s \implies pSum\ t\ x \leq s$   
 ⟨proof⟩

**lemma** *ascending-chain-sup-left:*

$ascending\text{-}chain\ t \implies test\text{-}seq\ t \implies ascending\text{-}chain\ (\lambda n . -p \sqcup t\ n) \wedge test\text{-}seq\ (\lambda n . -p \sqcup t\ n)$   
 ⟨proof⟩

**lemma** *ascending-chain-sup-right:*

$ascending\text{-}chain\ t \implies test\text{-}seq\ t \implies ascending\text{-}chain\ (\lambda n . t\ n \sqcup -p) \wedge test\text{-}seq\ (\lambda n . t\ n \sqcup -p)$   
 ⟨proof⟩

**lemma** *ascending-chain-mult-left:*

$ascending\text{-}chain\ t \implies test\text{-}seq\ t \implies ascending\text{-}chain\ (\lambda n . -p * t\ n) \wedge test\text{-}seq\ (\lambda n . -p * t\ n)$   
 ⟨proof⟩

**lemma** *ascending-chain-mult-right:*

$ascending\text{-}chain\ t \implies test\text{-}seq\ t \implies ascending\text{-}chain\ (\lambda n . t\ n * -p) \wedge test\text{-}seq\ (\lambda n . t\ n * -p)$   
 ⟨proof⟩

**lemma** *descending-chain-sup-left:*

$descending\text{-}chain\ t \implies test\text{-}seq\ t \implies descending\text{-}chain\ (\lambda n . -p \sqcup t\ n) \wedge test\text{-}seq\ (\lambda n . -p \sqcup t\ n)$   
 ⟨proof⟩

**lemma** *descending-chain-sup-right:*

$descending\text{-}chain\ t \implies test\text{-}seq\ t \implies descending\text{-}chain\ (\lambda n . t\ n \sqcup -p) \wedge test\text{-}seq\ (\lambda n . t\ n \sqcup -p)$   
 ⟨proof⟩

**lemma** *descending-chain-mult-left:*

$descending\text{-}chain\ t \implies test\text{-}seq\ t \implies descending\text{-}chain\ (\lambda n . -p * t\ n) \wedge test\text{-}seq\ (\lambda n . -p * t\ n)$   
 ⟨proof⟩

**lemma** *descending-chain-mult-right:*

$descending\text{-}chain\ t \implies test\text{-}seq\ t \implies descending\text{-}chain\ (\lambda n . t\ n * -p) \wedge test\text{-}seq\ (\lambda n . t\ n * -p)$

$\langle proof \rangle$

end

end

## 11 Test Iterings

theory *Test-Iterings*

imports *Stone-Kleene-Relation-Algebras.Iterings Tests*

begin

class *test-itering* = *itering* + *tests* + *while* +

assumes *while-def*:  $p \star y = (p * y)^\circ * -p$

begin

lemma *wnf-lemma-5*:

$(-p \sqcup -q) * (-q * x \sqcup --q * y) = -q * x \sqcup --q * -p * y$

$\langle proof \rangle$

lemma *test-case-split-left-equal*:

$-z * x = -z * y \implies --z * x = --z * y \implies x = y$

$\langle proof \rangle$

lemma *preserves-equation*:

$-y * x \leq x * -y \iff -y * x = -y * x * -y$

$\langle proof \rangle$

[Theorem 5](#)

lemma *preserve-test*:

$-y * x \leq x * -y \implies -y * x^\circ = -y * x^\circ * -y$

$\langle proof \rangle$

[Theorem 5](#)

lemma *import-test*:

$-y * x \leq x * -y \implies -y * x^\circ = -y * (-y * x)^\circ$

$\langle proof \rangle$

definition *ite* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$  ( $\langle - \triangleleft - \triangleright - \rangle$  [58,58,58] 57)

where  $x \triangleleft p \triangleright y \equiv p * x \sqcup -p * y$

definition *it* ::  $'a \Rightarrow 'a \Rightarrow 'a$  ( $\langle - \triangleright - \rangle$  [58,58] 57)

where  $p \triangleright x \equiv p * x \sqcup -p$

definition *assigns* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$

**where** *assigns*  $x \ p \ q \equiv x = x * (p * q \sqcup -p * -q)$

**definition** *preserves*  $:: 'a \Rightarrow 'a \Rightarrow \text{bool}$

**where** *preserves*  $x \ p \equiv p * x \leq x * p \wedge -p * x \leq x * -p$

**lemma** *ite-neg*:

$x \triangleleft -p \triangleright y = y \triangleleft --p \triangleright x$   
(*proof*)

**lemma** *ite-import-true*:

$x \triangleleft -p \triangleright y = -p * x \triangleleft -p \triangleright y$   
(*proof*)

**lemma** *ite-import-false*:

$x \triangleleft -p \triangleright y = x \triangleleft -p \triangleright --p * y$   
(*proof*)

**lemma** *ite-import-true-false*:

$x \triangleleft -p \triangleright y = -p * x \triangleleft -p \triangleright --p * y$   
(*proof*)

**lemma** *ite-context-true*:

$-p * (x \triangleleft -p \triangleright y) = -p * x$   
(*proof*)

**lemma** *ite-context-false*:

$--p * (x \triangleleft -p \triangleright y) = --p * y$   
(*proof*)

**lemma** *ite-context-import*:

$-p * (x \triangleleft -q \triangleright y) = -p * (x \triangleleft -p * -q \triangleright y)$   
(*proof*)

**lemma** *ite-conjunction*:

$(x \triangleleft -q \triangleright y) \triangleleft -p \triangleright y = x \triangleleft -p * -q \triangleright y$   
(*proof*)

**lemma** *ite-disjunction*:

$x \triangleleft -p \triangleright (x \triangleleft -q \triangleright y) = x \triangleleft -p \sqcup -q \triangleright y$   
(*proof*)

**lemma** *wnf-lemma-6*:

$(-p \sqcup -q) * (x \triangleleft --p * -q \triangleright y) = (-p \sqcup -q) * (y \triangleleft -p \triangleright x)$   
(*proof*)

**lemma** *it-ite*:

$-p \triangleright x = x \triangleleft -p \triangleright 1$   
(*proof*)

**lemma** *it-neg*:

$$\neg\neg p \triangleright x = 1 \triangleleft \neg p \triangleright x$$

*<proof>*

**lemma** *it-import-true*:

$$\neg p \triangleright x = \neg p \triangleright \neg p * x$$

*<proof>*

**lemma** *it-context-true*:

$$\neg p * (\neg p \triangleright x) = \neg p * x$$

*<proof>*

**lemma** *it-context-false*:

$$\neg\neg p * (\neg p \triangleright x) = \neg\neg p$$

*<proof>*

**lemma** *while-unfold-it*:

$$\neg p * x = \neg p \triangleright x * (\neg p * x)$$

*<proof>*

**lemma** *while-context-false*:

$$\neg\neg p * (\neg p * x) = \neg\neg p$$

*<proof>*

**lemma** *while-context-true*:

$$\neg p * (\neg p * x) = \neg p * x * (\neg p * x)$$

*<proof>*

**lemma** *while-zero*:

$$\text{bot} * x = 1$$

*<proof>*

**lemma** *wnf-lemma-7*:

$$1 * (\text{bot} * 1) = 1$$

*<proof>*

**lemma** *while-import-condition*:

$$\neg p * x = \neg p * \neg p * x$$

*<proof>*

**lemma** *while-import-condition-2*:

$$\neg p * \neg q * x = \neg p * \neg q * \neg p * x$$

*<proof>*

**lemma** *wnf-lemma-8*:

$$\neg r * (\neg p \sqcup \neg\neg p * \neg q) * (x \triangleleft \neg\neg p * \neg q \triangleright y) = \neg r * (\neg p \sqcup \neg q) * (y \triangleleft \neg p \triangleright x)$$

*<proof>*

Theorem 6 - see Theorem 31 on page 329 of Back and von Wright, Acta

[Informatica 36:295-334, 1999](#)

**lemma** *split-merge-loops*:

**assumes**  $--p * y \leq y * --p$

**shows**  $(-p \sqcup -q) * (x \triangleleft -p \triangleright y) = (-p * x) * (-q * y)$

*<proof>*

**lemma** *assigns-same*:

*assigns*  $x$   $(-p)$   $(-p)$

*<proof>*

**lemma** *preserves-equation-test*:

*preserves*  $x$   $(-p) \longleftrightarrow -p * x = -p * x * -p \wedge --p * x = --p * x * --p$

*<proof>*

**lemma** *preserves-test*:

*preserves*  $(-q)$   $(-p)$

*<proof>*

**lemma** *preserves-zero*:

*preserves* *bot*  $(-p)$

*<proof>*

**lemma** *preserves-one*:

*preserves* *1*  $(-p)$

*<proof>*

**lemma** *preserves-sup*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $y$   $(-p) \implies$  *preserves*  $(x \sqcup y)$   $(-p)$

*<proof>*

**lemma** *preserves-mult*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $y$   $(-p) \implies$  *preserves*  $(x * y)$   $(-p)$

*<proof>*

**lemma** *preserves-ite*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $y$   $(-p) \implies$  *preserves*  $(x \triangleleft -q \triangleright y)$   $(-p)$

*<proof>*

**lemma** *preserves-it*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $(-q \triangleright x)$   $(-p)$

*<proof>*

**lemma** *preserves-circ*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $(x^\circ)$   $(-p)$

*<proof>*

**lemma** *preserves-while*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $(-q * x)$   $(-p)$

*<proof>*



**lemma** *preserves-test-neg*:

*preserves*  $x$   $(-p) \implies$  *preserves*  $x$   $(--p)$   
*<proof>*

**lemma** *preserves-import-circ*:

*preserves*  $x$   $(-p) \implies -p * x^\circ = -p * (-p * x)^\circ$   
*<proof>*

**lemma** *preserves-simulate*:

*preserves*  $x$   $(-p) \implies -p * x^\circ = -p * x^\circ * -p$   
*<proof>*

**lemma** *preserves-import-ite*:

**assumes** *preserves*  $z$   $(-p)$   
**shows**  $z * (x \triangleleft -p \triangleright y) = z * x \triangleleft -p \triangleright z * y$   
*<proof>*

**lemma** *preserves-while-context*:

*preserves*  $x$   $(-p) \implies -p * (-q \star x) = -p * (-p * -q \star x)$   
*<proof>*

**lemma** *while-ite-context-false*:

**assumes** *preserves*  $y$   $(-p)$   
**shows**  $--p * (-p \sqcup -q \star (x \triangleleft -p \triangleright y)) = --p * (-q \star y)$   
*<proof>*

### Theorem 7.1

**lemma** *while-ite-norm*:

**assumes** *assigns*  $z$   $(-p)$   $(-q)$   
**and** *preserves*  $x1$   $(-q)$   
**and** *preserves*  $x2$   $(-q)$   
**and** *preserves*  $y1$   $(-q)$   
**and** *preserves*  $y2$   $(-q)$   
**shows**  $z * (x1 * (-r1 \star y1) \triangleleft -p \triangleright x2 * (-r2 \star y2)) = z * (x1 \triangleleft -q \triangleright x2) * ((-q * -r1 \sqcup -q * -r2) \star (y1 \triangleleft -q \triangleright y2))$   
*<proof>*

**lemma** *while-it-norm*:

*assigns*  $z$   $(-p)$   $(-q) \implies$  *preserves*  $x$   $(-q) \implies$  *preserves*  $y$   $(-q) \implies z * (-p \triangleright x * (-r \star y)) = z * (-q \triangleright x) * (-q * -r \star y)$   
*<proof>*

**lemma** *while-else-norm*:

*assigns*  $z$   $(-p)$   $(-q) \implies$  *preserves*  $x$   $(-q) \implies$  *preserves*  $y$   $(-q) \implies z * (1 \triangleleft -p \triangleright x * (-r \star y)) = z * (1 \triangleleft -q \triangleright x) * (-q * -r \star y)$   
*<proof>*

**lemma** *while-while-pre-norm*:

$$-p \star x \star (-q \star y) = -p \triangleright x \star (-p \sqcup -q \star (y \triangleleft -q \triangleright x))$$

*<proof>*

### Theorem 7.2

**lemma** *while-while-norm*:

$$\text{assigns } z \ (-p) \ (-r) \implies \text{preserves } x \ (-r) \implies \text{preserves } y \ (-r) \implies z \star (-p \star x \star (-q \star y)) = z \star (-r \triangleright x) \star (-r \star (-p \sqcup -q) \star (y \triangleleft -q \triangleright x))$$

*<proof>*

**lemma** *while-seq-replace*:

$$\text{assigns } z \ (-p) \ (-q) \implies z \star (-p \star x \star z) \star y = z \star (-q \star x \star z) \star y$$

*<proof>*

**lemma** *while-ite-replace*:

$$\text{assigns } z \ (-p) \ (-q) \implies z \star (x \triangleleft -p \triangleright y) = z \star (x \triangleleft -q \triangleright y)$$

*<proof>*

**lemma** *while-post-norm-an*:

**assumes** *preserves*  $y \ (-p)$   
**shows**  $(-p \star x) \star y = y \triangleleft -p \triangleright (-p \star x \star (-p \triangleright y))$

*<proof>*

**lemma** *while-post-norm*:

$$\text{preserves } y \ (-p) \implies (-p \star x) \star y = -p \star x \star (1 \triangleleft -p \triangleright y) \triangleleft -p \triangleright y$$

*<proof>*

**lemma** *wnf-lemma-9*:

**assumes** *assigns*  $z \ (-p) \ (-q)$   
**and** *preserves*  $x1 \ (-q)$   
**and** *preserves*  $y1 \ (-q)$   
**and** *preserves*  $x2 \ (-q)$   
**and** *preserves*  $y2 \ (-q)$   
**and** *preserves*  $x2 \ (-p)$   
**and** *preserves*  $y2 \ (-p)$   
**shows**  $z \star (x1 \triangleleft -q \triangleright x2) \star (-q \star -p \sqcup -r \star (y1 \triangleleft -q \star -p \triangleright y2)) = z \star (x1 \triangleleft -p \triangleright x2) \star (-p \sqcup -r \star (y1 \triangleleft -p \triangleright y2))$

*<proof>*

### Theorem 7.3

**lemma** *while-seq-norm*:

**assumes** *assigns*  $z1 \ (-r1) \ (-q)$   
**and** *preserves*  $x2 \ (-q)$   
**and** *preserves*  $y2 \ (-q)$   
**and** *preserves*  $z2 \ (-q)$   
**and**  $z1 \star z2 = z2 \star z1$   
**and** *assigns*  $z2 \ (-q) \ (-r)$   
**and** *preserves*  $y1 \ (-r)$   
**and** *preserves*  $z1 \ (-r)$   
**and** *preserves*  $x2 \ (-r)$

**and preserves  $y^2$  ( $-r$ )**  
**shows**  $x1 * z1 * z2 * (-r1 \star y1 * z1) * x2 * (-r2 \star y2) = x1 * z1 * z2 * (y1 * z1 * (1 \triangleleft -q \triangleright x2) \triangleleft -q \triangleright x2) * (-q \sqcup -r2 \star (y1 * z1 * (1 \triangleleft -q \triangleright x2) \triangleleft -q \triangleright y2))$   
 <proof>

**end**

**end**

## 12 N-Semirings

**theory** *N-Semirings*

**imports** *Test-Iterings Omega-Algebras*

**begin**

**class** *n-semiring* = *bounded-idempotent-left-zero-semiring* + *n* + *L* +  
**assumes** *n-bot* :  $n(\text{bot}) = \text{bot}$   
**assumes** *n-top* :  $n(\text{top}) = 1$   
**assumes** *n-dist-sup* :  $n(x \sqcup y) = n(x) \sqcup n(y)$   
**assumes** *n-export* :  $n(n(x) * y) = n(x) * n(y)$   
**assumes** *n-sub-mult-bot*:  $n(x) = n(x * \text{bot}) * n(x)$   
**assumes** *n-L-split* :  $x * n(y) * L = x * \text{bot} \sqcup n(x * y) * L$   
**assumes** *n-split* :  $x \leq x * \text{bot} \sqcup n(x * L) * \text{top}$

**begin**

**lemma** *n-sub-one*:

$n(x) \leq 1$   
 <proof>

[Theorem 15](#)

**lemma** *n-isotone*:

$x \leq y \implies n(x) \leq n(y)$   
 <proof>

**lemma** *n-mult-idempotent*:

$n(x) * n(x) = n(x)$   
 <proof>

[Theorem 15.3](#)

**lemma** *n-mult-bot*:

$n(x) = n(x * \text{bot})$   
 <proof>

**lemma** *n-mult-left-upper-bound*:

$n(x) \leq n(x * y)$   
 <proof>

**lemma** *n-mult-right-bot*:

$$n(x) * bot = bot$$

*<proof>*

**Theorem 15.9**

**lemma** *n-mult-n*:

$$n(x * n(y)) = n(x)$$

*<proof>*

**lemma** *n-mult-left-absorb-sup*:

$$n(x) * (n(x) \sqcup n(y)) = n(x)$$

*<proof>*

**lemma** *n-mult-right-absorb-sup*:

$$(n(x) \sqcup n(y)) * n(y) = n(y)$$

*<proof>*

**lemma** *n-sup-left-absorb-mult*:

$$n(x) \sqcup n(x) * n(y) = n(x)$$

*<proof>*

**lemma** *n-sup-right-absorb-mult*:

$$n(x) * n(y) \sqcup n(y) = n(y)$$

*<proof>*

**lemma** *n-mult-commutative*:

$$n(x) * n(y) = n(y) * n(x)$$

*<proof>*

**lemma** *n-sup-left-dist-mult*:

$$n(x) \sqcup n(y) * n(z) = (n(x) \sqcup n(y)) * (n(x) \sqcup n(z))$$

*<proof>*

**lemma** *n-sup-right-dist-mult*:

$$n(x) * n(y) \sqcup n(z) = (n(x) \sqcup n(z)) * (n(y) \sqcup n(z))$$

*<proof>*

**lemma** *n-order*:

$$n(x) \leq n(y) \iff n(x) * n(y) = n(x)$$

*<proof>*

**lemma** *n-mult-left-lower-bound*:

$$n(x) * n(y) \leq n(x)$$

*<proof>*

**lemma** *n-mult-right-lower-bound*:

$$n(x) * n(y) \leq n(y)$$

*<proof>*

**lemma** *n-mult-least-upper-bound*:

$$n(x) \leq n(y) \wedge n(x) \leq n(z) \longleftrightarrow n(x) \leq n(y) * n(z)$$

*<proof>*

**lemma** *n-mult-left-divisibility*:

$$n(x) \leq n(y) \longleftrightarrow (\exists z . n(x) = n(y) * n(z))$$

*<proof>*

**lemma** *n-mult-right-divisibility*:

$$n(x) \leq n(y) \longleftrightarrow (\exists z . n(x) = n(z) * n(y))$$

*<proof>*

### Theorem 15.1

**lemma** *n-one*:

$$n(1) = \text{bot}$$

*<proof>*

**lemma** *n-split-equal*:

$$x \sqcup n(x * L) * \text{top} = x * \text{bot} \sqcup n(x * L) * \text{top}$$

*<proof>*

**lemma** *n-split-top*:

$$x * \text{top} \leq x * \text{bot} \sqcup n(x * L) * \text{top}$$

*<proof>*

### Theorem 15.2

**lemma** *n-L*:

$$n(L) = 1$$

*<proof>*

### Theorem 15.5

**lemma** *n-split-L*:

$$x * L = x * \text{bot} \sqcup n(x * L) * L$$

*<proof>*

**lemma** *n-n-L*:

$$n(n(x) * L) = n(x)$$

*<proof>*

**lemma** *n-L-decreasing*:

$$n(x) * L \leq x$$

*<proof>*

### Theorem 15.10

**lemma** *n-galois*:

$$n(x) \leq n(y) \longleftrightarrow n(x) * L \leq y$$

*<proof>*

### Theorem 15.6

**lemma** *split-L*:  
 $x * L \leq x * bot \sqcup L$   
 ⟨proof⟩

[Theorem 15.7](#)

**lemma** *L-left-zero*:  
 $L * x = L$   
 ⟨proof⟩

[Theorem 15.8](#)

**lemma** *n-mult*:  
 $n(x * n(y) * L) = n(x * y)$   
 ⟨proof⟩

**lemma** *n-mult-top*:  
 $n(x * n(y) * top) = n(x * y)$   
 ⟨proof⟩

[Theorem 15.4](#)

**lemma** *n-top-L*:  
 $n(x * top) = n(x * L)$   
 ⟨proof⟩

**lemma** *n-top-split*:  
 $x * n(y) * top \leq x * bot \sqcup n(x * y) * top$   
 ⟨proof⟩

**lemma** *n-mult-right-upper-bound*:  
 $n(x * y) \leq n(z) \iff n(x) \leq n(z) \wedge x * n(y) * L \leq x * bot \sqcup n(z) * L$   
 ⟨proof⟩

**lemma** *n-preserves-equation*:  
 $n(y) * x \leq x * n(y) \iff n(y) * x = n(y) * x * n(y)$   
 ⟨proof⟩

**definition** *ni* :: 'a ⇒ 'a  
 where *ni* x = n(x) \* L

**lemma** *ni-bot*:  
 $ni(bot) = bot$   
 ⟨proof⟩

**lemma** *ni-one*:  
 $ni(1) = bot$   
 ⟨proof⟩

**lemma** *ni-L*:  
 $ni(L) = L$   
 ⟨proof⟩

**lemma** *ni-top*:

$$ni(top) = L$$

*<proof>*

**lemma** *ni-dist-sup*:

$$ni(x \sqcup y) = ni(x) \sqcup ni(y)$$

*<proof>*

**lemma** *ni-mult-bot*:

$$ni(x) = ni(x * bot)$$

*<proof>*

**lemma** *ni-split*:

$$x * ni(y) = x * bot \sqcup ni(x * y)$$

*<proof>*

**lemma** *ni-decreasing*:

$$ni(x) \leq x$$

*<proof>*

**lemma** *ni-isotone*:

$$x \leq y \implies ni(x) \leq ni(y)$$

*<proof>*

**lemma** *ni-mult-left-upper-bound*:

$$ni(x) \leq ni(x * y)$$

*<proof>*

**lemma** *ni-idempotent*:

$$ni(ni(x)) = ni(x)$$

*<proof>*

**lemma** *ni-below-L*:

$$ni(x) \leq L$$

*<proof>*

**lemma** *ni-left-zero*:

$$ni(x) * y = ni(x)$$

*<proof>*

**lemma** *ni-split-L*:

$$x * L = x * bot \sqcup ni(x * L)$$

*<proof>*

**lemma** *ni-top-L*:

$$ni(x * top) = ni(x * L)$$

*<proof>*

**lemma** *ni-galois*:

$$ni(x) \leq ni(y) \longleftrightarrow ni(x) \leq y$$

*<proof>*

**lemma** *ni-mult*:

$$ni(x * ni(y)) = ni(x * y)$$

*<proof>*

**lemma** *ni-n-order*:

$$ni(x) \leq ni(y) \longleftrightarrow n(x) \leq n(y)$$

*<proof>*

**lemma** *ni-n-equal*:

$$ni(x) = ni(y) \longleftrightarrow n(x) = n(y)$$

*<proof>*

**lemma** *ni-mult-right-upper-bound*:

$$ni(x * y) \leq ni(z) \longleftrightarrow ni(x) \leq ni(z) \wedge x * ni(y) \leq x * bot \sqcup ni(z)$$

*<proof>*

**lemma** *n-ni*:

$$n(ni(x)) = n(x)$$

*<proof>*

**lemma** *ni-n*:

$$ni(n(x)) = bot$$

*<proof>*

**lemma** *ni-n-galois*:

$$n(x) \leq n(y) \longleftrightarrow ni(x) \leq y$$

*<proof>*

**lemma** *n-mult-ni*:

$$n(x * ni(y)) = n(x * y)$$

*<proof>*

**lemma** *ni-mult-n*:

$$ni(x * n(y)) = ni(x)$$

*<proof>*

**lemma** *ni-export*:

$$ni(n(x) * y) = n(x) * ni(y)$$

*<proof>*

**lemma** *ni-mult-top*:

$$ni(x * n(y) * top) = ni(x * y)$$

*<proof>*

**lemma** *ni-n-bot*:



$ni(x) = bot \longleftrightarrow n(x) = bot$   
 $\langle proof \rangle$

**lemma** *ni-n-L*:

$ni(x) = L \longleftrightarrow n(x) = 1$   
 $\langle proof \rangle$

**proposition** *n-bot* :  $n(bot) = bot$   $\langle proof \rangle$

**proposition** *n-top* :  $n(top) = 1$   $\langle proof \rangle$

**proposition** *n-dist-sup* :  $n(x \sqcup y) = n(x) \sqcup n(y)$   $\langle proof \rangle$

**proposition** *n-export* :  $n(n(x) * y) = n(x) * n(y)$   $\langle proof \rangle$

**proposition** *n-sub-mult-bot*:  $n(x) = n(x * bot) * n(x)$   $\langle proof \rangle$

**proposition** *n-L-split* :  $x * n(y) * L = x * bot \sqcup n(x * y) * L$   $\langle proof \rangle$

**proposition** *n-split* :  $x \leq x * bot \sqcup n(x * L) * top$   $\langle proof \rangle$

**end**

**typedef** (overloaded) *'a nImage* =  $\{ x::'a::n\text{-semiring} . (\exists y::'a . x = n(y)) \}$   
 $\langle proof \rangle$

**lemma** *simp-nImage*[*simp*]:

$\exists y . Rep\text{-}nImage\ x = n(y)$   
 $\langle proof \rangle$

**setup-lifting** *type-definition-nImage*

**Theorem 15**

**instantiation** *nImage* :: (*n-semiring*) *bounded-idempotent-semiring*  
**begin**

**lift-definition** *sup-nImage* :: *'a nImage*  $\Rightarrow$  *'a nImage*  $\Rightarrow$  *'a nImage* **is** *sup*  
 $\langle proof \rangle$

**lift-definition** *times-nImage* :: *'a nImage*  $\Rightarrow$  *'a nImage*  $\Rightarrow$  *'a nImage* **is** *times*  
 $\langle proof \rangle$

**lift-definition** *bot-nImage* :: *'a nImage* **is** *bot*  
 $\langle proof \rangle$

**lift-definition** *one-nImage* :: *'a nImage* **is** *1*  
 $\langle proof \rangle$

**lift-definition** *top-nImage* :: *'a nImage* **is** *1*  
 $\langle proof \rangle$

**lift-definition** *less-eq-nImage* :: *'a nImage*  $\Rightarrow$  *'a nImage*  $\Rightarrow$  *bool* **is** *less-eq*  $\langle proof \rangle$

**lift-definition** *less-nImage* :: *'a nImage*  $\Rightarrow$  *'a nImage*  $\Rightarrow$  *bool* **is** *less*  $\langle proof \rangle$

**instance**

*<proof>*

**end**

**Theorem 15**

**instantiation** *nImage* :: (*n-semiring*) *bounded-distrib-lattice*

**begin**

**lift-definition** *inf-nImage* :: 'a *nImage*  $\Rightarrow$  'a *nImage*  $\Rightarrow$  'a *nImage* **is times**

*<proof>*

**instance**

*<proof>*

**end**

**class** *n-itering* = *bounded-itering* + *n-semiring*

**begin**

**lemma** *mult-L-circ*:

$$(x * L)^\circ = 1 \sqcup x * L$$

*<proof>*

**lemma** *mult-L-circ-mult*:

$$(x * L)^\circ * y = y \sqcup x * L$$

*<proof>*

**lemma** *circ-L*:

$$L^\circ = L \sqcup 1$$

*<proof>*

**lemma** *circ-n-L*:

$$x^\circ * n(x) * L = x^\circ * \text{bot}$$

*<proof>*

**lemma** *n-circ-left-unfold*:

$$n(x^\circ) = n(x * x^\circ)$$

*<proof>*

**lemma** *ni-circ*:

$$ni(x)^\circ = 1 \sqcup ni(x)$$

*<proof>*

**lemma** *circ-ni*:

$$x^\circ * ni(x) = x^\circ * \text{bot}$$

*<proof>*

**lemma** *ni-circ-left-unfold*:

$$ni(x^\circ) = ni(x * x^\circ)$$

*<proof>*

**lemma** *n-circ-import*:

$$n(y) * x \leq x * n(y) \implies n(y) * x^\circ = n(y) * (n(y) * x)^\circ$$

*<proof>*

**end**

**class** *n-omega-itering* = *left-omega-conway-semiring* + *n-itering* +

**assumes** *circ-circ*:  $x^{\circ\circ} = L \sqcup x^*$

**begin**

**lemma** *L-below-one-circ*:

$$L \leq 1^\circ$$

*<proof>*

**lemma** *circ-below-L-sup-star*:

$$x^\circ \leq L \sqcup x^*$$

*<proof>*

**lemma** *L-sup-circ-sup-star*:

$$L \sqcup x^\circ = L \sqcup x^*$$

*<proof>*

**lemma** *circ-one-L*:

$$1^\circ = L \sqcup 1$$

*<proof>*

**lemma** *one-circ-zero*:

$$L = 1^\circ * bot$$

*<proof>*

**lemma** *circ-not-simulate*:

$$(\forall x y z . x * z \leq z * y \longrightarrow x^\circ * z \leq z * y^\circ) \longrightarrow 1 = bot$$

*<proof>*

**lemma** *star-circ-L*:

$$x^{*\circ} = L \sqcup x^*$$

*<proof>*

**lemma** *circ-circ-2*:

$$x^{\circ\circ} = L \sqcup x^\circ$$

*<proof>*

**lemma** *circ-sup-6*:

$$L \sqcup (x \sqcup y)^\circ = (x^\circ * y^\circ)^\circ$$

*<proof>*

**lemma** *circ-sup-7*:

$$(x^\circ * y^\circ)^\circ = L \sqcup (x \sqcup y)^*$$

*<proof>*

**end**

**class** *n-omega-algebra-2* = *bounded-left-zero-omega-algebra* + *n-semiring* + *Omega* +

**assumes** *Omega-def*:  $x^\Omega = n(x^\omega) * L \sqcup x^*$

**begin**

**lemma** *mult-L-star*:

$$(x * L)^* = 1 \sqcup x * L$$

*<proof>*

**lemma** *mult-L-omega*:

$$(x * L)^\omega = x * L$$

*<proof>*

**lemma** *mult-L-sup-star*:

$$(x * L \sqcup y)^* = y^* \sqcup y^* * x * L$$

*<proof>*

**lemma** *mult-L-sup-omega*:

$$(x * L \sqcup y)^\omega = y^\omega \sqcup y^* * x * L$$

*<proof>*

**lemma** *mult-L-sup-circ*:

$$(x * L \sqcup y)^\Omega = n(y^\omega) * L \sqcup y^* \sqcup y^* * x * L$$

*<proof>*

**lemma** *circ-sup-n*:

$$(x^\Omega * y)^\Omega * x^\Omega = n((x^* * y)^\omega) * L \sqcup ((x^* * y)^* * x^* \sqcup (x^* * y)^* * n(x^\omega) * L)$$

*<proof>*

### Theorem 20.6

**lemma** *n-omega-induct*:

$$n(y) \leq n(x * y \sqcup z) \implies n(y) \leq n(x^\omega \sqcup x^* * z)$$

*<proof>*

**lemma** *n-Omega-left-unfold*:

$$1 \sqcup x * x^\Omega = x^\Omega$$

*<proof>*

**lemma** *n-Omega-circ-sup*:

$$(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$$

*<proof>*

**lemma** *n-Omega-circ-simulate-right-sup*:

**assumes**  $z * x \leq y * y^\Omega * z \sqcup w$

**shows**  $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$

*<proof>*

**lemma** *n-Omega-circ-simulate-left-sup*:

**assumes**  $x * z \leq z * y^\Omega \sqcup w$

**shows**  $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$

*<proof>*

**end**

[Theorem 2.6](#) and [Theorem 19](#)

**sublocale** *n-omega-algebra-2* < *nL-omega*: *itering* **where** *circ* = *Omega*

*<proof>*

**sublocale** *n-omega-algebra-2* < *nL-omega*: *n-omega-itering* **where** *circ* = *Omega*

*<proof>*

**sublocale** *n-omega-algebra-2* < *nL-omega*: *left-zero-kleene-conway-semiring*

**where** *circ* = *Omega* *<proof>*

**sublocale** *n-omega-algebra-2* < *nL-star*: *left-omega-conway-semiring* **where** *circ*

= *star* *<proof>*

**context** *n-omega-algebra-2*

**begin**

**lemma** *circ-sup-8*:

$n((x^* * y)^* * x^\omega) * L \leq (x^* * y)^\Omega * x^\Omega$

*<proof>*

**lemma** *n-split-omega-omega*:

$x^\omega \leq x^\omega * \text{bot} \sqcup n(x^\omega) * \text{top}$

*<proof>*

[Theorem 20.1](#)

**lemma** *n-below-n-star*:

$n(x) \leq n(x^*)$

*<proof>*

[Theorem 20.2](#)

**lemma** *n-star-below-n-omega*:

$n(x^*) \leq n(x^\omega)$

*<proof>*

**lemma** *n-below-n-omega*:

$n(x) \leq n(x^\omega)$

*<proof>*

### Theorem 20.4

**lemma** *star-n-L*:

$$x^* * n(x) * L = x^* * \text{bot}$$

*<proof>*

**lemma** *star-L-split*:

**assumes**  $y \leq z$

**and**  $x * z * L \leq x * \text{bot} \sqcup z * L$

**shows**  $x^* * y * L \leq x^* * \text{bot} \sqcup z * L$

*<proof>*

**lemma** *star-L-split-same*:

$$x * y * L \leq x * \text{bot} \sqcup y * L \implies x^* * y * L = x^* * \text{bot} \sqcup y * L$$

*<proof>*

**lemma** *star-n-L-split-equal*:

$$n(x * y) \leq n(y) \implies x^* * n(y) * L = x^* * \text{bot} \sqcup n(y) * L$$

*<proof>*

**lemma** *n-star-mult*:

$$n(x * y) \leq n(y) \implies n(x^* * y) = n(x^*) \sqcup n(y)$$

*<proof>*

### Theorem 20.3

**lemma** *n-omega-mult*:

$$n(x^\omega * y) = n(x^\omega)$$

*<proof>*

**lemma** *n-star-left-unfold*:

$$n(x^*) = n(x * x^*)$$

*<proof>*

**lemma** *ni-star-below-ni-omega*:

$$ni(x^*) \leq ni(x^\omega)$$

*<proof>*

**lemma** *ni-below-ni-omega*:

$$ni(x) \leq ni(x^\omega)$$

*<proof>*

**lemma** *ni-star*:

$$ni(x)^* = 1 \sqcup ni(x)$$

*<proof>*

**lemma** *ni-omega*:

$$ni(x)^\omega = ni(x)$$

*<proof>*

**lemma** *ni-omega-induct*:

$$ni(y) \leq ni(x * y \sqcup z) \implies ni(y) \leq ni(x^\omega \sqcup x^* * z)$$

*<proof>*

**lemma** *star-ni*:

$$x^* * ni(x) = x^* * bot$$

*<proof>*

**lemma** *star-ni-split-equal*:

$$ni(x * y) \leq ni(y) \implies x^* * ni(y) = x^* * bot \sqcup ni(y)$$

*<proof>*

**lemma** *ni-star-mult*:

$$ni(x * y) \leq ni(y) \implies ni(x^* * y) = ni(x^*) \sqcup ni(y)$$

*<proof>*

**lemma** *ni-omega-mult*:

$$ni(x^\omega * y) = ni(x^\omega)$$

*<proof>*

**lemma** *ni-star-left-unfold*:

$$ni(x^*) = ni(x * x^*)$$

*<proof>*

**lemma** *n-star-import*:

**assumes**  $n(y) * x \leq x * n(y)$   
**shows**  $n(y) * x^* = n(y) * (n(y) * x)^*$

*<proof>*

**lemma** *n-omega-export*:

$$n(y) * x \leq x * n(y) \implies n(y) * x^\omega = (n(y) * x)^\omega$$

*<proof>*

**lemma** *n-omega-import*:

$$n(y) * x \leq x * n(y) \implies n(y) * x^\omega = n(y) * (n(y) * x)^\omega$$

*<proof>*

### Theorem 20.5

**lemma** *star-n-omega-top*:

$$x^* * n(x^\omega) * top = x^* * bot \sqcup n(x^\omega) * top$$

*<proof>*

**proposition** *n-star-induct-sup*:  $n(z \sqcup x * y) \leq n(y) \implies n(x^* * z) \leq n(y)$

*<proof>*

**end**

**end**

## 13 Boolean N-Semirings

**theory** *N-Semirings-Boolean*

**imports** *N-Semirings*

**begin**

**class** *an* =

**fixes** *an* :: 'a  $\Rightarrow$  'a

**class** *an-semiring* = *bounded-idempotent-left-zero-semiring* + *L* + *n* + *an* + *uminus* +

**assumes** *an-complement*:  $an(x) \sqcup n(x) = 1$

**assumes** *an-dist-sup* :  $an(x \sqcup y) = an(x) * an(y)$

**assumes** *an-export* :  $an(an(x) * y) = n(x) \sqcup an(y)$

**assumes** *an-mult-zero* :  $an(x) = an(x * bot)$

**assumes** *an-L-split* :  $x * n(y) * L = x * bot \sqcup n(x * y) * L$

**assumes** *an-split* :  $an(x * L) * x \leq x * bot$

**assumes** *an-uminus* :  $-x = an(x * L)$

**begin**

[Theorem 21](#)

**lemma** *n-an-def*:

$n(x) = an(an(x) * L)$

*<proof>*

[Theorem 21](#)

**lemma** *an-complement-bot*:

$an(x) * n(x) = bot$

*<proof>*

[Theorem 21](#)

**lemma** *an-n-def*:

$an(x) = n(an(x) * L)$

*<proof>*

**lemma** *an-case-split-left*:

$an(z) * x \leq y \wedge n(z) * x \leq y \longleftrightarrow x \leq y$

*<proof>*

**lemma** *an-case-split-right*:

$x * an(z) \leq y \wedge x * n(z) \leq y \longleftrightarrow x \leq y$

*<proof>*

**lemma** *split-sub*:

$x * y \leq z \sqcup x * top$

*<proof>*

[Theorem 21](#)



**subclass** *n-semiring*  
⟨*proof*⟩

**lemma** *n-complement-bot*:  
 $n(x) * an(x) = bot$   
⟨*proof*⟩

**lemma** *an-bot*:  
 $an(bot) = 1$   
⟨*proof*⟩

**lemma** *an-one*:  
 $an(1) = 1$   
⟨*proof*⟩

**lemma** *an-L*:  
 $an(L) = bot$   
⟨*proof*⟩

**lemma** *an-top*:  
 $an(top) = bot$   
⟨*proof*⟩

**lemma** *an-export-n*:  
 $an(n(x) * y) = an(x) \sqcup an(y)$   
⟨*proof*⟩

**lemma** *n-export-an*:  
 $n(an(x) * y) = an(x) * n(y)$   
⟨*proof*⟩

**lemma** *n-an-mult-commutative*:  
 $n(x) * an(y) = an(y) * n(x)$   
⟨*proof*⟩

**lemma** *an-mult-commutative*:  
 $an(x) * an(y) = an(y) * an(x)$   
⟨*proof*⟩

**lemma** *an-mult-idempotent*:  
 $an(x) * an(x) = an(x)$   
⟨*proof*⟩

**lemma** *an-sub-one*:  
 $an(x) \leq 1$   
⟨*proof*⟩

### Theorem 21

**lemma** *an-antitone*:

$$x \leq y \implies an(y) \leq an(x)$$

*<proof>*

**lemma** *an-mult-left-upper-bound:*

$$an(x * y) \leq an(x)$$

*<proof>*

**lemma** *an-mult-right-zero:*

$$an(x) * bot = bot$$

*<proof>*

**lemma** *n-mult-an:*

$$n(x * an(y)) = n(x)$$

*<proof>*

**lemma** *an-mult-n:*

$$an(x * n(y)) = an(x)$$

*<proof>*

**lemma** *an-mult-an:*

$$an(x * an(y)) = an(x)$$

*<proof>*

**lemma** *an-mult-left-absorb-sup:*

$$an(x) * (an(x) \sqcup an(y)) = an(x)$$

*<proof>*

**lemma** *an-mult-right-absorb-sup:*

$$(an(x) \sqcup an(y)) * an(y) = an(y)$$

*<proof>*

**lemma** *an-sup-left-absorb-mult:*

$$an(x) \sqcup an(x) * an(y) = an(x)$$

*<proof>*

**lemma** *an-sup-right-absorb-mult:*

$$an(x) * an(y) \sqcup an(y) = an(y)$$

*<proof>*

**lemma** *an-sup-left-dist-mult:*

$$an(x) \sqcup an(y) * an(z) = (an(x) \sqcup an(y)) * (an(x) \sqcup an(z))$$

*<proof>*

**lemma** *an-sup-right-dist-mult:*

$$an(x) * an(y) \sqcup an(z) = (an(x) \sqcup an(z)) * (an(y) \sqcup an(z))$$

*<proof>*

**lemma** *an-n-order:*

$$an(x) \leq an(y) \iff n(y) \leq n(x)$$

*<proof>*

**lemma** *an-order:*

$$an(x) \leq an(y) \longleftrightarrow an(x) * an(y) = an(x)$$

*<proof>*

**lemma** *an-mult-left-lower-bound:*

$$an(x) * an(y) \leq an(x)$$

*<proof>*

**lemma** *an-mult-right-lower-bound:*

$$an(x) * an(y) \leq an(y)$$

*<proof>*

**lemma** *an-n-mult-left-lower-bound:*

$$an(x) * n(y) \leq an(x)$$

*<proof>*

**lemma** *an-n-mult-right-lower-bound:*

$$an(x) * n(y) \leq n(y)$$

*<proof>*

**lemma** *n-an-mult-left-lower-bound:*

$$n(x) * an(y) \leq n(x)$$

*<proof>*

**lemma** *n-an-mult-right-lower-bound:*

$$n(x) * an(y) \leq an(y)$$

*<proof>*

**lemma** *an-mult-least-upper-bound:*

$$an(x) \leq an(y) \wedge an(x) \leq an(z) \longleftrightarrow an(x) \leq an(y) * an(z)$$

*<proof>*

**lemma** *an-mult-left-divisibility:*

$$an(x) \leq an(y) \longleftrightarrow (\exists z . an(x) = an(y) * an(z))$$

*<proof>*

**lemma** *an-mult-right-divisibility:*

$$an(x) \leq an(y) \longleftrightarrow (\exists z . an(x) = an(z) * an(y))$$

*<proof>*

**lemma** *an-split-top:*

$$an(x * L) * x * top \leq x * bot$$

*<proof>*

**lemma** *an-n-L:*

$$an(n(x) * L) = an(x)$$

*<proof>*

**lemma** *an-galois*:

$$an(y) \leq an(x) \longleftrightarrow n(x) * L \leq y$$

*<proof>*

**lemma** *an-mult*:

$$an(x * n(y) * L) = an(x * y)$$

*<proof>*

**lemma** *n-mult-top*:

$$an(x * n(y) * top) = an(x * y)$$

*<proof>*

**lemma** *an-n-equal*:

$$an(x) = an(y) \longleftrightarrow n(x) = n(y)$$

*<proof>*

**lemma** *an-top-L*:

$$an(x * top) = an(x * L)$$

*<proof>*

**lemma** *an-case-split-left-equal*:

$$an(z) * x = an(z) * y \implies n(z) * x = n(z) * y \implies x = y$$

*<proof>*

**lemma** *an-case-split-right-equal*:

$$x * an(z) = y * an(z) \implies x * n(z) = y * n(z) \implies x = y$$

*<proof>*

**lemma** *an-equal-complement*:

$$n(x) \sqcup an(y) = 1 \wedge n(x) * an(y) = bot \longleftrightarrow an(x) = an(y)$$

*<proof>*

**lemma** *n-equal-complement*:

$$n(x) \sqcup an(y) = 1 \wedge n(x) * an(y) = bot \longleftrightarrow n(x) = n(y)$$

