Algebras for correctness of sequential computations

Walter Guttmann, University of Canterbury

2012.12.20

theory *ACSC*

imports *Main MonoBoolTranAlgebra/Assertion-Algebra*

begin

class *mult* = *times*

begin

notation
  times (**infixl** · *70*) **and**
  times (**infixl** ; *70*)

end

class *neg* = *uminus*

begin

no-notation
  uminus (− - [*81*] *80*)

notation
  uminus (− - [*80*] *80*)

end

context *ord*

begin

definition *ascending-chain* :: $(nat \Rightarrow 'a) \Rightarrow bool$
  where *ascending-chain* $f \leftrightarrow (\forall n\ .\ f\ n \leq f\ (Suc\ n))$

definition *descending-chain* :: $(nat \Rightarrow 'a) \Rightarrow bool$
  where *descending-chain* $f \leftrightarrow (\forall n\ .\ f\ (Suc\ n) \leq f\ n)$

definition *directed* $X \leftrightarrow X \neq \{\} \wedge (\forall x{\in}X\ .\ \forall y{\in}X\ .\ \exists z{\in}X\ .\ x \leq z \wedge y \leq z)$

definition *codirected* $X \leftrightarrow X \neq \{\} \wedge (\forall x{\in}X\ .\ \forall y{\in}X\ .\ \exists z{\in}X\ .\ z \leq x \wedge z \leq y)$

definition *chain* $X \leftrightarrow (\forall x{\in}X\ .\ \forall y{\in}X\ .\ x \leq y \vee y \leq x)$

end

context *order*

begin

lemma *ascending-chain-k*: *ascending-chain* $f \rightarrow f\ n \leq f\ (n + k)$
  apply (*induct k*)
  apply *simp*
  apply (*metis add-Suc-right ascending-chain-def order-trans*)
  done

lemma *ascending-chain-isotone*: *ascending-chain* $f \wedge m \leq n \rightarrow f\ m \leq f\ n$
  by (*metis ascending-chain-k le-iff-add*)

lemma *ascending-chain-comparable*: *ascending-chain* $f \rightarrow f\ n \leq f\ m \vee f\ m \leq f\ n$
  by (*metis nat-le-linear ascending-chain-isotone*)

lemma *ascending-chain-chain*: *ascending-chain* $f \rightarrow$ *chain* (*range* f)
  unfolding *chain-def*
  apply *simp*
  apply (*smt ascending-chain-comparable*)
  done

lemma *chain-directed*: $X \neq \{\} \wedge$ *chain* $X \rightarrow$ *directed* X
  by (*metis chain-def directed-def*)

lemma *ascending-chain-directed*: *ascending-chain* $f \rightarrow$ *directed* (*range* f)
  by (*metis UNIV-not-empty ascending-chain-chain chain-directed empty-is-image*)

lemma *descending-chain-k*: *descending-chain* $f \rightarrow f\ (n + k) \leq f\ n$
  apply (*induct k*)

apply *simp*
apply (*metis add-Suc-right descending-chain-def order-trans*)
done

lemma *descending-chain-antitone*: *descending-chain f* $\wedge$ $m \leq n \rightarrow f\ n \leq f\ m$
by (*metis descending-chain-k le-iff-add*)

lemma *descending-chain-comparable*: *descending-chain f* $\rightarrow f\ n \leq f\ m \vee f\ m \leq f\ n$
by (*metis nat-le-linear descending-chain-antitone*)

lemma *descending-chain-chain*: *descending-chain f* $\rightarrow$ *chain* (*range f*)
unfolding *chain-def*
apply *simp*
apply (*smt descending-chain-comparable*)
done

lemma *chain-codirected*: $X \neq \{\}$ $\wedge$ *chain X* $\rightarrow$ *codirected X*
by (*metis chain-def codirected-def*)

lemma *descending-chain-codirected*: *descending-chain f* $\rightarrow$ *codirected* (*range f*)
by (*metis UNIV-not-empty descending-chain-chain chain-codirected empty-is-image*)

end

context *complete-lattice*

begin

lemma *sup-Sup*: **assumes** *nonempty*: $A \neq \{\}$
shows *sup x* (*Sup A*) = *Sup* ((*sup x*) ‘ *A*)
apply (*rule antisym*)
apply (*metis Sup-mono Sup-upper2 assms ex-in-conv imageI le-supI sup-ge1 sup-ge2*)
apply (*smt Sup-least Sup-upper image-iff le-iff-sup sup.commute sup-ge1 sup-left-commute*)
done