*<proof>*

**lemma** *an-shunting*:

$$an(z) * x \leq y \longleftrightarrow x \leq y \sqcup n(z) * top$$

*<proof>*

**lemma** *an-shunting-an*:

$$an(z) * an(x) \leq an(y) \longleftrightarrow an(x) \leq n(z) \sqcup an(y)$$

*<proof>*

**lemma** *an-L-zero*:

$$an(x * L) * x = an(x * L) * x * bot$$

*<proof>*

**lemma** *n-plus-complement-intro-n*:  
 $n(x) \sqcup an(x) * n(y) = n(x) \sqcup n(y)$   
 ⟨proof⟩

**lemma** *n-plus-complement-intro-an*:  
 $n(x) \sqcup an(x) * an(y) = n(x) \sqcup an(y)$   
 ⟨proof⟩

**lemma** *an-plus-complement-intro-n*:  
 $an(x) \sqcup n(x) * n(y) = an(x) \sqcup n(y)$   
 ⟨proof⟩

**lemma** *an-plus-complement-intro-an*:  
 $an(x) \sqcup n(x) * an(y) = an(x) \sqcup an(y)$   
 ⟨proof⟩

**lemma** *n-mult-complement-intro-n*:  
 $n(x) * (an(x) \sqcup n(y)) = n(x) * n(y)$   
 ⟨proof⟩

**lemma** *n-mult-complement-intro-an*:  
 $n(x) * (an(x) \sqcup an(y)) = n(x) * an(y)$   
 ⟨proof⟩

**lemma** *an-mult-complement-intro-n*:  
 $an(x) * (n(x) \sqcup n(y)) = an(x) * n(y)$   
 ⟨proof⟩

**lemma** *an-mult-complement-intro-an*:  
 $an(x) * (n(x) \sqcup an(y)) = an(x) * an(y)$   
 ⟨proof⟩

**lemma** *an-preserves-equation*:  
 $an(y) * x \leq x * an(y) \longleftrightarrow an(y) * x = an(y) * x * an(y)$   
 ⟨proof⟩

**lemma** *wnf-lemma-1*:  
 $(n(p * L) * n(q * L) \sqcup an(p * L) * an(r * L)) * n(p * L) = n(p * L) * n(q * L)$   
 ⟨proof⟩

**lemma** *wnf-lemma-2*:  
 $(n(p * L) * n(q * L) \sqcup an(r * L) * an(q * L)) * n(q * L) = n(p * L) * n(q * L)$   
 ⟨proof⟩

**lemma** *wnf-lemma-3*:  
 $(n(p * L) * n(r * L) \sqcup an(p * L) * an(q * L)) * an(p * L) = an(p * L) * an(q * L)$   
 ⟨proof⟩

**lemma** *wnf-lemma-4*:

$$(n(r * L) * n(q * L) \sqcup an(p * L) * an(q * L)) * an(q * L) = an(p * L) * an(q * L)$$

*<proof>*

**lemma** *wnf-lemma-5*:

$$n(p \sqcup q) * (n(q) * x \sqcup an(q) * y) = n(q) * x \sqcup an(q) * n(p) * y$$

*<proof>*

**definition** *ani* :: 'a  $\Rightarrow$  'a

**where** *ani* x  $\equiv$  an(x) \* L

**lemma** *ani-bot*:

$$ani(bot) = L$$

*<proof>*

**lemma** *ani-one*:

$$ani(1) = L$$

*<proof>*

**lemma** *ani-L*:

$$ani(L) = bot$$

*<proof>*

**lemma** *ani-top*:

$$ani(top) = bot$$

*<proof>*

**lemma** *ani-complement*:

$$ani(x) \sqcup ni(x) = L$$

*<proof>*

**lemma** *ani-mult-zero*:

$$ani(x) = ani(x * bot)$$

*<proof>*

**lemma** *ani-antitone*:

$$y \leq x \implies ani(x) \leq ani(y)$$

*<proof>*

**lemma** *ani-mult-left-upper-bound*:

$$ani(x * y) \leq ani(x)$$

*<proof>*

**lemma** *ani-involutive*:

$$ani(ani(x)) = ni(x)$$

*<proof>*

**lemma** *ani-below-L*:

$ani(x) \leq L$   
 $\langle proof \rangle$

**lemma** *ani-left-zero*:  
 $ani(x) * y = ani(x)$   
 $\langle proof \rangle$

**lemma** *ani-top-L*:  
 $ani(x * top) = ani(x * L)$   
 $\langle proof \rangle$

**lemma** *ani-ni-order*:  
 $ani(x) \leq ani(y) \longleftrightarrow ni(y) \leq ni(x)$   
 $\langle proof \rangle$

**lemma** *ani-ni-equal*:  
 $ani(x) = ani(y) \longleftrightarrow ni(x) = ni(y)$   
 $\langle proof \rangle$

**lemma** *ni-ani*:  
 $ni(ani(x)) = ani(x)$   
 $\langle proof \rangle$

**lemma** *ani-ni*:  
 $ani(ni(x)) = ani(x)$   
 $\langle proof \rangle$

**lemma** *ani-mult*:  
 $ani(x * ni(y)) = ani(x * y)$   
 $\langle proof \rangle$

**lemma** *ani-an-order*:  
 $ani(x) \leq ani(y) \longleftrightarrow an(x) \leq an(y)$   
 $\langle proof \rangle$

**lemma** *ani-an-equal*:  
 $ani(x) = ani(y) \longleftrightarrow an(x) = an(y)$   
 $\langle proof \rangle$

**lemma** *n-mult-ani*:  
 $n(x) * ani(x) = bot$   
 $\langle proof \rangle$

**lemma** *an-mult-ni*:  
 $an(x) * ni(x) = bot$   
 $\langle proof \rangle$

**lemma** *n-mult-ni*:  
 $n(x) * ni(x) = ni(x)$

*<proof>*

**lemma** *an-mult-ani*:

$$an(x) * ani(x) = ani(x)$$

*<proof>*

**lemma** *ani-ni-meet*:

$$x \leq ani(y) \implies x \leq ni(y) \implies x = bot$$

*<proof>*

**lemma** *ani-galois*:

$$ani(x) \leq y \iff ni(x \sqcup y) = L$$

*<proof>*

**lemma** *an-ani*:

$$an(ani(x)) = n(x)$$

*<proof>*

**lemma** *n-ani*:

$$n(ani(x)) = an(x)$$

*<proof>*

**lemma** *an-ni*:

$$an(ni(x)) = an(x)$$

*<proof>*

**lemma** *ani-an*:

$$ani(an(x)) = L$$

*<proof>*

**lemma** *ani-n*:

$$ani(n(x)) = L$$

*<proof>*

**lemma** *ni-an*:

$$ni(an(x)) = bot$$

*<proof>*

**lemma** *ani-mult-n*:

$$ani(x * n(y)) = ani(x)$$

*<proof>*

**lemma** *ani-mult-an*:

$$ani(x * an(y)) = ani(x)$$

*<proof>*

**lemma** *ani-export-n*:

$$ani(n(x) * y) = ani(x) \sqcup ani(y)$$

*<proof>*



**lemma** *ani-export-an*:  
 $ani(an(x) * y) = ni(x) \sqcup ani(y)$   
 ⟨proof⟩

**lemma** *ni-export-an*:  
 $ni(an(x) * y) = an(x) * ni(y)$   
 ⟨proof⟩

**lemma** *ani-mult-top*:  
 $ani(x * n(y) * top) = ani(x * y)$   
 ⟨proof⟩

**lemma** *ani-an-bot*:  
 $ani(x) = bot \longleftrightarrow an(x) = bot$   
 ⟨proof⟩

**lemma** *ani-an-L*:  
 $ani(x) = L \longleftrightarrow an(x) = 1$   
 ⟨proof⟩

### Theorem 21

**subclass** *tests*  
 ⟨proof⟩

**end**

**class** *an-itering* = *n-itering* + *an-semiring* + *while* +  
**assumes** *while-circ-def*:  $p * y = (p * y)^\circ * -p$   
**begin**

**subclass** *test-itering*  
 ⟨proof⟩

**lemma** *an-circ-left-unfold*:  
 $an(x^\circ) = an(x * x^\circ)$   
 ⟨proof⟩

**lemma** *an-circ-x-n-circ*:  
 $an(x^\circ) * x * n(x^\circ) \leq x * bot$   
 ⟨proof⟩

**lemma** *an-circ-invariant*:  
 $an(x^\circ) * x \leq x * an(x^\circ)$   
 ⟨proof⟩

**lemma** *ani-circ*:  
 $ani(x)^\circ = 1 \sqcup ani(x)$   
 ⟨proof⟩

**lemma** *ani-circ-left-unfold*:

$$\text{ani}(x^\circ) = \text{ani}(x * x^\circ)$$

*<proof>*

**lemma** *an-circ-import*:

$$\text{an}(y) * x \leq x * \text{an}(y) \implies \text{an}(y) * x^\circ = \text{an}(y) * (\text{an}(y) * x)^\circ$$

*<proof>*

**lemma** *preserves-L*:

$$\text{preserves } L \ (-p)$$

*<proof>*

**end**

**class** *an-omega-algebra* = *n-omega-algebra-2* + *an-semiring* + *while* +  
 **assumes** *while-Omega-def*:  $p * y = (p * y)^\Omega * -p$

**begin**

**lemma** *an-split-omega-omega*:

$$\text{an}(x^\omega) * x^\omega \leq x^\omega * \text{bot}$$

*<proof>*

**lemma** *an-omega-below-an-star*:

$$\text{an}(x^\omega) \leq \text{an}(x^*)$$

*<proof>*

**lemma** *an-omega-below-an*:

$$\text{an}(x^\omega) \leq \text{an}(x)$$

*<proof>*

**lemma** *an-omega-induct*:

$$\text{an}(x * y \sqcup z) \leq \text{an}(y) \implies \text{an}(x^\omega \sqcup x^* * z) \leq \text{an}(y)$$

*<proof>*

**lemma** *an-star-mult*:

$$\text{an}(y) \leq \text{an}(x * y) \implies \text{an}(x^* * y) = \text{an}(x^*) * \text{an}(y)$$

*<proof>*

**lemma** *an-omega-mult*:

$$\text{an}(x^\omega * y) = \text{an}(x^\omega)$$

*<proof>*

**lemma** *an-star-left-unfold*:

$$\text{an}(x^*) = \text{an}(x * x^*)$$

*<proof>*

**lemma** *an-star-x-n-star*:

$$\text{an}(x^*) * x * \text{an}(x^*) \leq x * \text{bot}$$

*<proof>*

**lemma** *an-star-invariant:*

$$an(x^*) * x \leq x * an(x^*)$$

*<proof>*

**lemma** *n-an-star-unfold-invariant:*

$$n(an(x^*) * x^\omega) \leq an(x) * n(x * an(x^*) * x^\omega)$$

*<proof>*

**lemma** *ani-omega-below-ani-star:*

$$ani(x^\omega) \leq ani(x^*)$$

*<proof>*

**lemma** *ani-omega-below-ani:*

$$ani(x^\omega) \leq ani(x)$$

*<proof>*

**lemma** *ani-star:*

$$ani(x)^* = 1 \sqcup ani(x)$$

*<proof>*

**lemma** *ani-omega:*

$$ani(x)^\omega = ani(x) * L$$

*<proof>*

**lemma** *ani-omega-induct:*

$$ani(x * y \sqcup z) \leq ani(y) \implies ani(x^\omega \sqcup x^* * z) \leq ani(y)$$

*<proof>*

**lemma** *ani-omega-mult:*

$$ani(x^\omega * y) = ani(x^\omega)$$

*<proof>*

**lemma** *ani-star-left-unfold:*

$$ani(x^*) = ani(x * x^*)$$

*<proof>*

**lemma** *an-star-import:*

$$an(y) * x \leq x * an(y) \implies an(y) * x^* = an(y) * (an(y) * x)^*$$

*<proof>*

**lemma** *an-omega-export:*

$$an(y) * x \leq x * an(y) \implies an(y) * x^\omega = (an(y) * x)^\omega$$

*<proof>*

**lemma** *an-omega-import:*

$$an(y) * x \leq x * an(y) \implies an(y) * x^\omega = an(y) * (an(y) * x)^\omega$$

*<proof>*

**end**

**Theorem 22**

**sublocale** *an-omega-algebra* < *nL-omega: an-itering* **where** *circ = Omega*  
⟨*proof*⟩

**context** *an-omega-algebra*  
**begin**

**lemma** *preserves-star*:  
*nL-omega.preserves*  $x$   $(-p) \implies nL-omega.preserves$   $(x^*)$   $(-p)$   
⟨*proof*⟩

**end**

**end**

## 14 Modal N-Semirings

**theory** *N-Semirings-Modal*

**imports** *N-Semirings-Boolean*

**begin**

**class** *n-diamond-semiring* = *n-semiring* + *diamond* +  
**assumes** *ndiamond-def*:  $|x>y = n(x * y * L)$   
**begin**

**lemma** *diamond-x-bot*:  
 $|x>bot = n(x)$   
⟨*proof*⟩

**lemma** *diamond-x-1*:  
 $|x>1 = n(x * L)$   
⟨*proof*⟩

**lemma** *diamond-x-L*:  
 $|x>L = n(x * L)$   
⟨*proof*⟩

**lemma** *diamond-x-top*:  
 $|x>top = n(x * L)$   
⟨*proof*⟩

**lemma** *diamond-x-n*:  
 $|x>n(y) = n(x * y)$   
⟨*proof*⟩

**lemma** *diamond-bot-y*:

$$|bot\rangle y = bot$$

*<proof>*

**lemma** *diamond-1-y*:

$$|1\rangle y = n(y * L)$$

*<proof>*

**lemma** *diamond-1-n*:

$$|1\rangle n(y) = n(y)$$

*<proof>*

**lemma** *diamond-L-y*:

$$|L\rangle y = 1$$

*<proof>*

**lemma** *diamond-top-y*:

$$|top\rangle y = 1$$

*<proof>*

**lemma** *diamond-n-y*:

$$|n(x)\rangle y = n(x) * n(y * L)$$

*<proof>*

**lemma** *diamond-n-bot*:

$$|n(x)\rangle bot = bot$$

*<proof>*

**lemma** *diamond-n-1*:

$$|n(x)\rangle 1 = n(x)$$

*<proof>*

**lemma** *diamond-n-n*:

$$|n(x)\rangle n(y) = n(x) * n(y)$$

*<proof>*

**lemma** *diamond-n-n-same*:

$$|n(x)\rangle n(x) = n(x)$$

*<proof>*

### Theorem 23.1

**lemma** *diamond-left-dist-sup*:

$$|x \sqcup y\rangle z = |x\rangle z \sqcup |y\rangle z$$

*<proof>*

### Theorem 23.2

**lemma** *diamond-right-dist-sup*:

$$|x\rangle (y \sqcup z) = |x\rangle y \sqcup |x\rangle z$$

*<proof>*

### Theorem 23.3

**lemma** *diamond-associative:*

$$|x * y \rangle z = |x \rangle (y * z)$$

*<proof>*

### Theorem 23.3

**lemma** *diamond-left-mult:*

$$|x * y \rangle z = |x \rangle |y \rangle z$$

*<proof>*

**lemma** *diamond-right-mult:*

$$|x \rangle (y * z) = |x \rangle |y \rangle z$$

*<proof>*

**lemma** *diamond-n-export:*

$$|n(x) * y \rangle z = n(x) * |y \rangle z$$

*<proof>*

**lemma** *diamond-diamond-export:*

$$||x \rangle y \rangle z = |x \rangle y * |z \rangle 1$$

*<proof>*

**lemma** *diamond-left-isotone:*

$$x \leq y \implies |x \rangle z \leq |y \rangle z$$

*<proof>*

**lemma** *diamond-right-isotone:*

$$y \leq z \implies |x \rangle y \leq |x \rangle z$$

*<proof>*

**lemma** *diamond-isotone:*

$$w \leq y \implies x \leq z \implies |w \rangle x \leq |y \rangle z$$

*<proof>*

**definition** *ndiamond-L* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a ( $\langle || - \rangle \rightarrow [50,90] 95$ )

**where**  $||x \rangle y \equiv n(x * y) * L$

**lemma** *ndiamond-to-L:*

$$||x \rangle y = |x \rangle n(y) * L$$

*<proof>*

**lemma** *ndiamond-from-L:*

$$|x \rangle y = n(||x \rangle (y * L))$$

*<proof>*

**lemma** *diamond-L-ni:*

$$||x \rangle y = ni(x * y)$$

*<proof>*

**lemma** *diamond-L-associative:*

$$\|x * y\gg z = \|x\gg(y * z)$$

*<proof>*

**lemma** *diamond-L-left-mult:*

$$\|x * y\gg z = \|x\gg\|y\gg z$$

*<proof>*

**lemma** *diamond-L-right-mult:*

$$\|x\gg(y * z) = \|x\gg\|y\gg z$$

*<proof>*

**lemma** *diamond-L-left-dist-sup:*

$$\|x \sqcup y\gg z = \|x\gg z \sqcup \|y\gg z$$

*<proof>*

**lemma** *diamond-L-x-ni:*

$$\|x\gg ni(y) = ni(x * y)$$

*<proof>*

**lemma** *diamond-L-left-isotone:*

$$x \leq y \implies \|x\gg z \leq \|y\gg z$$

*<proof>*

**lemma** *diamond-L-right-isotone:*

$$y \leq z \implies \|x\gg y \leq \|x\gg z$$

*<proof>*

**lemma** *diamond-L-isotone:*

$$w \leq y \implies x \leq z \implies \|w\gg x \leq \|y\gg z$$

*<proof>*

**end**

**class** *n-box-semiring* = *n-diamond-semiring* + *an-semiring* + *box* +

**assumes** *nbox-def*:  $|x]y = an(x * an(y * L) * L)$

**begin**

[Theorem 23.8](#)

**lemma** *box-diamond:*

$$|x]y = an(|x>an(y * L) * L)$$

*<proof>*

[Theorem 23.4](#)

**lemma** *diamond-box:*

$$|x>y = an(|x]an(y * L) * L)$$

*<proof>*

**lemma** *box-x-bot*:  
 $|x]bot = an(x * L)$   
*<proof>*

**lemma** *box-x-1*:  
 $|x]1 = an(x)$   
*<proof>*

**lemma** *box-x-L*:  
 $|x]L = an(x)$   
*<proof>*

**lemma** *box-x-top*:  
 $|x]top = an(x)$   
*<proof>*

**lemma** *box-x-n*:  
 $|x]n(y) = an(x * an(y) * L)$   
*<proof>*

**lemma** *box-x-an*:  
 $|x]an(y) = an(x * y)$   
*<proof>*

**lemma** *box-bot-y*:  
 $|bot]y = 1$   
*<proof>*

**lemma** *box-1-y*:  
 $|1]y = n(y * L)$   
*<proof>*

**lemma** *box-1-n*:  
 $|1]n(y) = n(y)$   
*<proof>*

**lemma** *box-1-an*:  
 $|1]an(y) = an(y)$   
*<proof>*

**lemma** *box-L-y*:  
 $|L]y = bot$   
*<proof>*

**lemma** *box-top-y*:  
 $|top]y = bot$   
*<proof>*



**lemma** *box-n-y*:  
 $|n(x)]y = an(x) \sqcup n(y * L)$   
*<proof>*

**lemma** *box-an-y*:  
 $|an(x)]y = n(x) \sqcup n(y * L)$   
*<proof>*

**lemma** *box-n-bot*:  
 $|n(x)]bot = an(x)$   
*<proof>*

**lemma** *box-an-bot*:  
 $|an(x)]bot = n(x)$   
*<proof>*

**lemma** *box-n-1*:  
 $|n(x)]1 = 1$   
*<proof>*

**lemma** *box-an-1*:  
 $|an(x)]1 = 1$   
*<proof>*

**lemma** *box-n-n*:  
 $|n(x)]n(y) = an(x) \sqcup n(y)$   
*<proof>*

**lemma** *box-an-n*:  
 $|an(x)]n(y) = n(x) \sqcup n(y)$   
*<proof>*

**lemma** *box-n-an*:  
 $|n(x)]an(y) = an(x) \sqcup an(y)$   
*<proof>*

**lemma** *box-an-an*:  
 $|an(x)]an(y) = n(x) \sqcup an(y)$   
*<proof>*

**lemma** *box-n-n-same*:  
 $|n(x)]n(x) = 1$   
*<proof>*

**lemma** *box-an-an-same*:  
 $|an(x)]an(x) = 1$   
*<proof>*

[Theorem 23.5](#)

**lemma** *box-left-dist-sup*:

$$|x \sqcup y]z = |x]z * |y]z$$

*<proof>*

**lemma** *box-right-dist-sup*:

$$|x](y \sqcup z) = an(x * an(y * L) * an(z * L) * L)$$

*<proof>*

**lemma** *box-associative*:

$$|x * y]z = an(x * y * an(z * L) * L)$$

*<proof>*

### Theorem 23.7

**lemma** *box-left-mult*:

$$|x * y]z = |x]|y]z$$

*<proof>*

**lemma** *box-right-mult*:

$$|x](y * z) = an(x * an(y * z * L) * L)$$

*<proof>*

### Theorem 23.6

**lemma** *box-right-mult-n-n*:

$$|x](n(y) * n(z)) = |x]n(y) * |x]n(z)$$

*<proof>*

**lemma** *box-right-mult-an-n*:

$$|x](an(y) * n(z)) = |x]an(y) * |x]n(z)$$

*<proof>*

**lemma** *box-right-mult-n-an*:

$$|x](n(y) * an(z)) = |x]n(y) * |x]an(z)$$

*<proof>*

**lemma** *box-right-mult-an-an*:

$$|x](an(y) * an(z)) = |x]an(y) * |x]an(z)$$

*<proof>*

**lemma** *box-n-export*:

$$|n(x) * y]z = an(x) \sqcup |y]z$$

*<proof>*

**lemma** *box-an-export*:

$$|an(x) * y]z = n(x) \sqcup |y]z$$

*<proof>*

**lemma** *box-left-antitone*:

$$y \leq x \implies |x]z \leq |y]z$$

*<proof>*

**lemma** *box-right-isotone*:

$$y \leq z \implies |x]y \leq |x]z$$

*<proof>*

**lemma** *box-antitone-isotone*:

$$y \leq w \implies x \leq z \implies |w]x \leq |y]z$$

*<proof>*

**definition** *nbox-L* :: 'a  $\Rightarrow$  'a  $(\langle | - \rangle \rightarrow [50,90] 95)$

**where**  $|x]y \equiv an(x * an(y) * L) * L$

**lemma** *nbox-to-L*:

$$|x]y = |x]n(y) * L$$

*<proof>*

**lemma** *nbox-from-L*:

$$|x]y = n(|x](y * L))$$

*<proof>*

**lemma** *diamond-x-an*:

$$|x > an(y) = n(x * an(y) * L)$$

*<proof>*

**lemma** *diamond-1-an*:

$$|1 > an(y) = an(y)$$

*<proof>*

**lemma** *diamond-an-y*:

$$|an(x) > y = an(x) * n(y * L)$$

*<proof>*

**lemma** *diamond-an-bot*:

$$|an(x) > bot = bot$$

*<proof>*

**lemma** *diamond-an-1*:

$$|an(x) > 1 = an(x)$$

*<proof>*

**lemma** *diamond-an-n*:

$$|an(x) > n(y) = an(x) * n(y)$$

*<proof>*

**lemma** *diamond-n-an*:

$$|n(x) > an(y) = n(x) * an(y)$$

*<proof>*

**lemma** *diamond-an-an*:

$|an(x) \rangle an(y) = an(x) * an(y)$   
 $\langle proof \rangle$

**lemma** *diamond-an-an-same*:

$|an(x) \rangle an(x) = an(x)$   
 $\langle proof \rangle$

**lemma** *diamond-an-export*:

$|an(x) * y \rangle z = an(x) * |y \rangle z$   
 $\langle proof \rangle$

**lemma** *box-ani*:

$|x]y = an(x * ani(y * L))$   
 $\langle proof \rangle$

**lemma** *box-x-n-ani*:

$|x]n(y) = an(x * ani(y))$   
 $\langle proof \rangle$

**lemma** *box-L-ani*:

$\|x\|y = ani(x * ani(y))$   
 $\langle proof \rangle$

**lemma** *box-L-left-mult*:

$\|x * y\|z = \|x\|\|y\|z$   
 $\langle proof \rangle$

**lemma** *diamond-x-an-ani*:

$|x \rangle an(y) = n(x * ani(y))$   
 $\langle proof \rangle$

**lemma** *box-L-left-antitone*:

$y \leq x \implies \|x\|z \leq \|y\|z$   
 $\langle proof \rangle$

**lemma** *box-L-right-isotone*:

$y \leq z \implies \|x\|y \leq \|x\|z$   
 $\langle proof \rangle$

**lemma** *box-L-antitone-isotone*:

$y \leq w \implies x \leq z \implies \|w\|x \leq \|y\|z$   
 $\langle proof \rangle$

**end**

**class** *n-box-omega-algebra* = *n-box-semiring* + *an-omega-algebra*

**begin**

**lemma** *diamond-omega*:

$$|x^\omega \rangle y = |x^\omega \rangle z$$

*<proof>*

**lemma** *box-omega:*

$$|x^\omega \rangle y = |x^\omega \rangle z$$

*<proof>*

**lemma** *an-box-omega-induct:*

$$|x \rangle an(y) * n(z * L) \leq an(y) \implies |x^\omega \sqcup x^* \rangle z \leq an(y)$$

*<proof>*

**lemma** *n-box-omega-induct:*

$$|x \rangle n(y) * n(z * L) \leq n(y) \implies |x^\omega \sqcup x^* \rangle z \leq n(y)$$

*<proof>*

**lemma** *an-box-omega-induct-an:*

$$|x \rangle an(y) * an(z) \leq an(y) \implies |x^\omega \sqcup x^* \rangle an(z) \leq an(y)$$

*<proof>*

### Theorem 23.13

**lemma** *n-box-omega-induct-n:*

$$|x \rangle n(y) * n(z) \leq n(y) \implies |x^\omega \sqcup x^* \rangle n(z) \leq n(y)$$

*<proof>*

**lemma** *n-diamond-omega-induct:*

$$n(y) \leq |x \rangle n(y) \sqcup n(z * L) \implies n(y) \leq |x^\omega \sqcup x^* \rangle z$$

*<proof>*

**lemma** *an-diamond-omega-induct:*

$$an(y) \leq |x \rangle an(y) \sqcup n(z * L) \implies an(y) \leq |x^\omega \sqcup x^* \rangle z$$

*<proof>*

### Theorem 23.9

**lemma** *n-diamond-omega-induct-n:*

$$n(y) \leq |x \rangle n(y) \sqcup n(z) \implies n(y) \leq |x^\omega \sqcup x^* \rangle n(z)$$

*<proof>*

**lemma** *an-diamond-omega-induct-an:*

$$an(y) \leq |x \rangle an(y) \sqcup an(z) \implies an(y) \leq |x^\omega \sqcup x^* \rangle an(z)$$

*<proof>*

**lemma** *box-segerberg-an:*

$$|x^\omega \sqcup x^* \rangle an(y) = an(y) * |x^\omega \sqcup x^* \rangle (n(y) \sqcup |x \rangle an(y))$$

*<proof>*

### Theorem 23.16

**lemma** *box-segerberg-n:*

$$|x^\omega \sqcup x^* \rangle n(y) = n(y) * |x^\omega \sqcup x^* \rangle (an(y) \sqcup |x \rangle n(y))$$

*<proof>*

**lemma** *diamond-segerberg-an:*

$$|x^\omega \sqcup x^* \rangle an(y) = an(y) \sqcup |x^\omega \sqcup x^* \rangle (n(y) * |x \rangle an(y))$$

*\langle proof \rangle*

**Theorem 23.12**

**lemma** *diamond-segerberg-n:*

$$|x^\omega \sqcup x^* \rangle n(y) = n(y) \sqcup |x^\omega \sqcup x^* \rangle (an(y) * |x \rangle n(y))$$

*\langle proof \rangle*

**Theorem 23.11**

**lemma** *diamond-star-unfold-n:*

$$|x^* \rangle n(y) = n(y) \sqcup |an(y) * x \rangle |x^* \rangle n(y)$$

*\langle proof \rangle*

**lemma** *diamond-star-unfold-an:*

$$|x^* \rangle an(y) = an(y) \sqcup |n(y) * x \rangle |x^* \rangle an(y)$$

*\langle proof \rangle*

**Theorem 23.15**

**lemma** *box-star-unfold-n:*

$$|x^* \rangle n(y) = n(y) * |n(y) * x \rangle |x^* \rangle n(y)$$

*\langle proof \rangle*

**lemma** *box-star-unfold-an:*

$$|x^* \rangle an(y) = an(y) * |an(y) * x \rangle |x^* \rangle an(y)$$

*\langle proof \rangle*

**Theorem 23.10**

**lemma** *diamond-omega-unfold-n:*

$$|x^\omega \sqcup x^* \rangle n(y) = n(y) \sqcup |an(y) * x \rangle |x^\omega \sqcup x^* \rangle n(y)$$

*\langle proof \rangle*

**lemma** *diamond-omega-unfold-an:*

$$|x^\omega \sqcup x^* \rangle an(y) = an(y) \sqcup |n(y) * x \rangle |x^\omega \sqcup x^* \rangle an(y)$$

*\langle proof \rangle*

**Theorem 23.14**

**lemma** *box-omega-unfold-n:*

$$|x^\omega \sqcup x^* \rangle n(y) = n(y) * |n(y) * x \rangle |x^\omega \sqcup x^* \rangle n(y)$$

*\langle proof \rangle*

**lemma** *box-omega-unfold-an:*

$$|x^\omega \sqcup x^* \rangle an(y) = an(y) * |an(y) * x \rangle |x^\omega \sqcup x^* \rangle an(y)$$

*\langle proof \rangle*

**lemma** *box-cut-iteration-an:*

$$|x^\omega \sqcup x^* \rangle an(y) = |(an(y) * x)^\omega \sqcup (an(y) * x)^* \rangle an(y)$$

*\langle proof \rangle*

**lemma** *box-cut-iteration-n*:

$$|x^\omega \sqcup x^*]n(y) = |(n(y) * x)^\omega \sqcup (n(y) * x)^*]n(y)$$

*<proof>*

**lemma** *diamond-cut-iteration-an*:

$$|x^\omega \sqcup x^*>an(y) = |(n(y) * x)^\omega \sqcup (n(y) * x)^*>an(y)$$

*<proof>*

**lemma** *diamond-cut-iteration-n*:

$$|x^\omega \sqcup x^*>n(y) = |(an(y) * x)^\omega \sqcup (an(y) * x)^*>n(y)$$

*<proof>*

**lemma** *ni-diamond-omega-induct*:

$$ni(y) \leq \|x\gg ni(y) \sqcup ni(z) \implies ni(y) \leq \|x^\omega \sqcup x^*\gg z$$

*<proof>*

**lemma** *ani-diamond-omega-induct*:

$$ani(y) \leq \|x\gg ani(y) \sqcup ni(z) \implies ani(y) \leq \|x^\omega \sqcup x^*\gg z$$

*<proof>*

**lemma** *n-diamond-omega-L*:

$$|n(x^\omega) * L>y = |x^\omega>y$$

*<proof>*

**lemma** *n-diamond-loop*:

$$|x^\Omega>y = |x^\omega \sqcup x^*>y$$

*<proof>*

### Theorem 24.1

**lemma** *cut-iteration-loop*:

$$|x^\Omega>n(y) = |(an(y) * x)^\Omega>n(y)$$

*<proof>*

**lemma** *cut-iteration-while-loop*:

$$|x^\Omega>n(y) = |(an(y) * x)^\Omega * n(y)>n(y)$$

*<proof>*

### Theorem 24.1

**lemma** *cut-iteration-while-loop-2*:

$$|x^\Omega>n(y) = |an(y) * x>n(y)$$

*<proof>*

**lemma** *modal-while*:

**assumes**  $-q * -p * L \leq x * -p * L \wedge -p \leq -q \sqcup -r$   
**shows**  $-p \leq |n((-q * x)^\omega) * L \sqcup (-q * x)^* * --q>(-r)$

*<proof>*

**lemma** *modal-while-loop*:

$-q * -p * L \leq x * -p * L \wedge -p \leq -q \sqcup -r \implies -p \leq |(-q * x)^\Omega * -q \rangle (-r)$   
 <proof>

**Theorem 24.2**

**lemma** *modal-while-loop-2*:

$-q * -p * L \leq x * -p * L \wedge -p \leq -q \sqcup -r \implies -p \leq |-q * x \rangle (-r)$   
 <proof>

**lemma** *modal-while-2*:

**assumes**  $-p * L \leq x * -p * L$

**shows**  $-p \leq |n((-q * x)^\omega) * L \sqcup (-q * x)^* * -q \rangle (-q)$

<proof>

**end**

**class** *n-modal-omega-algebra* = *n-box-omega-algebra* +

**assumes** *n-star-induct*:  $n(x * y) \leq n(y) \longrightarrow n(x^* * y) \leq n(y)$

**begin**

**lemma** *n-star-induct-sup*:

$n(z \sqcup x * y) \leq n(y) \implies n(x^* * z) \leq n(y)$   
 <proof>

**lemma** *n-star-induct-star*:

$n(x * y) \leq n(y) \implies n(x^*) \leq n(y)$   
 <proof>

**lemma** *n-star-induct-iff*:

$n(x * y) \leq n(y) \iff n(x^* * y) \leq n(y)$   
 <proof>

**lemma** *n-star-bot*:

$n(x) = \text{bot} \iff n(x^*) = \text{bot}$   
 <proof>

**lemma** *n-diamond-star-induct*:

$|x \rangle n(y) \leq n(y) \implies |x^* \rangle n(y) \leq n(y)$   
 <proof>

**lemma** *n-diamond-star-induct-sup*:

$|x \rangle n(y) \sqcup n(z) \leq n(y) \implies |x^* \rangle n(z) \leq n(y)$   
 <proof>

**lemma** *n-diamond-star-induct-iff*:

$|x \rangle n(y) \leq n(y) \iff |x^* \rangle n(y) \leq n(y)$   
 <proof>

**lemma** *an-star-induct*:



$$an(y) \leq an(x * y) \implies an(y) \leq an(x^* * y)$$

*<proof>*

**lemma** *an-star-induct-sup:*

$$an(y) \leq an(z \sqcup x * y) \implies an(y) \leq an(x^* * z)$$

*<proof>*

**lemma** *an-star-induct-star:*

$$an(y) \leq an(x * y) \implies an(y) \leq an(x^*)$$

*<proof>*

**lemma** *an-star-induct-iff:*

$$an(y) \leq an(x * y) \longleftrightarrow an(y) \leq an(x^* * y)$$

*<proof>*

**lemma** *an-star-one:*

$$an(x) = 1 \longleftrightarrow an(x^*) = 1$$

*<proof>*

**lemma** *an-box-star-induct:*

$$an(y) \leq |x]an(y) \implies an(y) \leq |x^*]an(y)$$

*<proof>*

**lemma** *an-box-star-induct-sup:*

$$an(y) \leq |x]an(y) * an(z) \implies an(y) \leq |x^*]an(z)$$

*<proof>*

**lemma** *an-box-star-induct-iff:*

$$an(y) \leq |x]an(y) \longleftrightarrow an(y) \leq |x^*]an(y)$$

*<proof>*

**lemma** *box-star-segerberg-an:*

$$|x^*]an(y) = an(y) * |x^*](n(y) \sqcup |x]an(y))$$

*<proof>*

**lemma** *box-star-segerberg-n:*

$$|x^*]n(y) = n(y) * |x^*](an(y) \sqcup |x]n(y))$$

*<proof>*

**lemma** *diamond-segerberg-an:*

$$|x^*>an(y) = an(y) \sqcup |x^*>(n(y) * |x>an(y))$$

*<proof>*

**lemma** *diamond-star-segerberg-n:*

$$|x^*>n(y) = n(y) \sqcup |x^*>(an(y) * |x>n(y))$$

*<proof>*

**lemma** *box-cut-star-iteration-an:*

$$|x^*]an(y) = |(an(y) * x)^*]an(y)$$

*<proof>*

**lemma** *box-cut-star-iteration-n:*

$$|x^*]n(y) = |(n(y) * x)^*]n(y)$$

*<proof>*

**lemma** *diamond-cut-star-iteration-an:*

$$|x^*>an(y) = |(n(y) * x)^*>an(y)$$

*<proof>*

**lemma** *diamond-cut-star-iteration-n:*

$$|x^*>n(y) = |(an(y) * x)^*>n(y)$$

*<proof>*

**lemma** *ni-star-induct:*

$$ni(x * y) \leq ni(y) \implies ni(x^* * y) \leq ni(y)$$

*<proof>*

**lemma** *ni-star-induct-sup:*

$$ni(z \sqcup x * y) \leq ni(y) \implies ni(x^* * z) \leq ni(y)$$

*<proof>*

**lemma** *ni-star-induct-star:*

$$ni(x * y) \leq ni(y) \implies ni(x^*) \leq ni(y)$$

*<proof>*

**lemma** *ni-star-induct-iff:*

$$ni(x * y) \leq ni(y) \iff ni(x^* * y) \leq ni(y)$$

*<proof>*

**lemma** *ni-star-bot:*

$$ni(x) = bot \iff ni(x^*) = bot$$

*<proof>*

**lemma** *ni-diamond-star-induct:*

$$\|x\gg ni(y) \leq ni(y) \implies \|x^*\gg ni(y) \leq ni(y)$$

*<proof>*

**lemma** *ni-diamond-star-induct-sup:*

$$\|x\gg ni(y) \sqcup ni(z) \leq ni(y) \implies \|x^*\gg ni(z) \leq ni(y)$$

*<proof>*

**lemma** *ni-diamond-star-induct-iff:*

$$\|x\gg ni(y) \leq ni(y) \iff \|x^*\gg ni(y) \leq ni(y)$$

*<proof>*

**lemma** *ani-star-induct:*

$$ani(y) \leq ani(x * y) \implies ani(y) \leq ani(x^* * y)$$

*<proof>*

**lemma** *ani-star-induct-sup*:

$$ani(y) \leq ani(z \sqcup x * y) \implies ani(y) \leq ani(x^* * z)$$

*<proof>*

**lemma** *ani-star-induct-star*:

$$ani(y) \leq ani(x * y) \implies ani(y) \leq ani(x^*)$$

*<proof>*

**lemma** *ani-star-induct-iff*:

$$ani(y) \leq ani(x * y) \iff ani(y) \leq ani(x^* * y)$$

*<proof>*

**lemma** *ani-star-L*:

$$ani(x) = L \iff ani(x^*) = L$$

*<proof>*

**lemma** *ani-box-star-induct*:

$$ani(y) \leq \|x\|ani(y) \implies ani(y) \leq \|x^*\|ani(y)$$

*<proof>*

**lemma** *ani-box-star-induct-iff*:

$$ani(y) \leq \|x\|ani(y) \iff ani(y) \leq \|x^*\|ani(y)$$

*<proof>*

**lemma** *ani-box-star-induct-sup*:

$$ani(y) \leq \|x\|ani(y) \implies ani(y) \leq ani(z) \implies ani(y) \leq \|x^*\|ani(z)$$

*<proof>*

end

end

## 15 Approximation

**theory** *Approximation*

**imports** *Stone-Kleene-Relation-Algebras.Iterings*

**begin**

**nitpick-params** [*timeout = 600*]

**class** *apx* =

**fixes** *apx* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  bool (**infix**  $\langle \sqsubseteq \rangle$  50)

**class** *apx-order* = *apx* +

**assumes** *apx-reflexive*:  $x \sqsubseteq x$

**assumes** *apx-antisymmetric*:  $x \sqsubseteq y \wedge y \sqsubseteq x \longrightarrow x = y$

**assumes** *apx-transitive*:  $x \sqsubseteq y \wedge y \sqsubseteq z \longrightarrow x \sqsubseteq z$

**sublocale** *apx-order* < *apx: order* **where** *less-eq* = *apx* **and** *less* =  $\lambda x y . x \sqsubseteq y \wedge \neg y \sqsubseteq x$   
 <proof>

**context** *apx-order*  
**begin**

**abbreviation** *the-apx-least-fixpoint* ::  $('a \Rightarrow 'a) \Rightarrow 'a$  ( $\langle \kappa \rightarrow [201] 200$ ) **where**  
 $\kappa f \equiv \text{apx.the-least-fixpoint } f$

**abbreviation** *the-apx-least-prefixpoint* ::  $('a \Rightarrow 'a) \Rightarrow 'a$  ( $\langle p\kappa \rightarrow [201] 200$ )  
**where**  $p\kappa f \equiv \text{apx.the-least-prefixpoint } f$

**definition** *is-apx-meet* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool}$  **where** *is-apx-meet*  $x y z$   
 $\equiv z \sqsubseteq x \wedge z \sqsubseteq y \wedge (\forall w . w \sqsubseteq x \wedge w \sqsubseteq y \longrightarrow w \sqsubseteq z)$

**definition** *has-apx-meet* ::  $'a \Rightarrow 'a \Rightarrow \text{bool}$  **where** *has-apx-meet*  $x y \equiv$   
 $\exists z . \text{is-apx-meet } x y z$

**definition** *the-apx-meet* ::  $'a \Rightarrow 'a \Rightarrow 'a$  (**infixl**  $\langle \Delta \rangle$  66) **where**  $x \Delta y \equiv \text{THE } z$   
 $. \text{is-apx-meet } x y z$

**lemma** *apx-meet-unique*:  
*has-apx-meet*  $x y \Longrightarrow \exists! z . \text{is-apx-meet } x y z$   
 <proof>

**lemma** *apx-meet*:  
**assumes** *has-apx-meet*  $x y$   
**shows** *is-apx-meet*  $x y (x \Delta y)$   
 <proof>

**lemma** *apx-greatest-lower-bound*:  
*has-apx-meet*  $x y \Longrightarrow (w \sqsubseteq x \wedge w \sqsubseteq y \longleftrightarrow w \sqsubseteq x \Delta y)$   
 <proof>

**lemma** *apx-meet-same*:  
*is-apx-meet*  $x y z \Longrightarrow z = x \Delta y$   
 <proof>

**lemma** *apx-meet-char*:  
*is-apx-meet*  $x y z \longleftrightarrow \text{has-apx-meet } x y \wedge z = x \Delta y$   
 <proof>

**end**

**class** *apx-biorder* = *apx-order* + *order*  
**begin**

**lemma** *mu-below-kappa*:  
*has-least-fixpoint*  $f \Longrightarrow \text{apx.has-least-fixpoint } f \Longrightarrow \mu f \leq \kappa f$

*<proof>*

**lemma** *kappa-below-nu*:

*has-greatest-fixpoint f*  $\implies$  *apx.has-least-fixpoint f*  $\implies$   $\kappa f \leq \nu f$

*<proof>*

**lemma** *kappa-apx-below-mu*:

*has-least-fixpoint f*  $\implies$  *apx.has-least-fixpoint f*  $\implies$   $\kappa f \sqsubseteq \mu f$

*<proof>*

**lemma** *kappa-apx-below-nu*:

*has-greatest-fixpoint f*  $\implies$  *apx.has-least-fixpoint f*  $\implies$   $\kappa f \sqsubseteq \nu f$

*<proof>*

**end**

**class** *apx-semiring* = *apx-biorder* + *idempotent-left-semiring* + *L* +

**assumes** *apx-L-least*:  $L \sqsubseteq x$

**assumes** *sup-apx-left-isotone*:  $x \sqsubseteq y \longrightarrow x \sqcup z \sqsubseteq y \sqcup z$

**assumes** *mult-apx-left-isotone*:  $x \sqsubseteq y \longrightarrow x * z \sqsubseteq y * z$

**assumes** *mult-apx-right-isotone*:  $x \sqsubseteq y \longrightarrow z * x \sqsubseteq z * y$

**begin**

**lemma** *sup-apx-right-isotone*:

$x \sqsubseteq y \implies z \sqcup x \sqsubseteq z \sqcup y$

*<proof>*

**lemma** *sup-apx-isotone*:

$w \sqsubseteq y \implies x \sqsubseteq z \implies w \sqcup x \sqsubseteq y \sqcup z$

*<proof>*

**lemma** *mult-apx-isotone*:

$w \sqsubseteq y \implies x \sqsubseteq z \implies w * x \sqsubseteq y * z$

*<proof>*

**lemma** *affine-apx-isotone*:

*apx.isotone* ( $\lambda x . y * x \sqcup z$ )

*<proof>*

**end**

**end**

## 16 Strict Recursion

**theory** *Recursion-Strict*

**imports** *N-Semirings Approximation*

**begin**

**class** *semiring-apx* = *n-semiring* + *apx* +  
  **assumes** *apx-def*:  $x \sqsubseteq y \longleftrightarrow x \leq y \sqcup n(x) * L \wedge y \leq x \sqcup n(x) * top$   
**begin**

**lemma** *apx-n-order-reverse*:  
   $y \sqsubseteq x \implies n(x) \leq n(y)$   
   $\langle proof \rangle$

**lemma** *apx-n-order*:  
   $x \sqsubseteq y \implies y \sqsubseteq x \implies n(x) = n(y)$   
   $\langle proof \rangle$

**lemma** *apx-transitive*:  
  **assumes**  $x \sqsubseteq y$   
    **and**  $y \sqsubseteq z$   
  **shows**  $x \sqsubseteq z$   
   $\langle proof \rangle$

[Theorem 16.1](#)

**subclass** *apx-biorder*  
   $\langle proof \rangle$

**lemma** *sup-apx-left-isotone*:  
  **assumes**  $x \sqsubseteq y$   
  **shows**  $x \sqcup z \sqsubseteq y \sqcup z$   
   $\langle proof \rangle$

**lemma** *mult-apx-left-isotone*:  
  **assumes**  $x \sqsubseteq y$   
  **shows**  $x * z \sqsubseteq y * z$   
   $\langle proof \rangle$

**lemma** *mult-apx-right-isotone*:  
  **assumes**  $x \sqsubseteq y$   
  **shows**  $z * x \sqsubseteq z * y$   
   $\langle proof \rangle$

[Theorem 16.1](#) and [Theorem 16.2](#)

**subclass** *apx-semiring*  
   $\langle proof \rangle$

[Theorem 16.2](#)

**lemma** *ni-apx-isotone*:  
   $x \sqsubseteq y \implies ni(x) \sqsubseteq ni(y)$   
   $\langle proof \rangle$

[Theorem 17](#)

**definition**  $\text{kappa-apx-meet} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{kappa-apx-meet } f \equiv \text{apx.has-least-fixpoint } f \wedge \text{has-apx-meet } (\mu f) (\nu f)$   
 $\wedge \kappa f = \mu f \Delta \nu f$

**definition**  $\text{kappa-mu-nu} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{kappa-mu-nu } f \equiv \text{apx.has-least-fixpoint } f \wedge \kappa f = \mu f \sqcup n(\nu f) * L$

**definition**  $\text{nu-below-mu-nu} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{nu-below-mu-nu } f \equiv \nu f \leq \mu f \sqcup n(\nu f) * \text{top}$

**definition**  $\text{mu-nu-apx-nu} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{mu-nu-apx-nu } f \equiv \mu f \sqcup n(\nu f) * L \sqsubseteq \nu f$

**definition**  $\text{mu-nu-apx-meet} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{mu-nu-apx-meet } f \equiv \text{has-apx-meet } (\mu f) (\nu f) \wedge \mu f \Delta \nu f = \mu f \sqcup n(\nu f) * L$

**definition**  $\text{apx-meet-below-nu} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$   
**where**  $\text{apx-meet-below-nu } f \equiv \text{has-apx-meet } (\mu f) (\nu f) \wedge \mu f \Delta \nu f \leq \nu f$

**lemma**  $\text{mu-below-l}$ :  
 $\mu f \leq \mu f \sqcup n(\nu f) * L$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{l-below-nu}$ :  
 $\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \mu f \sqcup n(\nu f) * L \leq \nu f$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{n-l-nu}$ :  
 $\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow n(\mu f \sqcup n(\nu f) * L) = n(\nu f)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{l-apx-mu}$ :  
 $\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \mu f \sqcup n(\nu f) * L \sqsubseteq \mu f$   
 $\langle \text{proof} \rangle$

[Theorem 17.4 implies Theorem 17.5](#)

**lemma**  $\text{nu-below-mu-nu-mu-nu-apx-nu}$ :  
 $\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \text{nu-below-mu-nu } f \Longrightarrow$   
 $\text{mu-nu-apx-nu } f$   
 $\langle \text{proof} \rangle$

[Theorem 17.5 implies Theorem 17.6](#)

**lemma**  $\text{mu-nu-apx-nu-mu-nu-apx-meet}$ :  
**assumes**  $\text{has-least-fixpoint } f$   
**and**  $\text{has-greatest-fixpoint } f$   
**and**  $\text{mu-nu-apx-nu } f$   
**shows**  $\text{mu-nu-apx-meet } f$   
 $\langle \text{proof} \rangle$

Theorem 17.6 implies Theorem 17.7

**lemma** *mu-nu-apx-meet-apx-meet-below-nu*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *mu-nu-apx-meet f*  $\implies$   
*apx-meet-below-nu f*  
(proof)

Theorem 17.7 implies Theorem 17.4

**lemma** *apx-meet-below-nu-nu-below-mu-nu*:  
**assumes** *apx-meet-below-nu f*  
**shows** *nu-below-mu-nu f*  
(proof)

Theorem 17.1 implies Theorem 17.2

**lemma** *has-apx-least-fixpoint-kappa-apx-meet*:  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *apx.has-least-fixpoint f*  
**shows** *kappa-apx-meet f*  
(proof)

Theorem 17.2 implies Theorem 17.7

**lemma** *kappa-apx-meet-apx-meet-below-nu*:  
*has-greatest-fixpoint f*  $\implies$  *kappa-apx-meet f*  $\implies$  *apx-meet-below-nu f*  
(proof)

Theorem 17.7 implies Theorem 17.3

**lemma** *apx-meet-below-nu-kappa-mu-nu*:  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *isotone f*  
**and** *apx.isotone f*  
**and** *apx-meet-below-nu f*  
**shows** *kappa-mu-nu f*  
(proof)

Theorem 17.3 implies Theorem 17.1

**lemma** *kappa-mu-nu-has-apx-least-fixpoint*:  
*kappa-mu-nu f*  $\implies$  *apx.has-least-fixpoint f*  
(proof)

Theorem 17.4 implies Theorem 17.3

**lemma** *nu-below-mu-nu-kappa-mu-nu*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *isotone f*  $\implies$  *apx.isotone f*  
 $\implies$  *nu-below-mu-nu f*  $\implies$  *kappa-mu-nu f*  
(proof)

Theorem 17.3 implies Theorem 17.4

**lemma** *kappa-mu-nu-nu-below-mu-nu*:



*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu f*  $\implies$   
*nu-below-mu-nu f*  
 ⟨*proof*⟩

**definition** *kappa-mu-nu-ni* :: ('a  $\Rightarrow$  'a)  $\Rightarrow$  bool  
**where** *kappa-mu-nu-ni f*  $\equiv$  *apx.has-least-fixpoint f*  $\wedge$   $\kappa$  *f* =  $\mu$  *f*  $\sqcup$  *ni*( $\nu$  *f*)

**lemma** *kappa-mu-nu-ni-kappa-mu-nu*:  
*kappa-mu-nu-ni f*  $\longleftrightarrow$  *kappa-mu-nu f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-kappa-mu-nu-ni*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *isotone f*  $\implies$  *apx.isotone f*  
 $\implies$  *nu-below-mu-nu f*  $\implies$  *kappa-mu-nu-ni f*  
 ⟨*proof*⟩

**lemma** *kappa-mu-nu-ni-nu-below-mu-nu*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu-ni f*  $\implies$   
*nu-below-mu-nu f*  
 ⟨*proof*⟩

**end**

**class** *itering-apx* = *n-itering* + *semiring-apx*  
**begin**

[Theorem 16.3](#)

**lemma** *circ-apx-isotone*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $x^\circ \sqsubseteq y^\circ$   
 ⟨*proof*⟩

**end**

**class** *omega-algebra-apx* = *n-omega-algebra-2* + *semiring-apx*

**sublocale** *omega-algebra-apx* < *star*: *itering-apx* **where** *circ* = *star* ⟨*proof*⟩

**sublocale** *omega-algebra-apx* < *nL-omega*: *itering-apx* **where** *circ* = *Omega*  
 ⟨*proof*⟩

**context** *omega-algebra-apx*  
**begin**

[Theorem 16.4](#)

**lemma** *omega-apx-isotone*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $x^\omega \sqsubseteq y^\omega$   
 ⟨*proof*⟩

**end**

**class** *omega-algebra-apx-extra* = *omega-algebra-apx* +  
  **assumes** *n-split-omega*:  $x^\omega \leq x^* * \text{bot} \sqcup n(x^\omega) * \text{top}$   
**begin**

**lemma** *omega-n-star*:  
   $x^\omega \sqcup n(x^*) * \text{top} \leq x^* * n(x^\omega) * \text{top}$   
  ⟨*proof*⟩

**lemma** *n-omega-zero*:  
   $n(x^\omega) = \text{bot} \iff n(x^*) = \text{bot} \wedge x^\omega \leq x^* * \text{bot}$   
  ⟨*proof*⟩

**lemma** *n-split-nu-mu*:  
   $y^\omega \sqcup y^* * z \leq y^* * z \sqcup n(y^\omega \sqcup y^* * z) * \text{top}$   
  ⟨*proof*⟩

**lemma** *loop-exists*:  
   $\nu (\lambda x . y * x \sqcup z) \leq \mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * \text{top}$   
  ⟨*proof*⟩

**lemma** *loop-apx-least-fixpoint*:  
   $\text{apx.is-least-fixpoint} (\lambda x . y * x \sqcup z) (\mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)))$   
  \* *L*  
  ⟨*proof*⟩

**lemma** *loop-has-apx-least-fixpoint*:  
   $\text{apx.has-least-fixpoint} (\lambda x . y * x \sqcup z)$   
  ⟨*proof*⟩

**lemma** *loop-semantic*:  
   $\kappa (\lambda x . y * x \sqcup z) = \mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * L$   
  ⟨*proof*⟩

**lemma** *loop-apx-least-fixpoint-ni*:  
   $\text{apx.is-least-fixpoint} (\lambda x . y * x \sqcup z) (\mu (\lambda x . y * x \sqcup z) \sqcup \text{ni}(\nu (\lambda x . y * x \sqcup z)))$   
  ⟨*proof*⟩

**lemma** *loop-semantic-ni*:  
   $\kappa (\lambda x . y * x \sqcup z) = \mu (\lambda x . y * x \sqcup z) \sqcup \text{ni}(\nu (\lambda x . y * x \sqcup z))$   
  ⟨*proof*⟩

### Theorem 18

**lemma** *loop-semantic-kappa-mu-nu*:  
   $\kappa (\lambda x . y * x \sqcup z) = n(y^\omega) * L \sqcup y^* * z$   
  ⟨*proof*⟩

**end**

**class** *omega-algebra-apx-extra-2* = *omega-algebra-apx* +  
  **assumes** *omega-n-star*:  $x^\omega \leq x^* * n(x^\omega) * top$   
**begin**

**subclass** *omega-algebra-apx-extra*  
    ⟨*proof*⟩

**end**

**end**

## 17 N-Algebras

**theory** *N-Algebras*

**imports** *Stone-Kleene-Relation-Algebras.Iterings Base Lattice-Ordered-Semirings*

**begin**

**class** *C-left-n-algebra* = *bounded-idempotent-left-semiring* +  
*bounded-distrib-lattice* + *n* + *L*  
**begin**

**abbreviation** *C* :: 'a ⇒ 'a **where** *C* *x* ≡ *n(L) \* top* □ *x*

  AACP Theorem 3.38

**lemma** *C-isotone*:  
   $x \leq y \longrightarrow C\ x \leq C\ y$   
  ⟨*proof*⟩

  AACP Theorem 3.40

**lemma** *C-decreasing*:  
   $C\ x \leq x$   
  ⟨*proof*⟩

**end**

**class** *left-n-algebra* = *C-left-n-algebra* +  
  **assumes** *n-dist-n-add* :  $n(x) \sqcup n(y) = n(n(x) * top \sqcup y)$   
  **assumes** *n-export* :  $n(x) * n(y) = n(n(x) * y)$   
  **assumes** *n-left-upper-bound* :  $n(x) \leq n(x \sqcup y)$   
  **assumes** *n-nL-meet-L-nL0* :  $n(L) * x = (x \sqcap L) \sqcup n(L * bot) * x$   
  **assumes** *n-n-L-split-n-n-L-L* :  $x * n(y) * L = x * bot \sqcup n(x * n(y) * L) * L$   
  **assumes** *n-sub-nL* :  $n(x) \leq n(L)$   
  **assumes** *n-L-decreasing* :  $n(x) * L \leq x$   
  **assumes** *n-L-T-meet-mult-combined*:  $C\ (x * y) * z \leq C\ x * y * C\ z$   
  **assumes** *n-n-top-split-n-top* :  $x * n(y) * top \leq x * bot \sqcup n(x * y) * top$

**assumes** *n-top-meet-L-below-L* :  $x * top * y \sqcap L \leq x * L * y$   
**begin**

**subclass** *lattice-ordered-pre-left-semiring*  $\langle proof \rangle$

**lemma** *n-L-T-meet-mult-below*:

$C (x * y) \leq C x * y$   
 $\langle proof \rangle$

[AACP Theorem 3.41](#)

**lemma** *n-L-T-meet-mult-propagate*:

$C x * y \leq x * C y$   
 $\langle proof \rangle$

[AACP Theorem 3.43](#)

**lemma** *C-n-mult-closed*:

$C (n(x) * y) = n(x) * y$   
 $\langle proof \rangle$

[AACP Theorem 3.40](#)

**lemma** *meet-L-below-C*:

$x \sqcap L \leq C x$   
 $\langle proof \rangle$

[AACP Theorem 3.42](#)

**lemma** *n-L-T-meet-mult*:

$C (x * y) = C x * y$   
 $\langle proof \rangle$

[AACP Theorem 3.42](#)

**lemma** *C-mult-propagate*:

$C x * y = C x * C y$   
 $\langle proof \rangle$

[AACP Theorem 3.32](#)

**lemma** *meet-L-below-n-L*:

$x \sqcap L \leq n(L) * x$   
 $\langle proof \rangle$

[AACP Theorem 3.27](#)

**lemma** *n-vector-meet-L*:

$x * top \sqcap L \leq x * L$   
 $\langle proof \rangle$

**lemma** *n-right-upper-bound*:

$n(x) \leq n(y \sqcup x)$   
 $\langle proof \rangle$

[AACP Theorem 3.1](#)

**lemma** *n-isotone*:

$$x \leq y \implies n(x) \leq n(y)$$

*<proof>*

**lemma** *n-add-left-zero*:

$$n(\text{bot}) \sqcup n(x) = n(x)$$

*<proof>*

AACP Theorem 3.13

**lemma** *n-mult-right-zero-L*:

$$n(x) * \text{bot} \leq L$$

*<proof>*

**lemma** *n-add-left-top*:

$$n(\text{top}) \sqcup n(x) = n(\text{top})$$

*<proof>*

AACP Theorem 3.18

**lemma** *n-n-L*:

$$n(n(x) * L) = n(x)$$

*<proof>*

**lemma** *n-mult-transitive*:

$$n(x) * n(x) \leq n(x)$$

*<proof>*

**lemma** *n-mult-left-absorb-add-sub*:

$$n(x) * (n(x) \sqcup n(y)) \leq n(x)$$

*<proof>*

AACP Theorem 3.21

**lemma** *n-mult-left-lower-bound*:

$$n(x) * n(y) \leq n(x)$$

*<proof>*

AACP Theorem 3.20

**lemma** *n-mult-left-zero*:

$$n(\text{bot}) * n(x) = n(\text{bot})$$

*<proof>*

**lemma** *n-mult-right-one*:

$$n(x) * n(\text{top}) = n(x)$$

*<proof>*

**lemma** *n-L-increasing*:

$$n(x) \leq n(n(x) * L)$$

*<proof>*

AACP Theorem 3.2

**lemma** *n-galois*:

$$n(x) \leq n(y) \longleftrightarrow n(x) * L \leq y$$

*<proof>*

**lemma** *n-add-n-top*:

$$n(x \sqcup n(x) * top) = n(x)$$

*<proof>*

AACP Theorem 3.6

**lemma** *n-L-below-nL-top*:

$$L \leq n(L) * top$$

*<proof>*

AACP Theorem 3.4

**lemma** *n-less-eq-char-n*:

$$x \leq y \longleftrightarrow x \leq y \sqcup L \wedge C x \leq y \sqcup n(y) * top$$

*<proof>*

AACP Theorem 3.31

**lemma** *n-L-decreasing-meet-L*:

$$n(x) * L \leq x \sqcap L$$

*<proof>*

AACP Theorem 3.5

**lemma** *n-zero-L-zero*:

$$n(bot) * L = bot$$

*<proof>*

**lemma** *n-L-top-below-L*:

$$L * top \leq L$$

*<proof>*

AACP Theorem 3.9

**lemma** *n-L-top-L*:

$$L * top = L$$

*<proof>*

AACP Theorem 3.10

**lemma** *n-L-below-L*:

$$L * x \leq L$$

*<proof>*

AACP Theorem 3.7

**lemma** *n-nL-nT*:

$$n(L) = n(top)$$

*<proof>*

AACP Theorem 3.8

**lemma** *n-L-L*:

$$n(L) * L = L$$

*<proof>*

**lemma** *n-top-L:*

$$n(top) * L = L$$

*<proof>*

AACP Theorem 3.23

**lemma** *n-n-L-split-n-L:*

$$x * n(y) * L \leq x * bot \sqcup n(x * y) * L$$

*<proof>*

AACP Theorem 3.12

**lemma** *n-L-split-n-L-L:*

$$x * L = x * bot \sqcup n(x * L) * L$$

*<proof>*

AACP Theorem 3.11

**lemma** *n-L-split-L:*

$$x * L \leq x * bot \sqcup L$$

*<proof>*

AACP Theorem 3.24

**lemma** *n-split-top:*

$$x * n(y) * top \leq x * y \sqcup n(x * y) * top$$

*<proof>*

AACP Theorem 3.9

**lemma** *n-L-L-L:*

$$L * L = L$$

*<proof>*

AACP Theorem 3.9

**lemma** *n-L-top-L-L:*

$$L * top * L = L$$

*<proof>*

AACP Theorem 3.19

**lemma** *n-n-nL:*

$$n(x) = n(x) * n(L)$$

*<proof>*

**lemma** *n-L-mult-idempotent:*

$$n(L) * n(L) = n(L)$$

*<proof>*

AACP Theorem 3.22

**lemma** *n-n-L-n:*

$$n(x * n(y) * L) \leq n(x * y)$$

*<proof>*

**AACP Theorem 3.3**

**lemma** *n-less-eq-char:*

$$x \leq y \iff x \leq y \sqcup L \wedge x \leq y \sqcup n(y) * top$$

*<proof>*

**AACP Theorem 3.28**

**lemma** *n-top-meet-L-split-L:*

$$x * top * y \sqcap L \leq x * bot \sqcup L * y$$

*<proof>*

**AACP Theorem 3.29**

**lemma** *n-top-meet-L-L-meet-L:*

$$x * top * y \sqcap L = x * L * y \sqcap L$$

*<proof>*

**lemma** *n-n-top-below-n-L:*

$$n(x * top) \leq n(x * L)$$

*<proof>*

**AACP Theorem 3.14**

**lemma** *n-n-top-n-L:*

$$n(x * top) = n(x * L)$$

*<proof>*

**AACP Theorem 3.30**

**lemma** *n-meet-L-0-below-0-meet-L:*

$$(x \sqcap L) * bot \leq x * bot \sqcap L$$

*<proof>*

**AACP Theorem 3.15**

**lemma** *n-n-L-below-L:*

$$n(x) * L \leq x * L$$

*<proof>*

**lemma** *n-n-L-below-n-L-L:*

$$n(x) * L \leq n(x * L) * L$$

*<proof>*

**AACP Theorem 3.16**

**lemma** *n-below-n-L:*

$$n(x) \leq n(x * L)$$

*<proof>*

**AACP Theorem 3.17**

**lemma** *n-below-n-L-mult:*

$$n(x) \leq n(L) * n(x)$$

*<proof>*



### AACP Theorem 3.33

**lemma** *n-meet-L-below*:

$$n(x) \sqcap L \leq x$$

*<proof>*

### AACP Theorem 3.35

**lemma** *n-meet-L-top-below-n-L*:

$$(n(x) \sqcap L) * top \leq n(x) * L$$

*<proof>*

### AACP Theorem 3.34

**lemma** *n-meet-L-top-below*:

$$(n(x) \sqcap L) * top \leq x$$

*<proof>*

### AACP Theorem 3.36

**lemma** *n-n-meet-L*:

$$n(x) = n(x \sqcap L)$$

*<proof>*

**lemma** *n-T-below-n-meet*:

$$n(x) * top = n(C x) * top$$

*<proof>*

### AACP Theorem 3.44

**lemma** *n-C*:

$$n(C x) = n(x)$$

*<proof>*

### AACP Theorem 3.37

**lemma** *n-T-meet-L*:

$$n(x) * top \sqcap L = n(x) * L$$

*<proof>*

### AACP Theorem 3.39

**lemma** *n-L-top-meet-L*:

$$C L = L$$

*<proof>*

**end**

**class** *n-algebra* = *left-n-algebra* + *idempotent-left-zero-semiring*  
**begin**

**proposition** *n-dist-n-add* :  $n(x) \sqcup n(y) = n(n(x) * top \sqcup y)$  *<proof>*

**proposition** *n-export* :  $n(x) * n(y) = n(n(x) * y)$  *<proof>*

**proposition** *n-left-upper-bound* :  $n(x) \leq n(x \sqcup y)$  *<proof>*

**proposition** *n-nL-meet-L-nL0* :  $n(L) * x = (x \sqcap L) \sqcup n(L * bot) * x$   
 $\langle proof \rangle$

**proposition** *n-n-L-split-n-n-L-L* :  $x * n(y) * L = x * bot \sqcup n(x * n(y) * L) * L$   
 $\langle proof \rangle$

**proposition** *n-sub-nL* :  $n(x) \leq n(L)$   $\langle proof \rangle$

**proposition** *n-L-decreasing* :  $n(x) * L \leq x$   $\langle proof \rangle$

**proposition** *n-L-T-meet-mult-combined*:  $C (x * y) * z \leq C x * y * C z$   $\langle proof \rangle$

**proposition** *n-n-top-split-n-top* :  $x * n(y) * top \leq x * bot \sqcup n(x * y) * top$   
 $\langle proof \rangle$

**proposition** *n-top-meet-L-below-L* :  $x * top * y \sqcap L \leq x * L * y$   $\langle proof \rangle$

### AACP Theorem 3.25

**lemma** *n-top-split-0*:

$n(x) * top * y \leq x * y \sqcup n(x * bot) * top$   
 $\langle proof \rangle$

### AACP Theorem 3.26

**lemma** *n-top-split*:

$n(x) * top * y \leq x * y \sqcup n(x * y) * top$   
 $\langle proof \rangle$

**proposition** *n-zero*:  $n(bot) = bot$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-one*:  $n(1) = bot$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-nL-one*:  $n(L) = 1$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-nT-one*:  $n(top) = 1$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-n-zero*:  $n(x) = n(x * bot)$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-dist-add*:  $n(x) \sqcup n(y) = n(x \sqcup y)$  **nitpick** [*expect=genuine,card=4*]  $\langle proof \rangle$

**proposition** *n-L-split*:  $x * n(y) * L = x * bot \sqcup n(x * y) * L$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition** *n-split*:  $x \leq x * bot \sqcup n(x * L) * top$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *n-mult-top-1*:  $n(x * y) \leq n(x * n(y) * top)$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition** *l91-1*:  $n(L) * x \leq n(x * top) * top$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition** *meet-domain-top*:  $x \sqcap n(y) * top = n(y) * x$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition** *meet-domain-2*:  $x \sqcap n(y) * top \leq n(L) * x$  **nitpick** [*expect=genuine,card=4*]  $\langle proof \rangle$

**proposition** *n-nL-top-n-top-meet-L-top-2*:  $n(L) * x * top \leq n(x * top \sqcap L) * top$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition** *n-nL-top-n-top-meet-L-top-1*:  $n(x * top \sqcap L) * top \leq n(L) * x * top$  **nitpick** [*expect=genuine,card=2*]  $\langle proof \rangle$

**proposition** *l9*:  $x * bot \sqcap L \leq n(x * L) * L$  **nitpick** [*expect=genuine,card=4*]  $\langle proof \rangle$

**proposition** *l18-2*:  $n(x * L) * L \leq n(x) * L$  **nitpick** [*expect=genuine,card=3*]  $\langle proof \rangle$

**proposition l51-1:**  $n(x) * L \leq (x \sqcap L) * bot$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition l51-2:**  $(x \sqcap L) * bot \leq n(x) * L$  **nitpick** [*expect=genuine,card=4*]  
 ⟨*proof*⟩

**proposition n-split-equal:**  $x \sqcup n(x * L) * top = x * bot \sqcup n(x * L) * top$   
**nitpick** [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition n-split-top:**  $x * top \leq x * bot \sqcup n(x * L) * top$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition n-mult:**  $n(x * n(y) * L) = n(x * y)$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition n-mult-1:**  $n(x * y) \leq n(x * n(y) * L)$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition n-mult-top:**  $n(x * n(y) * top) = n(x * y)$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition n-mult-right-upper-bound:**  $n(x * y) \leq n(z) \longleftrightarrow n(x) \leq n(z) \wedge x * n(y) * L \leq x * bot \sqcup n(z) * L$  **nitpick** [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition meet-domain:**  $x \sqcap n(y) * z = n(y) * (x \sqcap z)$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition meet-domain-1:**  $x \sqcap n(y) * z \leq n(y) * x$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition meet-domain-top-3:**  $x \sqcap n(y) * top \leq n(y) * x$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition n-n-top-n-top-split-n-n-top-top:**  $n(x) * top \sqcup x * n(y) * top = x * bot \sqcup n(x * n(y) * top) * top$  **nitpick** [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition n-n-top-n-top-split-n-n-top-top-1:**  $x * bot \sqcup n(x * n(y) * top) * top \leq n(x) * top \sqcup x * n(y) * top$  **nitpick** [*expect=genuine,card=5*] ⟨*proof*⟩

**proposition n-n-top-n-top-split-n-n-top-top-2:**  $n(x) * top \sqcup x * n(y) * top \leq x * bot \sqcup n(x * n(y) * top) * top$  **nitpick** [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition n-nL-top-n-top-meet-L-top:**  $n(L) * x * top = n(x * top \sqcap L) * top$  **nitpick** [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition l18:**  $n(x) * L = n(x * L) * L$  **nitpick** [*expect=genuine,card=3*]  
 ⟨*proof*⟩

**proposition l22:**  $x * bot \sqcap L = n(x) * L$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition l22-1:**  $x * bot \sqcap L = n(x * L) * L$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition l22-2:**  $x \sqcap L = n(x) * L$  **nitpick** [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition l22-3:**  $x \sqcap L = n(x * L) * L$  **nitpick** [*expect=genuine,card=3*]  
 ⟨*proof*⟩

**proposition l22-4:**  $x \sqcap L \leq n(x) * L$  **nitpick** [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition l22-5:**  $x * bot \sqcap L \leq n(x) * L$  **nitpick** [*expect=genuine,card=4*]  
 ⟨*proof*⟩

**proposition l23:**  $x * top \sqcap L = n(x) * L$  **nitpick** [*expect=genuine,card=3*]  
 ⟨*proof*⟩

**proposition l51:**  $n(x) * L = (x \sqcap L) * bot$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition l91:**  $x = x * top \longrightarrow n(L) * x \leq n(x) * top$  **nitpick**  
 [*expect=genuine,card=3*] ⟨*proof*⟩

**proposition** *l92*:  $x = x * top \longrightarrow n(L) * x \leq n(x \sqcap L) * top$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition**  $x \sqcap L \leq n(x) * top$  **nitpick**  $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *n-meet-comp*:  $n(x) \sqcap n(y) \leq n(x) * n(y)$  **nitpick**  
 $[expect=genuine,card=3]$   $\langle proof \rangle$

**proposition** *n-n-meet-L-n-zero*:  $n(x) = (n(x) \sqcap L) \sqcup n(x * bot)$   $\langle proof \rangle$

**proposition** *n-below-n-zero*:  $n(x) \leq x \sqcup n(x * bot)$   $\langle proof \rangle$

**proposition** *n-n-top-split-n-L-n-zero-top*:  $n(x) * top = n(x) * L \sqcup n(x * bot) * top$   $\langle proof \rangle$

**proposition** *n-meet-L-0-0-meet-L*:  $(x \sqcap L) * bot = x * bot \sqcap L$   $\langle proof \rangle$

**end**

**end**

## 18 Recursion

**theory** *Recursion*

**imports** *Approximation N-Algebras*

**begin**

**class** *n-algebra-apx* = *n-algebra* + *apx* +  
**assumes** *apx-def*:  $x \sqsubseteq y \longleftrightarrow x \leq y \sqcup L \wedge C y \leq x \sqcup n(x) * top$   
**begin**

**lemma** *apx-transitive-2*:

**assumes**  $x \sqsubseteq y$   
**and**  $y \sqsubseteq z$   
**shows**  $x \sqsubseteq z$   
 $\langle proof \rangle$

**lemma** *apx-meet-L*:

**assumes**  $y \sqsubseteq x$   
**shows**  $x \sqcap L \leq y \sqcap L$   
 $\langle proof \rangle$

[AACP Theorem 4.1](#)

**subclass** *apx-biorder*

$\langle proof \rangle$

**lemma** *sup-apx-left-isotone-2*:

**assumes**  $x \sqsubseteq y$   
**shows**  $x \sqcup z \sqsubseteq y \sqcup z$   
 $\langle proof \rangle$

**lemma** *mult-apx-left-isotone-2*:

**assumes**  $x \sqsubseteq y$   
**shows**  $x * z \sqsubseteq y * z$   
 $\langle proof \rangle$

**lemma** *mult-apx-right-isotone-2*:

**assumes**  $x \sqsubseteq y$   
**shows**  $z * x \sqsubseteq z * y$   
 $\langle proof \rangle$

[AACP Theorem 4.1 and Theorem 4.2](#)

**subclass** *apx-semiring*

$\langle proof \rangle$

[AACP Theorem 4.2](#)

**lemma** *meet-L-apx-isotone*:

$x \sqsubseteq y \implies x \sqcap L \sqsubseteq y \sqcap L$   
 $\langle proof \rangle$

[AACP Theorem 4.2](#)

**lemma** *n-L-apx-isotone*:

**assumes**  $x \sqsubseteq y$   
**shows**  $n(x) * L \sqsubseteq n(y) * L$   
 $\langle proof \rangle$

**definition** *kappa-apx-meet* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $kappa\text{-apx-meet } f \equiv apx.has\text{-least-fixpoint } f \wedge has\text{-apx-meet } (\mu f) (\nu f)$   
 $\wedge \kappa f = \mu f \Delta \nu f$

**definition** *kappa-mu-nu* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $kappa\text{-mu-nu } f \equiv apx.has\text{-least-fixpoint } f \wedge \kappa f = \mu f \sqcup (\nu f \sqcap L)$

**definition** *nu-below-mu-nu* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $nu\text{-below-mu-nu } f \equiv C (\nu f) \leq \mu f \sqcup (\nu f \sqcap L) \sqcup n(\nu f) * top$

**definition** *nu-below-mu-nu-2* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $nu\text{-below-mu-nu-2 } f \equiv C (\nu f) \leq \mu f \sqcup (\nu f \sqcap L) \sqcup n(\mu f \sqcup (\nu f \sqcap L))$   
 $* top$

**definition** *mu-nu-apx-nu* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $mu\text{-nu-apx-nu } f \equiv \mu f \sqcup (\nu f \sqcap L) \sqsubseteq \nu f$

**definition** *mu-nu-apx-meet* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $mu\text{-nu-apx-meet } f \equiv has\text{-apx-meet } (\mu f) (\nu f) \wedge \mu f \Delta \nu f = \mu f \sqcup (\nu f \sqcap L)$

**definition** *apx-meet-below-nu* ::  $('a \Rightarrow 'a) \Rightarrow bool$

**where**  $apx\text{-meet-below-nu } f \equiv has\text{-apx-meet } (\mu f) (\nu f) \wedge \mu f \Delta \nu f \leq \nu f$

**lemma** *mu-below-l*:

$\mu f \leq \mu f \sqcup (\nu f \sqcap L)$   
(proof)

**lemma** *l-below-nu*:

*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies \mu f \sqcup (\nu f \sqcap L) \leq \nu f$   
(proof)

**lemma** *n-l-nu*:

*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies (\mu f \sqcup (\nu f \sqcap L)) \sqcap L = \nu f$   
 $\sqcap L$   
(proof)

**lemma** *l-apx-mu*:

$\mu f \sqcup (\nu f \sqcap L) \sqsubseteq \mu f$   
(proof)

AACP Theorem 4.8 implies Theorem 4.9

**lemma** *nu-below-mu-nu-nu-below-mu-nu-2*:

**assumes** *nu-below-mu-nu f*  
**shows** *nu-below-mu-nu-2 f*  
(proof)

AACP Theorem 4.9 implies Theorem 4.8

**lemma** *nu-below-mu-nu-2-nu-below-mu-nu*:

**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *nu-below-mu-nu-2 f*  
**shows** *nu-below-mu-nu f*  
(proof)

**lemma** *nu-below-mu-nu-equivalent*:

*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies (nu-below-mu-nu f \longleftrightarrow nu-below-mu-nu-2 f)$   
(proof)

AACP Theorem 4.9 implies Theorem 4.10

**lemma** *nu-below-mu-nu-2-mu-nu-apx-nu*:

**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *nu-below-mu-nu-2 f*  
**shows** *mu-nu-apx-nu f*  
(proof)

AACP Theorem 4.10 implies Theorem 4.11

**lemma** *mu-nu-apx-nu-mu-nu-apx-meet*:

**assumes** *mu-nu-apx-nu f*  
**shows** *mu-nu-apx-meet f*  
(proof)

AACP Theorem 4.11 implies Theorem 4.12

**lemma** *mu-nu-apx-meet-apx-meet-below-nu*:  
 $has\text{-}least\text{-}fixpoint\ f \implies has\text{-}greatest\text{-}fixpoint\ f \implies mu\text{-}nu\text{-}apx\text{-}meet\ f \implies$   
 $apx\text{-}meet\text{-}below\text{-}nu\ f$   
 ⟨proof⟩

AACP Theorem 4.12 implies Theorem 4.9

**lemma** *apx-meet-below-nu-nu-below-mu-nu-2*:  
**assumes**  $apx\text{-}meet\text{-}below\text{-}nu\ f$   
**shows**  $nu\text{-}below\text{-}mu\text{-}nu\text{-}2\ f$   
 ⟨proof⟩

AACP Theorem 4.5 implies Theorem 4.6

**lemma** *has-apx-least-fixpoint-kappa-apx-meet*:  
**assumes**  $has\text{-}least\text{-}fixpoint\ f$   
**and**  $has\text{-}greatest\text{-}fixpoint\ f$   
**and**  $apx.has\text{-}least\text{-}fixpoint\ f$   
**shows**  $kappa\text{-}apx\text{-}meet\ f$   
 ⟨proof⟩

AACP Theorem 4.6 implies Theorem 4.12

**lemma** *kappa-apx-meet-apx-meet-below-nu*:  
 $has\text{-}greatest\text{-}fixpoint\ f \implies kappa\text{-}apx\text{-}meet\ f \implies apx\text{-}meet\text{-}below\text{-}nu\ f$   
 ⟨proof⟩

AACP Theorem 4.12 implies Theorem 4.7

**lemma** *apx-meet-below-nu-kappa-mu-nu*:  
**assumes**  $has\text{-}least\text{-}fixpoint\ f$   
**and**  $has\text{-}greatest\text{-}fixpoint\ f$   
**and**  $isotone\ f$   
**and**  $apx.isotone\ f$   
**and**  $apx\text{-}meet\text{-}below\text{-}nu\ f$   
**shows**  $kappa\text{-}mu\text{-}nu\ f$   
 ⟨proof⟩

AACP Theorem 4.7 implies Theorem 4.5

**lemma** *kappa-mu-nu-has-apx-least-fixpoint*:  
 $kappa\text{-}mu\text{-}nu\ f \implies apx.has\text{-}least\text{-}fixpoint\ f$   
 ⟨proof⟩

AACP Theorem 4.8 implies Theorem 4.7

**lemma** *nu-below-mu-nu-kappa-mu-nu*:  
 $has\text{-}least\text{-}fixpoint\ f \implies has\text{-}greatest\text{-}fixpoint\ f \implies isotone\ f \implies apx.isotone\ f$   
 $\implies nu\text{-}below\text{-}mu\text{-}nu\ f \implies kappa\text{-}mu\text{-}nu\ f$   
 ⟨proof⟩

AACP Theorem 4.7 implies Theorem 4.8

**lemma** *kappa-mu-nu-nu-below-mu-nu*:  
 $has\text{-}least\text{-}fixpoint\ f \implies has\text{-}greatest\text{-}fixpoint\ f \implies kappa\text{-}mu\text{-}nu\ f \implies$   
 $nu\text{-}below\text{-}mu\text{-}nu\ f$

*<proof>*

**definition**  $\text{kappa-mu-nu-L} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$

**where**  $\text{kappa-mu-nu-L } f \equiv \text{apx.has-least-fixpoint } f \wedge \kappa f = \mu f \sqcup n(\nu f) * L$

**definition**  $\text{nu-below-mu-nu-L} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$

**where**  $\text{nu-below-mu-nu-L } f \equiv C(\nu f) \leq \mu f \sqcup n(\nu f) * \text{top}$

**definition**  $\text{mu-nu-apx-nu-L} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$

**where**  $\text{mu-nu-apx-nu-L } f \equiv \mu f \sqcup n(\nu f) * L \sqsubseteq \nu f$

**definition**  $\text{mu-nu-apx-meet-L} :: ('a \Rightarrow 'a) \Rightarrow \text{bool}$

**where**  $\text{mu-nu-apx-meet-L } f \equiv \text{has-apx-meet } (\mu f) (\nu f) \wedge \mu f \Delta \nu f = \mu f \sqcup n(\nu f) * L$

**lemma**  $n\text{-below-l}$ :

$x \sqcup n(y) * L \leq x \sqcup (y \sqcap L)$

*<proof>*

**lemma**  $n\text{-equal-l}$ :

**assumes**  $\text{nu-below-mu-nu-L } f$

**shows**  $\mu f \sqcup n(\nu f) * L = \mu f \sqcup (\nu f \sqcap L)$

*<proof>*

[AACP Theorem 4.14 implies Theorem 4.8](#)

**lemma**  $\text{nu-below-mu-nu-L-nu-below-mu-nu}$ :

$\text{nu-below-mu-nu-L } f \Longrightarrow \text{nu-below-mu-nu } f$

*<proof>*

[AACP Theorem 4.14 implies Theorem 4.13](#)

**lemma**  $\text{nu-below-mu-nu-L-kappa-mu-nu-L}$ :

$\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \text{isotone } f \Longrightarrow \text{apx.isotone } f$

$\Longrightarrow \text{nu-below-mu-nu-L } f \Longrightarrow \text{kappa-mu-nu-L } f$

*<proof>*

[AACP Theorem 4.14 implies Theorem 4.15](#)

**lemma**  $\text{nu-below-mu-nu-L-mu-nu-apx-nu-L}$ :

$\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \text{nu-below-mu-nu-L } f \Longrightarrow$

$\text{mu-nu-apx-nu-L } f$

*<proof>*

[AACP Theorem 4.14 implies Theorem 4.16](#)

**lemma**  $\text{nu-below-mu-nu-L-mu-nu-apx-meet-L}$ :

$\text{has-least-fixpoint } f \Longrightarrow \text{has-greatest-fixpoint } f \Longrightarrow \text{nu-below-mu-nu-L } f \Longrightarrow$

$\text{mu-nu-apx-meet-L } f$

*<proof>*

[AACP Theorem 4.15 implies Theorem 4.14](#)

**lemma**  $\text{mu-nu-apx-nu-L-nu-below-mu-nu-L}$ :



**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *mu-nu-apx-nu-L f*  
**shows** *nu-below-mu-nu-L f*  
 ⟨*proof*⟩

AACP Theorem 4.13 implies Theorem 4.15

**lemma** *kappa-mu-nu-L-mu-nu-apx-nu-L:*  
*has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu-L f*  $\implies$  *mu-nu-apx-nu-L f*  
 ⟨*proof*⟩

AACP Theorem 4.16 implies Theorem 4.15

**lemma** *mu-nu-apx-meet-L-mu-nu-apx-nu-L:*  
*mu-nu-apx-meet-L f*  $\implies$  *mu-nu-apx-nu-L f*  
 ⟨*proof*⟩

AACP Theorem 4.13 implies Theorem 4.14

**lemma** *kappa-mu-nu-L-nu-below-mu-nu-L:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu-L f*  $\implies$   
*nu-below-mu-nu-L f*  
 ⟨*proof*⟩

**proposition** *nu-below-mu-nu-nu-below-mu-nu-L: nu-below-mu-nu f*  $\longrightarrow$   
*nu-below-mu-nu-L f* **nitpick** [*expect=genuine,card=3*] ⟨*proof*⟩

**lemma** *unfold-fold-1:*  
*isotone f*  $\implies$  *has-least-prefixpoint f*  $\implies$  *apx.has-least-fixpoint f*  $\implies$   $f(x) \leq x$   
 $\implies \kappa f \leq x \sqcup L$   
 ⟨*proof*⟩

**lemma** *unfold-fold-2:*  
**assumes** *isotone f*  
**and** *apx.isotone f*  
**and** *has-least-prefixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *apx.has-least-fixpoint f*  
**and**  $f(x) \leq x$   
**and**  $\kappa f \sqcap L \leq x \sqcap L$   
**shows**  $\kappa f \leq x$   
 ⟨*proof*⟩

**end**

**class** *n-algebra-apx-2 = n-algebra + apx +*  
**assumes** *apx-def: x*  $\sqsubseteq$  *y*  $\longleftrightarrow$   $x \leq y \sqcup L \wedge y \leq x \sqcup n(x) * top$   
**begin**

**lemma** *apx-transitive-2:*  
**assumes**  $x \sqsubseteq y$

**and**  $y \sqsubseteq z$   
**shows**  $x \sqsubseteq z$   
 ⟨*proof*⟩

**lemma** *apx-meet-L*:  
**assumes**  $y \sqsubseteq x$   
**shows**  $x \sqcap L \leq y \sqcap L$   
 ⟨*proof*⟩

AACP Theorem 4.1

**subclass** *apx-biorder*  
 ⟨*proof*⟩

**lemma** *sup-apx-left-isotone-2*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $x \sqcup z \sqsubseteq y \sqcup z$   
 ⟨*proof*⟩

**lemma** *mult-apx-left-isotone-2*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $x * z \sqsubseteq y * z$   
 ⟨*proof*⟩

**lemma** *mult-apx-right-isotone-2*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $z * x \sqsubseteq z * y$   
 ⟨*proof*⟩

**end**

**end**

## 19 N-Omega-Algebras

**theory** *N-Omega-Algebras*

**imports** *Omega-Algebras Recursion*

**begin**

**class** *itering-apx* = *bounded-itering* + *n-algebra-apx*  
**begin**

**lemma** *circ-L*:  
 $L^\circ = L \sqcup 1$   
 ⟨*proof*⟩

**lemma** *C-circ-import*:  
 $C(x^\circ) \leq (C x)^\circ$

*<proof>*

[AACP Theorem 4.3 and Theorem 4.4](#)

**lemma** *circ-apx-isotone*:

**assumes**  $x \sqsubseteq y$

**shows**  $x^\circ \sqsubseteq y^\circ$

*<proof>*

**end**

**class** *n-omega-algebra-1* = *bounded-left-zero-omega-algebra* + *n-algebra-apx* + *Omega* +

**assumes** *Omega-def*:  $x^\Omega = n(x^\omega) * L \sqcup x^*$

**begin**

[AACP Theorem 8.13](#)

**lemma** *C-omega-export*:

$C(x^\omega) = (C x)^\omega$

*<proof>*

[AACP Theorem 8.2](#)

**lemma** *L-mult-star*:

$L * x^* = L$

*<proof>*

[AACP Theorem 8.3](#)

**lemma** *mult-L-star*:

$(x * L)^* = 1 \sqcup x * L$

*<proof>*

**lemma** *mult-L-omega-below*:

$(x * L)^\omega \leq x * L$

*<proof>*

[AACP Theorem 8.5](#)