lemma *sup-SUP*: $Y \neq \{\} \rightarrow$ *sup x* (*SUP y*:*Y* . *f y*) = (*SUP y*:*Y*. *sup x* (*f y*))
unfolding *SUP-def*
apply *rule*
apply (*subst sup-Sup*)
apply (*smt empty-is-image*)
apply (*metis SUP-def SUP-image*)
done

lemma *inf-Inf*: **assumes** *nonempty*: $A \neq \{\}$
shows *inf x* (*Inf A*) = *Inf* ((*inf x*) ‘ *A*)
apply (*rule antisym*)
apply (*smt Inf-greatest Inf-lower image-iff le-iff-inf inf.commute inf-le1 inf-left-commute*)

apply (*metis Inf-mono Inf-lower2 assms ex-in-conv imageI le-infI inf-le1 inf-le2*)
done

lemma *inf-INF*: $Y \neq \{\} \rightarrow inf\ x\ (INF\ y{:}Y\ .\ f\ y) = (INF\ y{:}Y.\ inf\ x\ (f\ y))$
unfolding *INF-def*
apply *rule*
apply (*subst inf-Inf*)
apply (*smt empty-is-image*)
apply (*metis INF-def INF-image*)
done

lemma *SUP-image-id*[*simp*]: $(SUP\ x{:}f`A\ .\ x) = (SUP\ x{:}A\ .\ f\ x)$
by *simp*

lemma *INF-image-id*[*simp*]: $(INF\ x{:}f`A\ .\ x) = (INF\ x{:}A\ .\ f\ x)$
by *simp*

end

lemma *image-Collect-2*: $f\ `\ \{\ g\ x\ |\ x\ .\ P\ x\ \} = \{\ f\ (g\ x)\ |\ x\ .\ P\ x\ \}$
by *auto*

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

begin

definition *one-fun* :: $'a \Rightarrow 'a$
where *one-fun-def*: *one-fun* $\equiv$ *id*

definition *times-fun* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a)$
where *times-fun-def*: *times-fun* $\equiv$ *comp*

instance
by *intro-classes*

end

lemma *id-power*: $id\hat{}n = id$
apply (*induct n*)
apply (*metis one-fun-def power-0*)
by (*simp add*: *times-fun-def*)

lemma *power-zero-id*: $f\hat{}0 = id$
by (*metis one-fun-def power-0*)

lemma *power-succ-unfold*: $f\hat{}Suc\ n = f \circ f\hat{}n$
by (*metis power-Suc times-fun-def*)

lemma *power-succ-unfold-ext*: $(f\hat{}Suc\ n)\ x = f\ ((f\hat{}n)\ x)$
by (*metis o-apply power-succ-unfold*)

context *order*

begin

definition *isotone* :: $('a \Rightarrow 'a) \Rightarrow bool$
where $isotone\ f \leftrightarrow (\forall x\ y\ .\ x \leq y \rightarrow f(x) \leq f(y))$

definition *is-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}fixpoint\ f\ x \leftrightarrow f(x) = x$
definition *is-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}prefixpoint\ f\ x \leftrightarrow f(x) \leq x$
definition *is-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}postfixpoint\ f\ x \leftrightarrow f(x) \geq x$
definition *is-least-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}least\text{-}fixpoint\ f\ x \leftrightarrow f(x) = x \wedge (\forall y\ .\ f(y) = y \rightarrow x \leq y)$
definition *is-greatest-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}greatest\text{-}fixpoint\ f\ x \leftrightarrow f(x) = x \wedge (\forall y\ .\ f(y) = y \rightarrow x \geq y)$
definition *is-least-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}least\text{-}prefixpoint\ f\ x \leftrightarrow f(x) \leq x \wedge (\forall y\ .\ f(y) \leq y \rightarrow x \leq y)$
definition *is-greatest-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}greatest\text{-}postfixpoint\ f\ x \leftrightarrow f(x) \geq x \wedge (\forall y\ .\ f(y) \geq y \rightarrow x \geq y)$
definition *has-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}fixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}fixpoint\ f\ x)$
definition *has-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}prefixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}prefixpoint\ f\ x)$
definition *has-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}postfixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}postfixpoint\ f\ x)$
definition *has-least-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}least\text{-}fixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}least\text{-}fixpoint\ f\ x)$
definition *has-greatest-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}greatest\text{-}fixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}greatest\text{-}fixpoint\ f\ x)$
definition *has-least-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}least\text{-}prefixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}least\text{-}prefixpoint\ f\ x)$
definition *has-greatest-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where** $has\text{-}greatest\text{-}postfixpoint\ f \leftrightarrow (\exists x\ .\ is\text{-}greatest\text{-}postfixpoint\ f\ x)$
definition *the-least-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a\ (\mu\ -\ [201]\ 200)$ **where** $\mu\ f = (THE\ x\ .\ is\text{-}least\text{-}fixpoint\ f\ x)$
definition *the-greatest-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a\ (\nu\ -\ [201]\ 200)$ **where** $\nu\ f = (THE\ x\ .\ is\text{-}greatest\text{-}fixpoint\ f\ x)$
definition *the-least-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a\ (p\mu\ -\ [201]\ 200)$ **where** $p\mu\ f = (THE\ x\ .\ is\text{-}least\text{-}prefixpoint\ f\ x)$
definition *the-greatest-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a\ (p\nu\ -\ [201]\ 200)$ **where** $p\nu\ f = (THE\ x\ .\ is\text{-}greatest\text{-}postfixpoint\ f\ x)$