**lemma** *mult-L-sup-star*:

$(x * L \sqcup y)^* = y^* \sqcup y^* * x * L$

*<proof>*

**lemma** *mult-L-sup-omega-below*:

$(x * L \sqcup y)^\omega \leq y^\omega \sqcup y^* * x * L$

*<proof>*

**lemma** *n-Omega-isotone*:

$x \leq y \implies x^\Omega \leq y^\Omega$

*<proof>*

**lemma** *n-star-below-Omega*:

$x^* \leq x^\Omega$

*<proof>*

**lemma** *mult-L-star-mult-below*:

$$(x * L)^* * y \leq y \sqcup x * L$$

*<proof>*

**end**

**sublocale** *n-omega-algebra-1 < star: iterating-apx* **where** *circ = star* *<proof>*

**class** *n-omega-algebra* = *n-omega-algebra-1* + *n-algebra-apx* +

**assumes** *n-split-omega-mult*:  $C(x^\omega) \leq x^* * n(x^\omega) * top$

**assumes** *tarski*:  $x * L \leq x * L * x * L$

**begin**

[AACP Theorem 8.4](#)

**lemma** *mult-L-omega*:

$$(x * L)^\omega = x * L$$

*<proof>*

[AACP Theorem 8.6](#)

**lemma** *mult-L-sup-omega*:

$$(x * L \sqcup y)^\omega = y^\omega \sqcup y^* * x * L$$

*<proof>*

[AACP Theorem 8.1](#)

**lemma** *tarski-mult-top-idempotent*:

$$x * L = x * L * x * L$$

*<proof>*

[AACP Theorem 8.7](#)

**lemma** *n-below-n-omega*:

$$n(x) \leq n(x^\omega)$$

*<proof>*

[AACP Theorem 8.14](#)

**lemma** *n-split-omega-sup-zero*:

$$C(x^\omega) \leq x^* * bot \sqcup n(x^\omega) * top$$

*<proof>*

**lemma** *n-split-omega-sup*:

$$C(x^\omega) \leq x^* \sqcup n(x^\omega) * top$$

*<proof>*

[AACP Theorem 8.12](#)

**lemma** *n-dist-omega-star*:

$$n(y^\omega \sqcup y^* * z) = n(y^\omega) \sqcup n(y^* * z)$$

*<proof>*

**lemma** *mult-L-sup-circ-below*:

$$(x * L \sqcup y)^\Omega \leq n(y^\omega) * L \sqcup y^* \sqcup y^* * x * L$$

*<proof>*

**lemma** *n-mult-omega-L-below-zero*:

$$n(y * x^\omega) * L \leq y * x^* * bot \sqcup y * n(x^\omega) * L$$

*<proof>*

AACP Theorem 8.10

**lemma** *n-mult-omega-L-star-zero*:

$$y * x^* * bot \sqcup n(y * x^\omega) * L = y * x^* * bot \sqcup y * n(x^\omega) * L$$

*<proof>*

AACP Theorem 8.11

**lemma** *n-mult-omega-L-star*:

$$y * x^* \sqcup n(y * x^\omega) * L = y * x^* \sqcup y * n(x^\omega) * L$$

*<proof>*

**lemma** *n-mult-omega-L-below*:

$$n(y * x^\omega) * L \leq y * x^* \sqcup y * n(x^\omega) * L$$

*<proof>*

**lemma** *n-omega-L-below-zero*:

$$n(x^\omega) * L \leq x * x^* * bot \sqcup x * n(x^\omega) * L$$

*<proof>*

**lemma** *n-omega-L-below*:

$$n(x^\omega) * L \leq x^* \sqcup x * n(x^\omega) * L$$

*<proof>*

**lemma** *n-omega-L-star-zero*:

$$x * x^* * bot \sqcup n(x^\omega) * L = x * x^* * bot \sqcup x * n(x^\omega) * L$$

*<proof>*

AACP Theorem 8.8

**lemma** *n-omega-L-star*:

$$x^* \sqcup n(x^\omega) * L = x^* \sqcup x * n(x^\omega) * L$$

*<proof>*

AACP Theorem 8.9

**lemma** *n-omega-L-star-zero-star*:

$$x^* * bot \sqcup n(x^\omega) * L = x^* * bot \sqcup x^* * n(x^\omega) * L$$

*<proof>*

AACP Theorem 8.8

**lemma** *n-omega-L-star-star*:

$$x^* \sqcup n(x^\omega) * L = x^* \sqcup x^* * n(x^\omega) * L$$

*<proof>*

**lemma** *n-Omega-left-unfold*:

$$1 \sqcup x * x^\Omega = x^\Omega$$

*<proof>*

**lemma** *n-Omega-left-slide*:

$$(x * y)^\Omega * x \leq x * (y * x)^\Omega$$

*<proof>*

**lemma** *n-Omega-sup-1*:

$$(x \sqcup y)^\Omega = x^\Omega * (y * x^\Omega)^\Omega$$

*<proof>*

**end**

**sublocale** *n-omega-algebra* < *nL-omega: left-zero-conway-semiring* **where** *circ = Omega*

*<proof>*

**context** *n-omega-algebra*

**begin**

[AACP Theorem 8.16](#)

**lemma** *omega-apx-isotone*:

**assumes**  $x \sqsubseteq y$

**shows**  $x^\omega \sqsubseteq y^\omega$

*<proof>*

**lemma** *combined-apx-left-isotone*:

$$x \sqsubseteq y \implies n(x^\omega) * L \sqcup x^* * z \sqsubseteq n(y^\omega) * L \sqcup y^* * z$$

*<proof>*

**lemma** *combined-apx-left-isotone-2*:

$$x \sqsubseteq y \implies (x^\omega \sqcap L) \sqcup x^* * z \sqsubseteq (y^\omega \sqcap L) \sqcup y^* * z$$

*<proof>*

**lemma** *combined-apx-right-isotone*:

$$y \sqsubseteq z \implies n(x^\omega) * L \sqcup x^* * y \sqsubseteq n(x^\omega) * L \sqcup x^* * z$$

*<proof>*

**lemma** *combined-apx-right-isotone-2*:

$$y \sqsubseteq z \implies (x^\omega \sqcap L) \sqcup x^* * y \sqsubseteq (x^\omega \sqcap L) \sqcup x^* * z$$

*<proof>*

**lemma** *combined-apx-isotone*:

$$x \sqsubseteq y \implies w \sqsubseteq z \implies n(x^\omega) * L \sqcup x^* * w \sqsubseteq n(y^\omega) * L \sqcup y^* * z$$

*<proof>*

**lemma** *combined-apx-isotone-2*:

$$x \sqsubseteq y \implies w \sqsubseteq z \implies (x^\omega \sqcap L) \sqcup x^* * w \sqsubseteq (y^\omega \sqcap L) \sqcup y^* * z$$

*<proof>*

**lemma** *n-split-nu-mu*:

$$C (y^\omega \sqcup y^* * z) \leq y^* * z \sqcup n(y^\omega \sqcup y^* * z) * top$$

*<proof>*

**lemma** *n-split-nu-mu-2*:

$$C (y^\omega \sqcup y^* * z) \leq y^* * z \sqcup ((y^\omega \sqcup y^* * z) \sqcap L) \sqcup n(y^\omega \sqcup y^* * z) * top$$

*<proof>*

**lemma** *loop-exists*:

$$C (\nu (\lambda x . y * x \sqcup z)) \leq \mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * top$$

*<proof>*

**lemma** *loop-exists-2*:

$$C (\nu (\lambda x . y * x \sqcup z)) \leq \mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * top$$

*<proof>*

**lemma** *loop-apx-least-fixpoint*:

$$apx.is-least-fixpoint (\lambda x . y * x \sqcup z) (\mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * L)$$

*<proof>*

**lemma** *loop-apx-least-fixpoint-2*:

$$apx.is-least-fixpoint (\lambda x . y * x \sqcup z) (\mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L))$$

*<proof>*

**lemma** *loop-has-apx-least-fixpoint*:

$$apx.has-least-fixpoint (\lambda x . y * x \sqcup z)$$

*<proof>*

**lemma** *loop-semantic*:

$$\kappa (\lambda x . y * x \sqcup z) = \mu (\lambda x . y * x \sqcup z) \sqcup n(\nu (\lambda x . y * x \sqcup z)) * L$$

*<proof>*

**lemma** *loop-semantic-2*:

$$\kappa (\lambda x . y * x \sqcup z) = \mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L)$$

*<proof>*

[AACP Theorem 8.15](#)

**lemma** *loop-semantic-kappa-mu-nu*:

$$\kappa (\lambda x . y * x \sqcup z) = n(y^\omega) * L \sqcup y^* * z$$

*<proof>*

[AACP Theorem 8.15](#)

**lemma** *loop-semantic-kappa-mu-nu-2*:  
 $\kappa (\lambda x . y * x \sqcup z) = (y^\omega \sqcap L) \sqcup y^* * z$   
 ⟨proof⟩

AACP Theorem 8.16

**lemma** *loop-semantic-apx-left-isotone*:  
 $w \sqsubseteq y \implies \kappa (\lambda x . w * x \sqcup z) \sqsubseteq \kappa (\lambda x . y * x \sqcup z)$   
 ⟨proof⟩

AACP Theorem 8.16

**lemma** *loop-semantic-apx-right-isotone*:  
 $w \sqsubseteq z \implies \kappa (\lambda x . y * x \sqcup w) \sqsubseteq \kappa (\lambda x . y * x \sqcup z)$   
 ⟨proof⟩

**lemma** *loop-semantic-apx-isotone*:  
 $v \sqsubseteq y \implies w \sqsubseteq z \implies \kappa (\lambda x . v * x \sqcup w) \sqsubseteq \kappa (\lambda x . y * x \sqcup z)$   
 ⟨proof⟩

end

end

## 20 N-Omega Binary Iterings

**theory** *N-Omega-Binary-Iterings*

**imports** *N-Omega-Algebras Binary-Iterings-Strict*

**begin**

**sublocale** *extended-binary-itering* < *left-zero-conway-semiring* **where** *circ* =  $(\lambda x . x * 1)$   
 ⟨proof⟩

**class** *binary-itering-apx* = *bounded-binary-itering* + *n-algebra-apx*  
**begin**

**lemma** *C-while-import*:  
 $C (x * z) = C (C x * z)$   
 ⟨proof⟩

**lemma** *C-while-preserve*:  
 $C (x * z) = C (x * C z)$   
 ⟨proof⟩

**lemma** *C-while-import-preserve*:  
 $C (x * z) = C (C x * C z)$   
 ⟨proof⟩



**lemma** *while-L-L*:

$$L \star L = L$$

*<proof>*

**lemma** *while-L-below-sup*:

$$L \star x \leq x \sqcup L$$

*<proof>*

**lemma** *while-L-split*:

$$x \star L \leq (x \star y) \sqcup L$$

*<proof>*

**lemma** *while-n-while-top-split*:

$$x \star (n(x \star y) \star \text{top}) \leq (x \star \text{bot}) \sqcup n(x \star y) \star \text{top}$$

*<proof>*

**lemma** *circ-apx-right-isotone*:

**assumes**  $x \sqsubseteq y$

**shows**  $z \star x \sqsubseteq z \star y$

*<proof>*

**end**

**class** *extended-binary-itering-apx* = *binary-itering-apx* +  
*bounded-extended-binary-itering* +

**assumes** *n-below-while-zero*:  $n(x) \leq n(x \star \text{bot})$

**begin**

**lemma** *circ-apx-right-isotone*:

**assumes**  $x \sqsubseteq y$

**shows**  $x \star z \sqsubseteq y \star z$

*<proof>*

**proposition** *while-top*:  $\text{top} \star x = L \sqcup \text{top} \star x$  *<proof>*

**proposition** *while-one-top*:  $1 \star x = L \sqcup x$  *<proof>*

**proposition** *while-unfold-below-1*:  $x = y \star x \implies x \leq y \star 1$  *<proof>*

**proposition** *while-square-1*:  $x \star 1 = (x \star x) \star (x \sqcup 1)$  *<proof>*

**proposition** *while-absorb-below-one*:  $y \star x \leq x \implies y \star x \leq 1 \star x$  *<proof>*

**proposition** *while-mult-L*:  $(x \star L) \star z = z \sqcup x \star L$  *<proof>*

**proposition** *tarski-top-omega-below-2*:  $x \star L \leq (x \star L) \star \text{bot}$  *<proof>*

**proposition** *tarski-top-omega-2*:  $x \star L = (x \star L) \star \text{bot}$  *<proof>*

**proposition** *while-separate-right-plus*:  $y \star x \leq x \star (x \star (1 \sqcup y)) \sqcup 1 \implies y \star (x \star z) \leq x \star (y \star z)$  *<proof>*

**proposition**  $y \star (x \star 1) \leq x \star (y \star 1) \implies (x \sqcup y) \star 1 = x \star (y \star 1)$  *<proof>*

**proposition**  $y \star x \leq (1 \sqcup x) \star (y \star 1) \implies (x \sqcup y) \star 1 = x \star (y \star 1)$  *<proof>*

**end**

**class** *n-omega-algebra-binary* = *n-omega-algebra* + *while* +  
**assumes** *while-def*:  $x \star y = n(x^\omega) * L \sqcup x^* * y$   
**begin**

**lemma** *while-omega-inf-L-star*:  
 $x \star y = (x^\omega \sqcap L) \sqcup x^* * y$   
*<proof>*

**lemma** *while-one-mult-while-below-1*:  
 $(y \star 1) * (y \star v) \leq y \star v$   
*<proof>*

**lemma** *star-below-while*:  
 $x^* * y \leq x \star y$   
*<proof>*

**subclass** *bounded-binary-itering*  
*<proof>*

**lemma** *while-top*:  
 $top \star x = L \sqcup top * x$   
*<proof>*

**lemma** *while-one-top*:  
 $1 \star x = L \sqcup x$   
*<proof>*

**lemma** *while-finite-associative*:  
 $x^\omega = bot \implies (x \star y) * z = x \star (y * z)$   
*<proof>*

**lemma** *while-while-one*:  
 $y \star (x \star 1) = n(y^\omega) * L \sqcup y^* * n(x^\omega) * L \sqcup y^* * x^*$   
*<proof>*

### AACP Theorem 8.17

**subclass** *bounded-extended-binary-itering*  
*<proof>*

**subclass** *extended-binary-itering-apx*  
*<proof>*

**lemma** *while-simulate-4-plus*:  
**assumes**  $y * x \leq x * (x \star (1 \sqcup y))$   
**shows**  $y * x * x^* \leq x * (x \star (1 \sqcup y))$   
*<proof>*

**lemma** *while-simulate-4-omega*:  
**assumes**  $y * x \leq x * (x \star (1 \sqcup y))$

**shows**  $y * x^\omega \leq x^\omega$   
*<proof>*

**lemma** *while-square-1*:  
 $x * 1 = (x * x) * (x \sqcup 1)$   
*<proof>*

**lemma** *while-absorb-below-one*:  
 $y * x \leq x \implies y * x \leq 1 * x$   
*<proof>*

**lemma** *while-mult-L*:  
 $(x * L) * z = z \sqcup x * L$   
*<proof>*

**lemma** *tarski-top-omega-below-2*:  
 $x * L \leq (x * L) * \text{bot}$   
*<proof>*

**lemma** *tarski-top-omega-2*:  
 $x * L = (x * L) * \text{bot}$   
*<proof>*

**proposition** *while-sub-mult-one*:  $x * (1 * y) \leq 1 * x$  **nitpick**  
*[expect=genuine,card=3] <proof>*

**proposition** *while-unfold-below*:  $x = z \sqcup y * x \longrightarrow x \leq y * z$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition** *while-loop-is-greatest-postfixpoint*: *is-greatest-postfixpoint*  $(\lambda x . y * x \sqcup z)$   $(y * z)$  **nitpick** *[expect=genuine,card=2] <proof>*

**proposition** *while-loop-is-greatest-fixpoint*: *is-greatest-fixpoint*  $(\lambda x . y * x \sqcup z)$   $(y * z)$  **nitpick** *[expect=genuine,card=2] <proof>*

**proposition** *while-denest-3*:  $(x * w) * x^\omega = (x * w)^\omega$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition** *while-mult-top*:  $(x * \text{top}) * z = z \sqcup x * \text{top}$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition** *tarski-below-top-omega*:  $x \leq (x * L)^\omega$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition** *tarski-mult-omega-omega*:  $(x * y^\omega)^\omega = x * y^\omega$  **nitpick**  
*[expect=genuine,card=3] <proof>*

**proposition** *tarski-below-top-omega-2*:  $x \leq (x * L) * \text{bot}$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition**  $1 = (x * \text{bot}) * 1$  **nitpick** *[expect=genuine,card=3] <proof>*

**proposition** *tarski*:  $x = \text{bot} \vee \text{top} * x * \text{top} = \text{top}$  **nitpick**  
*[expect=genuine,card=3] <proof>*

**proposition**  $(x \sqcup y) * z = ((x * 1) * y) * ((x * 1) * z)$  **nitpick**  
*[expect=genuine,card=2] <proof>*

**proposition** *while-top-2*:  $\text{top} * z = \text{top} * z$  **nitpick** *[expect=genuine,card=2] <proof>*

**proposition** *while-mult-top-2*:  $(x * \text{top}) * z = z \sqcup x * \text{top} * z$  **nitpick**

```

[expect=genuine,card=2] <proof>
proposition while-one-mult:  $(x \star 1) * x = x \star x$  nitpick
[expect=genuine,card=4] <proof>
proposition  $(x \star 1) * y = x \star y$  nitpick [expect=genuine,card=2] <proof>
proposition while-associative:  $(x \star y) * z = x \star (y * z)$  nitpick
[expect=genuine,card=2] <proof>
proposition while-back-loop-is-fixpoint: is-fixpoint  $(\lambda x . x * y \sqcup z) (z * (y \star 1))$ 
nitpick [expect=genuine,card=4] <proof>
proposition  $1 \sqcup x * \text{bot} = x \star 1$  nitpick [expect=genuine,card=3] <proof>
proposition  $x = x * (x \star 1)$  nitpick [expect=genuine,card=3] <proof>
proposition  $x * (x \star 1) = x \star 1$  nitpick [expect=genuine,card=2] <proof>
proposition  $x \star 1 = x \star (1 \star 1)$  nitpick [expect=genuine,card=3] <proof>
proposition  $(x \sqcup y) \star 1 = (x \star (y \star 1)) \star 1$  nitpick [expect=genuine,card=3]
<proof>
proposition  $z \sqcup y * x = x \implies y \star z \leq x$  nitpick [expect=genuine,card=2]
<proof>
proposition  $y * x = x \implies y \star x \leq x$  nitpick [expect=genuine,card=2] <proof>
proposition  $z \sqcup x * y = x \implies z * (y \star 1) \leq x$  nitpick
[expect=genuine,card=3] <proof>
proposition  $x * y = x \implies x * (y \star 1) \leq x$  nitpick [expect=genuine,card=3]
<proof>
proposition  $x * z = z * y \implies x \star z \leq z * (y \star 1)$  nitpick
[expect=genuine,card=2] <proof>

proposition while-unfold-below-1:  $x = y * x \implies x \leq y \star 1$  nitpick
[expect=genuine,card=3] <proof>
proposition  $x^\omega \leq x^\omega * x^\omega$  <proof>
proposition tarski-omega-idempotent:  $x^{\omega\omega} = x^\omega$  <proof>

```

**end**

```

class n-omega-algebra-binary-strict = n-omega-algebra-binary + circ +
  assumes L-left-zero:  $L * x = L$ 
  assumes circ-def:  $x^\circ = n(x^\omega) * L \sqcup x^\star$ 
begin

```

```

subclass strict-binary-itering
  <proof>

```

**end**

**end**

## 21 N-Relation-Algebras

**theory** N-Relation-Algebras

**imports** Stone-Relation-Algebras.Relation-Algebras N-Omega-Algebras

**begin**

**context** *bounded-distrib-allegory*

**begin**

**subclass** *lattice-ordered-pre-left-semiring*  $\langle$ *proof* $\rangle$

**end**

[Theorem 37](#)

**sublocale** *relation-algebra*  $<$  *n-algebra* **where**  $sup = sup$  **and**  $bot = bot$  **and**  $top = top$  **and**  $inf = inf$  **and**  $n = N$  **and**  $L = top$   
 $\langle$ *proof* $\rangle$

**sublocale** *relation-algebra*  $<$  *n-algebra-apx* **where**  $sup = sup$  **and**  $bot = bot$  **and**  $top = top$  **and**  $inf = inf$  **and**  $n = N$  **and**  $L = top$  **and**  $apx = greater-eq$   
 $\langle$ *proof* $\rangle$

**no-notation**

*inverse-divide* (**infixl**  $\langle$ '/ $\rangle$  70)

**notation**

*divide* (**infixl**  $\langle$ '/ $\rangle$  70)

**class** *left-residuated-relation-algebra* = *relation-algebra* + *inverse* +  
**assumes** *lres-def*:  $x / y = -(-x * y^T)$

**begin**

[Theorem 32.1](#)

**subclass** *residuated-pre-left-semiring*

$\langle$ *proof* $\rangle$

**end**

**context** *left-residuated-relation-algebra*

**begin**

[Theorem 32.3](#)

**lemma** *lres-mult-lres-lres*:

$x / (z * y) = (x / y) / z$

$\langle$ *proof* $\rangle$

[Theorem 32.5](#)

**lemma** *lres-dist-inf*:

$(x \sqcap y) / z = (x / z) \sqcap (y / z)$

$\langle$ *proof* $\rangle$

[Theorem 32.6](#)

**lemma** *lres-add-export-vector*:

**assumes** *vector x*  
**shows**  $(x \sqcup y) / z = x \sqcup (y / z)$   
 <proof>

**Theorem 32.7**

**lemma** *lres-top-vector*:  
*vector (x / top)*  
 <proof>

**Theorem 32.10**

**lemma** *lres-top-export-inf-mult*:  
 $((x / top) \sqcap y) * z = (x / top) \sqcap (y * z)$   
 <proof>

**lemma** *N-lres*:  
 $N(x) = x / top \sqcap 1$   
 <proof>

**end**

**class** *complete-relation-algebra* = *relation-algebra* + *complete-lattice*  
**begin**

**definition** *mu* ::  $('a \Rightarrow 'a) \Rightarrow 'a$  **where**  $mu\ f \equiv Inf\ \{ y . f\ y \leq y \}$   
**definition** *nu* ::  $('a \Rightarrow 'a) \Rightarrow 'a$  **where**  $nu\ f \equiv Sup\ \{ y . y \leq f\ y \}$

**lemma** *mu-lower-bound*:  
 $f\ x \leq x \implies mu\ f \leq x$   
 <proof>

**lemma** *mu-greatest-lower-bound*:  
 $(\forall y . f\ y \leq y \longrightarrow x \leq y) \implies x \leq mu\ f$   
 <proof>

**lemma** *mu-unfold-1*:  
 $isotone\ f \implies f\ (mu\ f) \leq mu\ f$   
 <proof>

**lemma** *mu-unfold-2*:  
 $isotone\ f \implies mu\ f \leq f\ (mu\ f)$   
 <proof>

**lemma** *mu-unfold*:  
 $isotone\ f \implies mu\ f = f\ (mu\ f)$   
 <proof>

**lemma** *mu-const*:  
 $mu\ (\lambda x . y) = y$   
 <proof>

**lemma** *mu-lpfp*:  
*isotone f*  $\implies$  *is-least-prefixpoint f (mu f)*  
 ⟨*proof*⟩

**lemma** *mu-lfp*:  
*isotone f*  $\implies$  *is-least-fixpoint f (mu f)*  
 ⟨*proof*⟩

**lemma** *mu-pmu*:  
*isotone f*  $\implies$   $p\mu f = mu f$   
 ⟨*proof*⟩

**lemma** *mu-mu*:  
*isotone f*  $\implies$   $\mu f = mu f$   
 ⟨*proof*⟩

**end**

**class** *omega-relation-algebra* = *relation-algebra* + *star* + *omega* +  
**assumes** *ra-star-left-unfold* :  $1 \sqcup y * y^* \leq y^*$   
**assumes** *ra-star-left-induct* :  $z \sqcup y * x \leq x \longrightarrow y^* * z \leq x$   
**assumes** *ra-star-right-induct* :  $z \sqcup x * y \leq x \longrightarrow z * y^* \leq x$   
**assumes** *ra-omega-unfold* :  $y^\omega = y * y^\omega$   
**assumes** *ra-omega-induct* :  $x \leq z \sqcup y * x \longrightarrow x \leq y^\omega \sqcup y^* * z$   
**begin**

**subclass** *bounded-omega-algebra*  
 ⟨*proof*⟩

**end**

[Theorem 38](#)

**sublocale** *omega-relation-algebra* < *n-omega-algebra* **where** *sup* = *sup* **and** *bot*  
 = *bot* **and** *top* = *top* **and** *inf* = *inf* **and** *n* = *N* **and** *L* = *top* **and** *apx* =  
*greater-eq* **and** *Omega* =  $\lambda x . N(x^\omega) * top \sqcup x^*$   
 ⟨*proof*⟩

**end**

## 22 Domain

**theory** *Domain*

**imports** *Stone-Relation-Algebras.Semirings Tests*

**begin**

**context** *idempotent-left-semiring*

**begin**

**sublocale** *ils*: *il-semiring* **where** *inf* = *times* **and** *sup* = *sup* **and** *bot* = *bot* **and**  
*less-eq* = *less-eq* **and** *less* = *less* **and** *top* = 1  
⟨*proof*⟩

**end**

**class** *left-zero-domain-semiring* = *idempotent-left-zero-semiring* + *dom* +  
**assumes** *d-restrict*:  $x \sqcup d(x) * x = d(x) * x$   
**assumes** *d-mult-d* :  $d(x * y) = d(x * d(y))$   
**assumes** *d-plus-one*:  $d(x) \sqcup 1 = 1$   
**assumes** *d-zero* :  $d(bot) = bot$   
**assumes** *d-dist-sup*:  $d(x \sqcup y) = d(x) \sqcup d(y)$

**begin**

Many lemmas in this class are taken from Georg Struth's theories.

**lemma** *d-restrict-equals*:

$x = d(x) * x$   
⟨*proof*⟩

**lemma** *d-idempotent*:

$d(d(x)) = d(x)$   
⟨*proof*⟩

**lemma** *d-fixpoint*:

$(\exists y . x = d(y)) \longleftrightarrow x = d(x)$   
⟨*proof*⟩

**lemma** *d-type*:

$\forall P . (\forall x . x = d(x) \longrightarrow P(x)) \longleftrightarrow (\forall x . P(d(x)))$   
⟨*proof*⟩

**lemma** *d-mult-sub*:

$d(x * y) \leq d(x)$   
⟨*proof*⟩

**lemma** *d-sub-one*:

$x \leq 1 \implies x \leq d(x)$   
⟨*proof*⟩

**lemma** *d-strict*:

$d(x) = bot \longleftrightarrow x = bot$   
⟨*proof*⟩

**lemma** *d-one*:

$d(1) = 1$   
⟨*proof*⟩



**lemma** *d-below-one*:

$$d(x) \leq 1$$

*<proof>*

**lemma** *d-isotone*:

$$x \leq y \implies d(x) \leq d(y)$$

*<proof>*

**lemma** *d-plus-left-upper-bound*:

$$d(x) \leq d(x \sqcup y)$$

*<proof>*

**lemma** *d-export*:

$$d(d(x) * y) = d(x) * d(y)$$

*<proof>*

**lemma** *d-mult-idempotent*:

$$d(x) * d(x) = d(x)$$

*<proof>*

**lemma** *d-commutative*:

$$d(x) * d(y) = d(y) * d(x)$$

*<proof>*

**lemma** *d-least-left-preserver*:

$$x \leq d(y) * x \iff d(x) \leq d(y)$$

*<proof>*

**lemma** *d-weak-locality*:

$$x * y = \text{bot} \iff x * d(y) = \text{bot}$$

*<proof>*

**lemma** *d-sup-closed*:

$$d(d(x) \sqcup d(y)) = d(x) \sqcup d(y)$$

*<proof>*

**lemma** *d-mult-closed*:

$$d(d(x) * d(y)) = d(x) * d(y)$$

*<proof>*

**lemma** *d-mult-left-lower-bound*:

$$d(x) * d(y) \leq d(x)$$

*<proof>*

**lemma** *d-mult-greatest-lower-bound*:

$$d(x) \leq d(y) * d(z) \iff d(x) \leq d(y) \wedge d(x) \leq d(z)$$

*<proof>*

**lemma** *d-mult-left-absorb-sup*:

$d(x) * (d(x) \sqcup d(y)) = d(x)$   
 ⟨proof⟩

**lemma** *d-sup-left-absorb-mult*:

$d(x) \sqcup d(x) * d(y) = d(x)$   
 ⟨proof⟩

**lemma** *d-sup-left-dist-mult*:

$d(x) \sqcup d(y) * d(z) = (d(x) \sqcup d(y)) * (d(x) \sqcup d(z))$   
 ⟨proof⟩

**lemma** *d-order*:

$d(x) \leq d(y) \iff d(x) = d(x) * d(y)$   
 ⟨proof⟩

**lemma** *d-mult-below*:

$d(x) * y \leq y$   
 ⟨proof⟩

**lemma** *d-preserves-equation*:

$d(y) * x \leq x * d(y) \iff d(y) * x = d(y) * x * d(y)$   
 ⟨proof⟩

**end**

**class** *left-zero-antidomain-semiring* = *idempotent-left-zero-semiring* + *dom* + *uminus* +

**assumes** *a-restrict* :  $-x * x = \text{bot}$

**assumes** *a-plus-mult-d*:  $-(x * y) \sqcup -(x * --y) = -(x * --y)$

**assumes** *a-complement* :  $--x \sqcup -x = 1$

**assumes** *d-def* :  $d(x) = --x$

**begin**

**sublocale** *aa*: *a-algebra* **where** *minus* =  $\lambda x y . -(x \sqcup y)$  **and** *uminus* = *uminus* **and** *inf* = *times* **and** *sup* = *sup* **and** *bot* = *bot* **and** *less-eq* = *less-eq* **and** *less* = *less* **and** *top* = 1

⟨proof⟩

**subclass** *left-zero-domain-semiring*

⟨proof⟩

**subclass** *tests*

⟨proof⟩

Many lemmas in this class are taken from Georg Struth's theories.

**notation**

*uminus* (⟨a⟩)

**lemma** *a-greatest-left-absorber*:

$$a(x) * y = \text{bot} \iff a(x) \leq a(y)$$

*<proof>*

**lemma** *a-mult-d*:

$$a(x * y) = a(x * d(y))$$

*<proof>*

**lemma** *a-d-closed*:

$$d(a(x)) = a(x)$$

*<proof>*

**lemma** *a-plus-left-lower-bound*:

$$a(x \sqcup y) \leq a(x)$$

*<proof>*

**lemma** *a-mult-sup*:

$$a(x) * (y \sqcup x) = a(x) * y$$

*<proof>*

**lemma** *a-3*:

$$a(x) * a(y) * d(x \sqcup y) = \text{bot}$$

*<proof>*

**lemma** *a-export*:

$$a(a(x) * y) = d(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-fixpoint*:

$$\forall x . (a(x) = x \implies (\forall y . y = \text{bot}))$$

*<proof>*

**lemma** *a-strict*:

$$a(x) = 1 \iff x = \text{bot}$$

*<proof>*

**lemma** *d-complement-zero*:

$$d(x) * a(x) = \text{bot}$$

*<proof>*

**lemma** *a-complement-zero*:

$$a(x) * d(x) = \text{bot}$$

*<proof>*

**lemma** *a-shunting-zero*:

$$a(x) * d(y) = \text{bot} \iff a(x) \leq a(y)$$

*<proof>*

**lemma** *a-antitone*:

$$x \leq y \implies a(y) \leq a(x)$$

*<proof>*

**lemma** *a-mult-deMorgan:*

$$a(a(x) * a(y)) = d(x \sqcup y)$$

*<proof>*

**lemma** *a-mult-deMorgan-1:*

$$a(a(x) * a(y)) = d(x) \sqcup d(y)$$

*<proof>*

**lemma** *a-mult-deMorgan-2:*

$$a(d(x) * d(y)) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-plus-deMorgan:*

$$a(a(x) \sqcup a(y)) = d(x) * d(y)$$

*<proof>*

**lemma** *a-plus-deMorgan-1:*

$$a(d(x) \sqcup d(y)) = a(x) * a(y)$$

*<proof>*

**lemma** *a-mult-left-upper-bound:*

$$a(x) \leq a(x * y)$$

*<proof>*

**lemma** *d-a-closed:*

$$a(d(x)) = a(x)$$

*<proof>*

**lemma** *a-export-d:*

$$a(d(x) * y) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-7:*

$$d(x) * a(d(y) \sqcup d(z)) = d(x) * a(y) * a(z)$$

*<proof>*

**lemma** *d-a-shunting:*

$$d(x) * a(y) \leq d(z) \iff d(x) \leq d(z) \sqcup d(y)$$

*<proof>*

**lemma** *d-d-shunting:*

$$d(x) * d(y) \leq d(z) \iff d(x) \leq d(z) \sqcup a(y)$$

*<proof>*

**lemma** *d-cancellation-1:*

$$d(x) \leq d(y) \sqcup (d(x) * a(y))$$

*<proof>*

**lemma** *d-cancellation-2*:

$$(d(z) \sqcup d(y)) * a(y) \leq d(z)$$

*<proof>*

**lemma** *a-sup-closed*:

$$d(a(x) \sqcup a(y)) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-mult-closed*:

$$d(a(x) * a(y)) = a(x) * a(y)$$

*<proof>*

**lemma** *d-a-shunting-zero*:

$$d(x) * a(y) = \text{bot} \iff d(x) \leq d(y)$$

*<proof>*

**lemma** *d-d-shunting-zero*:

$$d(x) * d(y) = \text{bot} \iff d(x) \leq a(y)$$

*<proof>*

**lemma** *d-compl-intro*:

$$d(x) \sqcup d(y) = d(x) \sqcup a(x) * d(y)$$

*<proof>*

**lemma** *a-compl-intro*:

$$a(x) \sqcup a(y) = a(x) \sqcup d(x) * a(y)$$

*<proof>*

**lemma** *kat-2*:

$$y * a(z) \leq a(x) * y \implies d(x) * y * a(z) = \text{bot}$$

*<proof>*

**lemma** *kat-3*:

$$d(x) * y * a(z) = \text{bot} \implies d(x) * y = d(x) * y * d(z)$$

*<proof>*

**lemma** *kat-4*:

$$d(x) * y = d(x) * y * d(z) \implies d(x) * y \leq y * d(z)$$

*<proof>*

**lemma** *kat-2-equiv*:

$$y * a(z) \leq a(x) * y \iff d(x) * y * a(z) = \text{bot}$$

*<proof>*

**lemma** *kat-4-equiv*:

$$d(x) * y = d(x) * y * d(z) \iff d(x) * y \leq y * d(z)$$

*<proof>*

**lemma** *kat-3-equiv-opp*:

$$a(z) * y * d(x) = \text{bot} \longleftrightarrow y * d(x) = d(z) * y * d(x)$$

*<proof>*

**lemma** *kat-4-equiv-opp*:

$$y * d(x) = d(z) * y * d(x) \longleftrightarrow y * d(x) \leq d(z) * y$$

*<proof>*

**lemma** *d-restrict-iff*:

$$(x \leq y) \longleftrightarrow (x \leq d(x) * y)$$

*<proof>*

**lemma** *d-restrict-iff-1*:

$$(d(x) * y \leq z) \longleftrightarrow (d(x) * y \leq d(x) * z)$$

*<proof>*

**end**

**end**

## 23 Domain Iterings

**theory** *Domain-Iterings*

**imports** *Domain Lattice-Ordered-Semirings Omega-Algebras*

**begin**

**class** *domain-semiring-lattice* = *left-zero-domain-semiring* +  
*lattice-ordered-pre-left-semiring*

**begin**

**subclass** *bounded-idempotent-left-zero-semiring* *<proof>*

**lemma** *d-top*:

$$d(\text{top}) = 1$$

*<proof>*

**lemma** *mult-domain-top*:

$$x * d(y) * \text{top} \leq d(x * y) * \text{top}$$

*<proof>*

**lemma** *domain-meet-domain*:

$$d(x \sqcap d(y) * z) \leq d(y)$$

*<proof>*

**lemma** *meet-domain*:

$$x \sqcap d(y) * z = d(y) * (x \sqcap z)$$

*<proof>*

**lemma** *meet-intro-domain*:

$$x \sqcap y = d(y) * x \sqcap y$$

*<proof>*

**lemma** *meet-domain-top*:

$$x \sqcap d(y) * top = d(y) * x$$

*<proof>*

**proposition**  $d(x) = x * top \sqcap 1$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**lemma** *d-galois*:

$$d(x) \leq d(y) \iff x \leq d(y) * top$$

*<proof>*

**lemma** *vector-meet*:

$$x * top \sqcap y \leq d(x) * y$$

*<proof>*

**end**

**class** *domain-semiring-lattice-L* = *domain-semiring-lattice* + *L* +

**assumes** *l1*:  $x * L = x * bot \sqcup d(x) * L$

**assumes** *l2*:  $d(L) * x \leq x * d(L)$

**assumes** *l3*:  $d(L) * top \leq L \sqcup d(L * bot) * top$

**assumes** *l4*:  $L * top \leq L$

**assumes** *l5*:  $x * bot \sqcap L \leq (x \sqcap L) * bot$

**begin**

**lemma** *l8*:

$$(x \sqcap L) * bot \leq x * bot \sqcap L$$

*<proof>*

**lemma** *l9*:

$$x * bot \sqcap L \leq d(x * bot) * L$$

*<proof>*

**lemma** *l10*:

$$L * L = L$$

*<proof>*

**lemma** *l11*:

$$d(x) * L \leq x * L$$

*<proof>*

**lemma** *l12*:

$$d(x * bot) * L \leq x * bot$$

*<proof>*

**lemma l13:**

$$d(x * bot) * L \leq x$$

*<proof>*

**lemma l14:**

$$x * L \leq x * bot \sqcup L$$

*<proof>*

**lemma l15:**

$$x * d(y) * L = x * bot \sqcup d(x * y) * L$$

*<proof>*

**lemma l16:**

$$x * top \sqcap L \leq x * L$$

*<proof>*

**lemma l17:**

$$d(x) * L \leq d(x * L) * L$$

*<proof>*

**lemma l18:**

$$d(x) * L = d(x * L) * L$$

*<proof>*

**lemma l19:**

$$d(x * top * bot) * L \leq d(x * L) * L$$

*<proof>*

**lemma l20:**

$$x \leq y \iff x \leq y \sqcup L \wedge x \leq y \sqcup d(y * bot) * top$$

*<proof>*

**lemma l21:**

$$d(x * bot) * L \leq x * bot \sqcap L$$

*<proof>*

**lemma l22:**

$$x * bot \sqcap L = d(x * bot) * L$$

*<proof>*

**lemma l23:**

$$x * top \sqcap L = d(x) * L$$

*<proof>*

**lemma l29:**

$$L * d(L) = L$$

*<proof>*

**lemma l30:**



$d(L) * x \leq (x \sqcap L) \sqcup d(L * bot) * x$   
*<proof>*

**lemma** *l31*:

$d(L) * x = (x \sqcap L) \sqcup d(L * bot) * x$   
*<proof>*

**lemma** *l40*:

$L * x \leq L$   
*<proof>*

**lemma** *l41*:

$L * top = L$   
*<proof>*

**lemma** *l50*:

$x * bot \sqcap L = (x \sqcap L) * bot$   
*<proof>*

**lemma** *l51*:

$d(x * bot) * L = (x \sqcap L) * bot$   
*<proof>*

**lemma** *l90*:

$L * top * L = L$   
*<proof>*

**lemma** *l91*:

**assumes**  $x = x * top$   
**shows**  $d(L * bot) * x \leq d(x * bot) * top$   
*<proof>*

**lemma** *l92*:

**assumes**  $x = x * top$   
**shows**  $d(L * bot) * x \leq d((x \sqcap L) * bot) * top$   
*<proof>*

**end**

**class** *domain-itering-lattice-L* = *bounded-itering* + *domain-semiring-lattice-L*  
**begin**

**lemma** *mult-L-circ*:

$(x * L)^\circ = 1 \sqcup x * L$   
*<proof>*

**lemma** *mult-L-circ-mult-below*:

$(x * L)^\circ * y \leq y \sqcup x * L$   
*<proof>*

**lemma** *circ-L*:

$$L^\circ = L \sqcup 1$$

*<proof>*

**lemma** *circ-d0-L*:

$$x^\circ * d(x * \text{bot}) * L = x^\circ * \text{bot}$$

*<proof>*

**lemma** *d0-circ-left-unfold*:

$$d(x^\circ * \text{bot}) = d(x * x^\circ * \text{bot})$$

*<proof>*

**lemma** *d-circ-import*:

$$d(y) * x \leq x * d(y) \implies d(y) * x^\circ = d(y) * (d(y) * x)^\circ$$

*<proof>*

**end**

**class** *domain-omega-algebra-lattice-L* = *bounded-left-zero-omega-algebra* +  
*domain-semiring-lattice-L*

**begin**

**lemma** *mult-L-star*:

$$(x * L)^* = 1 \sqcup x * L$$

*<proof>*

**lemma** *mult-L-omega*:

$$(x * L)^\omega \leq x * L$$

*<proof>*

**lemma** *mult-L-sup-star*:

$$(x * L \sqcup y)^* = y^* \sqcup y^* * x * L$$

*<proof>*

**lemma** *mult-L-sup-omega*:

$$(x * L \sqcup y)^\omega \leq y^\omega \sqcup y^* * x * L$$

*<proof>*

**end**

**sublocale** *domain-omega-algebra-lattice-L* < *dL-star*: *itering* **where** *circ* = *star*  
*<proof>*

**sublocale** *domain-omega-algebra-lattice-L* < *dL-star*: *domain-itering-lattice-L*  
**where** *circ* = *star* *<proof>*

**context** *domain-omega-algebra-lattice-L*

**begin**

**lemma** *d0-star-below-d0-omega*:

$$d(x^* * bot) \leq d(x^\omega * bot)$$

*<proof>*

**lemma** *d0-below-d0-omega*:

$$d(x * bot) \leq d(x^\omega * bot)$$

*<proof>*

**lemma** *star-L-split*:

**assumes**  $y \leq z$

**and**  $x * z * L \leq x * bot \sqcup z * L$

**shows**  $x^* * y * L \leq x^* * bot \sqcup z * L$

*<proof>*

**lemma** *star-L-split-same*:

$$x * y * L \leq x * bot \sqcup y * L \implies x^* * y * L = x^* * bot \sqcup y * L$$

*<proof>*

**lemma** *star-d-L-split-equal*:

$$d(x * y) \leq d(y) \implies x^* * d(y) * L = x^* * bot \sqcup d(y) * L$$

*<proof>*

**lemma** *d0-omega-mult*:

$$d(x^\omega * y * bot) = d(x^\omega * bot)$$

*<proof>*

**lemma** *d-omega-export*:

$$d(y) * x \leq x * d(y) \implies d(y) * x^\omega = (d(y) * x)^\omega$$

*<proof>*

**lemma** *d-omega-import*:

$$d(y) * x \leq x * d(y) \implies d(y) * x^\omega = d(y) * (d(y) * x)^\omega$$

*<proof>*

**lemma** *star-d-omega-top*:

$$x^* * d(x^\omega) * top = x^* * bot \sqcup d(x^\omega) * top$$

*<proof>*

**lemma** *omega-meet-L*:

$$x^\omega \sqcap L = d(x^\omega) * L$$

*<proof>*

**proposition** *d-star-mult*:  $d(x * y) \leq d(y) \implies d(x^* * y) = d(x^* * bot) \sqcup d(y)$   
*<proof>*

**proposition** *d0-split-omega-omega*:  $x^\omega \leq x^\omega * bot \sqcup d(x^\omega \sqcap L) * top$  **nitpick**  
[*expect=genuine,card=2*] *<proof>*

**end**

end

## 24 Domain Recursion

theory *Domain-Recursion*

imports *Domain-Iterings Approximation*

begin

class *domain-semiring-lattice-apx* = *domain-semiring-lattice-L* + *apx* +  
 assumes *apx-def*:  $x \sqsubseteq y \iff x \leq y \sqcup L \wedge d(L) * y \leq x \sqcup d(x * \text{bot}) * \text{top}$   
begin

lemma *apx-transitive*:

assumes  $x \sqsubseteq y$   
 and  $y \sqsubseteq z$   
 shows  $x \sqsubseteq z$   
*<proof>*

lemma *apx-meet-L*:

assumes  $y \sqsubseteq x$   
 shows  $x \sqcap L \leq y \sqcap L$   
*<proof>*

lemma *sup-apx-left-isotone*:

assumes  $x \sqsubseteq y$   
 shows  $x \sqcup z \sqsubseteq y \sqcup z$   
*<proof>*

subclass *apx-biorder*

*<proof>*

lemma *mult-apx-left-isotone*:

assumes  $x \sqsubseteq y$   
 shows  $x * z \sqsubseteq y * z$   
*<proof>*

lemma *mult-apx-right-isotone*:

assumes  $x \sqsubseteq y$   
 shows  $z * x \sqsubseteq z * y$   
*<proof>*

subclass *apx-semiring*

*<proof>*

lemma *meet-L-apx-isotone*:

$x \sqsubseteq y \implies x \sqcap L \sqsubseteq y \sqcap L$

*<proof>*

**definition** *kappa-apx-meet* :: ('a ⇒ 'a) ⇒ bool

**where** *kappa-apx-meet* f ≡ *apx.has-least-fixpoint* f ∧ *has-apx-meet* (μ f) (ν f)  
∧ κ f = μ f Δ ν f

**definition** *kappa-mu-nu* :: ('a ⇒ 'a) ⇒ bool

**where** *kappa-mu-nu* f ≡ *apx.has-least-fixpoint* f ∧ κ f = μ f ⊔ (ν f ⊓ L)

**definition** *nu-below-mu-nu* :: ('a ⇒ 'a) ⇒ bool

**where** *nu-below-mu-nu* f ≡ d(L) \* ν f ≤ μ f ⊔ (ν f ⊓ L) ⊔ d(ν f \* bot) \* top

**definition** *nu-below-mu-nu-2* :: ('a ⇒ 'a) ⇒ bool

**where** *nu-below-mu-nu-2* f ≡ d(L) \* ν f ≤ μ f ⊔ (ν f ⊓ L) ⊔ d((μ f ⊔ (ν f ⊓ L)) \* bot) \* top

**definition** *mu-nu-apx-nu* :: ('a ⇒ 'a) ⇒ bool

**where** *mu-nu-apx-nu* f ≡ μ f ⊔ (ν f ⊓ L) ⊑ ν f

**definition** *mu-nu-apx-meet* :: ('a ⇒ 'a) ⇒ bool

**where** *mu-nu-apx-meet* f ≡ *has-apx-meet* (μ f) (ν f) ∧ μ f Δ ν f = μ f ⊔ (ν f ⊓ L)

**definition** *apx-meet-below-nu* :: ('a ⇒ 'a) ⇒ bool

**where** *apx-meet-below-nu* f ≡ *has-apx-meet* (μ f) (ν f) ∧ μ f Δ ν f ≤ ν f

**lemma** *mu-below-l*:

μ f ≤ μ f ⊔ (ν f ⊓ L)

*<proof>*

**lemma** *l-below-nu*:

*has-least-fixpoint* f ⇒ *has-greatest-fixpoint* f ⇒ μ f ⊔ (ν f ⊓ L) ≤ ν f

*<proof>*

**lemma** *n-l-nu*:

*has-least-fixpoint* f ⇒ *has-greatest-fixpoint* f ⇒ (μ f ⊔ (ν f ⊓ L)) ⊓ L = ν f

*<proof>*

**lemma** *l-apx-mu*:

μ f ⊔ (ν f ⊓ L) ⊑ μ f

*<proof>*

**lemma** *nu-below-mu-nu-nu-below-mu-nu-2*:

**assumes** *nu-below-mu-nu* f

**shows** *nu-below-mu-nu-2* f

*<proof>*

**lemma** *nu-below-mu-nu-2-nu-below-mu-nu*:

**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *nu-below-mu-nu-2 f*  
**shows** *nu-below-mu-nu f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-equivalent:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  (*nu-below-mu-nu f*  $\iff$   
*nu-below-mu-nu-2 f*)  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-2-mu-nu-apx-nu:*  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *nu-below-mu-nu-2 f*  
**shows** *mu-nu-apx-nu f*  
 ⟨*proof*⟩

**lemma** *mu-nu-apx-nu-mu-nu-apx-meet:*  
**assumes** *mu-nu-apx-nu f*  
**shows** *mu-nu-apx-meet f*  
 ⟨*proof*⟩

**lemma** *mu-nu-apx-meet-apx-meet-below-nu:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *mu-nu-apx-meet f*  $\implies$   
*apx-meet-below-nu f*  
 ⟨*proof*⟩

**lemma** *apx-meet-below-nu-nu-below-mu-nu-2:*  
**assumes** *apx-meet-below-nu f*  
**shows** *nu-below-mu-nu-2 f*  
 ⟨*proof*⟩

**lemma** *has-apx-least-fixpoint-kappa-apx-meet:*  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *apx.has-least-fixpoint f*  
**shows** *kappa-apx-meet f*  
 ⟨*proof*⟩

**lemma** *kappa-apx-meet-apx-meet-below-nu:*  
*has-greatest-fixpoint f*  $\implies$  *kappa-apx-meet f*  $\implies$  *apx-meet-below-nu f*  
 ⟨*proof*⟩

**lemma** *apx-meet-below-nu-kappa-mu-nu:*  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *isotone f*  
**and** *apx.isotone f*

**and** *apx-meet-below-nu f*  
**shows** *kappa-mu-nu f*  
 ⟨*proof*⟩

**lemma** *kappa-mu-nu-has-apx-least-fixpoint:*  
*kappa-mu-nu f*  $\implies$  *apx.has-least-fixpoint f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-kappa-mu-nu:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *isotone f*  $\implies$  *apx.isotone f*  
 $\implies$  *nu-below-mu-nu f*  $\implies$  *kappa-mu-nu f*  
 ⟨*proof*⟩

**lemma** *kappa-mu-nu-nu-below-mu-nu:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu f*  $\implies$   
*nu-below-mu-nu f*  
 ⟨*proof*⟩

**definition** *kappa-mu-nu-L* :: (*'a*  $\Rightarrow$  *'a*)  $\Rightarrow$  *bool*  
**where** *kappa-mu-nu-L f*  $\equiv$  *apx.has-least-fixpoint f*  $\wedge$   $\kappa f = \mu f \sqcup d(\nu f * bot) * L$

**definition** *nu-below-mu-nu-L* :: (*'a*  $\Rightarrow$  *'a*)  $\Rightarrow$  *bool*  
**where** *nu-below-mu-nu-L f*  $\equiv$   $d(L) * \nu f \leq \mu f \sqcup d(\nu f * bot) * top$

**definition** *mu-nu-apx-nu-L* :: (*'a*  $\Rightarrow$  *'a*)  $\Rightarrow$  *bool*  
**where** *mu-nu-apx-nu-L f*  $\equiv$   $\mu f \sqcup d(\nu f * bot) * L \sqsubseteq \nu f$

**definition** *mu-nu-apx-meet-L* :: (*'a*  $\Rightarrow$  *'a*)  $\Rightarrow$  *bool*  
**where** *mu-nu-apx-meet-L f*  $\equiv$  *has-apx-meet* ( $\mu f$ ) ( $\nu f$ )  $\wedge$   $\mu f \Delta \nu f = \mu f \sqcup d(\nu f * bot) * L$

**lemma** *n-below-l:*  
 $x \sqcup d(y * bot) * L \leq x \sqcup (y \sqcap L)$   
 ⟨*proof*⟩

**lemma** *n-equal-l:*  
**assumes** *nu-below-mu-nu-L f*  
**shows**  $\mu f \sqcup d(\nu f * bot) * L = \mu f \sqcup (\nu f \sqcap L)$   
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-L-nu-below-mu-nu:*  
*nu-below-mu-nu-L f*  $\implies$  *nu-below-mu-nu f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-L-kappa-mu-nu-L:*  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *isotone f*  $\implies$  *apx.isotone f*  
 $\implies$  *nu-below-mu-nu-L f*  $\implies$  *kappa-mu-nu-L f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-L-mu-nu-apx-nu-L*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *nu-below-mu-nu-L f*  $\implies$   
*mu-nu-apx-nu-L f*  
 ⟨*proof*⟩

**lemma** *nu-below-mu-nu-L-mu-nu-apx-meet-L*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *nu-below-mu-nu-L f*  $\implies$   
*mu-nu-apx-meet-L f*  
 ⟨*proof*⟩

**lemma** *mu-nu-apx-nu-L-nu-below-mu-nu-L*:  
**assumes** *has-least-fixpoint f*  
**and** *has-greatest-fixpoint f*  
**and** *mu-nu-apx-nu-L f*  
**shows** *nu-below-mu-nu-L f*  
 ⟨*proof*⟩

**lemma** *kappa-mu-nu-L-mu-nu-apx-nu-L*:  
*has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu-L f*  $\implies$  *mu-nu-apx-nu-L f*  
 ⟨*proof*⟩

**lemma** *mu-nu-apx-meet-L-mu-nu-apx-nu-L*:  
*mu-nu-apx-meet-L f*  $\implies$  *mu-nu-apx-nu-L f*  
 ⟨*proof*⟩

**lemma** *kappa-mu-nu-L-nu-below-mu-nu-L*:  
*has-least-fixpoint f*  $\implies$  *has-greatest-fixpoint f*  $\implies$  *kappa-mu-nu-L f*  $\implies$   
*nu-below-mu-nu-L f*  
 ⟨*proof*⟩

**end**

**class** *itering-apx* = *domain-itering-lattice-L* + *domain-semiring-lattice-apx*  
**begin**

**lemma** *circ-apx-isotone*:  
**assumes**  $x \sqsubseteq y$   
**shows**  $x^\circ \sqsubseteq y^\circ$   
 ⟨*proof*⟩

**end**

**class** *omega-algebra-apx* = *domain-omega-algebra-lattice-L* +  
*domain-semiring-lattice-apx*

**sublocale** *omega-algebra-apx* < *star*: *itering-apx* **where** *circ* = *star* ⟨*proof*⟩

**context** *omega-algebra-apx*



**begin**

**lemma** *omega-apx-isotone*:

**assumes**  $x \sqsubseteq y$   
**shows**  $x^\omega \sqsubseteq y^\omega$

*<proof>*

**lemma** *combined-apx-isotone*:

$x \sqsubseteq y \implies (x^\omega \sqcap L) \sqcup x^* * z \sqsubseteq (y^\omega \sqcap L) \sqcup y^* * z$

*<proof>*

**lemma** *d-split-nu-mu*:

$d(L) * (y^\omega \sqcup y^* * z) \leq y^* * z \sqcup ((y^\omega \sqcup y^* * z) \sqcap L) \sqcup d((y^\omega \sqcup y^* * z) * bot) * top$

*<proof>*

**lemma** *loop-exists*:

$d(L) * \nu (\lambda x . y * x \sqcup z) \leq \mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L) \sqcup d(\nu (\lambda x . y * x \sqcup z) * bot) * top$

*<proof>*

**lemma** *loop-apx-least-fixpoint*:

*apx.is-least-fixpoint*  $(\lambda x . y * x \sqcup z) (\mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L))$

*<proof>*

**lemma** *loop-has-apx-least-fixpoint*:

*apx.has-least-fixpoint*  $(\lambda x . y * x \sqcup z)$

*<proof>*

**lemma** *loop-semantic*:

$\kappa (\lambda x . y * x \sqcup z) = \mu (\lambda x . y * x \sqcup z) \sqcup (\nu (\lambda x . y * x \sqcup z) \sqcap L)$

*<proof>*

**lemma** *loop-semantic-kappa-mu-nu*:

$\kappa (\lambda x . y * x \sqcup z) = (y^\omega \sqcap L) \sqcup y^* * z$

*<proof>*

**lemma** *loop-semantic-kappa-mu-nu-domain*:

$\kappa (\lambda x . y * x \sqcup z) = d(y^\omega) * L \sqcup y^* * z$

*<proof>*

**lemma** *loop-semantic-apx-isotone*:

$w \sqsubseteq y \implies \kappa (\lambda x . w * x \sqcup z) \sqsubseteq \kappa (\lambda x . y * x \sqcup z)$

*<proof>*

**end**

**end**

## 25 Extended Designs

**theory** *Extended-Designs*

**imports** *Omega-Algebras Domain*

**begin**

**class** *domain-semiring-L-below* = *left-zero-domain-semiring* + *L* +

**assumes** *L-left-zero-below*:  $L * x \leq L$

**assumes** *mult-L-split*:  $x * L = x * bot \sqcup d(x) * L$

**begin**

**lemma** *d-zero-mult-L*:

$d(x * bot) * L \leq x$

*<proof>*

**lemma** *mult-L*:

$x * L \leq x * bot \sqcup L$

*<proof>*

**lemma** *d-mult-L*:

$d(x) * L \leq x * L$

*<proof>*

**lemma** *d-L-split*:

$x * d(y) * L = x * bot \sqcup d(x * y) * L$

*<proof>*

**lemma** *d-mult-mult-L*:

$d(x * y) * L \leq x * d(y) * L$

*<proof>*

**lemma** *L-L*:

$L * L = L$

*<proof>*

**end**

**class** *antidomain-semiring-L* = *left-zero-antidomain-semiring* + *L* +

**assumes** *d-zero-mult-L*:  $d(x * bot) * L \leq x$

**assumes** *d-L-zero* :  $d(L * bot) = 1$

**assumes** *mult-L* :  $x * L \leq x * bot \sqcup L$

**begin**

**lemma** *L-left-zero*:

$L * x = L$

*<proof>*

**subclass** *domain-semiring-L-below*  
 ⟨*proof*⟩

**end**

**class** *ed-below* = *bounded-left-zero-omega-algebra* + *domain-semiring-L-below* +  
*Omega* +  
**assumes** *Omega-def*:  $x^\Omega = d(x^\omega) * L \sqcup x^*$   
**begin**

**lemma** *Omega-isotone*:  
 $x \leq y \implies x^\Omega \leq y^\Omega$   
 ⟨*proof*⟩

**lemma** *star-below-Omega*:  
 $x^* \leq x^\Omega$   
 ⟨*proof*⟩

**lemma** *one-below-Omega*:  
 $1 \leq x^\Omega$   
 ⟨*proof*⟩

**lemma** *L-left-zero-star*:  
 $L * x^* = L$   
 ⟨*proof*⟩

**lemma** *L-left-zero-Omega*:  
 $L * x^\Omega = L$   
 ⟨*proof*⟩

**lemma** *mult-L-star*:  
 $(x * L)^* = 1 \sqcup x * L$   
 ⟨*proof*⟩

**lemma** *mult-L-omega-below*:  
 $(x * L)^\omega \leq x * L$   
 ⟨*proof*⟩

**lemma** *mult-L-sup-star*:  
 $(x * L \sqcup y)^* = y^* \sqcup y^* * x * L$   
 ⟨*proof*⟩

**lemma** *mult-L-sup-omega-below*:  
 $(x * L \sqcup y)^\omega \leq y^\omega \sqcup y^* * x * L$   
 ⟨*proof*⟩

**lemma** *mult-L-sup-circ*:  
 $(x * L \sqcup y)^\Omega = d(y^\omega) * L \sqcup y^* \sqcup y^* * x * L$   
 ⟨*proof*⟩

**lemma** *circ-sup-d*:

$(x^\Omega * y)^\Omega * x^\Omega = d((x^* * y)^\omega) * L \sqcup ((x^* * y)^* * x^* \sqcup (x^* * y)^* * d(x^\omega) * L)$   
*<proof>*

**proposition** *mult-L-omega*:  $(x * L)^\omega = x * L$  **nitpick** [*expect=genuine,card=5*]  
*<proof>*

**proposition** *mult-L-sup-omega*:  $(x * L \sqcup y)^\omega = y^\omega \sqcup y^* * x * L$  **nitpick**  
[*expect=genuine,card=5*] *<proof>*

**proposition** *d-Omega-circ-simulate-right-plus*:  $z * x \leq y * y^\Omega * z \sqcup w \implies z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$  **nitpick** [*expect=genuine,card=4*] *<proof>*

**proposition** *d-Omega-circ-simulate-left-plus*:  $x * z \leq z * y^\Omega \sqcup w \implies x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$  **nitpick** [*expect=genuine,card=3*] *<proof>*

**end**

**class** *ed* = *ed-below* +  
  **assumes** *L-left-zero*:  $L * x = L$   
**begin**

**lemma** *mult-L-omega*:

$(x * L)^\omega = x * L$   
*<proof>*

**lemma** *mult-L-sup-omega*:

$(x * L \sqcup y)^\omega = y^\omega \sqcup y^* * x * L$   
*<proof>*

**lemma** *d-Omega-circ-simulate-right-plus*:

**assumes**  $z * x \leq y * y^\Omega * z \sqcup w$   
  **shows**  $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$   
*<proof>*

**lemma** *d-Omega-circ-simulate-left-plus*:

**assumes**  $x * z \leq z * y^\Omega \sqcup w$   
  **shows**  $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$   
*<proof>*

**end**

[Theorem 2.5](#) and [Theorem 50.4](#)

**sublocale** *ed* < *ed-omega*: *itering where circ = Omega*  
*<proof>*

**sublocale** *ed* < *ed-star*: *itering where circ = star* *<proof>*

**class** *ed-2* = *ed-below* + *antidomain-semiring-L* + *Omega*  
**begin**

```
subclass ed
  <proof>
```

```
end
```

```
end
```

## 26 Relative Domain

```
theory Relative-Domain
```

```
imports Tests
```

```
begin
```

```
class Z =
  fixes Z :: 'a
```

```
class relative-domain-semiring = idempotent-left-semiring + dom + Z +
  assumes d-restrict :  $x \leq d(x) * x \sqcup Z$ 
  assumes d-mult-d :  $d(x * y) = d(x) * d(y)$ 
  assumes d-below-one:  $d(x) \leq 1$ 
  assumes d-Z :  $d(Z) = bot$ 
  assumes d-dist-sup :  $d(x \sqcup y) = d(x) \sqcup d(y)$ 
  assumes d-export :  $d(d(x) * y) = d(x) * d(y)$ 
```

```
begin
```

```
lemma d-plus-one:
   $d(x) \sqcup 1 = 1$ 
  <proof>
```

[Theorem 44.2](#)

```
lemma d-zero:
   $d(bot) = bot$ 
  <proof>
```

[Theorem 44.3](#)

```
lemma d-idempotent:
   $d(d(x)) = d(x)$ 
  <proof>
```

```
lemma d-fixpoint:
   $(\exists y . x = d(y)) \longleftrightarrow x = d(x)$ 
  <proof>
```

```
lemma d-type:
   $\forall P . (\forall x . x = d(x) \longrightarrow P(x)) \longleftrightarrow (\forall x . P(d(x)))$ 
  <proof>
```

#### Theorem 44.4

**lemma** *d-mult-sub*:

$$d(x * y) \leq d(x)$$

*<proof>*

**lemma** *d-sub-one*:

$$x \leq 1 \implies x \leq d(x) \sqcup Z$$

*<proof>*

**lemma** *d-one*:

$$d(1) \sqcup Z = 1 \sqcup Z$$

*<proof>*

#### Theorem 44.8

**lemma** *d-strict*:

$$d(x) = \text{bot} \iff x \leq Z$$

*<proof>*

#### Theorem 44.1

**lemma** *d-isotone*:

$$x \leq y \implies d(x) \leq d(y)$$

*<proof>*

**lemma** *d-plus-left-upper-bound*:

$$d(x) \leq d(x \sqcup y)$$

*<proof>*

**lemma** *d-mult-idempotent*:

$$d(x) * d(x) = d(x)$$

*<proof>*

#### Theorem 44.12

**lemma** *d-least-left-preserver*:

$$x \leq d(y) * x \sqcup Z \iff d(x) \leq d(y)$$

*<proof>*

#### Theorem 44.9

**lemma** *d-weak-locality*:

$$x * y \leq Z \iff x * d(y) \leq Z$$

*<proof>*

**lemma** *d-sup-closed*:

$$d(d(x) \sqcup d(y)) = d(x) \sqcup d(y)$$

*<proof>*

**lemma** *d-mult-closed*:

$$d(d(x) * d(y)) = d(x) * d(y)$$

*<proof>*

**lemma** *d-mult-left-lower-bound*:

$$d(x) * d(y) \leq d(x)$$

*<proof>*

**lemma** *d-mult-left-absorb-sup*:

$$d(x) * (d(x) \sqcup d(y)) = d(x)$$

*<proof>*

**lemma** *d-sup-left-absorb-mult*:

$$d(x) \sqcup d(x) * d(y) = d(x)$$

*<proof>*

**lemma** *d-commutative*:

$$d(x) * d(y) = d(y) * d(x)$$

*<proof>*

**lemma** *d-mult-greatest-lower-bound*:

$$d(x) \leq d(y) * d(z) \iff d(x) \leq d(y) \wedge d(x) \leq d(z)$$

*<proof>*

**lemma** *d-sup-left-dist-mult*:

$$d(x) \sqcup d(y) * d(z) = (d(x) \sqcup d(y)) * (d(x) \sqcup d(z))$$

*<proof>*

**lemma** *d-order*:

$$d(x) \leq d(y) \iff d(x) = d(x) * d(y)$$

*<proof>*

[Theorem 44.6](#)

**lemma** *Z-mult-decreasing*:

$$Z * x \leq Z$$

*<proof>*

[Theorem 44.5](#)

**lemma** *d-below-d-one*:

$$d(x) \leq d(1)$$

*<proof>*

[Theorem 44.7](#)

**lemma** *d-relative-Z*:

$$d(x) * x \sqcup Z = x \sqcup Z$$

*<proof>*

**lemma** *Z-left-zero-above-one*:

$$1 \leq x \implies Z * x = Z$$

*<proof>*

[Theorem 44.11](#)

**lemma** *kat-4*:

$d(x) * y = d(x) * y * d(z) \implies d(x) * y \leq y * d(z)$   
 ⟨proof⟩

**lemma** *kat-4-equiv*:

$d(x) * y = d(x) * y * d(z) \longleftrightarrow d(x) * y \leq y * d(z)$   
 ⟨proof⟩

**lemma** *kat-4-equiv-opp*:

$y * d(x) = d(z) * y * d(x) \longleftrightarrow y * d(x) \leq d(z) * y$   
 ⟨proof⟩

### Theorem 44.10

**lemma** *d-restrict-iff-1*:

$d(x) * y \leq z \longleftrightarrow d(x) * y \leq d(x) * z$   
 ⟨proof⟩

**proposition** *d-restrict* :  $x \leq d(x) * x \sqcup Z$  ⟨proof⟩

**proposition** *d-mult-d* :  $d(x * y) = d(x * d(y))$  ⟨proof⟩

**proposition** *d-below-one*:  $d(x) \leq 1$  ⟨proof⟩

**proposition** *d-Z* :  $d(Z) = \text{bot}$  ⟨proof⟩

**proposition** *d-dist-sup* :  $d(x \sqcup y) = d(x) \sqcup d(y)$  ⟨proof⟩

**proposition** *d-export* :  $d(d(x) * y) = d(x) * d(y)$  ⟨proof⟩

end

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

**lemma** *simp-dImage[simp]*:

$\exists y . \text{Rep-dImage } x = d(y)$   
 ⟨proof⟩

**setup-lifting** *type-definition-dImage*

### Theorem 44

**instantiation** *dImage* :: (*relative-domain-semiring*) *bounded-distrib-lattice*  
 begin

**lift-definition** *sup-dImage* :: *'a dImage*  $\Rightarrow$  *'a dImage*  $\Rightarrow$  *'a dImage* **is** *sup*  
 ⟨proof⟩

**lift-definition** *inf-dImage* :: *'a dImage*  $\Rightarrow$  *'a dImage*  $\Rightarrow$  *'a dImage* **is** *times*  
 ⟨proof⟩

**lift-definition** *bot-dImage* :: *'a dImage* **is** *bot*  
 ⟨proof⟩



```

lift-definition top-dImage :: 'a dImage is d(1)
  ⟨proof⟩

lift-definition less-eq-dImage :: 'a dImage ⇒ 'a dImage ⇒ bool is less-eq ⟨proof⟩

lift-definition less-dImage :: 'a dImage ⇒ 'a dImage ⇒ bool is less ⟨proof⟩

instance
  ⟨proof⟩

end

class bounded-relative-domain-semiring = relative-domain-semiring +
  bounded-idempotent-left-semiring
begin

lemma Z-top:
   $Z * top = Z$ 
  ⟨proof⟩

lemma d-restrict-top:
   $x \leq d(x) * top \sqcup Z$ 
  ⟨proof⟩

proposition d-one-one:  $d(1) = 1$  nitpick [expect=genuine,card=2] ⟨proof⟩

end

class relative-domain-semiring-split = relative-domain-semiring +
  assumes split-Z:  $x * (y \sqcup Z) \leq x * y \sqcup Z$ 
begin

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

end

class relative-antidomain-semiring = idempotent-left-semiring + dom + Z +
  uminus +
  assumes a-restrict :  $-x * x \leq Z$ 
  assumes a-mult-d :  $-(x * y) = -(x * --y)$ 
  assumes a-complement:  $-x * --x = bot$ 
  assumes a-Z :  $-Z = 1$ 
  assumes a-export :  $-(--x * y) = -x \sqcup -y$ 
  assumes a-dist-sup :  $-(x \sqcup y) = -x * -y$ 
  assumes d-def :  $d(x) = --x$ 
begin

```

**notation***uminus* ( $\langle a \rangle$ )[Theorem 45.7](#)**lemma** *a-complement-one*:

$$--x \sqcup -x = 1$$

*\langle proof \rangle*[Theorem 45.5](#) and [Theorem 45.6](#)**lemma** *a-d-closed*:

$$d(a(x)) = a(x)$$

*\langle proof \rangle***lemma** *a-below-one*:

$$a(x) \leq 1$$

*\langle proof \rangle***lemma** *a-export-a*:

$$a(a(x) * y) = d(x) \sqcup a(y)$$

*\langle proof \rangle***lemma** *a-sup-absorb*:

$$(x \sqcup a(y)) * a(a(y)) = x * a(a(y))$$

*\langle proof \rangle*[Theorem 45.10](#)**lemma** *a-greatest-left-absorber*:

$$a(x) * y \leq Z \iff a(x) \leq a(y)$$

*\langle proof \rangle***lemma** *a-plus-left-lower-bound*:

$$a(x \sqcup y) \leq a(x)$$

*\langle proof \rangle*[Theorem 45.2](#)**subclass** *relative-domain-semiring**\langle proof \rangle*[Theorem 45.1](#)**subclass** *tests**\langle proof \rangle***lemma** *a-plus-mult-d*:

$$-(x * y) \sqcup -(x * --y) = -(x * --y)$$

*\langle proof \rangle***lemma** *a-mult-d-2*:

$$a(x * y) = a(x * d(y))$$

*\langle proof \rangle*

**lemma a-3:**

$$a(x) * a(y) * d(x \sqcup y) = \text{bot}$$

*<proof>*

**lemma a-fixpoint:**

$$\forall x . (a(x) = x \longrightarrow (\forall y . y = \text{bot}))$$

*<proof>*

### Theorem 45.9

**lemma a-strict:**

$$a(x) = 1 \iff x \leq Z$$

*<proof>*

**lemma d-complement-zero:**

$$d(x) * a(x) = \text{bot}$$

*<proof>*

**lemma a-complement-zero:**

$$a(x) * d(x) = \text{bot}$$

*<proof>*

**lemma a-shunting-zero:**

$$a(x) * d(y) = \text{bot} \iff a(x) \leq a(y)$$

*<proof>*

**lemma a-antitone:**

$$x \leq y \implies a(y) \leq a(x)$$

*<proof>*

**lemma a-mult-deMorgan:**

$$a(a(x) * a(y)) = d(x \sqcup y)$$

*<proof>*

**lemma a-mult-deMorgan-1:**

$$a(a(x) * a(y)) = d(x) \sqcup d(y)$$

*<proof>*

**lemma a-mult-deMorgan-2:**

$$a(d(x) * d(y)) = a(x) \sqcup a(y)$$

*<proof>*

**lemma a-plus-deMorgan:**

$$a(a(x) \sqcup a(y)) = d(x) * d(y)$$

*<proof>*

**lemma a-plus-deMorgan-1:**

$$a(d(x) \sqcup d(y)) = a(x) * a(y)$$

*<proof>*

### Theorem 45.8

**lemma** *a-mult-left-upper-bound:*

$$a(x) \leq a(x * y)$$

*<proof>*

### Theorem 45.6

**lemma** *d-a-closed:*

$$a(d(x)) = a(x)$$

*<proof>*

**lemma** *a-export-d:*

$$a(d(x) * y) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-7:*

$$d(x) * a(d(y) \sqcup d(z)) = d(x) * a(y) * a(z)$$

*<proof>*

**lemma** *d-a-shunting:*

$$d(x) * a(y) \leq d(z) \iff d(x) \leq d(z) \sqcup d(y)$$

*<proof>*

**lemma** *d-d-shunting:*

$$d(x) * d(y) \leq d(z) \iff d(x) \leq d(z) \sqcup a(y)$$

*<proof>*

**lemma** *d-cancellation-1:*

$$d(x) \leq d(y) \sqcup (d(x) * a(y))$$

*<proof>*

**lemma** *d-cancellation-2:*

$$(d(z) \sqcup d(y)) * a(y) \leq d(z)$$

*<proof>*

**lemma** *a-sup-closed:*

$$d(a(x) \sqcup a(y)) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *a-mult-closed:*

$$d(a(x) * a(y)) = a(x) * a(y)$$

*<proof>*

**lemma** *d-a-shunting-zero:*

$$d(x) * a(y) = \text{bot} \iff d(x) \leq d(y)$$

*<proof>*

**lemma** *d-d-shunting-zero:*

$$d(x) * d(y) = \text{bot} \iff d(x) \leq a(y)$$

*<proof>*

**lemma** *d-compl-intro*:

$$d(x) \sqcup d(y) = d(x) \sqcup a(x) * d(y)$$

*<proof>*

**lemma** *a-compl-intro*:

$$a(x) \sqcup a(y) = a(x) \sqcup d(x) * a(y)$$

*<proof>*

**lemma** *kat-2*:

$$y * a(z) \leq a(x) * y \implies d(x) * y * a(z) = \text{bot}$$

*<proof>*

[Theorem 45.4](#)

**lemma** *kat-2-equiv*:

$$y * a(z) \leq a(x) * y \iff d(x) * y * a(z) = \text{bot}$$

*<proof>*

**lemma** *kat-3-equiv-opp*:

$$a(z) * y * d(x) = \text{bot} \iff y * d(x) = d(z) * y * d(x)$$

*<proof>*

[Theorem 45.4](#)

**lemma** *kat-3-equiv-opp-2*:

$$d(z) * y * a(x) = \text{bot} \iff y * a(x) = a(z) * y * a(x)$$

*<proof>*

**lemma** *kat-equiv-6*:

$$d(x) * y * a(z) = d(x) * y * \text{bot} \iff d(x) * y * a(z) \leq y * \text{bot}$$

*<proof>*

**lemma** *d-one-one*:

$$d(1) = 1$$

*<proof>*

**lemma** *case-split-left-sup*:

$$-p * x \leq y \wedge \neg\neg p * x \leq z \implies x \leq y \sqcup z$$

*<proof>*

**lemma** *test-mult-left-sub-dist-shunt*:

$$-p * (\neg\neg p * x \sqcup Z) \leq Z$$

*<proof>*

**lemma** *test-mult-left-dist-shunt*:

$$-p * (\neg\neg p * x \sqcup Z) = -p * Z$$

*<proof>*

**proposition** *a-restrict* :  $-x * x \leq Z$  *<proof>*

**proposition** *a-mult-d* :  $-(x * y) = -(x * --y)$  *<proof>*  
**proposition** *a-complement*:  $-x * --x = \text{bot}$  *<proof>*  
**proposition** *a-Z* :  $-Z = 1$  *<proof>*  
**proposition** *a-export* :  $-(-x * y) = -x \sqcup -y$  *<proof>*  
**proposition** *a-dist-sup* :  $-(x \sqcup y) = -x * -y$  *<proof>*  
**proposition** *d-def* :  $d(x) = --x$  *<proof>*

**end**

**typedef** (overloaded) 'a aImage = { x::'a::relative-antidomain-semiring .  
 ( $\exists y::'a . x = a(y)$ ) }  
*<proof>*

**lemma** *simp-aImage*[*simp*]:  
 $\exists y . \text{Rep-aImage } x = a(y)$   
*<proof>*

**setup-lifting** *type-definition-aImage*

[Theorem 45.3](#)

**instantiation** *aImage* :: (relative-antidomain-semiring) boolean-algebra  
**begin**

**lift-definition** *sup-aImage* :: 'a aImage  $\Rightarrow$  'a aImage  $\Rightarrow$  'a aImage **is** *sup*  
*<proof>*

**lift-definition** *inf-aImage* :: 'a aImage  $\Rightarrow$  'a aImage  $\Rightarrow$  'a aImage **is** *times*  
*<proof>*

**lift-definition** *minus-aImage* :: 'a aImage  $\Rightarrow$  'a aImage  $\Rightarrow$  'a aImage **is**  $\lambda x y . x$   
 $* a(y)$   
*<proof>*

**lift-definition** *uminus-aImage* :: 'a aImage  $\Rightarrow$  'a aImage **is** *a*  
*<proof>*

**lift-definition** *bot-aImage* :: 'a aImage **is** *bot*  
*<proof>*

**lift-definition** *top-aImage* :: 'a aImage **is** *1*  
*<proof>*

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

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

**instance**  
*<proof>*

**end**

**class** *bounded-relative-antidomain-semiring* = *relative-antidomain-semiring* +  
*bounded-idempotent-left-semiring*  
**begin**

**subclass** *bounded-relative-domain-semiring*  $\langle$ *proof* $\rangle$

**lemma** *a-top*:  
 $a(\text{top}) = \text{bot}$   
 $\langle$ *proof* $\rangle$

**lemma** *d-top*:  
 $d(\text{top}) = 1$   
 $\langle$ *proof* $\rangle$

**lemma** *shunting-top-1*:  
 $-p * x \leq y \implies x \leq --p * \text{top} \sqcup y$   
 $\langle$ *proof* $\rangle$

**lemma** *shunting-Z*:  
 $-p * x \leq Z \iff x \leq --p * \text{top} \sqcup Z$   
 $\langle$ *proof* $\rangle$

**proposition** *a-left-dist-sup*:  $-p * (y \sqcup z) = -p * y \sqcup -p * z$  **nitpick**  
 $[expect=genuine,card=7]$   $\langle$ *proof* $\rangle$

**proposition** *shunting-top*:  $-p * x \leq y \iff x \leq --p * \text{top} \sqcup y$  **nitpick**  
 $[expect=genuine,card=7]$   $\langle$ *proof* $\rangle$

**end**

**class** *relative-left-zero-antidomain-semiring* = *relative-antidomain-semiring* +  
*idempotent-left-zero-semiring*  
**begin**

**lemma** *kat-3*:  
 $d(x) * y * a(z) = \text{bot} \implies d(x) * y = d(x) * y * d(z)$   
 $\langle$ *proof* $\rangle$

**lemma** *a-a-below*:  
 $a(a(x)) * y \leq y$   
 $\langle$ *proof* $\rangle$

**lemma** *kat-equiv-5*:  
 $d(x) * y \leq y * d(z) \iff d(x) * y * a(z) = d(x) * y * \text{bot}$   
 $\langle$ *proof* $\rangle$

**lemma** *case-split-right-sup*:  
 $x * -p \leq y \implies x * --p \leq z \implies x \leq y \sqcup z$

```

    <proof>

end

class bounded-relative-left-zero-antidomain-semiring =
  relative-left-zero-antidomain-semiring + bounded-idempotent-left-zero-semiring
begin

lemma shunting-top:
   $-p * x \leq y \iff x \leq --p * top \sqcup y$ 
  <proof>

end

end

```

## 27 Relative Modal Operators

```

theory Relative-Modal

imports Relative-Domain

begin

class relative-diamond-semiring = relative-domain-semiring + diamond +
  assumes diamond-def:  $|x>y = d(x * y)$ 
begin

lemma diamond-x-1:
   $|x>1 = d(x)$ 
  <proof>

lemma diamond-x-d:
   $|x>d(y) = d(x * y)$ 
  <proof>

lemma diamond-x-und:
   $|x>d(y) = |x>y$ 
  <proof>

lemma diamond-d-closed:
   $|x>y = d(|x>y)$ 
  <proof>

  Theorem 46.11

lemma diamond-bot-y:
   $|bot>y = bot$ 
  <proof>

```



**lemma** *diamond-1-y*:

$$|1\rangle y = d(y)$$

*\langle proof \rangle*

[Theorem 46.12](#)

**lemma** *diamond-1-d*:

$$|1\rangle d(y) = d(y)$$

*\langle proof \rangle*

[Theorem 46.10](#)

**lemma** *diamond-d-y*:

$$|d(x)\rangle y = d(x) * d(y)$$

*\langle proof \rangle*

[Theorem 46.11](#)

**lemma** *diamond-d-bot*:

$$|d(x)\rangle bot = bot$$

*\langle proof \rangle*

[Theorem 46.12](#)

**lemma** *diamond-d-1*:

$$|d(x)\rangle 1 = d(x)$$

*\langle proof \rangle*

**lemma** *diamond-d-d*:

$$|d(x)\rangle d(y) = d(x) * d(y)$$

*\langle proof \rangle*

[Theorem 46.12](#)

**lemma** *diamond-d-d-same*:

$$|d(x)\rangle d(x) = d(x)$$

*\langle proof \rangle*

[Theorem 46.2](#)

**lemma** *diamond-left-dist-sup*:

$$|x \sqcup y\rangle z = |x\rangle z \sqcup |y\rangle z$$

*\langle proof \rangle*

[Theorem 46.3](#)

**lemma** *diamond-right-sub-dist-sup*:

$$|x\rangle y \sqcup |x\rangle z \leq |x\rangle (y \sqcup z)$$

*\langle proof \rangle*

[Theorem 46.4](#)

**lemma** *diamond-associative*:

$$|x * y\rangle z = |x\rangle (y * z)$$

*\langle proof \rangle*

[Theorem 46.4](#)

**lemma** *diamond-left-mult:*

$$|x * y\rangle z = |x\rangle |y\rangle z$$

*\langle proof \rangle*

**lemma** *diamond-right-mult:*

$$|x\rangle (y * z) = |x\rangle |y\rangle z$$

*\langle proof \rangle*

**Theorem 46.6**

**lemma** *diamond-d-export:*

$$|d(x) * y\rangle z = d(x) * |y\rangle z$$

*\langle proof \rangle*

**lemma** *diamond-diamond-export:*

$$||x\rangle y\rangle z = |x\rangle y * |z\rangle 1$$

*\langle proof \rangle*

**Theorem 46.1**

**lemma** *diamond-left-isotone:*

$$x \leq y \implies |x\rangle z \leq |y\rangle z$$

*\langle proof \rangle*

**Theorem 46.1**

**lemma** *diamond-right-isotone:*

$$y \leq z \implies |x\rangle y \leq |x\rangle z$$

*\langle proof \rangle*

**lemma** *diamond-isotone:*

$$w \leq y \implies x \leq z \implies |w\rangle x \leq |y\rangle z$$

*\langle proof \rangle*

**lemma** *diamond-left-upper-bound:*

$$|x\rangle y \leq |x \sqcup z\rangle y$$

*\langle proof \rangle*

**lemma** *diamond-right-upper-bound:*

$$|x\rangle y \leq |x\rangle (y \sqcup z)$$

*\langle proof \rangle*

**lemma** *diamond-lower-bound-right:*

$$|x\rangle (d(y) * d(z)) \leq |x\rangle d(y)$$

*\langle proof \rangle*

**lemma** *diamond-lower-bound-left:*

$$|x\rangle (d(y) * d(z)) \leq |x\rangle d(z)$$

*\langle proof \rangle*

**Theorem 46.5**

**lemma** *diamond-right-sub-dist-mult:*

$|x>(d(y) * d(z)) \leq |x>d(y) * |x>d(z)$   
 ⟨proof⟩

Theorem 46.13

**lemma** *diamond-demodalisation-1*:

$d(x) * |y>z \leq Z \iff d(x) * y * d(z) \leq Z$   
 ⟨proof⟩

Theorem 46.14

**lemma** *diamond-demodalisation-3*:

$|x>y \leq d(z) \iff x * d(y) \leq d(z) * x \sqcup Z$   
 ⟨proof⟩

Theorem 46.6

**lemma** *diamond-d-export-2*:

$|d(x) * y>z = d(x) * |d(x) * y>z$   
 ⟨proof⟩

Theorem 46.7

**lemma** *diamond-d-promote*:

$|x * d(y)>z = |x * d(y)>(d(y) * z)$   
 ⟨proof⟩

Theorem 46.8

**lemma** *diamond-d-import-iff*:

$d(x) \leq |y>z \iff d(x) \leq |d(x) * y>z$   
 ⟨proof⟩

Theorem 46.9

**lemma** *diamond-d-import-iff-2*:

$d(x) * d(y) \leq |z>w \iff d(x) * d(y) \leq |d(y) * z>w$   
 ⟨proof⟩

end

**class** *relative-box-semiring* = *relative-diamond-semiring* +  
*relative-antidomain-semiring* + *box* +

**assumes** *box-def*:  $|x]y = a(x * a(y))$

**begin**

Theorem 47.1

**lemma** *box-diamond*:

$|x]y = a(|x>a(y))$   
 ⟨proof⟩

Theorem 47.2

**lemma** *diamond-box*:

$|x>y = a(|x]a(y))$   
 ⟨proof⟩

**lemma** *box-x-bot*:

$$|x]bot = a(x)$$

*<proof>*

**lemma** *box-x-1*:

$$|x]1 = a(x * bot)$$

*<proof>*

**lemma** *box-x-d*:

$$|x]d(y) = a(x * a(y))$$

*<proof>*

**lemma** *box-x-und*:

$$|x]d(y) = |x]y$$

*<proof>*

**lemma** *box-x-a*:

$$|x]a(y) = a(x * y)$$

*<proof>*

[Theorem 47.15](#)

**lemma** *box-bot-y*:

$$|bot]y = 1$$

*<proof>*

**lemma** *box-1-y*:

$$|1]y = d(y)$$

*<proof>*

[Theorem 47.16](#)

**lemma** *box-1-d*:

$$|1]d(y) = d(y)$$

*<proof>*

**lemma** *box-1-a*:

$$|1]a(y) = a(y)$$

*<proof>*

**lemma** *box-d-y*:

$$|d(x)]y = a(x) \sqcup d(y)$$

*<proof>*

**lemma** *box-a-y*:

$$|a(x)]y = d(x) \sqcup d(y)$$

*<proof>*

[Theorem 47.14](#)

**lemma** *box-d-bot*:

$$|d(x)]bot = a(x)$$

*<proof>*

**lemma** *box-a-bot*:

$$|a(x)]bot = d(x)$$

*<proof>*

[Theorem 47.15](#)

**lemma** *box-d-1*:

$$|d(x)]1 = 1$$

*<proof>*

**lemma** *box-a-1*:

$$|a(x)]1 = 1$$

*<proof>*

[Theorem 47.13](#)

**lemma** *box-d-d*:

$$|d(x)]d(y) = a(x) \sqcup d(y)$$

*<proof>*

**lemma** *box-a-d*:

$$|a(x)]d(y) = d(x) \sqcup d(y)$$

*<proof>*

**lemma** *box-d-a*:

$$|d(x)]a(y) = a(x) \sqcup a(y)$$

*<proof>*

**lemma** *box-a-a*:

$$|a(x)]a(y) = d(x) \sqcup a(y)$$

*<proof>*

[Theorem 47.15](#)

**lemma** *box-d-d-same*:

$$|d(x)]d(x) = 1$$

*<proof>*

**lemma** *box-a-a-same*:

$$|a(x)]a(x) = 1$$

*<proof>*

[Theorem 47.16](#)

**lemma** *box-d-below-box*:

$$d(x) \leq |d(y)]d(x)$$

*<proof>*

**lemma** *box-d-closed*:

$$|x]y = d(|x]y)$$

$\langle proof \rangle$

**lemma** *box-deMorgan-1*:

$$a(|x]y) = |x>a(y)$$

$\langle proof \rangle$

**lemma** *box-deMorgan-2*:

$$a(|x>y) = |x]a(y)$$

$\langle proof \rangle$

### Theorem 47.5

**lemma** *box-left-dist-sup*:

$$|x \sqcup y]z = |x]z * |y]z$$

$\langle proof \rangle$

**lemma** *box-right-dist-sup*:

$$|x](y \sqcup z) = a(x * a(y) * a(z))$$

$\langle proof \rangle$

**lemma** *box-associative*:

$$|x * y]z = a(x * y * a(z))$$

$\langle proof \rangle$

### Theorem 47.6

**lemma** *box-left-mult*:

$$|x * y]z = |x]|y]z$$

$\langle proof \rangle$

**lemma** *box-right-mult*:

$$|x](y * z) = a(x * a(y * z))$$

$\langle proof \rangle$

### Theorem 47.7

**lemma** *box-right-submult-d-d*:

$$|x](d(y) * d(z)) \leq |x]d(y) * |x]d(z)$$

$\langle proof \rangle$

**lemma** *box-right-submult-a-d*:

$$|x](a(y) * d(z)) \leq |x]a(y) * |x]d(z)$$

$\langle proof \rangle$

**lemma** *box-right-submult-d-a*:

$$|x](d(y) * a(z)) \leq |x]d(y) * |x]a(z)$$

$\langle proof \rangle$

**lemma** *box-right-submult-a-a*:

$$|x](a(y) * a(z)) \leq |x]a(y) * |x]a(z)$$

$\langle proof \rangle$

### Theorem 47.8

**lemma** *box-d-export*:

$$|d(x) * y]z = a(x) \sqcup |y]z$$

*<proof>*

**lemma** *box-a-export*:

$$|a(x) * y]z = d(x) \sqcup |y]z$$

*<proof>*

#### Theorem 47.4

**lemma** *box-left-antitone*:

$$y \leq x \implies |x]z \leq |y]z$$

*<proof>*

#### Theorem 47.3

**lemma** *box-right-isotone*:

$$y \leq z \implies |x]y \leq |x]z$$

*<proof>*

**lemma** *box-antitone-isotone*:

$$y \leq w \implies x \leq z \implies |w]x \leq |y]z$$

*<proof>*

**lemma** *diamond-1-a*:

$$|1 > a(y) = a(y)$$

*<proof>*

**lemma** *diamond-a-y*:

$$|a(x) > y = a(x) * d(y)$$

*<proof>*

**lemma** *diamond-a-bot*:

$$|a(x) > bot = bot$$

*<proof>*

**lemma** *diamond-a-1*:

$$|a(x) > 1 = a(x)$$

*<proof>*

**lemma** *diamond-a-d*:

$$|a(x) > d(y) = a(x) * d(y)$$

*<proof>*

**lemma** *diamond-d-a*:

$$|d(x) > a(y) = d(x) * a(y)$$

*<proof>*

**lemma** *diamond-a-a*:

$$|a(x) > a(y) = a(x) * a(y)$$

*<proof>*

**lemma** *diamond-a-a-same*:

$$|a(x) \rangle a(x) = a(x)$$

*<proof>*

**lemma** *diamond-a-export*:

$$|a(x) * y \rangle z = a(x) * |y \rangle z$$

*<proof>*

**lemma** *a-box-a-a*:

$$a(p) * |a(p) \rangle a(q) = a(p) * a(q)$$

*<proof>*

**lemma** *box-left-lower-bound*:

$$|x \sqcup y \rangle z \leq |x \rangle z$$

*<proof>*

**lemma** *box-right-upper-bound*:

$$|x \rangle y \leq |x \rangle (y \sqcup z)$$

*<proof>*

**lemma** *box-lower-bound-right*:

$$|x \rangle (d(y) * d(z)) \leq |x \rangle d(y)$$

*<proof>*

**lemma** *box-lower-bound-left*:

$$|x \rangle (d(y) * d(z)) \leq |x \rangle d(z)$$

*<proof>*

[Theorem 47.9](#)

**lemma** *box-d-import*:

$$d(x) * |y \rangle z = d(x) * |d(x) * y \rangle z$$

*<proof>*

[Theorem 47.10](#)

**lemma** *box-d-promote*:

$$|x * d(y) \rangle z = |x * d(y) \rangle (d(y) * z)$$

*<proof>*

[Theorem 47.11](#)

**lemma** *box-d-import-iff*:

$$d(x) \leq |y \rangle z \iff d(x) \leq |d(x) * y \rangle z$$

*<proof>*

[Theorem 47.12](#)

**lemma** *box-d-import-iff-2*:

$$d(x) * d(y) \leq |z \rangle w \iff d(x) * d(y) \leq |d(y) * z \rangle w$$

*<proof>*

[Theorem 47.20](#)



**lemma** *box-demodalisation-2*:  
 $-p \leq |y|(-q) \longleftrightarrow -p * y * --q \leq Z$   
 $\langle proof \rangle$

**lemma** *box-right-sub-dist-sup*:  
 $|x]d(y) \sqcup |x]d(z) \leq |x](d(y) \sqcup d(z))$   
 $\langle proof \rangle$

**lemma** *box-diff-var*:  
 $|x](d(y) \sqcup a(z)) * |x]d(z) \leq |x]d(z)$   
 $\langle proof \rangle$

[Theorem 47.19](#)

**lemma** *diamond-demodalisation-2*:  
 $|x>y \leq d(z) \longleftrightarrow a(z) * x * d(y) \leq Z$   
 $\langle proof \rangle$

[Theorem 47.17](#)

**lemma** *box-below-Z*:  
 $(|x]y) * x * a(y) \leq Z$   
 $\langle proof \rangle$

[Theorem 47.18](#)

**lemma** *box-partial-correctness*:  
 $|x]1 = 1 \longleftrightarrow x * bot \leq Z$   
 $\langle proof \rangle$

**lemma** *diamond-split*:  
 $|x>y = d(z) * |x>y \sqcup a(z) * |x>y$   
 $\langle proof \rangle$

**lemma** *box-import-shunting*:  
 $-p * -q \leq |x](-r) \longleftrightarrow -q \leq |-p * x](-r)$   
 $\langle proof \rangle$

**proposition** *box-dist-mult*:  $|x](d(y) * d(z)) = |x](d(y)) * |x](d(z))$  **nitpick**  
 $[expect=genuine,card=6] \langle proof \rangle$

**proposition** *box-demodalisation-3*:  $d(x) \leq |y]d(z) \longrightarrow d(x) * y \leq y * d(z) \sqcup Z$   
**nitpick**  $[expect=genuine,card=6] \langle proof \rangle$

**proposition** *fbox-diff*:  $|x](d(y) \sqcup a(z)) \leq |x]y \sqcup a(|x]z)$  **nitpick**  
 $[expect=genuine,card=6] \langle proof \rangle$

**proposition** *diamond-diff*:  $|x>y * a(|x>z) \leq |x>(d(y) * a(z))$  **nitpick**  
 $[expect=genuine,card=6] \langle proof \rangle$

**proposition** *diamond-diff-var*:  $|x>d(y) \leq |x>(d(y) * a(z)) \sqcup |x>d(z)$  **nitpick**  
 $[expect=genuine,card=6] \langle proof \rangle$

**end**

**class** *relative-left-zero-diamond-semiring* = *relative-diamond-semiring* +  
*relative-domain-semiring* + *idempotent-left-zero-semiring*

**begin**

**lemma** *diamond-right-dist-sup*:

$$|x\rangle(y \sqcup z) = |x\rangle y \sqcup |x\rangle z$$

*<proof>*

**end**

**class** *relative-left-zero-box-semiring* = *relative-box-semiring* +  
*relative-left-zero-antidomain-semiring*

**begin**

**subclass** *relative-left-zero-diamond-semiring* *<proof>*

**lemma** *box-right-mult-d-d*:

$$|x](d(y) * d(z)) = |x]d(y) * |x]d(z)$$

*<proof>*

**lemma** *box-right-mult-a-d*:

$$|x](a(y) * d(z)) = |x]a(y) * |x]d(z)$$

*<proof>*

**lemma** *box-right-mult-d-a*:

$$|x](d(y) * a(z)) = |x]d(y) * |x]a(z)$$

*<proof>*

**lemma** *box-right-mult-a-a*:

$$|x](a(y) * a(z)) = |x]a(y) * |x]a(z)$$

*<proof>*

**lemma** *box-demodalisation-3*:

**assumes**  $d(x) \leq |y]d(z)$   
**shows**  $d(x) * y \leq y * d(z) \sqcup Z$   
*<proof>*

**lemma** *fbox-diff*:

$$|x](d(y) \sqcup a(z)) \leq |x]y \sqcup a(|x]z)$$

*<proof>*

**lemma** *diamond-diff-var*:

$$|x\rangle d(y) \leq |x\rangle(d(y) * a(z)) \sqcup |x\rangle d(z)$$

*<proof>*

**lemma** *diamond-diff*:

$$|x\rangle y * a(|x\rangle z) \leq |x\rangle(d(y) * a(z))$$

*<proof>*

**end**

end

## 28 Complete Tests

theory *Complete-Tests*

imports *Tests*

begin

class *complete-tests* = *tests* + *Sup* + *Inf* +  
 assumes *sup-test*: *test-set*  $A \longrightarrow \text{Sup } A = \text{--} \text{Sup } A$   
 assumes *sup-upper*: *test-set*  $A \wedge x \in A \longrightarrow x \leq \text{Sup } A$   
 assumes *sup-least*: *test-set*  $A \wedge (\forall x \in A . x \leq -y) \longrightarrow \text{Sup } A \leq -y$   
begin

lemma *Sup-isotone*:

*test-set*  $B \Longrightarrow A \subseteq B \Longrightarrow \text{Sup } A \leq \text{Sup } B$   
{*proof*}

lemma *mult-right-dist-sup*:

assumes *test-set*  $A$   
 shows  $\text{Sup } A * -p = \text{Sup } \{ x * -p \mid x . x \in A \}$   
{*proof*}

lemma *mult-left-dist-sup*:

assumes *test-set*  $A$   
 shows  $-p * \text{Sup } A = \text{Sup } \{ -p * x \mid x . x \in A \}$   
{*proof*}

definition *Sum* ::  $(\text{nat} \Rightarrow 'a) \Rightarrow 'a$

where  $\text{Sum } f \equiv \text{Sup } \{ f n \mid n::\text{nat} . \text{True} \}$

lemma *Sum-test*:

*test-seq*  $t \Longrightarrow \text{Sum } t = \text{--} \text{Sum } t$   
{*proof*}

lemma *Sum-upper*:

*test-seq*  $t \Longrightarrow t x \leq \text{Sum } t$   
{*proof*}

lemma *Sum-least*:

*test-seq*  $t \Longrightarrow (\forall n . t n \leq -p) \Longrightarrow \text{Sum } t \leq -p$   
{*proof*}

lemma *mult-right-dist-Sum*:

*test-seq*  $t \Longrightarrow (\forall n . t n * -p \leq -q) \Longrightarrow \text{Sum } t * -p \leq -q$   
{*proof*}

**lemma** *mult-left-dist-Sum*:  
 $test\text{-}seq\ t \implies (\forall n . -p * t\ n \leq -q) \implies -p * Sum\ t \leq -q$   
 ⟨proof⟩

**lemma** *pSum-below-Sum*:  
 $test\text{-}seq\ t \implies pSum\ t\ m \leq Sum\ t$   
 ⟨proof⟩

**lemma** *pSum-sup*:  
**assumes** *test-seq t*  
**shows**  $pSum\ t\ m = Sup\ \{ t\ i \mid i . i \in \{..<m\}\}$   
 ⟨proof⟩

**definition** *Prod* ::  $(nat \Rightarrow 'a) \Rightarrow 'a$   
**where**  $Prod\ f \equiv Inf\ \{ f\ n \mid n::nat . True \}$

**lemma** *Sum-range*:  
 $Sum\ f = Sup\ (range\ f)$   
 ⟨proof⟩

**lemma** *Prod-range*:  
 $Prod\ f = Inf\ (range\ f)$   
 ⟨proof⟩

end

end

## 29 Complete Domain

**theory** *Complete-Domain*

**imports** *Relative-Domain Complete-Tests*

**begin**

**class** *complete-antidomain-semiring* = *relative-antidomain-semiring* +  
*complete-tests* +  
**assumes** *a-dist-Sum*:  $ascending\text{-}chain\ f \longrightarrow -(Sum\ f) = Prod\ (\lambda n . -f\ n)$   
**assumes** *a-dist-Prod*:  $descending\text{-}chain\ f \longrightarrow -(Prod\ f) = Sum\ (\lambda n . -f\ n)$   
**begin**

**lemma** *a-ascending-chain*:  
 $ascending\text{-}chain\ f \implies descending\text{-}chain\ (\lambda n . -f\ n)$   
 ⟨proof⟩

**lemma** *a-descending-chain*:  
 $descending\text{-}chain\ f \implies ascending\text{-}chain\ (\lambda n . -f\ n)$   
 ⟨proof⟩

**lemma** *d-dist-Sum*:  
*ascending-chain*  $f \implies d(\text{Sum } f) = \text{Sum } (\lambda n . d(f\ n))$   
 ⟨*proof*⟩

**lemma** *d-dist-Prod*:  
*descending-chain*  $f \implies d(\text{Prod } f) = \text{Prod } (\lambda n . d(f\ n))$   
 ⟨*proof*⟩

**end**

**end**

## 30 Preconditions

**theory** *Preconditions*

**imports** *Tests*

**begin**

**class** *pre* =  
**fixes** *pre* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (**infixr** «» 55)

**class** *precondition* = *tests* + *pre* +  
**assumes** *pre-closed*:  $x \ll -q = \neg\neg(x \ll -q)$   
**assumes** *pre-seq*:  $x * y \ll -q = x \ll y \ll -q$   
**assumes** *pre-lower-bound-right*:  $x \ll -p * -q \leq x \ll -q$   
**assumes** *pre-one-increasing*:  $-q \leq 1 \ll -q$

**begin**

[Theorem 39.2](#)

**lemma** *pre-sub-distr*:  
 $x \ll -p * -q \leq (x \ll -p) * (x \ll -q)$   
 ⟨*proof*⟩

[Theorem 39.5](#)

**lemma** *pre-below-one*:  
 $x \ll -p \leq 1$   
 ⟨*proof*⟩

**lemma** *pre-lower-bound-left*:  
 $x \ll -p * -q \leq x \ll -p$   
 ⟨*proof*⟩

[Theorem 39.1](#)

**lemma** *pre-iso*:  
 $-p \leq -q \implies x \ll -p \leq x \ll -q$   
 ⟨*proof*⟩

Theorem 39.4 and Theorem 40.9

**lemma** *pre-below-pre-one*:

$$x \ll -p \leq x \ll 1$$

*<proof>*

Theorem 39.3

**lemma** *pre-seq-below-pre-one*:

$$x * y \ll 1 \leq x \ll 1$$

*<proof>*

Theorem 39.6

**lemma** *pre-compose*:

$$-p \leq x \ll -q \implies -q \leq y \ll -r \implies -p \leq x * y \ll -r$$

*<proof>*

**proposition** *pre-test-test*:  $-p * (-p \ll -q) = -p * -q$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *pre-test-promote*:  $-p \ll -q = -p \ll -p * -q$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *pre-test*:  $-p \ll -q = \neg\neg p \sqcup -q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-test*:  $-p \ll -q = -p * -q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-distr-mult*:  $x \ll -p * -q = (x \ll -p) * (x \ll -q)$  **nitpick**

[*expect=genuine,card=4*] *<proof>*

**proposition** *pre-distr-plus*:  $x \ll -p \sqcup -q = (x \ll -p) * (x \ll -q)$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**end**

**class** *precondition-test-test* = *precondition* +

**assumes** *pre-test-test*:  $-p * (-p \ll -q) = -p * -q$

**begin**

**lemma** *pre-one*:

$$1 \ll -p = -p$$

*<proof>*

**lemma** *pre-import*:

$$-p * (x \ll -q) = -p * (-p * x \ll -q)$$

*<proof>*

**lemma** *pre-import-composition*:

$$-p * (-p * x * y \ll -q) = -p * (x \ll y \ll -q)$$

*<proof>*

**lemma** *pre-import-equiv*:

$$-p \leq x \ll -q \iff -p \leq -p * x \ll -q$$

*<proof>*

**lemma** *pre-import-equiv-mult*:

$$-p*-q \leq x\ll-s \longleftrightarrow -p*-q \leq -q*x\ll-s$$

*<proof>*

**proposition** *pre-test-promote*:  $-p\ll-q = -p\ll-p*-q$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *pre-test*:  $-p\ll-q = --p\sqcup-q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-test*:  $-p\ll-q = -p*-q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-distr-mult*:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$  **nitpick**

[*expect=genuine,card=4*] *<proof>*

**proposition** *pre-distr-plus*:  $x\ll-p\sqcup-q = (x\ll-p)*(x\ll-q)$  **nitpick**

[*expect=2*] *<proof>*

**end**

**class** *precondition-promote* = *precondition* +

**assumes** *pre-test-promote*:  $-p\ll-q = -p\ll-p*-q$

**begin**

**lemma** *pre-mult-test-promote*:

$$x*-p\ll-q = x*-p\ll-p*-q$$

*<proof>*

**proposition** *pre-test-test*:  $-p*(-p\ll-q) = -p*-q$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**proposition** *pre-test*:  $-p\ll-q = --p\sqcup-q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-test*:  $-p\ll-q = -p*-q$  **nitpick** [*expect=genuine,card=2*]

*<proof>*

**proposition** *pre-distr-mult*:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$  **nitpick**

[*expect=genuine,card=4*] *<proof>*

**proposition** *pre-distr-plus*:  $x\ll-p\sqcup-q = (x\ll-p)*(x\ll-q)$  **nitpick**

[*expect=genuine,card=2*] *<proof>*

**end**

**class** *precondition-test-box* = *precondition* +

**assumes** *pre-test*:  $-p\ll-q = --p\sqcup-q$

**begin**

**lemma** *pre-test-neg*:

$$--p*(-p\ll-q) = --p$$

*<proof>*

**lemma** *pre-bot*:

$$bot\ll-q = 1$$

```

    <proof>

lemma pre-export:
   $-p*x\ll-q = --p\sqcup(x\ll-q)$ 
  <proof>

lemma pre-neg-mult:
   $--p \leq -p*x\ll-q$ 
  <proof>

lemma pre-test-test-same:
   $-p\ll-p = 1$ 
  <proof>

lemma test-below-pre-test-mult:
   $-q \leq -p\ll-p*-q$ 
  <proof>

lemma test-below-pre-test:
   $-q \leq -p\ll-q$ 
  <proof>

lemma test-below-pre-test-2:
   $--p \leq -p\ll-q$ 
  <proof>

lemma pre-test-bot:
   $-p\ll-bot = --p$ 
  <proof>

lemma pre-test-one:
   $-p\ll 1 = 1$ 
  <proof>

subclass precondition-test-test
  <proof>

subclass precondition-promote
  <proof>

proposition pre-test:  $-p\ll-q = -p*-q$  nitpick [expect=genuine,card=2]
  <proof>
proposition pre-distr-mult:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$  <proof>
proposition pre-distr-plus:  $x\ll-p\sqcup-q = (x\ll-p)*(x\ll-q)$  nitpick
  [expect=genuine,card=2] <proof>

end

class precondition-test-diamond = precondition +

```



**assumes** *pre-test*:  $-p \ll -q = -p * -q$   
**begin**

**lemma** *pre-test-neg*:  
 $--p * (-p \ll -q) = bot$   
(*proof*)

**lemma** *pre-bot*:  
 $bot \ll -q = bot$   
(*proof*)

**lemma** *pre-export*:  
 $-p * x \ll -q = -p * (x \ll -q)$   
(*proof*)

**lemma** *pre-neg-mult*:  
 $-p * x \ll -q \leq -p$   
(*proof*)

**lemma** *pre-test-test-same*:  
 $-p \ll -p = -p$   
(*proof*)

**lemma** *test-above-pre-test-plus*:  
 $--p \ll -p \sqcup -q \leq -q$   
(*proof*)

**lemma** *test-above-pre-test*:  
 $-p \ll -q \leq -q$   
(*proof*)

**lemma** *test-above-pre-test-2*:  
 $-p \ll -q \leq -p$   
(*proof*)

**lemma** *pre-test-bot*:  
 $-p \ll bot = bot$   
(*proof*)

**lemma** *pre-test-one*:  
 $-p \ll 1 = -p$   
(*proof*)

**subclass** *precondition-test-test*  
(*proof*)

**subclass** *precondition-promote*  
(*proof*)

**proposition** *pre-test*:  $-p\ll-q = --p\sqcup-q$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition** *pre-distr-mult*:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$  **nitpick**  
 [*expect=genuine,card=6*] ⟨*proof*⟩

**proposition** *pre-distr-plus*:  $x\ll-p\sqcup-q = (x\ll-p)*(x\ll-q)$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**end**

**class** *precondition-distr-mult* = *precondition* +  
**assumes** *pre-distr-mult*:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$   
**begin**

**proposition** *pre-test-test*:  $-p*(-p\ll-q) = -p*-q$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition** *pre-test-promote*:  $-p\ll-q = -p\ll-p*-q$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition** *pre-test*:  $-p\ll-q = --p\sqcup-q$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition** *pre-test*:  $-p\ll-q = -p*-q$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition** *pre-distr-plus*:  $x\ll-p\sqcup-q = (x\ll-p)*(x\ll-q)$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**end**

**class** *precondition-distr-plus* = *precondition* +  
**assumes** *pre-distr-plus*:  $x\ll-p\sqcup-q = (x\ll-p)\sqcup(x\ll-q)$   
**begin**

**proposition** *pre-test-test*:  $-p*(-p\ll-q) = -p*-q$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition** *pre-test-promote*:  $-p\ll-q = -p\ll-p*-q$  **nitpick**  
 [*expect=genuine,card=2*] ⟨*proof*⟩

**proposition** *pre-test*:  $-p\ll-q = --p\sqcup-q$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition** *pre-test*:  $-p\ll-q = -p*-q$  **nitpick** [*expect=genuine,card=2*]  
 ⟨*proof*⟩

**proposition** *pre-distr-mult*:  $x\ll-p*-q = (x\ll-p)*(x\ll-q)$  **nitpick**  
 [*expect=genuine,card=4*] ⟨*proof*⟩

**end**

**end**

## 31 Hoare Calculus

**theory** *Hoare*

**imports** *Complete-Tests Preconditions*

**begin**

**class** *ite* =

**fixes** *ite* :: 'a ⇒ 'a ⇒ 'a ⇒ 'a (⟨- ◁ - ▷ -⟩ [58,58,58] 57)

**class** *hoare-triple* =

**fixes** *hoare-triple* :: 'a ⇒ 'a ⇒ 'a ⇒ bool (⟨- ⟦ - ⟧ -⟩ [54,54,54] 53)

**class** *ifthenelse* = *precondition* + *ite* +

**assumes** *ite-pre*:  $x \triangleleft - p \triangleright y \ll - q = \neg p^*(x \ll - q) \sqcup \neg \neg p^*(y \ll - q)$

**begin**

[Theorem 40.2](#)

**lemma** *ite-pre-then*:

$\neg p^*(x \triangleleft - p \triangleright y \ll - q) = \neg p^*(x \ll - q)$

*<proof>*

[Theorem 40.3](#)

**lemma** *ite-pre-else*:

$\neg \neg p^*(x \triangleleft - p \triangleright y \ll - q) = \neg \neg p^*(y \ll - q)$

*<proof>*

**lemma** *ite-import-mult-then*:

$\neg p^*-q \leq x \ll -r \implies \neg p^*-q \leq x \triangleleft - p \triangleright y \ll -r$

*<proof>*

**lemma** *ite-import-mult-else*:

$\neg \neg p^*-q \leq y \ll -r \implies \neg \neg p^*-q \leq x \triangleleft - p \triangleright y \ll -r$

*<proof>*

[Theorem 40.1](#)

**lemma** *ite-import-mult*:

$\neg p^*-q \leq x \ll -r \implies \neg \neg p^*-q \leq y \ll -r \implies -q \leq x \triangleleft - p \triangleright y \ll -r$

*<proof>*

**end**

**class** *whiledo* = *ifthenelse* + *while* +

**assumes** *while-pre*:  $\neg p^*x \ll -q = \neg p^*(x \ll - p^*x \ll -q) \sqcup \neg \neg p^*-q$

**assumes** *while-post*:  $\neg p^*x \ll -q = \neg p^*x \ll \neg \neg p^*-q$

**begin**

[Theorem 40.4](#)

**lemma** *while-pre-then*:

$\neg p^*(\neg p^*x \ll -q) = \neg p^*(x \ll - p^*x \ll -q)$

*<proof>*

[Theorem 40.5](#)

**lemma** *while-pre-else*:

$$--p*(-p*x«-q) = --p*-q$$

*<proof>*

[Theorem 40.6](#)

**lemma** *while-pre-sub-1*:

$$-p*x«-q \leq x*(-p*x)\triangleleft-p\rangle 1 «-q$$

*<proof>*

[Theorem 40.7](#)

**lemma** *while-pre-sub-2*:

$$-p*x«-q \leq x\triangleleft-p\rangle 1 «-p*x«-q$$

*<proof>*

[Theorem 40.8](#)

**lemma** *while-pre-compl*:

$$--p \leq -p*x«--p$$

*<proof>*

**lemma** *while-pre-compl-one*:

$$--p \leq -p*x«1$$

*<proof>*

[Theorem 40.10](#)

**lemma** *while-export-equiv*:

$$-q \leq -p*x«1 \iff -p*-q \leq -p*x«1$$

*<proof>*

**lemma** *nat-test-pre*:

**assumes** *nat-test t s*

**and**  $-q \leq s$

**and**  $\forall n . t \ n*-p*-q \leq x«pSum \ t \ n*-q$

**shows**  $-q \leq -p*x«--p*-q$

*<proof>*

**lemma** *nat-test-pre-1*:

**assumes** *nat-test t s*

**and**  $-r \leq s$

**and**  $-r \leq -q$

**and**  $\forall n . t \ n*-p*-q \leq x«pSum \ t \ n*-q$

**shows**  $-r \leq -p*x«--p*-q$

*<proof>*

**lemma** *nat-test-pre-2*:

**assumes** *nat-test t s*

**and**  $-r \leq s$

**and**  $\forall n . t \ n*-p \leq x«pSum \ t \ n$

**shows**  $-r \leq -p*x«1$

*<proof>*

```

lemma nat-test-pre-3:
  assumes nat-test t s
    and  $-q \leq s$ 
    and  $\forall n . t \ n^* - p^* - q \leq x \ll pSum \ t \ n^* - q$ 
  shows  $-q \leq -p^* x \ll 1$ 
  <proof>

definition aL :: 'a
  where  $aL \equiv 1^* 1 \ll 1$ 

lemma aL-test:
   $aL = --aL$ 
  <proof>

end

class atoms = tests +
  fixes Atomic-program :: 'a set
  fixes Atomic-test :: 'a set
  assumes one-atomic-program:  $1 \in Atomic-program$ 
  assumes zero-atomic-test:  $bot \in Atomic-test$ 
  assumes atomic-test-test:  $p \in Atomic-test \longrightarrow p = --p$ 

class while-program = whiledo + atoms + power
begin

inductive-set Test-expression :: 'a set
  where atom-test:  $p \in Atomic-test \Longrightarrow p \in Test-expression$ 
    | neg-test:  $p \in Test-expression \Longrightarrow -p \in Test-expression$ 
    | conj-test:  $p \in Test-expression \Longrightarrow q \in Test-expression \Longrightarrow p^*q \in Test-expression$ 

lemma test-expression-test:
   $p \in Test-expression \Longrightarrow p = --p$ 
  <proof>

lemma disj-test:
   $p \in Test-expression \Longrightarrow q \in Test-expression \Longrightarrow p \sqcup q \in Test-expression$ 
  <proof>

lemma zero-test-expression:
   $bot \in Test-expression$ 
  <proof>

lemma one-test-expression:
   $1 \in Test-expression$ 
  <proof>

```

**lemma** *pSum-test-expression*:

$(\forall n . t \ n \in \text{Test-expression}) \implies p\text{Sum } t \ m \in \text{Test-expression}$

*<proof>*

**inductive-set** *While-program* :: 'a set

**where** *atom-prog*:  $x \in \text{Atomic-program} \implies x \in \text{While-program}$

| *seq-prog*:  $x \in \text{While-program} \implies y \in \text{While-program} \implies x*y \in \text{While-program}$

| *cond-prog*:  $p \in \text{Test-expression} \implies x \in \text{While-program} \implies y \in \text{While-program} \implies x\langle p \rangle y \in \text{While-program}$

| *while-prog*:  $p \in \text{Test-expression} \implies x \in \text{While-program} \implies p\star x \in \text{While-program}$

**lemma** *one-while-program*:

$1 \in \text{While-program}$

*<proof>*

**lemma** *power-while-program*:

$x \in \text{While-program} \implies x^{\wedge}m \in \text{While-program}$

*<proof>*

**inductive-set** *Pre-expression* :: 'a set

**where** *test-pre*:  $p \in \text{Test-expression} \implies p \in \text{Pre-expression}$

| *neg-pre*:  $p \in \text{Pre-expression} \implies \neg p \in \text{Pre-expression}$

| *conj-pre*:  $p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies p*q \in \text{Pre-expression}$

| *pre-pre*:  $p \in \text{Pre-expression} \implies x \in \text{While-program} \implies x\ll p \in \text{Pre-expression}$

**lemma** *pre-expression-test*:

$p \in \text{Pre-expression} \implies p = \neg\neg p$

*<proof>*

**lemma** *disj-pre*:

$p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies p\sqcup q \in \text{Pre-expression}$

*<proof>*

**lemma** *zero-pre-expression*:

$\text{bot} \in \text{Pre-expression}$

*<proof>*

**lemma** *one-pre-expression*:

$1 \in \text{Pre-expression}$

*<proof>*

**lemma** *pSum-pre-expression*:

$(\forall n . t \ n \in \text{Pre-expression}) \implies p\text{Sum } t \ m \in \text{Pre-expression}$

*<proof>*

**lemma** *aL-pre-expression*:

$aL \in \text{Pre-expression}$

$\langle \text{proof} \rangle$

**end**

**class** *hoare-calculus* = *while-program* + *complete-tests*

**begin**

**definition** *tfun* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$

**where**  $\text{tfun } p \ x \ q \ r \equiv p \sqcup (x \ll q * r)$

**lemma** *tfun-test*:

$p = \text{---}p \Longrightarrow q = \text{---}q \Longrightarrow r = \text{---}r \Longrightarrow \text{tfun } p \ x \ q \ r = \text{---}\text{tfun } p \ x \ q \ r$

$\langle \text{proof} \rangle$

**lemma** *tfun-pre-expression*:

$x \in \text{While-program} \Longrightarrow p \in \text{Pre-expression} \Longrightarrow q \in \text{Pre-expression} \Longrightarrow r \in \text{Pre-expression} \Longrightarrow \text{tfun } p \ x \ q \ r \in \text{Pre-expression}$

$\langle \text{proof} \rangle$

**lemma** *tfun-iso*:

$p = \text{---}p \Longrightarrow q = \text{---}q \Longrightarrow r = \text{---}r \Longrightarrow s = \text{---}s \Longrightarrow r \leq s \Longrightarrow \text{tfun } p \ x \ q \ r \leq \text{tfun } p \ x \ q \ s$

$\langle \text{proof} \rangle$

**definition** *tseq* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{nat} \Rightarrow 'a$

**where**  $\text{tseq } p \ x \ q \ r \ m \equiv (\text{tfun } p \ x \ q \ ^{\wedge} m) \ r$

**lemma** *tseq-test*:

$p = \text{---}p \Longrightarrow q = \text{---}q \Longrightarrow r = \text{---}r \Longrightarrow \text{tseq } p \ x \ q \ r \ m = \text{---}\text{tseq } p \ x \ q \ r \ m$

$\langle \text{proof} \rangle$

**lemma** *tseq-test-seq*:

$p = \text{---}p \Longrightarrow q = \text{---}q \Longrightarrow r = \text{---}r \Longrightarrow \text{test-seq } (\text{tseq } p \ x \ q \ r)$

$\langle \text{proof} \rangle$

**lemma** *tseq-pre-expression*:

$x \in \text{While-program} \Longrightarrow p \in \text{Pre-expression} \Longrightarrow q \in \text{Pre-expression} \Longrightarrow r \in \text{Pre-expression} \Longrightarrow \text{tseq } p \ x \ q \ r \ m \in \text{Pre-expression}$

$\langle \text{proof} \rangle$

**definition** *tsum* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$

**where**  $\text{tsum } p \ x \ q \ r \equiv \text{Sum } (\text{tseq } p \ x \ q \ r)$

**lemma** *tsum-test*:

$p = \text{---}p \Longrightarrow q = \text{---}q \Longrightarrow r = \text{---}r \Longrightarrow \text{tsum } p \ x \ q \ r = \text{---}\text{tsum } p \ x \ q \ r$

$\langle \text{proof} \rangle$

**lemma** *t-fun-test*:

$$q = \text{--}q \implies \text{tfun } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) = \text{--} \text{tfun } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL))$$

*<proof>*

**lemma** *t-fun-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies \text{tfun } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) \in \text{Pre-expression}$$

*<proof>*

**lemma** *t-seq-test*:

$$q = \text{--}q \implies \text{tseq } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) \ m = \text{--} \text{tseq } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) \ m$$

*<proof>*

**lemma** *t-seq-test-seq*:

$$q = \text{--}q \implies \text{test-seq } (\text{tseq } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)))$$

*<proof>*

**lemma** *t-seq-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies \text{tseq } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) \ m \in \text{Pre-expression}$$

*<proof>*

**lemma** *t-sum-test*:

$$q = \text{--}q \implies \text{tsum } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL)) = \text{--} \text{tsum } (-p) \ x \ (p \star x \ll q) \ (-p \sqcup (x \ll (p \star x \ll q) \star aL))$$

*<proof>*

**definition** *tfun2* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a

where *tfun2* p q x r s  $\equiv$  p  $\sqcup$  q\*(x\*r\*s)

**lemma** *tfun2-test*:

$$p = \text{--}p \implies q = \text{--}q \implies r = \text{--}r \implies s = \text{--}s \implies \text{tfun2 } p \ q \ x \ r \ s = \text{--} \text{tfun2 } p \ q \ x \ r \ s$$

*<proof>*

**lemma** *tfun2-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies r \in \text{Pre-expression} \implies s \in \text{Pre-expression} \implies \text{tfun2 } p \ q \ x \ r \ s \in \text{Pre-expression}$$

*<proof>*

**lemma** *tfun2-iso*:

$$p = \text{--}p \implies q = \text{--}q \implies r = \text{--}r \implies s1 = \text{--}s1 \implies s2 = \text{--}s2 \implies s1 \leq s2 \implies \text{tfun2 } p \ q \ x \ r \ s1 \leq \text{tfun2 } p \ q \ x \ r \ s2$$

*<proof>*

**definition** *tseq2* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  nat  $\Rightarrow$  'a

where *tseq2* p q x r s m  $\equiv$  (tfun2 p q x r  $\hat{\ }m$ ) s



**lemma** *tseq2-test*:

$$p = \text{--}p \implies q = \text{--}q \implies r = \text{--}r \implies s = \text{--}s \implies \text{tseq2 } p \ q \ x \ r \ s \ m = \text{--} \text{tseq2 } p \ q \ x \ r \ s \ m$$

*<proof>*

**lemma** *tseq2-test-seq*:

$$p = \text{--}p \implies q = \text{--}q \implies r = \text{--}r \implies s = \text{--}s \implies \text{test-seq } (\text{tseq2 } p \ q \ x \ r \ s)$$

*<proof>*

**lemma** *tseq2-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies r \in \text{Pre-expression} \implies s \in \text{Pre-expression} \implies \text{tseq2 } p \ q \ x \ r \ s \ m \in \text{Pre-expression}$$

*<proof>*

**definition** *tsum2* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a  $\Rightarrow$  'a

$$\text{where } \text{tsum2 } p \ q \ x \ r \ s \equiv \text{Sum } (\text{tseq2 } p \ q \ x \ r \ s)$$

**lemma** *tsum2-test*:

$$p = \text{--}p \implies q = \text{--}q \implies r = \text{--}r \implies s = \text{--}s \implies \text{tsum2 } p \ q \ x \ r \ s = \text{--} \text{tsum2 } p \ q \ x \ r \ s$$

*<proof>*

**lemma** *tfun2-test*:

$$p = \text{--}p \implies q = \text{--}q \implies \text{tfun2 } (-p*q) \ p \ x \ (p*x\ll q) \\ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)) = \text{--} \text{tfun2 } (-p*q) \ p \ x \ (p*x\ll q) \\ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL))$$

*<proof>*

**lemma** *tfun2-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies \text{tfun2 } (-p*q) \ p \ x \ (p*x\ll q) \ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \in \text{Pre-expression}$$

*<proof>*

**lemma** *tseq2-test*:

$$p = \text{--}p \implies q = \text{--}q \implies \text{tseq2 } (-p*q) \ p \ x \ (p*x\ll q) \\ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \ m = \text{--} \text{tseq2 } (-p*q) \ p \ x \ (p*x\ll q) \\ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \ m$$

*<proof>*

**lemma** *tseq2-test-seq*:

$$p = \text{--}p \implies q = \text{--}q \implies \text{test-seq } (\text{tseq2 } (-p*q) \ p \ x \ (p*x\ll q) \\ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)))$$

*<proof>*

**lemma** *tseq2-pre-expression*:

$$x \in \text{While-program} \implies p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies \text{tseq2 } (-p*q) \ p \ x \ (p*x\ll q) \ (-p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \ m \in \text{Pre-expression}$$

*<proof>*

**lemma** *t-sum2-test*:

$p = \neg\neg p \implies q = \neg\neg q \implies tsum2 (-p*q) p x (p*x\ll q)$   
 $(\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) = \neg\neg tsum2 (-p*q) p x (p*x\ll q)$   
 $(\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL))$   
 ⟨proof⟩

**lemma** *t-seq2-below-t-seq*:

**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**shows**  $tseq2 (-p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) m \leq tseq (-p) x$   
 $(p*x\ll q) (\neg p \sqcup (x\ll(p*x\ll q)*aL)) m$   
 ⟨proof⟩

**lemma** *t-seq2-below-t-sum*:

$p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies tseq2$   
 $(\neg p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) m \leq tsum (-p) x (p*x\ll q)$   
 $(\neg p \sqcup (x\ll(p*x\ll q)*aL))$   
 ⟨proof⟩

**lemma** *t-sum2-below-t-sum*:

$p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies tsum2$   
 $(\neg p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \leq tsum (-p) x (p*x\ll q)$   
 $(\neg p \sqcup (x\ll(p*x\ll q)*aL))$   
 ⟨proof⟩

**lemma** *t-seq2-below-w*:

$p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies tseq2$   
 $(\neg p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) m \leq p*x\ll q$   
 ⟨proof⟩

**lemma** *t-sum2-below-w*:

$p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies tsum2$   
 $(\neg p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) \leq p*x\ll q$   
 ⟨proof⟩

**lemma** *t-sum2-w*:

**assumes**  $aL = 1$   
**and**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**and**  $x \in \text{While-program}$   
**shows**  $tsum2 (-p*q) p x (p*x\ll q) (\neg p*q \sqcup p*(x\ll(p*x\ll q)*aL)) = p*x\ll q$   
 ⟨proof⟩

**inductive** *derived-hoare-triple* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool} (\langle \cdot \mid \cdot \rangle \rightarrow [54, 54, 54] 53)$

**where** *atom-trip*:  $p \in \text{Pre-expression} \implies x \in \text{Atomic-program} \implies x\ll p \mid x)p$   
 | *seq-trip*:  $p \mid x)q \wedge q \mid y)r \implies p \mid x*y)r$   
 | *cond-trip*:  $p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies p*q \mid x)r \wedge$   
 $\neg p*q \mid y)r \implies q \mid x \langle p \triangleright y \rangle r$

$|$  *while-trip*:  $p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies \text{test-seq } t \wedge q \leq$   
 $\text{Sum } t \implies t \text{ } 0^*p^*q(x) \text{ } aL^*q \implies (\forall n > 0 . t \text{ } n^*p^*q(x) \text{ } p\text{Sum } t \text{ } n^*q) \implies$   
 $q(p^*x) - p^*q$   
 $|$  *cons-trip*:  $p \in \text{Pre-expression} \implies s \in \text{Pre-expression} \implies p \leq q \wedge q(x)r$   
 $\implies r \leq s \implies p(x)s$

**lemma** *derived-type*:

$p(x)q \implies p \in \text{Pre-expression} \wedge q \in \text{Pre-expression} \wedge x \in \text{While-program}$   
 $\langle \text{proof} \rangle$

**lemma** *cons-pre-trip*:

$p \in \text{Pre-expression} \implies q(y)r \implies p^*q(y)r$   
 $\langle \text{proof} \rangle$

**lemma** *cons-post-trip*:

$q \in \text{Pre-expression} \implies r \in \text{Pre-expression} \implies p(y)q^*r \longrightarrow p(y)r$   
 $\langle \text{proof} \rangle$

**definition** *valid-hoare-triple* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool} (\langle - \rangle \langle - \rangle \rightarrow [54, 54, 54] 53)$

**where**  $p(x)q \equiv (p \in \text{Pre-expression} \wedge q \in \text{Pre-expression} \wedge x \in \text{While-program}$   
 $\wedge p \leq x \ll q)$

**end**

**class** *hoare-calculus-sound* = *hoare-calculus* +

**assumes** *while-soundness*:  $-p^*-q \leq x \ll -q \longrightarrow aL^*-q \leq -p^*x \ll -q$

**begin**

**lemma** *while-soundness-0*:

$-p^*-q \leq x \ll -q \implies -q^*aL \leq -p^*x \ll -p^*-q$   
 $\langle \text{proof} \rangle$

**lemma** *while-soundness-1*:

**assumes** *test-seq*  $t$   
**and**  $-q \leq \text{Sum } t$   
**and**  $t \text{ } 0^*-p^*-q \leq x \ll aL^*-q$   
**and**  $\forall n > 0 . t \text{ } n^*-p^*-q \leq x \ll p\text{Sum } t \text{ } n^*-q$   
**shows**  $-q \leq -p^*x \ll -p^*-q$   
 $\langle \text{proof} \rangle$

**lemma** *while-soundness-2*:

**assumes** *test-seq*  $t$   
**and**  $-r \leq \text{Sum } t$   
**and**  $\forall n . t \text{ } n^*-p \leq x \ll p\text{Sum } t \text{ } n$   
**shows**  $-r \leq -p^*x \ll 1$   
 $\langle \text{proof} \rangle$

**theorem** *soundness*:

$p(x)q \implies p(x)q$

$\langle proof \rangle$

**end**

**class** *hoare-calculus-pre-complete* = *hoare-calculus* +  
**assumes** *aL-pre-import*:  $(x \ll -q) * aL \leq x \ll -q * aL$   
**assumes** *pre-right-dist-Sum*:  $x \in \text{While-program} \wedge \text{ascending-chain } t \wedge \text{test-seq } t \longrightarrow x \ll \text{Sum } t = \text{Sum } (\lambda n . x \ll t n)$   
**begin**

**lemma** *aL-pre-import-equal*:  
 $(x \ll -q) * aL = (x \ll -q * aL) * aL$   
 $\langle proof \rangle$

**lemma** *aL-pre-below-t-seq2*:  
**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**shows**  $(p \star x \ll q) * aL \leq \text{tseq2 } (-p * q) p x (p \star x \ll q) (-p * q \sqcup p * (x \ll (p \star x \ll q) * aL)) 0$   
 $\langle proof \rangle$

**lemma** *t-seq2-ascending*:  
**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**and**  $x \in \text{While-program}$   
**shows**  $\text{tseq2 } (-p * q) p x (p \star x \ll q) (-p * q \sqcup p * (x \ll (p \star x \ll q) * aL)) m \leq \text{tseq2 } (-p * q) p x (p \star x \ll q) (-p * q \sqcup p * (x \ll (p \star x \ll q) * aL)) (\text{Suc } m)$   
 $\langle proof \rangle$

**lemma** *t-seq2-ascending-chain*:  
 $p \in \text{Test-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies \text{ascending-chain } (\text{tseq2 } (-p * q) p x (p \star x \ll q) (-p * q \sqcup p * (x \ll (p \star x \ll q) * aL)))$   
 $\langle proof \rangle$

**end**

**class** *hoare-calculus-complete* = *hoare-calculus-pre-complete* +  
**assumes** *while-completeness*:  $-p * (x \ll -q) \leq -q \longrightarrow -p \star x \ll -q \leq -q \sqcup aL$   
**begin**

**lemma** *while-completeness-var*:  
**assumes**  $-p * (x \ll -q) \sqcup -r \leq -q$   
**shows**  $-p \star x \ll -r \leq -q \sqcup aL$   
 $\langle proof \rangle$

**lemma** *while-completeness-sum*:  
**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**and**  $x \in \text{While-program}$   
**shows**  $p \star x \ll q \leq \text{tsum } (-p) x (p \star x \ll q) (-p \sqcup (x \ll (p \star x \ll q) * aL))$

$\langle proof \rangle$

**lemma** *while-complete*:

**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**and**  $x \in \text{While-program}$   
**and**  $\forall r \in \text{Pre-expression} . x \ll r(x)r$   
**shows**  $p \star x \ll q(p \star x)q$

$\langle proof \rangle$

**lemma** *pre-completeness*:

$x \in \text{While-program} \implies q \in \text{Pre-expression} \implies x \ll q(x)q$   
 $\langle proof \rangle$

**theorem** *completeness*:

$p(x)q \implies p(x)q$   
 $\langle proof \rangle$

**end**

**class** *hoare-calculus-sound-complete* = *hoare-calculus-sound* +  
*hoare-calculus-complete*

**begin**

[Theorem 41](#)

**theorem** *soundness-completeness*:

$p(x)q \longleftrightarrow p(x)q$   
 $\langle proof \rangle$

**end**

**class** *hoare-rules* = *whiledo* + *complete-tests* + *hoare-triple* +

**assumes** *rule-pre*:  $x \ll -q\{x\}-q$   
**assumes** *rule-seq*:  $-p\{x\}-q \wedge -q\{y\}-r \longrightarrow -p\{x*y\}-r$   
**assumes** *rule-cond*:  $-p*-q\{x\}-r \wedge --p*-q\{y\}-r \longrightarrow -q\{x \triangleleft -p \triangleright y\}-r$   
**assumes** *rule-while*:  $\text{test-seq } t \wedge -q \leq \text{Sum } t \wedge t \ 0*-p*-q\{x\} \ aL*-q \wedge (\forall n > 0 . t \ n*-p*-q\{x\} \ p \ \text{Sum } t \ n*-q) \longrightarrow -q\{-p*x\}-p*-q$   
**assumes** *rule-cons*:  $-p \leq -q \wedge -q\{x\}-r \wedge -r \leq -s \longrightarrow -p\{x\}-s$   
**assumes** *rule-disj*:  $-p\{x\}-r \wedge -q\{x\}-s \longrightarrow -p \sqcup -q\{x\}-r \sqcup -s$

**begin**

**lemma** *rule-cons-pre*:

$-p \leq -q \implies -q\{x\}-r \implies -p\{x\}-r$   
 $\langle proof \rangle$

**lemma** *rule-cons-pre-mult*:

$-q\{x\}-r \implies -p*-q\{x\}-r$   
 $\langle proof \rangle$

**lemma** *rule-cons-pre-plus*:  
 $-p \sqcup -q\{x\} - r \implies -p\{x\} - r$   
 ⟨proof⟩

**lemma** *rule-cons-post*:  
 $-q\{x\} - r \implies -r \leq -s \implies -q\{x\} - s$   
 ⟨proof⟩

**lemma** *rule-cons-post-mult*:  
 $-q\{x\} - r * -s \implies -q\{x\} - s$   
 ⟨proof⟩

**lemma** *rule-cons-post-plus*:  
 $-q\{x\} - r \implies -q\{x\} - r \sqcup -s$   
 ⟨proof⟩

**lemma** *rule-disj-pre*:  
 $-p\{x\} - r \implies -q\{x\} - r \implies -p \sqcup -q\{x\} - r$   
 ⟨proof⟩

**end**

**class** *hoare-calculus-valid* = *hoare-calculus-sound-complete* + *hoare-triple* +  
**assumes** *hoare-triple-valid*:  $-p\{x\} - q \longleftrightarrow -p \leq x \ll -q$   
**begin**

**lemma** *valid-hoare-triple-same*:  
 $p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies p\{x\}q$   
 $= p(x)q$   
 ⟨proof⟩

**lemma** *derived-hoare-triple-same*:  
 $p \in \text{Pre-expression} \implies q \in \text{Pre-expression} \implies x \in \text{While-program} \implies p\{x\}q$   
 $= p(x)q$   
 ⟨proof⟩

**lemma** *valid-rule-disj*:  
**assumes**  $-p\{x\} - r$   
**and**  $-q\{x\} - s$   
**shows**  $-p \sqcup -q\{x\} - r \sqcup -s$   
 ⟨proof⟩

**subclass** *hoare-rules*  
 ⟨proof⟩

**lemma** *nat-test-rule-while*:  
 $\text{nat-test } t \ s \implies -q \leq s \implies (\forall n . t \ n * -p * -q\{x\} pSum \ t \ n * -q) \implies$   
 $-q\{-p * x\} - -p * -q$   
 ⟨proof⟩

**lemma** *test-seq-rule-while*:  
 $test\text{-}seq\ t \implies -q \leq Sum\ t \implies t\ 0^* - p^* - q \{x\} aL^* - q \implies (\forall n > 0 . t\ n^* - p^* - q \{x\} pSum\ t\ n^* - q) \implies -q \{-p^*x\} - -p^* - q$   
 ⟨proof⟩

**lemma** *rule-bot*:  
 $bot\{x\} - p$   
 ⟨proof⟩

**lemma** *rule-skip*:  
 $-p\{1\} - p$   
 ⟨proof⟩

**lemma** *rule-example-4*:  
 assumes *test-seq* *t*  
 and  $Sum\ t = 1$   
 and  $t\ 0^* - p1^* - p3 = bot$   
 and  $-p1\{z1\} - p1^* - p2$   
 and  $\forall n > 0 . t\ n^* - p1^* - p2^* - p3\{z2\} pSum\ t\ n^* - p1^* - p2$   
 shows  $-p1\{z1^*(-p3^*z2)\} - p2^* - -p3$   
 ⟨proof⟩

**end**

**class** *hoare-calculus-pc-2* = *hoare-calculus-sound* + *hoare-calculus-pre-complete* +  
 assumes *aL-one*:  $aL = 1$   
**begin**

**subclass** *hoare-calculus-sound-complete*  
 ⟨proof⟩

**lemma** *while-soundness-pc*:  
 assumes  $-p^* - q \leq x \ll -q$   
 shows  $-q \leq -p^*x \ll - -p^* - q$   
 ⟨proof⟩

**end**

**class** *hoare-calculus-pc* = *hoare-calculus-sound* + *hoare-calculus-pre-complete* +  
 assumes *pre-one-one*:  $x \ll 1 = 1$   
**begin**

**subclass** *hoare-calculus-pc-2*  
 ⟨proof⟩

**end**

**class** *hoare-calculus-pc-valid* = *hoare-calculus-pc* + *hoare-calculus-valid*

**begin**

**lemma** *rule-while-pc*:

$-p^* - q\{x\} - q \implies -q\{-p^*x\} - -p^* - q$   
(*proof*)

**lemma** *rule-alternation*:

$-p\{x\} - q \implies -q\{y\} - p \implies -p\{-r^*x^*y\} - -r^* - p$   
(*proof*)

**lemma** *rule-alternation-context*:

$-p\{v\} - p \implies -p\{w\} - q \implies -q\{x\} - q \implies -q\{y\} - p \wedge -p\{z\} - p \implies$   
 $-p\{-r^*v^*w^*x^*y^*z\} - -r^* - p$   
(*proof*)

**lemma** *rule-example-3*:

**assumes**  $-p^* - q\{x\} - -p^* - q$   
**and**  $- -p^* - r\{x\} - p^* - r$   
**and**  $-p^* - r\{y\} - p^* - q$   
**and**  $- -p^* - q\{z\} - -p^* - r$   
**shows**  $-p^* - q \sqcup - -p^* - r \{ -s^*x^*(y \triangleleft -p \triangleright z) \} - -s^*( -p^* - q \sqcup - -p^* - r)$   
(*proof*)

**end**

**class** *hoare-calculus-tc* = *hoare-calculus* + *precondition-test-test* +  
*precondition-distr-mult* +

**assumes** *while-bnd*:  $p \in \text{Test-expression} \wedge q \in \text{Pre-expression} \wedge x \in$   
*While-program*  $\longrightarrow p^*x \ll q \leq \text{Sum } (\lambda n . (p^*x) \hat{\ } n \ll \text{bot})$

**begin**

**lemma**

**assumes**  $p \in \text{Test-expression}$   
**and**  $q \in \text{Pre-expression}$   
**and**  $x \in \text{While-program}$   
**shows**  $p^*x \ll q \leq \text{tsum } (-p) x (p^*x \ll q) (-p \sqcup (x \ll (p^*x \ll q) * aL))$   
(*proof*)

**end**

**class** *complete-pre* = *complete-tests* + *precondition* + *power*

**begin**

**definition** *bnd* :: 'a  $\Rightarrow$  'a

**where**  $\text{bnd } x \equiv \text{Sup } \{ x \hat{\ } n \ll \text{bot} \mid n :: \text{nat} . \text{True} \}$

**lemma** *bnd-test-set*:

*test-set*  $\{ x \hat{\ } n \ll \text{bot} \mid n :: \text{nat} . \text{True} \}$   
(*proof*)



```

lemma bnd-test:
  bnd x = --bnd x
  ⟨proof⟩

lemma bnd-upper:
   $\widehat{x}^n \ll \text{bot} \leq \text{bnd } x$ 
  ⟨proof⟩

lemma bnd-least:
  assumes  $\forall n . \widehat{x}^n \ll \text{bot} \leq -p$ 
  shows  $\text{bnd } x \leq -p$ 
  ⟨proof⟩

lemma mult-right-dist-bnd:
  assumes  $\forall n . (\widehat{x}^n \ll \text{bot}) * -p \leq -q$ 
  shows  $\text{bnd } x * -p \leq -q$ 
  ⟨proof⟩

lemma tests-complete:
  nat-test ( $\lambda n . (-p * x) \widehat{\phantom{x}}^n \ll \text{bot}$ ) (bnd( $-p * x$ ))
  ⟨proof⟩

end

end

```

## 32 Hoare Calculus and Modal Operators

```

theory Hoare-Modal

```

```

imports Stone-Kleene-Relation-Algebras.Kleene-Algebras Complete-Domain
Hoare Relative-Modal

```

```

begin

```

```

class box-precondition = relative-box-semiring + pre +

```

```

  assumes pre-def:  $x \ll p = |x|p$ 

```

```

begin

```

```

  Theorem 47

```

```

subclass precondition

```

```

  ⟨proof⟩

```

```

subclass precondition-test-test

```

```

  ⟨proof⟩

```

```

subclass precondition-promote

```

```

  ⟨proof⟩

```

**subclass** *precondition-test-box*

*<proof>*

**lemma** *pre-Z*:

$-p \leq x \ll -q \iff -p * x * -q \leq Z$

*<proof>*

**lemma** *pre-left-dist-add*:

$x \sqcup y \ll -q = (x \ll -q) * (y \ll -q)$

*<proof>*

**lemma** *pre-left-antitone*:

$x \leq y \implies y \ll -q \leq x \ll -q$

*<proof>*

**lemma** *pre-promote-neg*:

$(x \ll -q) * x * -q \leq Z$

*<proof>*

**lemma** *pre-pc-Z*:

$x \ll 1 = 1 \iff x * \text{bot} \leq Z$

*<proof>*

**proposition** *pre-sub-promote*:  $(x \ll -q) * x \leq (x \ll -q) * x * -q \sqcup Z$  **nitpick**

[*expect=genuine,card=6*] *<proof>*

**proposition** *pre-promote*:  $(x \ll -q) * x \sqcup Z = (x \ll -q) * x * -q \sqcup Z$  **nitpick**

[*expect=genuine,card=6*] *<proof>*

**proposition** *pre-mult-sub-promote*:  $(x * y \ll -q) * x \leq (x * y \ll -q) * x * (y \ll -q) \sqcup Z$

**nitpick** [*expect=genuine,card=6*] *<proof>*

**proposition** *pre-mult-promote*:  $(x * y \ll -q) * x * (y \ll -q) \sqcup Z = (x * y \ll -q) * x \sqcup Z$

**nitpick** [*expect=genuine,card=6*] *<proof>*

**end**

**class** *left-zero-box-precondition* = *box-precondition* +

*relative-left-zero-antidomain-semiring*

**begin**

**lemma** *pre-sub-promote*:

$(x \ll -q) * x \leq (x \ll -q) * x * -q \sqcup Z$

*<proof>*

**lemma** *pre-promote*:

$(x \ll -q) * x \sqcup Z = (x \ll -q) * x * -q \sqcup Z$

*<proof>*

**lemma** *pre-mult-sub-promote*:

$(x * y \ll -q) * x \leq (x * y \ll -q) * x * (y \ll -q) \sqcup Z$

```

    <proof>

lemma pre-mult-promote-sub:
   $(x*y\ll-q) * x * (y\ll-q) \leq (x*y\ll-q) * x$ 
  <proof>

lemma pre-mult-promote:
   $(x*y\ll-q) * x * (y\ll-q) \sqcup Z = (x*y\ll-q) * x \sqcup Z$ 
  <proof>

end

class diamond-precondition = relative-box-semiring + pre +
  assumes pre-def:  $x\ll p = |x>p$ 
begin
  Theorem 47

  subclass precondition
  <proof>

  subclass precondition-test-test
  <proof>

  subclass precondition-promote
  <proof>

  subclass precondition-test-diamond
  <proof>

  lemma pre-left-dist-add:
   $x \sqcup y \ll -q = (x \ll -q) \sqcup (y \ll -q)$ 
  <proof>

  lemma pre-left-isotone:
   $x \leq y \implies x \ll -q \leq y \ll -q$ 
  <proof>

end

class box-while = box-precondition + bounded-left-conway-semiring + ite + while
+
  assumes ite-def:  $x \triangleleft p \triangleright y = p * x \sqcup -p * y$ 
  assumes while-def:  $p \star x = (p * x)^\circ * -p$ 
begin

  subclass bounded-relative-antidomain-semiring <proof>

  lemma Z-circ-left-zero:
   $Z * x^\circ = Z$ 

```

```

    <proof>

subclass ifthenelse
  <proof>

    Theorem 48.1

subclass whiledo
  <proof>

lemma pre-while-1:
   $-p*(-p*x)\ll 1 = -p*x\ll 1$ 
  <proof>

lemma aL-one-circ:
   $aL = a(1^\circ*bot)$ 
  <proof>

end

class diamond-while = diamond-precondition + bounded-left-conway-semiring +
ite + while +
  assumes ite-def:  $x\langle p\rangle y = p * x \sqcup -p * y$ 
  assumes while-def:  $p*x = (p * x)^\circ * -p$ 
begin

subclass bounded-relative-antidomain-semiring <proof>

lemma Z-circ-left-zero:
   $Z * x^\circ = Z$ 
  <proof>

subclass ifthenelse
  <proof>

    Theorem 48.2

subclass whiledo
  <proof>

lemma aL-one-circ:
   $aL = d(1^\circ*bot)$ 
  <proof>

end

class box-while-program = box-while + atoms
begin

subclass while-program <proof>

```

```

end

class diamond-while-program = diamond-while + atoms
begin

subclass while-program ⟨proof⟩

end

class box-hoare-calculus = box-while-program + complete-antidomain-semiring
begin

subclass hoare-calculus ⟨proof⟩

end

class diamond-hoare-calculus = diamond-while-program +
complete-antidomain-semiring
begin

subclass hoare-calculus ⟨proof⟩

end

class box-hoare-sound = box-hoare-calculus + relative-domain-semiring-split +
left-kleene-conway-semiring +
  assumes aL-circ: aL * xo ≤ x*
begin

lemma aL-circ-ext:
  |x*]y ≤ |aL * xo]y
  ⟨proof⟩

lemma box-star-induct:
  assumes -p ≤ |x](-p)
  shows -p ≤ |x*](-p)
  ⟨proof⟩

lemma box-circ-induct:
  -p ≤ |x](-p) ⇒ -p*aL ≤ |xo](-p)
  ⟨proof⟩

lemma a-while-soundness:
  assumes -p*-q ≤ |x](-q)
  shows aL*-q ≤ |(-p*x)o*--p](-q)
  ⟨proof⟩

subclass hoare-calculus-sound
  ⟨proof⟩

```

**end**

**class** *diamond-hoare-sound* = *diamond-hoare-calculus* +  
*left-kleene-conway-semiring* +  
  **assumes** *aL-circ*:  $aL * x^\circ \leq x^*$   
**begin**

**lemma** *aL-circ-equal*:  
   $aL * x^\circ = aL * x^*$   
  ⟨*proof*⟩

**lemma** *aL-zero*:  
   $aL = \text{bot}$   
  ⟨*proof*⟩

**subclass** *hoare-calculus-sound*  
  ⟨*proof*⟩

**end**

**class** *box-hoare-complete* = *box-hoare-calculus* + *left-kleene-conway-semiring* +  
  **assumes** *box-circ-induct-2*:  $-p * |x|(-q) \leq -q \longrightarrow |x^\circ|(-p) \leq -q \sqcup aL$   
  **assumes** *aL-zero-or-one*:  $aL = \text{bot} \vee aL = 1$   
  **assumes** *while-mult-left-dist-Prod*:  $x \in \text{While-program} \wedge \text{descending-chain } t \wedge$   
   $\text{test-seq } t \longrightarrow x * \text{Prod } t = \text{Prod } (\lambda n . x * t \ n)$   
**begin**

**subclass** *hoare-calculus-complete*  
  ⟨*proof*⟩

**end**

**class** *diamond-hoare-complete* = *diamond-hoare-calculus* +  
*relative-domain-semiring-split* + *left-kleene-conway-semiring* +  
  **assumes** *dL-circ*:  $-aL * x^\circ \leq x^*$   
  **assumes** *aL-zero-or-one*:  $aL = \text{bot} \vee aL = 1$   
  **assumes** *while-mult-left-dist-Sum*:  $x \in \text{While-program} \wedge \text{ascending-chain } t \wedge$   
   $\text{test-seq } t \longrightarrow x * \text{Sum } t = \text{Sum } (\lambda n . x * t \ n)$   
**begin**

**lemma** *diamond-star-induct-var*:  
  **assumes**  $|x > (d \ p) \leq d \ p$   
  **shows**  $|x^* > (d \ p) \leq d \ p$   
  ⟨*proof*⟩

**lemma** *diamond-star-induct*:  
   $d \ q \sqcup |x > (d \ p) \leq d \ p \implies |x^* > (d \ q) \leq d \ p$   
  ⟨*proof*⟩

**lemma** *while-completeness-1*:  
**assumes**  $-p*(x\ll-q) \leq -q$   
**shows**  $-p*x\ll-q \leq -q\sqcup aL$   
 $\langle proof \rangle$

**subclass** *hoare-calculus-complete*  
 $\langle proof \rangle$

**end**

**class** *box-hoare-valid* = *box-hoare-sound* + *box-hoare-complete* + *hoare-triple* +  
**assumes** *hoare-triple-def*:  $p\{x\}q \longleftrightarrow p \leq |x]q$   
**begin**

[Theorem 49.2](#)

**subclass** *hoare-calculus-valid*  
 $\langle proof \rangle$

**lemma** *rule-skip-valid*:  
 $-p\{1\}-p$   
 $\langle proof \rangle$

**end**

**class** *diamond-hoare-valid* = *diamond-hoare-sound* + *diamond-hoare-complete* +  
*hoare-triple* +  
**assumes** *hoare-triple-def*:  $p\{x\}q \longleftrightarrow p \leq |x>q$   
**begin**

**lemma** *circ-star-equal*:  
 $x^\circ = x^*$   
 $\langle proof \rangle$

[Theorem 49.1](#)

**subclass** *hoare-calculus-valid*  
 $\langle proof \rangle$

**end**

**class** *diamond-hoare-sound-2* = *diamond-hoare-calculus* +  
*left-kleene-conway-semiring* +  
**assumes** *diamond-circ-induct-2*:  $--p*-q \leq |x>(-q) \longrightarrow aL*-q \leq |x^\circ>(-p)$   
**begin**

**subclass** *hoare-calculus-sound*  
 $\langle proof \rangle$

**end**

```

class diamond-hoare-valid-2 = diamond-hoare-sound-2 +
diamond-hoare-complete + hoare-triple +
  assumes hoare-triple-def:  $p\{x\}q \longleftrightarrow p \leq |x>q$ 
begin

subclass hoare-calculus-valid
  <proof>

end

end

```

### 33 Pre-Post Specifications

```

theory Pre-Post

```

```

imports Preconditions

```

```

begin

```

```

class pre-post =
  fixes pre-post :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (infix <math>\vdash</math> 55)

```

```

class pre-post-spec-greatest = bounded-idempotent-left-semiring + precondition +
pre-post +
  assumes pre-post-galois:  $-p \leq x\ll-q \longleftrightarrow x \leq -p\vdash-q$ 
begin

```

[Theorem 42.1](#)

```

lemma post-pre-left-antitone:
   $x \leq y \Longrightarrow y\ll-q \leq x\ll-q$ 
  <proof>

```

```

lemma pre-left-sub-dist:
   $x\sqcup y\ll-q \leq x\ll-q$ 
  <proof>

```

[Theorem 42.2](#)

```

lemma pre-post-left-antitone:
   $-p \leq -q \Longrightarrow -q\vdash-r \leq -p\vdash-r$ 
  <proof>

```

```

lemma pre-post-left-sub-dist:
   $-p\sqcup -q\vdash-r \leq -p\vdash-r$ 
  <proof>

```

```

lemma pre-post-left-sup-dist:
   $-p\vdash-r \leq -p*\ll-q\vdash-r$ 

```



*<proof>*

**Theorem 42.5**

**lemma** *pre-pre-post*:

$$x \leq (x \ll -p) \dashv -p$$

*<proof>*

**Theorem 42.6**

**lemma** *pre-post-pre*:

$$-p \leq (-p \dashv -q) \ll -q$$

*<proof>*

**Theorem 42.8**

**lemma** *pre-post-zero-top*:

$$\text{bot} \dashv -q = \text{top}$$

*<proof>*

**Theorem 42.7**

**lemma** *pre-post-pre-one*:

$$(1 \dashv -q) \ll -q = 1$$

*<proof>*

**Theorem 42.3**

**lemma** *pre-post-right-isotone*:

$$-p \leq -q \implies -r \dashv -p \leq -r \dashv -q$$

*<proof>*

**lemma** *pre-post-right-sub-dist*:

$$-r \dashv -p \leq -r \dashv -p \sqcup -q$$

*<proof>*

**lemma** *pre-post-right-sup-dist*:

$$-r \dashv -p * -q \leq -r \dashv -p$$

*<proof>*

**Theorem 42.7**

**lemma** *pre-post-reflexive*:

$$1 \leq -p \dashv -p$$

*<proof>*

**Theorem 42.9**

**lemma** *pre-post-compose*:

$$-q \leq -r \implies (-p \dashv -q) * (-r \dashv -s) \leq -p \dashv -s$$

*<proof>*

**Theorem 42.10**

**lemma** *pre-post-compose-1*:

$$(-p \dashv -q) * (-q \dashv -r) \leq -p \dashv -r$$

*<proof>*

[Theorem 42.11](#)

**lemma** *pre-post-compose-2*:

$$(-p\vdash p)*(-p\vdash q) = -p\vdash q$$

*<proof>*

[Theorem 42.12](#)

**lemma** *pre-post-compose-3*:

$$(-p\vdash q)*(-q\vdash q) = -p\vdash q$$

*<proof>*

[Theorem 42.13](#)

**lemma** *pre-post-compose-4*:

$$(-p\vdash p)*(-p\vdash p) = -p\vdash p$$

*<proof>*

[Theorem 42.14](#)

**lemma** *pre-post-one-one*:

$$x\ll 1 = 1 \iff x \leq 1\vdash 1$$

*<proof>*

[Theorem 42.4](#)

**lemma** *post-pre-left-dist-sup*:

$$x\sqcup y\ll -q = (x\ll -q)*(y\ll -q)$$

*<proof>*

**proposition** *pre-post-right-dist-sup*:  $-p\vdash q\sqcup -r = (-p\vdash q) \sqcup (-p\vdash r)$  **nitpick**  
[*expect=genuine,card=4*] *<proof>*

**end**

**class** *pre-post-spec-greatest-2* = *pre-post-spec-greatest* + *precondition-test-test*  
**begin**

**subclass** *precondition-test-box*

*<proof>*

**lemma** *pre-post-seq-sub-associative*:

$$(-p\vdash q)*-r \leq -p\vdash q*-r$$

*<proof>*

**lemma** *pre-post-right-import-mult*:

$$(-p\vdash q)*-r = (-p\vdash q*-r)*-r$$

*<proof>*

**lemma** *seq-pre-post-sub-associative*:

$$-r*(-p\vdash q) \leq --r\sqcup -p\vdash q$$

*<proof>*

**lemma** *pre-post-left-import-sup*:

$$-r*(-p\vdash q) = -r*(-r\sqcup -p\vdash q)$$

*<proof>*

**lemma** *pre-post-import-same:*

$$-p*(-p\vdash q) = -p*(1\vdash q)$$

*<proof>*

**lemma** *pre-post-import-complement:*

$$--p*(-p\vdash q) = --p*top$$

*<proof>*

**lemma** *pre-post-export:*

$$-p\vdash q = (1\vdash q) \sqcup --p*top$$

*<proof>*

**lemma** *pre-post-left-dist-mult:*

$$-p*-q\vdash r = (-p\vdash r) \sqcup (-q\vdash r)$$

*<proof>*

**lemma** *pre-post-left-import-mult:*

$$-r*(-p\vdash q) = -r*(-r*-p\vdash q)$$

*<proof>*

**lemma** *pre-post-right-import-sup:*

$$(-p\vdash q)*-r = (-p\vdash q\sqcup --r)*-r$$

*<proof>*

**lemma** *pre-post-shunting:*

$$x \leq -p*-q\vdash r \iff -p*x \leq -q\vdash r$$

*<proof>*

**proposition** *pre-post-right-dist-sup:*  $-p\vdash q\sqcup -r = (-p\vdash q) \sqcup (-p\vdash r)$  *<proof>*

**end**

**class** *left-zero-pre-post-spec-greatest-2* = *pre-post-spec-greatest-2* +  
*bounded-idempotent-left-zero-semiring*  
**begin**

**lemma** *pre-post-right-dist-sup:*

$$-p\vdash q\sqcup -r = (-p\vdash q) \sqcup (-p\vdash r)$$

*<proof>*

**end**

**class** *havoc* =  
**fixes** *H* :: 'a

**class** *idempotent-left-semiring-H* = *bounded-idempotent-left-semiring* + *havoc* +

```

assumes H-zero :  $H * bot = bot$ 
assumes H-split:  $x \leq x * bot \sqcup H$ 
begin

lemma H-galois:
   $x * bot \leq y \iff x \leq y \sqcup H$ 
   $\langle proof \rangle$ 

lemma H-greatest-finite:
   $x * bot = bot \iff x \leq H$ 
   $\langle proof \rangle$ 

lemma H-reflexive:
   $1 \leq H$ 
   $\langle proof \rangle$ 

lemma H-transitive:
   $H = H * H$ 
   $\langle proof \rangle$ 

lemma T-split-H:
   $top * bot \sqcup H = top$ 
   $\langle proof \rangle$ 

proposition H * (x \sqcup y) = H * x \sqcup H * y nitpick [expect=genuine,card=6]
   $\langle proof \rangle$ 

end

class pre-post-spec-least = bounded-idempotent-left-semiring +
  precondition-test-test + precondition-promote + pre-post +
  assumes test-mult-right-distr-sup:  $-p * (x \sqcup y) = -p * x \sqcup -p * y$ 
  assumes pre-post-galois:  $-p \leq x \ll -q \iff -p \dashv -q \leq x$ 
begin

lemma shunting-top:
   $-p * x \leq y \iff x \leq y \sqcup --p * top$ 
   $\langle proof \rangle$ 

lemma post-pre-left-isotone:
   $x \leq y \implies x \ll -q \leq y \ll -q$ 
   $\langle proof \rangle$ 

lemma pre-left-sub-dist:
   $x \ll -q \leq x \sqcup y \ll -q$ 
   $\langle proof \rangle$ 

lemma pre-post-left-isotone:
   $-p \leq -q \implies -p \dashv -r \leq -q \dashv -r$ 

```

$\langle proof \rangle$

**lemma** *pre-post-left-sub-dist*:

$$-p\vdash r \leq -p\sqcup -q\vdash r$$

$\langle proof \rangle$

**lemma** *pre-post-left-sup-dist*:

$$-p* -q\vdash r \leq -p\vdash r$$

$\langle proof \rangle$

**lemma** *pre-pre-post*:

$$(x\ll -p)\vdash p \leq x$$

$\langle proof \rangle$

**lemma** *pre-post-pre*:

$$-p \leq (-p\vdash -q)\ll -q$$

$\langle proof \rangle$

**lemma** *pre-post-zero-top*:

$$bot\vdash -q = bot$$

$\langle proof \rangle$

**lemma** *pre-post-pre-one*:

$$(1\vdash -q)\ll -q = 1$$

$\langle proof \rangle$

**lemma** *pre-post-right-antitone*:

$$-p \leq -q \implies -r\vdash -q \leq -r\vdash -p$$

$\langle proof \rangle$

**lemma** *pre-post-right-sub-dist*:

$$-r\vdash -p\sqcup -q \leq -r\vdash -p$$

$\langle proof \rangle$

**lemma** *pre-post-right-sup-dist*:

$$-r\vdash -p \leq -r\vdash -p* -q$$

$\langle proof \rangle$

**lemma** *pre-top*:

$$top\ll -q = 1$$

$\langle proof \rangle$

**lemma** *pre-mult-top-increasing*:

$$-p \leq -p*top\ll -q$$

$\langle proof \rangle$

**lemma** *pre-post-below-mult-top*:

$$-p\vdash -q \leq -p*top$$

$\langle proof \rangle$

**lemma** *pre-post-import-complement*:

$$\begin{aligned} & \neg\neg p * (\neg p \dashv\vdash q) = \text{bot} \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-import-same*:

$$\begin{aligned} & \neg p * (\neg p \dashv\vdash q) = \neg p \dashv\vdash q \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-export*:

$$\begin{aligned} & \neg p \dashv\vdash q = \neg p * (1 \dashv\vdash q) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-seq-associative*:

$$\begin{aligned} & \neg r * (\neg p \dashv\vdash q) = \neg r * \neg p \dashv\vdash q \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-left-import-mult*:

$$\begin{aligned} & \neg r * (\neg p \dashv\vdash q) = \neg r * (\neg r * \neg p \dashv\vdash q) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *seq-pre-post-sub-associative*:

$$\begin{aligned} & \neg r * (\neg p \dashv\vdash q) \leq \neg\neg r \sqcup \neg p \dashv\vdash q \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-left-import-sup*:

$$\begin{aligned} & \neg r * (\neg p \dashv\vdash q) = \neg r * (\neg\neg r \sqcup \neg p \dashv\vdash q) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-left-dist-sup*:

$$\begin{aligned} & \neg p \sqcup \neg q \dashv\vdash r = (\neg p \dashv\vdash r) \sqcup (\neg q \dashv\vdash r) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-reflexive*:

$$\begin{aligned} & \neg p \dashv\vdash p \leq 1 \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-compose*:

$$\begin{aligned} & \neg q \leq \neg r \implies \neg p \dashv\vdash s \leq (\neg p \dashv\vdash q) * (\neg r \dashv\vdash s) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-compose-1*:

$$\begin{aligned} & \neg p \dashv\vdash r \leq (\neg p \dashv\vdash q) * (\neg q \dashv\vdash r) \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-compose-2*:

$$\begin{aligned} & (\neg p \dashv\vdash p) * (\neg p \dashv\vdash q) = \neg p \dashv\vdash q \\ & \langle \text{proof} \rangle \end{aligned}$$

**lemma** *pre-post-compose-3*:

$$(-p\vdash q)*(-q\vdash q) = -p\vdash q$$

*<proof>*

**lemma** *pre-post-compose-4*:

$$(-p\vdash p)*(-p\vdash p) = -p\vdash p$$

*<proof>*

**lemma** *pre-post-one-one*:

$$x\ll 1 = 1 \longleftrightarrow 1\vdash 1 \leq x$$

*<proof>*

**lemma** *pre-one-right*:

$$-p\ll 1 = -p$$

*<proof>*

**lemma** *pre-pre-one*:

$$x\ll -q = x*-q\ll 1$$

*<proof>*

**subclass** *precondition-test-diamond*

*<proof>*

**proposition** *pre-post-shunting*:  $x \leq -p*-q\vdash r \longleftrightarrow -p*x \leq -q\vdash r$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $(-p\vdash q)*-r = (-p\vdash q\sqcup -r)*-r$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $(-p\vdash q)*-r = (-p\vdash q\sqcup --r)*-r$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $(-p\vdash q)*-r = (-p\vdash q*-r)*-r$  **nitpick** [*expect=genuine,card=3*]

*<proof>*

**proposition**  $(-p\vdash q)*-r = (-p\vdash q*-r)*-r$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $-p\vdash q\sqcup -r = (-p\vdash q) \sqcup (-p\vdash r)$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition**  $-p\vdash q\sqcup -r = (-p\vdash q) * (-p\vdash r)$  **nitpick**

[*expect=genuine,card=3*] *<proof>*

**proposition** *pre-post-right-dist-mult*:  $-p\vdash q*-r = (-p\vdash q) * (-p\vdash r)$  *<proof>*

**proposition** *pre-post-right-dist-mult*:  $-p\vdash q*-r = (-p\vdash q) \sqcup (-p\vdash r)$  *<proof>*

**proposition** *post-pre-left-dist-sup*:  $x\sqcup y\ll -q = (x\ll -q) \sqcup (y\ll -q)$  *<proof>*

**end**

**class** *havoc-dual* =

**fixes** *Hd* :: 'a

**class** *idempotent-left-semiring-Hd* = *bounded-idempotent-left-semiring* +  
*havoc-dual* +

**assumes** *Hd-total*: *Hd* \* *top* = *top*

**assumes** *Hd-least*:  $x * top = top \longrightarrow Hd \leq x$   
**begin**

**lemma** *Hd-least-total*:  
 $x * top = top \longleftrightarrow Hd \leq x$   
 $\langle proof \rangle$

**lemma** *Hd-reflexive*:  
 $Hd \leq 1$   
 $\langle proof \rangle$

**lemma** *Hd-transitive*:  
 $Hd = Hd * Hd$   
 $\langle proof \rangle$

**end**

**class** *pre-post-spec-least-Hd* = *idempotent-left-semiring-Hd* + *pre-post-spec-least*  
+  
**assumes** *pre-one-mult-top*:  $(x \ll 1) * top = x * top$   
**begin**

**lemma** *Hd-pre-one*:  
 $Hd \ll 1 = 1$   
 $\langle proof \rangle$

**lemma** *pre-post-below-Hd*:  
 $1 \dashv 1 \leq Hd$   
 $\langle proof \rangle$

**lemma** *Hd-pre-post*:  
 $Hd = 1 \dashv 1$   
 $\langle proof \rangle$

**lemma** *top-left-zero*:  
 $top * x = top$   
 $\langle proof \rangle$

**lemma** *test-dual-test*:  
 $(-p \sqcup -p * top) * -p = -p \sqcup -p * top$   
 $\langle proof \rangle$

**lemma** *pre-zero-mult-top*:  
 $(x \ll bot) * top = x * bot$   
 $\langle proof \rangle$

**lemma** *pre-one-mult-Hd*:  
 $(x \ll 1) * Hd \leq x$   
 $\langle proof \rangle$



**lemma** *Hd-mult-pre-one*:

$$Hd*(x \ll 1) \leq x$$

*<proof>*

**lemma** *pre-post-one-def-1*:

**assumes**  $1 \leq x \ll -q$

**shows**  $Hd*(-q \sqcup - -q*top) \leq x$

*<proof>*

**lemma** *pre-post-one-def*:

$$1 \dashv -q = Hd*(-q \sqcup - -q*top)$$

*<proof>*

**lemma** *pre-post-def*:

$$-p \dashv -q = -p*Hd*(-q \sqcup - -q*top)$$

*<proof>*

**end**

**end**

## 34 Pre-Post Specifications and Modal Operators

**theory** *Pre-Post-Modal*

**imports** *Pre-Post Hoare-Modal*

**begin**

**class** *pre-post-spec-whiledo* = *pre-post-spec-greatest* + *whiledo*

**begin**

**lemma** *nat-test-pre-post*:

$$\begin{aligned} \text{nat-test } t \ s \implies -q \leq s \implies (\forall n . x \leq t \ n* -p* -q \dashv (pSum \ t \ n* -q)) \implies -p*x \\ \leq -q \dashv - -p* -q \end{aligned}$$

*<proof>*

**lemma** *nat-test-pre-post-2*:

$$\text{nat-test } t \ s \implies -r \leq s \implies (\forall n . x \leq t \ n* -p \dashv (pSum \ t \ n)) \implies -p*x \leq -r \dashv 1$$

*<proof>*

**end**

**class** *pre-post-spec-hoare* = *pre-post-spec-whiledo* + *hoare-calculus-sound*

**begin**

**lemma** *pre-post-while*:

$$x \leq -p* -q \dashv -q \implies -p*x \leq aL* -q \dashv -q$$

*<proof>*

### Theorem 43.1

**lemma** *while-soundness-3*:

$test\text{-}seq\ t \implies -q \leq Sum\ t \implies x \leq t \ 0 * -p * -q \dashv\!\! \dashv L * -q \implies (\forall n > 0 . x \leq t$

$n * -p * -q \dashv\!\! \dashv p Sum\ t\ n * -q) \implies -p * x \leq -q \dashv\!\! \dashv -p * -q$

*<proof>*

### Theorem 43.2

**lemma** *while-soundness-4*:

$test\text{-}seq\ t \implies -r \leq Sum\ t \implies (\forall n . x \leq t\ n * -p \dashv\!\! \dashv p Sum\ t\ n) \implies -p * x \leq -r \dashv\!\! \dashv 1$

*<proof>*

**end**

**class** *pre-post-spec-hoare-pc-2* = *pre-post-spec-hoare* + *hoare-calculus-pc-2*

**begin**

### Theorem 43.3

**lemma** *pre-post-while-pc*:

$x \leq -p * -q \dashv\!\! \dashv -q \implies -p * x \leq -q \dashv\!\! \dashv -p * -q$

*<proof>*

**end**

**class** *pre-post-spec-hoare-pc* = *pre-post-spec-hoare* + *hoare-calculus-pc*

**begin**

**subclass** *pre-post-spec-hoare-pc-2* *<proof>*

**lemma** *pre-post-one-one-top*:

$1 \dashv\!\! \dashv 1 = top$

*<proof>*

**end**

**class** *pre-post-spec-H* = *pre-post-spec-greatest* + *box-precondition* + *havoc* +

**assumes** *H-zero-2*:  $H * bot = bot$

**assumes** *H-split-2*:  $x \leq x * -q * top \sqcup H * --q$

**begin**

**subclass** *idempotent-left-semiring-H*

*<proof>*

**lemma** *pre-post-def-iff*:

$-p * x * --q \leq Z \iff x \leq Z \sqcup --p * top \sqcup H * -q$

*<proof>*

**lemma** *pre-post-def*:

```

    -p⊥-q = Z ⊔ --p*top ⊔ H*-q
    ⟨proof⟩

```

**end**

```

class pre-post-L = pre-post-spec-greatest + box-while + left-conway-semiring-L +
left-kleene-conway-semiring +

```

```

    assumes circ-below-L-add-star: x° ≤ L ⊔ x*

```

```

begin

```

```

    a loop does not abort if its body does not abort

```

```

    this avoids abortion from all states* alternatively from states in -r if -r
    is an invariant

```

```

lemma body-abort-loop:

```

```

    assumes Z = L

```

```

    and x ≤ -p⊥1

```

```

    shows -p*x ≤ 1⊥1

```

```

    ⟨proof⟩

```

**end**

```

class pre-post-spec-Hd = pre-post-spec-least + diamond-precondition +
idempotent-left-semiring-Hd +

```

```

    assumes d-mult-top: d(x) * top = x * top

```

```

begin

```

```

subclass pre-post-spec-least-Hd

```

```

    ⟨proof⟩

```

**end**

**end**

## 35 Monotonic Boolean Transformers

```

theory Monotonic-Boolean-Transformers

```

```

imports MonoBoolTranAlgebra.Assertion-Algebra Base

```

```

begin

```

```

no-notation inf (infixl <⊔> 70)

```

```

unbundle no uminus-syntax

```

```

context mbt-algebra

```

```

begin

```

```

lemma directed-left-mult:

```

*directed*  $Y \implies \text{directed } ((*) x \cdot Y)$   
*<proof>*

**lemma** *neg-assertion*:  
 $\text{neg-assert } x \in \text{assertion}$   
*<proof>*

**lemma** *assertion-neg-assert*:  
 $x \in \text{assertion} \iff x = \text{neg-assert } (\text{neg-assert } x)$   
*<proof>*

extend and dualise part of Viorel Preoteasa's theory

**definition** *assumption*  $\equiv \{x \mid 1 \leq x \wedge (x * \text{bot}) \sqcup (x \hat{\ } o) = x\}$

**definition** *neg-assume*  $(x::'a) \equiv (x \hat{\ } o * \text{top}) \sqcup 1$

**lemma** *neg-assume-assert*:  
 $\text{neg-assume } x = (\text{neg-assert } (x \hat{\ } o)) \hat{\ } o$   
*<proof>*

**lemma** *assert-iff-assume*:  
 $x \in \text{assertion} \iff x \hat{\ } o \in \text{assumption}$   
*<proof>*

**lemma** *assertion-iff-assumption-subseteq*:  
 $X \subseteq \text{assertion} \iff \text{dual } X \subseteq \text{assumption}$   
*<proof>*

**lemma** *assumption-iff-assertion-subseteq*:  
 $X \subseteq \text{assumption} \iff \text{dual } X \subseteq \text{assertion}$   
*<proof>*

**lemma** *assumption-prop*:  
 $x \in \text{assumption} \implies (x * \text{bot}) \sqcup 1 = x$   
*<proof>*

**lemma** *neg-assumption*:  
 $\text{neg-assume } x \in \text{assumption}$   
*<proof>*

**lemma** *assumption-neg-assume*:  
 $x \in \text{assumption} \iff x = \text{neg-assume } (\text{neg-assume } x)$   
*<proof>*

**lemma** *assumption-sup-comp-eq*:  
 $x \in \text{assumption} \implies y \in \text{assumption} \implies x \sqcup y = x * y$   
*<proof>*

**lemma** *sup-uminus-assume[simp]*:

$x \in \text{assumption} \implies x \sqcap \text{neg-assume } x = 1$   
(proof)

**lemma** *inf-uminus-assume*[simp]:  
 $x \in \text{assumption} \implies x \sqcup \text{neg-assume } x = \text{top}$   
(proof)

**lemma** *uminus-assumption*[simp]:  
 $x \in \text{assumption} \implies \text{neg-assume } x \in \text{assumption}$   
(proof)

**lemma** *uminus-uminus-assume*[simp]:  
 $x \in \text{assumption} \implies \text{neg-assume } (\text{neg-assume } x) = x$   
(proof)

**lemma** *sup-assumption*[simp]:  
 $x \in \text{assumption} \implies y \in \text{assumption} \implies x \sqcup y \in \text{assumption}$   
(proof)

**lemma** *comp-assumption*[simp]:  
 $x \in \text{assumption} \implies y \in \text{assumption} \implies x * y \in \text{assumption}$   
(proof)

**lemma** *inf-assumption*[simp]:  
 $x \in \text{assumption} \implies y \in \text{assumption} \implies x \sqcap y \in \text{assumption}$   
(proof)

**lemma** *assumption-comp-idempotent*[simp]:  
 $x \in \text{assumption} \implies x * x = x$   
(proof)

**lemma** *assumption-comp-idempotent-dual*[simp]:  
 $x \in \text{assumption} \implies (x \hat{\ } o) * (x \hat{\ } o) = x \hat{\ } o$   
(proof)

**lemma** *top-assumption*[simp]:  
 $\text{top} \in \text{assumption}$   
(proof)

**lemma** *one-assumption*[simp]:  
 $1 \in \text{assumption}$   
(proof)

**lemma** *assert-top*:  
 $\text{neg-assert } (\text{neg-assert } p) \hat{\ } o * \text{bot} = \text{neg-assert } p * \text{top}$   
(proof)

**lemma** *assume-bot*:  
 $\text{neg-assume } (\text{neg-assume } p) \hat{\ } o * \text{top} = \text{neg-assume } p * \text{bot}$

*<proof>*

**definition**  $wpb\ x \equiv (x * bot) \sqcup 1$

**lemma** *wpt-iff-wpb*:

$$wpb\ x = wpt\ (x \wedge o) \wedge o$$

*<proof>*

**lemma** *wpb-is-assumption[simp]*:

$$wpb\ x \in assumption$$

*<proof>*

**lemma** *wpb-comp*:

$$(wpb\ x) * x = x$$

*<proof>*

**lemma** *wpb-comp-2*:

$$wpb\ (x * y) = wpb\ (x * (wpb\ y))$$

*<proof>*

**lemma** *wpb-assumption[simp]*:

$$x \in assumption \implies wpb\ x = x$$

*<proof>*

**lemma** *wpb-choice*:

$$wpb\ (x \sqcup y) = wpb\ x \sqcup wpb\ y$$

*<proof>*

**lemma** *wpb-dual-assumption*:

$$x \in assumption \implies wpb\ (x \wedge o) = 1$$

*<proof>*

**lemma** *wpb-mono*:

$$x \leq y \implies wpb\ x \leq wpb\ y$$

*<proof>*

**lemma** *assumption-disjunctive*:

$$x \in assumption \implies x \in disjunctive$$

*<proof>*

**lemma** *assumption-conjunctive*:

$$x \in assumption \implies x \in conjunctive$$

*<proof>*

**lemma** *wpb-le-assumption*:

$$x \in assumption \implies x * y = y \implies x \leq wpb\ y$$

*<proof>*

**definition** *dual-omega* :: 'a  $\Rightarrow$  'a  $(\langle(- \wedge \mathcal{U})\rangle$  [81] 80)

**where**  $(x \hat{\cup}) \equiv (((x \hat{o}) \hat{\omega}) \hat{o})$

**lemma** *dual-omega-fix*:

$x \hat{\cup} = (x * (x \hat{\cup})) \sqcup 1$   
*<proof>*

**lemma** *dual-omega-comp-fix*:

$x \hat{\cup} * y = (x * (x \hat{\cup}) * y) \sqcup y$   
*<proof>*

**lemma** *dual-omega-greatest*:

$z \leq (x * z) \sqcup y \implies z \leq (x \hat{\cup}) * y$   
*<proof>*

**end**

**context** *post-mbt-algebra*

**begin**

**lemma** *post-antitone*:

**assumes**  $x \leq y$   
**shows**  $\text{post } y \leq \text{post } x$   
*<proof>*

**lemma** *post-assumption-below-one*:

$q \in \text{assumption} \implies \text{post } q \leq \text{post } 1$   
*<proof>*

**lemma** *post-assumption-above-one*:

$q \in \text{assumption} \implies \text{post } 1 \leq \text{post } (q \hat{o})$   
*<proof>*

**lemma** *post-assumption-below-dual*:

$q \in \text{assumption} \implies \text{post } q \leq \text{post } (q \hat{o})$   
*<proof>*

**lemma** *assumption-assertion-absorb*:

$q \in \text{assumption} \implies q * (q \hat{o}) = q$   
*<proof>*

**lemma** *post-dual-below-post-one*:

**assumes**  $q \in \text{assumption}$   
**shows**  $\text{post } (q \hat{o}) \leq \text{post } 1 * q$   
*<proof>*

**lemma** *post-below-post-one*:

$q \in \text{assumption} \implies \text{post } q \leq \text{post } 1 * q$   
*<proof>*

**end**

**context** *complete-mbt-algebra*  
**begin**

**lemma** *Inf-assumption[simp]*:  
 $X \subseteq \text{assumption} \implies \text{Inf } X \in \text{assumption}$   
(*proof*)

**definition** *continuous*  $x \equiv (\forall Y . \text{directed } Y \longrightarrow x * (\text{SUP } y \in Y . y) = (\text{SUP } y \in Y . x * y))$

**definition** *Continuous*  $\equiv \{ x . \text{continuous } x \}$

**lemma** *continuous-Continuous*:  
 $\text{continuous } x \longleftrightarrow x \in \text{Continuous}$   
(*proof*)

[Theorem 53.1](#)

**lemma** *one-continuous*:  
 $1 \in \text{Continuous}$   
(*proof*)

**lemma** *continuous-dist-ascending-chain*:  
**assumes**  $x \in \text{Continuous}$   
**and** *ascending-chain*  $f$   
**shows**  $x * (\text{SUP } n :: \text{nat} . f n) = (\text{SUP } n :: \text{nat} . x * f n)$   
(*proof*)

[Theorem 53.1](#)

**lemma** *assertion-continuous*:  
**assumes**  $x \in \text{assertion}$   
**shows**  $x \in \text{Continuous}$   
(*proof*)

[Theorem 53.1](#)

**lemma** *assumption-continuous*:  
**assumes**  $x \in \text{assumption}$   
**shows**  $x \in \text{Continuous}$   
(*proof*)

[Theorem 53.1](#)

**lemma** *mult-continuous*:  
**assumes**  $x \in \text{Continuous}$   
**and**  $y \in \text{Continuous}$   
**shows**  $x * y \in \text{Continuous}$   
(*proof*)

[Theorem 53.1](#)



**lemma** *sup-continuous*:

$x \in \text{Continuous} \implies y \in \text{Continuous} \implies x \sqcup y \in \text{Continuous}$

*<proof>*

[Theorem 53.1](#)

**lemma** *inf-continuous*:

**assumes**  $x \in \text{Continuous}$

**and**  $y \in \text{Continuous}$

**shows**  $x \sqcap y \in \text{Continuous}$

*<proof>*

[Theorem 53.1](#)

**lemma** *dual-star-continuous*:

**assumes**  $x \in \text{Continuous}$

**shows**  $x \hat{\otimes} \in \text{Continuous}$

*<proof>*

[Theorem 53.1](#)

**lemma** *omega-continuous*:

**assumes**  $x \in \text{Continuous}$

**shows**  $x \hat{\omega} \in \text{Continuous}$

*<proof>*

**definition** *co-continuous*  $x \equiv (\forall Y . \text{co-directed } Y \longrightarrow x * (\text{INF } y \in Y . y) = (\text{INF } y \in Y . x * y))$

**definition** *Co-continuous*  $\equiv \{ x . \text{co-continuous } x \}$

**lemma** *directed-dual*:

$\text{directed } X \longleftrightarrow \text{co-directed } (\text{dual } ' X)$

*<proof>*

**lemma** *dual-dual-image*:

$\text{dual } ' \text{dual } ' X = X$

*<proof>*

**lemma** *continuous-dual*:

$\text{continuous } x \longleftrightarrow \text{co-continuous } (x \hat{o})$

*<proof>*

**lemma** *co-continuous-Co-continuous*:

$\text{co-continuous } x \longleftrightarrow x \in \text{Co-continuous}$

*<proof>*

[Theorem 53.1](#) and [Theorem 53.2](#)

**lemma** *Continuous-dual*:

$x \in \text{Continuous} \longleftrightarrow x \hat{o} \in \text{Co-continuous}$

*<proof>*

[Theorem 53.2](#)

**lemma** *one-co-continuous*:

$1 \in Co\text{-continuous}$

$\langle proof \rangle$

**lemma** *ascending-chain-dual*:

$ascending\text{-chain } f \longleftrightarrow descending\text{-chain } (dual \circ f)$

$\langle proof \rangle$

**lemma** *co-continuous-dist-descending-chain*:

**assumes**  $x \in Co\text{-continuous}$

**and**  $descending\text{-chain } f$

**shows**  $x * (INF n::nat . f n) = (INF n::nat . x * f n)$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *assertion-co-continuous*:

$x \in assertion \implies x \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *assumption-co-continuous*:

$x \in assumption \implies x \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *mult-co-continuous*:

$x \in Co\text{-continuous} \implies y \in Co\text{-continuous} \implies x * y \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *sup-co-continuous*:

$x \in Co\text{-continuous} \implies y \in Co\text{-continuous} \implies x \sqcup y \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *inf-co-continuous*:

$x \in Co\text{-continuous} \implies y \in Co\text{-continuous} \implies x \sqcap y \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *dual-omega-co-continuous*:

$x \in Co\text{-continuous} \implies x \hat{\cup} \in Co\text{-continuous}$

$\langle proof \rangle$

[Theorem 53.2](#)

**lemma** *star-co-continuous*:

$x \in Co\text{-continuous} \implies x \hat{*} \in Co\text{-continuous}$

$\langle proof \rangle$

```

lemma dual-omega-iterate:
  assumes  $y \in Co\text{-continuous}$ 
  shows  $y \hat{\cup} * z = (INF\ n::nat . ((\lambda x . y * x \sqcup z) \hat{\ } n)\ top)$ 
  <proof>

lemma dual-omega-iterate-one:
   $y \in Co\text{-continuous} \implies y \hat{\cup} = (INF\ n::nat . ((\lambda x . y * x \sqcup 1) \hat{\ } n)\ top)$ 
  <proof>

subclass ccpo
  <proof>

end

class post-mbt-algebra-ext = post-mbt-algebra +
  assumes post-sub-fusion:  $post\ 1 * neg\text{-assume}\ q \leq post\ (neg\text{-assume}\ q \hat{\ } o)$ 
begin

lemma post-fusion:
   $post\ (neg\text{-assume}\ q \hat{\ } o) = post\ 1 * neg\text{-assume}\ q$ 
  <proof>

lemma post-dual-post-one:
   $q \in assumption \implies post\ 1 * q \leq post\ (q \hat{\ } o)$ 
  <proof>

end

instance MonoTran :: (complete-boolean-algebra) post-mbt-algebra-ext
  <proof>

class complete-mbt-algebra-ext = complete-mbt-algebra + post-mbt-algebra-ext

instance MonoTran :: (complete-boolean-algebra) complete-mbt-algebra-ext
  <proof>

end

```

## 36 Instances of Monotonic Boolean Transformers

**theory** *Monotonic-Boolean-Transformers-Instances*

**imports** *Monotonic-Boolean-Transformers Pre-Post-Modal*  
*General-Refinement-Algebras*

**begin**

**sublocale** *mbt-algebra* < *mbta*: *bounded-idempotent-left-semiring*  
*<proof>*

**sublocale** *mbt-algebra* < *mbta-dual: bounded-idempotent-left-semiring* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *top* = *bot*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta: bounded-general-refinement-algebra* **where** *star* = *dual-star* **and** *Omega* = *dual-omega*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta-dual: bounded-general-refinement-algebra* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *Omega* = *omega* **and** *top* = *bot*  $\langle$ *proof* $\rangle$

[Theorem 50.9\(b\)](#)

**sublocale** *mbt-algebra* < *mbta: left-conway-semiring-L* **where** *circ* = *dual-star* **and** *L* = *bot*  $\langle$ *proof* $\rangle$

[Theorem 50.8\(a\)](#)

**sublocale** *mbt-algebra* < *mbta-dual: left-conway-semiring-L* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *L* = *bot*  $\langle$ *proof* $\rangle$

[Theorem 50.8\(b\)](#)

**sublocale** *mbt-algebra* < *mbta-fix: left-conway-semiring-L* **where** *circ* = *dual-omega* **and** *L* = *top*  $\langle$ *proof* $\rangle$

[Theorem 50.9\(a\)](#)

**sublocale** *mbt-algebra* < *mbta-fix-dual: left-conway-semiring-L* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *L* = *top*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta: left-kleene-conway-semiring* **where** *circ* = *dual-star* **and** *star* = *dual-star*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta-dual: left-kleene-conway-semiring* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta-fix: left-kleene-conway-semiring* **where** *circ* = *dual-omega* **and** *star* = *dual-star*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta-fix-dual: left-kleene-conway-semiring* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top*  $\langle$ *proof* $\rangle$

**sublocale** *mbt-algebra* < *mbta*: tests **where** *uminus* = *neg-assert*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta-dual*: tests **where** *less* = *greater* **and** *less-eq* =  
*greater-eq* **and** *sup* = *inf* **and** *uminus* = *neg-assume* **and** *bot* = *top*  
 ⟨proof⟩

### Theorem 51.2

**sublocale** *mbt-algebra* < *mbta*: *bounded-relative-antidomain-semiring* **where** *d* =  
 $\lambda x . (x * top) \sqcap 1$  **and** *uminus* = *neg-assert* **and** *Z* = *bot*  
 ⟨proof⟩

### Theorem 51.1

**sublocale** *mbt-algebra* < *mbta-dual*: *bounded-relative-antidomain-semiring*  
**where** *d* =  $\lambda x . (x * bot) \sqcup 1$  **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and**  
*sup* = *inf* **and** *uminus* = *neg-assume* **and** *bot* = *top* **and** *top* = *bot* **and** *Z* = *top*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta*: *relative-domain-semiring-split* **where** *d* =  $\lambda x . (x$   
 $* top) \sqcap 1$  **and** *Z* = *bot*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta-dual*: *relative-domain-semiring-split* **where** *d* =  
 $\lambda x . (x * bot) \sqcup 1$  **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf*  
**and** *bot* = *top* **and** *Z* = *top*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta*: *diamond-while* **where** *box* =  $\lambda x y . neg-assert (x$   
 $* neg-assert y)$  **and** *circ* = *dual-star* **and** *d* =  $\lambda x . (x * top) \sqcap 1$  **and** *diamond* =  
 $\lambda x y . (x * y * top) \sqcap 1$  **and** *ite* =  $\lambda x p y . (p * x) \sqcup (neg-assert p * y)$  **and** *pre*  
 =  $\lambda x y . wpt (x * y)$  **and** *uminus* = *neg-assert* **and** *while* =  $\lambda p x . ((p * x) \hat{\ } \otimes)$   
 $* neg-assert p$  **and** *Z* = *bot*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta-dual*: *box-while* **where** *box* =  $\lambda x y . neg-assume$   
 $(x * neg-assume y)$  **and** *circ* = *omega* **and** *d* =  $\lambda x . (x * bot) \sqcup 1$  **and** *diamond*  
 =  $\lambda x y . (x * y * bot) \sqcup 1$  **and** *ite* =  $\lambda x p y . (p * x) \sqcap (neg-assume p * y)$  **and**  
*less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* =  $\lambda x y . wpb (x \hat{\ }$   
 $o * y)$  **and** *uminus* = *neg-assume* **and** *while* =  $\lambda p x . ((p * x) \hat{\ } \omega) * neg-assume$   
 $p$  **and** *bot* = *top* **and** *top* = *bot* **and** *Z* = *top*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta-fix*: *diamond-while* **where** *box* =  $\lambda x y . neg-assert$   
 $(x * neg-assert y)$  **and** *circ* = *dual-omega* **and** *d* =  $\lambda x . (x * top) \sqcap 1$  **and**  
*diamond* =  $\lambda x y . (x * y * top) \sqcap 1$  **and** *ite* =  $\lambda x p y . (p * x) \sqcup (neg-assert p * y)$   
**and** *pre* =  $\lambda x y . wpt (x * y)$  **and** *uminus* = *neg-assert* **and** *while* =  $\lambda p x .$   
 $((p * x) \hat{\ } \cup) * neg-assert p$  **and** *Z* = *bot*  
 ⟨proof⟩

**sublocale** *mbt-algebra* < *mbta-fix-dual*: *box-while* **where**  $box = \lambda x y .$   
*neg-assume*  $(x * \text{neg-assume } y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  
*diamond*  $= \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (\text{neg-assume } p * y)$   
**and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y .$   
 $wpb (x \hat{o} * y)$  **and**  $uminus = \text{neg-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{*}) *$   
*neg-assume*  $p$  **and**  $bot = top$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨*proof*⟩

**sublocale** *mbt-algebra* < *mbta-pre*: *box-while* **where**  $box = \lambda x y .$  *neg-assert*  $(x * \text{neg-assert } y)$   
**and**  $circ = dual-star$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and** *diamond*  $= \lambda x y . (x * y * top) \sqcap 1$   
**and**  $ite = \lambda x p y . (p * x) \sqcup (\text{neg-assert } p * y)$  **and**  $pre = \lambda x y . wpt (x \hat{o} * y)$   
**and**  $uminus = \text{neg-assert}$  **and**  $while = \lambda p x . ((p * x) \hat{\otimes}) * \text{neg-assert } p$  **and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *mbt-algebra* < *mbta-pre-dual*: *diamond-while* **where**  $box = \lambda x y .$   
*neg-assume*  $(x * \text{neg-assume } y)$  **and**  $circ = omega$  **and**  $d = \lambda x . (x * bot) \sqcup 1$   
**and** *diamond*  $= \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (\text{neg-assume } p * y)$   
**and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$   
**and**  $pre = \lambda x y . wpb (x * y)$  **and**  $uminus = \text{neg-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{\omega}) * \text{neg-assume } p$   
**and**  $bot = top$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨*proof*⟩

**sublocale** *mbt-algebra* < *mbta-pre-fix*: *box-while* **where**  $box = \lambda x y .$  *neg-assert*  $(x * \text{neg-assert } y)$   
**and**  $circ = dual-omega$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  
*diamond*  $= \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (\text{neg-assert } p * y)$   
**and**  $pre = \lambda x y . wpt (x \hat{o} * y)$  **and**  $uminus = \text{neg-assert}$  **and**  $while = \lambda p x . ((p * x) \hat{\cup}) * \text{neg-assert } p$   
**and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *mbt-algebra* < *mbta-pre-fix-dual*: *diamond-while* **where**  $box = \lambda x y .$   
*neg-assume*  $(x * \text{neg-assume } y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  
*diamond*  $= \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (\text{neg-assume } p * y)$   
**and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y .$   
 $wpb (x * y)$  **and**  $uminus = \text{neg-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{*}) *$   
*neg-assume*  $p$  **and**  $bot = top$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨*proof*⟩

**sublocale** *post-mbt-algebra* < *mbta*: *pre-post-spec-Hd* **where**  $box = \lambda x y .$   
*neg-assert*  $(x * \text{neg-assert } y)$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and** *diamond*  $= \lambda x y .$   
 $(x * y * top) \sqcap 1$  **and**  $pre = \lambda x y . wpt (x * y)$  **and**  $pre-post = \lambda p q . p * post q$   
**and**  $uminus = \text{neg-assert}$  **and**  $Hd = post 1$  **and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *post-mbt-algebra* < *mbta-dual*: *pre-post-spec-H* **where**  $box = \lambda x y .$   
*neg-assume*  $(x * \text{neg-assume } y)$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and** *diamond*  $= \lambda x y .$   
 $(x * y * bot) \sqcup 1$  **and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$   
**and**  $pre = \lambda x y . wpb (x \hat{o} * y)$  **and**  $pre-post = \lambda p q . (p \hat{o}) * post (q \hat{o})$

**and**  $uminus = neg\text{-}assume$  **and**  $bot = top$  **and**  $H = post\ 1$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}pre$ :  $pre\text{-}post\text{-}spec\text{-}H$  **where**  $box = \lambda x\ y . neg\text{-}assert\ (x * neg\text{-}assert\ y)$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x\ y . (x * y * top) \sqcap 1$  **and**  $pre = \lambda x\ y . wpt\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . p \hat{\ } o * (post\ q \hat{\ } o)$  **and**  $uminus = neg\text{-}assert$  **and**  $H = post\ 1 \hat{\ } o$  **and**  $Z = bot$   
 ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}pre\text{-}dual$ :  $pre\text{-}post\text{-}spec\text{-}Hd$  **where**  $box = \lambda x\ y . neg\text{-}assume\ (x * neg\text{-}assume\ y)$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x\ y . (x * y * bot) \sqcup 1$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x\ y . wpb\ (x * y)$  **and**  $pre\text{-}post = \lambda p\ q . p * (post\ (q \hat{\ } o) \hat{\ } o)$  **and**  $uminus = neg\text{-}assume$  **and**  $bot = top$  **and**  $Hd = post\ 1 \hat{\ } o$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}dual$ :  $pre\text{-}post\text{-}spec\text{-}whiledo$  **where**  $ite = \lambda x\ p\ y . (p * x) \sqcap (neg\text{-}assume\ p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x\ y . wpb\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . (p \hat{\ } o) * post\ (q \hat{\ } o)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p\ x . ((p * x) \hat{\ } \omega) * neg\text{-}assume\ p$  **and**  $bot = top$  **and**  $top = bot$  ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}fix\text{-}dual$ :  $pre\text{-}post\text{-}spec\text{-}whiledo$  **where**  $ite = \lambda x\ p\ y . (p * x) \sqcap (neg\text{-}assume\ p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x\ y . wpb\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . (p \hat{\ } o) * post\ (q \hat{\ } o)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p\ x . ((p * x) \hat{\ } \omega) * neg\text{-}assume\ p$  **and**  $bot = top$  **and**  $top = bot$  ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}pre$ :  $pre\text{-}post\text{-}spec\text{-}whiledo$  **where**  $ite = \lambda x\ p\ y . (p * x) \sqcup (neg\text{-}assert\ p * y)$  **and**  $pre = \lambda x\ y . wpt\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . p \hat{\ } o * (post\ q \hat{\ } o)$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p\ x . ((p * x) \hat{\ } \otimes) * neg\text{-}assert\ p$  ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}pre\text{-}fix$ :  $pre\text{-}post\text{-}spec\text{-}whiledo$  **where**  $ite = \lambda x\ p\ y . (p * x) \sqcup (neg\text{-}assert\ p * y)$  **and**  $pre = \lambda x\ y . wpt\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . p \hat{\ } o * (post\ q \hat{\ } o)$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p\ x . ((p * x) \hat{\ } \cup) * neg\text{-}assert\ p$  ⟨proof⟩

**sublocale**  $post\text{-}mbt\text{-}algebra < mbta\text{-}dual$ :  $pre\text{-}post\text{-}L$  **where**  $box = \lambda x\ y . neg\text{-}assume\ (x * neg\text{-}assume\ y)$  **and**  $circ = omega$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x\ y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x\ p\ y . (p * x) \sqcap (neg\text{-}assume\ p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x\ y . wpb\ (x \hat{\ } o * y)$  **and**  $pre\text{-}post = \lambda p\ q . (p \hat{\ } o) * post\ (q \hat{\ } o)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p\ x . ((p * x) \hat{\ } \omega) * neg\text{-}assume\ p$  **and**  $bot = top$  **and**  $L = bot$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨proof⟩

**sublocale** *post-mbt-algebra* < *mbta-fix-dual*: *pre-post-L* **where**  $box = \lambda x y .$   
*neg-assume*  $(x * \text{neg-assume } y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  
*diamond*  $= \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (\text{neg-assume } p *$   
 $y)$  **and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y .$   
 $wpb (x \hat{o} * y)$  **and**  $pre-post = \lambda p q . (p \hat{o}) * post (q \hat{o})$  **and**  $uminus =$   
 $neg-assume$  **and**  $while = \lambda p x . ((p * x) \hat{*}) * neg-assume p$  **and**  $bot = top$  **and**  
 $L = top$  **and**  $top = bot$  **and**  $Z = top$   
 ⟨*proof*⟩

**sublocale** *post-mbt-algebra* < *mbta-pre*: *pre-post-L* **where**  $box = \lambda x y .$   
*neg-assert*  $(x * \text{neg-assert } y)$  **and**  $circ = dual-star$  **and**  $d = \lambda x . (x * top) \sqcap 1$   
**and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (\text{neg-assert}$   
 $p * y)$  **and**  $pre = \lambda x y . wpt (x \hat{o} * y)$  **and**  $pre-post = \lambda p q . p \hat{o} * (post q \hat{o})$   
 $o)$  **and**  $star = dual-star$  **and**  $uminus = neg-assert$  **and**  $while = \lambda p x . ((p * x) \hat{\otimes})$   
 $* neg-assert p$  **and**  $L = bot$  **and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *post-mbt-algebra* < *mbta-pre-fix*: *pre-post-L* **where**  $box = \lambda x y .$   
*neg-assert*  $(x * \text{neg-assert } y)$  **and**  $circ = dual-omega$  **and**  $d = \lambda x . (x * top) \sqcap 1$   
**and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (\text{neg-assert}$   
 $p * y)$  **and**  $pre = \lambda x y . wpt (x \hat{o} * y)$  **and**  $pre-post = \lambda p q . p \hat{o} * (post q \hat{o})$   
 $o)$  **and**  $star = dual-star$  **and**  $uminus = neg-assert$  **and**  $while = \lambda p x . ((p * x) \hat{\cup})$   
 $* neg-assert p$  **and**  $L = top$  **and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta*: *complete-tests* **where**  $uminus =$   
 $neg-assert$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-dual*: *complete-tests* **where**  $less =$   
 $greater$  **and**  $less-eq = greater-eq$  **and**  $sup = inf$  **and**  $uminus = neg-assume$  **and**  
 $bot = top$  **and**  $Inf = Sup$  **and**  $Sup = Inf$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta*: *complete-antidomain-semiring* **where**  $d$   
 $= \lambda x . (x * top) \sqcap 1$  **and**  $uminus = neg-assert$  **and**  $Z = bot$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-dual*: *complete-antidomain-semiring*  
**where**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $less = greater$  **and**  $less-eq = greater-eq$  **and**  
 $sup = inf$  **and**  $uminus = neg-assume$  **and**  $bot = top$  **and**  $Inf = Sup$  **and**  $Sup =$   
 $Inf$  **and**  $Z = top$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta*: *diamond-while-program* **where**  $box =$   
 $\lambda x y . neg-assert (x * neg-assert y)$  **and**  $circ = dual-star$  **and**  $d = \lambda x . (x * top)$   
 $\sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup$   
 $(neg-assert p * y)$  **and**  $pre = \lambda x y . wpt (x * y)$  **and**  $uminus = neg-assert$  **and**  
 $while = \lambda p x . ((p * x) \hat{\otimes}) * neg-assert p$  **and**  $Atomic-program = Continuous$



**and** *Atomic-test* = *assertion* **and**  $Z = \text{bot}$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-dual*: *box-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assume } (x * \text{neg-assume } y)$  **and**  $\text{circ} = \text{omega}$  **and**  $d = \lambda x . (x * \text{bot}) \sqcup 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{bot}) \sqcup 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcap$   
 $(\text{neg-assume } p * y)$  **and**  $\text{less} = \text{greater}$  **and**  $\text{less-eq} = \text{greater-eq}$  **and**  $\text{sup} = \text{inf}$   
**and**  $\text{pre} = \lambda x y . \text{wpb } (x \hat{\ } o * y)$  **and**  $\text{uminus} = \text{neg-assume}$  **and**  $\text{while} = \lambda p x . ((p * x) \hat{\ } \omega) * \text{neg-assume } p$  **and**  $\text{bot} = \text{top}$  **and** *Atomic-program* = *Continuous*  
**and** *Atomic-test* = *assumption* **and**  $\text{top} = \text{bot}$  **and**  $Z = \text{top}$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-fix*: *diamond-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assert } (x * \text{neg-assert } y)$  **and**  $\text{circ} = \text{dual-omega}$  **and**  $d = \lambda x . (x * \text{top}) \sqcap 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{top}) \sqcap 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcup$   
 $(\text{neg-assert } p * y)$  **and**  $\text{pre} = \lambda x y . \text{wpt } (x * y)$  **and**  $\text{uminus} = \text{neg-assert}$  **and**  
 $\text{while} = \lambda p x . ((p * x) \hat{\ } \cup) * \text{neg-assert } p$  **and** *Atomic-program* =  
*Co-continuous* **and** *Atomic-test* = *assertion* **and**  $Z = \text{bot}$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-fix-dual*: *box-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assume } (x * \text{neg-assume } y)$  **and**  $\text{circ} = \text{star}$  **and**  $d = \lambda x . (x * \text{bot}) \sqcup 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{bot}) \sqcup 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcap$   
 $(\text{neg-assume } p * y)$  **and**  $\text{less} = \text{greater}$  **and**  $\text{less-eq} = \text{greater-eq}$  **and**  $\text{sup} = \text{inf}$   
**and**  $\text{pre} = \lambda x y . \text{wpb } (x \hat{\ } o * y)$  **and**  $\text{uminus} = \text{neg-assume}$  **and**  $\text{while} = \lambda p x . ((p * x) \hat{\ } *) * \text{neg-assume } p$  **and**  $\text{bot} = \text{top}$  **and** *Atomic-program* =  
*Co-continuous* **and** *Atomic-test* = *assumption* **and**  $\text{top} = \text{bot}$  **and**  $Z = \text{top}$   
 ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-pre*: *box-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assert } (x * \text{neg-assert } y)$  **and**  $\text{circ} = \text{dual-star}$  **and**  $d = \lambda x . (x * \text{top}) \sqcap 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{top}) \sqcap 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcup (\text{neg-assert } p * y)$  **and**  $\text{pre} = \lambda x y . \text{wpt } (x \hat{\ } o * y)$  **and**  $\text{uminus} = \text{neg-assert}$  **and**  $\text{while} = \lambda p x . ((p * x) \hat{\ } \otimes) * \text{neg-assert } p$  **and** *Atomic-program* = *Continuous* **and**  
*Atomic-test* = *assertion* **and**  $Z = \text{bot}$  ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-pre-dual*: *diamond-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assume } (x * \text{neg-assume } y)$  **and**  $\text{circ} = \text{omega}$  **and**  $d = \lambda x . (x * \text{bot}) \sqcup 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{bot}) \sqcup 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcap$   
 $(\text{neg-assume } p * y)$  **and**  $\text{less} = \text{greater}$  **and**  $\text{less-eq} = \text{greater-eq}$  **and**  $\text{sup} = \text{inf}$   
**and**  $\text{pre} = \lambda x y . \text{wpb } (x * y)$  **and**  $\text{uminus} = \text{neg-assume}$  **and**  $\text{while} = \lambda p x . ((p * x) \hat{\ } \omega) * \text{neg-assume } p$  **and**  $\text{bot} = \text{top}$  **and** *Atomic-program* = *Continuous* **and**  
*Atomic-test* = *assumption* **and**  $\text{top} = \text{bot}$  **and**  $Z = \text{top}$  ⟨*proof*⟩

**sublocale** *complete-mbt-algebra* < *mbta-pre-fix*: *box-while-program* **where**  $\text{box} = \lambda x y . \text{neg-assert } (x * \text{neg-assert } y)$  **and**  $\text{circ} = \text{dual-omega}$  **and**  $d = \lambda x . (x * \text{top}) \sqcap 1$  **and**  $\text{diamond} = \lambda x y . (x * y * \text{top}) \sqcap 1$  **and**  $\text{ite} = \lambda x p y . (p * x) \sqcup$   
 $(\text{neg-assert } p * y)$  **and**  $\text{pre} = \lambda x y . \text{wpt } (x \hat{\ } o * y)$  **and**  $\text{uminus} = \text{neg-assert}$   
**and**  $\text{while} = \lambda p x . ((p * x) \hat{\ } \cup) * \text{neg-assert } p$  **and** *Atomic-program* =

*Co-continuous and Atomic-test = assertion and Z = bot*  $\langle$ proof $\rangle$

**sublocale** *complete-mbt-algebra*  $<$  *mbta-pre-fix-dual: diamond-while-program*  
**where**  $box = \lambda x y . neg\text{-}assume (x * neg\text{-}assume y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-}assume p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb (x * y)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p x . ((p * x) \hat{*}) * neg\text{-}assume p$  **and**  $bot = top$  **and** *Atomic-program = Co-continuous and Atomic-test = assumption and top = bot and Z = top*  
 $\langle$ proof $\rangle$

### Theorem 52

**sublocale** *complete-mbt-algebra*  $<$  *mbta: diamond-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assert (x * neg\text{-}assert y)$  **and**  $circ = dual\text{-}star$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert p * y)$  **and**  $pre = \lambda x y . wpt (x * y)$  **and**  $star = dual\text{-}star$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\otimes}) * neg\text{-}assert p$  **and** *Atomic-program = Continuous and Atomic-test = assertion and Z = bot*  
 $\langle$ proof $\rangle$

### Theorem 52

**sublocale** *complete-mbt-algebra*  $<$  *mbta-dual: box-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assume (x * neg\text{-}assume y)$  **and**  $circ = omega$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-}assume p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb (x \hat{o} * y)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p x . ((p * x) \hat{\omega}) * neg\text{-}assume p$  **and**  $bot = top$  **and** *Atomic-program = Continuous and Atomic-test = assumption and Inf = Sup and Sup = Inf and top = bot and Z = top*  
 $\langle$ proof $\rangle$

### Theorem 52

**sublocale** *complete-mbt-algebra*  $<$  *mbta-fix: diamond-hoare-sound-2* **where**  $box = \lambda x y . neg\text{-}assert (x * neg\text{-}assert y)$  **and**  $circ = dual\text{-}omega$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert p * y)$  **and**  $pre = \lambda x y . wpt (x * y)$  **and**  $star = dual\text{-}star$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\cup}) * neg\text{-}assert p$  **and** *Atomic-program = Co-continuous and Atomic-test = assertion and Z = bot*  
 $\langle$ proof $\rangle$

### Theorem 52

**sublocale** *complete-mbt-algebra*  $<$  *mbta-fix-dual: box-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assume (x * neg\text{-}assume y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-}assume p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb (x \hat{o} * y)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p x . ((p * x) \hat{*}) * neg\text{-}assume p$  **and**  $bot = top$  **and** *Atomic-program = Co-continuous and Atomic-test = assumption and Inf = Sup and Sup = Inf and top = bot and Z = top*

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre: box-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assert (x * neg\text{-}assert y)$  **and**  $circ = dual\text{-}star$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert p * y)$  **and**  $pre = \lambda x y . wpt (x \hat{\ } o * y)$  **and**  $star = dual\text{-}star$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\ } \otimes) * neg\text{-}assert p$  **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and**  $Z = bot$

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre-dual: diamond-hoare-sound-2* **where**  $box = \lambda x y . neg\text{-}assume (x * neg\text{-}assume y)$  **and**  $circ = omega$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-}assume p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb (x * y)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p x . ((p * x) \hat{\ } \omega) * neg\text{-}assume p$  **and**  $bot = top$  **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  **and**  $Z = top$

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre-fix: box-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assert (x * neg\text{-}assert y)$  **and**  $circ = dual\text{-}omega$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert p * y)$  **and**  $pre = \lambda x y . wpt (x \hat{\ } o * y)$  **and**  $star = dual\text{-}star$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\ } \cup) * neg\text{-}assert p$  **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assertion* **and**  $Z = bot$

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre-fix-dual: diamond-hoare-sound* **where**  $box = \lambda x y . neg\text{-}assume (x * neg\text{-}assume y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-}assume p * y)$  **and**  $less = greater$  **and**  $less\text{-}eq = greater\text{-}eq$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb (x * y)$  **and**  $uminus = neg\text{-}assume$  **and**  $while = \lambda p x . ((p * x) \hat{\ } *) * neg\text{-}assume p$  **and**  $bot = top$  **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assumption* **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  **and**  $Z = top$

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta: diamond-hoare-valid* **where**  $box = \lambda x y . neg\text{-}assert (x * neg\text{-}assert y)$  **and**  $circ = dual\text{-}star$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $hoare\text{-}triple = \lambda p x q . p \leq wpt(x * q)$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert p * y)$  **and**  $pre = \lambda x y . wpt (x * y)$  **and**  $star = dual\text{-}star$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\ } \otimes) * neg\text{-}assert p$  **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and**  $Z = bot$

*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-dual*: *box-hoare-valid* **where**  $box = \lambda x y . neg\text{-assume } (x * neg\text{-assume } y)$  **and**  $circ = \omega$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $hoare\text{-triple} = \lambda p x q . wpb(x \hat{o} * q) \leq p$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-assume } p * y)$  **and**  $less = greater$  **and**  $less\text{-eq} = greater\text{-eq}$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb(x \hat{o} * y)$  **and**  $uminus = neg\text{-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{\omega}) * neg\text{-assume } p$  **and**  $bot = top$  **and**  $Atomic\text{-program} = Continuous$  **and**  $Atomic\text{-test} = assumption$  **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  **and**  $Z = top$   
*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre-fix-dual*: *diamond-hoare-valid* **where**  $box = \lambda x y . neg\text{-assume } (x * neg\text{-assume } y)$  **and**  $circ = star$  **and**  $d = \lambda x . (x * bot) \sqcup 1$  **and**  $diamond = \lambda x y . (x * y * bot) \sqcup 1$  **and**  $hoare\text{-triple} = \lambda p x q . wpb(x * q) \leq p$  **and**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-assume } p * y)$  **and**  $less = greater$  **and**  $less\text{-eq} = greater\text{-eq}$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb(x * y)$  **and**  $uminus = neg\text{-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{*}) * neg\text{-assume } p$  **and**  $bot = top$  **and**  $Atomic\text{-program} = Co\text{-continuous}$  **and**  $Atomic\text{-test} = assumption$  **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  **and**  $Z = top$   
*<proof>*

**Theorem 52**

**sublocale** *complete-mbt-algebra* < *mbta-pre-fix*: *box-hoare-valid* **where**  $box = \lambda x y . neg\text{-assert } (x * neg\text{-assert } y)$  **and**  $circ = dual\text{-omega}$  **and**  $d = \lambda x . (x * top) \sqcap 1$  **and**  $diamond = \lambda x y . (x * y * top) \sqcap 1$  **and**  $hoare\text{-triple} = \lambda p x q . p \leq wpt(x \hat{o} * q)$  **and**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-assert } p * y)$  **and**  $pre = \lambda x y . wpt(x \hat{o} * y)$  **and**  $star = dual\text{-star}$  **and**  $uminus = neg\text{-assert}$  **and**  $while = \lambda p x . ((p * x) \hat{\cup}) * neg\text{-assert } p$  **and**  $Atomic\text{-program} = Co\text{-continuous}$  **and**  $Atomic\text{-test} = assertion$  **and**  $Z = bot$   
*<proof>*

**sublocale** *complete-mbt-algebra* < *mbta-dual*: *pre-post-spec-hoare* **where**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-assume } p * y)$  **and**  $less = greater$  **and**  $less\text{-eq} = greater\text{-eq}$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb(x \hat{o} * y)$  **and**  $pre\text{-post} = \lambda p q . (p \hat{o}) * post(q \hat{o})$  **and**  $uminus = neg\text{-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{\omega}) * neg\text{-assume } p$  **and**  $bot = top$  **and**  $Atomic\text{-program} = Continuous$  **and**  $Atomic\text{-test} = assumption$  **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  *<proof>*

**sublocale** *complete-mbt-algebra* < *mbta-fix-dual*: *pre-post-spec-hoare* **where**  $ite = \lambda x p y . (p * x) \sqcap (neg\text{-assume } p * y)$  **and**  $less = greater$  **and**  $less\text{-eq} = greater\text{-eq}$  **and**  $sup = inf$  **and**  $pre = \lambda x y . wpb(x \hat{o} * y)$  **and**  $pre\text{-post} = \lambda p q . (p \hat{o}) * post(q \hat{o})$  **and**  $uminus = neg\text{-assume}$  **and**  $while = \lambda p x . ((p * x) \hat{*}) * neg\text{-assume } p$  **and**  $bot = top$  **and**  $Atomic\text{-program} = Co\text{-continuous}$  **and**  $Atomic\text{-test} = assumption$  **and**  $Inf = Sup$  **and**  $Sup = Inf$  **and**  $top = bot$  *<proof>*

**sublocale** *complete-mbt-algebra* < *mbta-pre*: *pre-post-spec-hoare* **where**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-assert } p * y)$  **and**  $pre = \lambda x y . wpt(x \hat{o} * y)$  **and**  $pre\text{-post}$

$= \lambda p q . p \hat{o} * (post\ q \hat{o})$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\otimes}) * neg\text{-}assert\ p$  **and**  $Atomic\text{-}program = Continuous$  **and**  $Atomic\text{-}test = assertion\ \langle proof \rangle$

**sublocale**  $complete\text{-}mbt\text{-}algebra < mbta\text{-}pre\text{-}fix: pre\text{-}post\text{-}spec\text{-}hoare$  **where**  $ite = \lambda x p y . (p * x) \sqcup (neg\text{-}assert\ p * y)$  **and**  $pre = \lambda x y . wpt\ (x \hat{o} * y)$  **and**  $pre\text{-}post = \lambda p q . p \hat{o} * (post\ q \hat{o})$  **and**  $uminus = neg\text{-}assert$  **and**  $while = \lambda p x . ((p * x) \hat{\cup}) * neg\text{-}assert\ p$  **and**  $Atomic\text{-}program = Co\text{-}continuous$  **and**  $Atomic\text{-}test = assertion\ \langle proof \rangle$

**end**

## References

- [1] R. Berghammer and W. Guttman. Closure, properties and closure properties of multirelations. In W. Kahl, M. Winter, and J. N. Oliveira, editors, *Relational and Algebraic Methods in Computer Science (RAM-iCS 2015)*, volume 9348 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2015.
- [2] R. Berghammer and W. Guttman. An algebraic approach to multirelations and their properties. *Journal of Logical and Algebraic Methods in Programming*, 88:45–63, 2017.
- [3] W. Guttman. General correctness algebra. In R. Berghammer, A. M. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science (RelMiCS/AKA 2009)*, volume 5827 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2009.
- [4] W. Guttman. Partial, total and general correctness. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction (MPC 2010)*, volume 6120 of *Lecture Notes in Computer Science*, pages 157–177. Springer, 2010.
- [5] W. Guttman. Unifying recursion in partial, total and general correctness. In S. Qin, editor, *Unifying Theories of Programming, Third International Symposium (UTP 2010)*, volume 6445 of *Lecture Notes in Computer Science*, pages 207–225. Springer, 2010.
- [6] W. Guttman. Fixpoints for general correctness. *Journal of Logic and Algebraic Programming*, 80(6):248–265, 2011.
- [7] W. Guttman. Towards a typed omega algebra. In H. de Swart, editor, *Relational and Algebraic Methods in Computer Science (RAM-iCS 2011)*, volume 6663 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2011.

- [8] W. Guttman. Algebras for iteration and infinite computations. *Acta Inf.*, 49(5):343–359, 2012.
- [9] W. Guttman. Typing theorems of omega algebra. *Journal of Logic and Algebraic Programming*, 81(6):643–659, 2012.
- [10] W. Guttman. Unifying correctness statements. In J. Gibbons and P. Nogueira, editors, *Mathematics of Program Construction (MPC 2012)*, volume 7342 of *Lecture Notes in Computer Science*, pages 198–219. Springer, 2012.
- [11] W. Guttman. Unifying lazy and strict computations. In W. Kahl and T. G. Griffin, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2012)*, volume 7560 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2012.
- [12] W. Guttman. Extended designs algebraically. *Sci. Comput. Programming*, 78(11):2064–2085, 2013.
- [13] W. Guttman. Algebras for correctness of sequential computations. *Sci. Comput. Programming*, 85(Part B):224–240, 2014.
- [14] W. Guttman. Extended conscriptions algebraically. In P. Höfner, P. Jipsen, W. Kahl, and M. E. Müller, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2014)*, volume 8428 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2014.
- [15] W. Guttman. Multirelations with infinite computations. *Journal of Logical and Algebraic Methods in Programming*, 83(2):194–211, 2014.
- [16] W. Guttman. *Algebras for Iteration, Infinite Executions and Correctness of Sequential Computations*. Habilitationsschrift, Universität Ulm, 2015.
- [17] W. Guttman. Infinite executions of lazy and strict computations. *Journal of Logical and Algebraic Methods in Programming*, 84(3):326–340, 2015.
- [18] W. Guttman. Isabelle/HOL theories of algebras for iteration, infinite executions and correctness of sequential computations. Technical Report TR-COSC 02/15, University of Canterbury, 2015.
- [19] W. Guttman. An algebraic approach to computations with progress. *Journal of Logical and Algebraic Methods in Programming*, 85(4):520–539, 2016.
- [20] W. Guttman, G. Struth, and T. Weber. Automating algebraic methods in Isabelle. In S. Qin and Z. Qiu, editors, *Formal Methods and*

*Software Engineering (ICFEM 2011)*, volume 6991 of *Lecture Notes in Computer Science*, pages 617–632. Springer, 2011.