lemma *least-fixpoint-unique*: $has\text{-}least\text{-}fixpoint\ f \rightarrow (\exists!x\ .\ is\text{-}least\text{-}fixpoint\ f\ x)$
by (*smt antisym has-least-fixpoint-def is-least-fixpoint-def*)

lemma *greatest-fixpoint-unique*: $has\text{-}greatest\text{-}fixpoint\ f \rightarrow (\exists!x\ .\ is\text{-}greatest\text{-}fixpoint\ f\ x)$
by (*smt antisym has-greatest-fixpoint-def is-greatest-fixpoint-def*)

lemma *least-prefixpoint-unique*: $has\text{-}least\text{-}prefixpoint\ f \rightarrow (\exists!x\ .\ is\text{-}least\text{-}prefixpoint\ f\ x)$
by (*smt antisym has-least-prefixpoint-def is-least-prefixpoint-def*)

lemma *greatest-postfixpoint-unique*: $has\text{-}greatest\text{-}postfixpoint\ f \rightarrow (\exists!x\ .\ is\text{-}greatest\text{-}postfixpoint\ f\ x)$
by (*smt antisym has-greatest-postfixpoint-def is-greatest-postfixpoint-def*)

lemma *least-fixpoint*: $has\text{-}least\text{-}fixpoint\ f \rightarrow is\text{-}least\text{-}fixpoint\ f\ (\mu\ f)$

proof
  assume *has-least-fixpoint f*
  hence *is-least-fixpoint f* (*THE x . is-least-fixpoint f x*)
    by (*smt least-fixpoint-unique theI'*)
  thus *is-least-fixpoint f* (*μ f*)
    by (*simp add*: *is-least-fixpoint-def the-least-fixpoint-def*)
qed

lemma *greatest-fixpoint*: *has-greatest-fixpoint f* ⟶ *is-greatest-fixpoint f* (*ν f*)
proof
  assume *has-greatest-fixpoint f*
  hence *is-greatest-fixpoint f* (*THE x . is-greatest-fixpoint f x*)
    by (*smt greatest-fixpoint-unique theI'*)
  thus *is-greatest-fixpoint f* (*ν f*)
    by (*simp add*: *is-greatest-fixpoint-def the-greatest-fixpoint-def*)
qed

lemma *least-prefixpoint*: *has-least-prefixpoint f* ⟶ *is-least-prefixpoint f* (*pμ f*)
proof
  assume *has-least-prefixpoint f*
  hence *is-least-prefixpoint f* (*THE x . is-least-prefixpoint f x*)
    by (*smt least-prefixpoint-unique theI'*)
  thus *is-least-prefixpoint f* (*pμ f*)
    by (*simp add*: *is-least-prefixpoint-def the-least-prefixpoint-def*)
qed

lemma *greatest-postfixpoint*: *has-greatest-postfixpoint f* ⟶ *is-greatest-postfixpoint f* (*pν f*)
proof
  assume *has-greatest-postfixpoint f*
  hence *is-greatest-postfixpoint f* (*THE x . is-greatest-postfixpoint f x*)
    by (*smt greatest-postfixpoint-unique theI'*)
  thus *is-greatest-postfixpoint f* (*pν f*)
    by (*simp add*: *is-greatest-postfixpoint-def the-greatest-postfixpoint-def*)
qed

lemma *least-fixpoint-same*: *is-least-fixpoint f x* ⟶ *x* = *μ f*
  by (*metis least-fixpoint least-fixpoint-unique has-least-fixpoint-def*)

lemma *greatest-fixpoint-same*: *is-greatest-fixpoint f x* ⟶ *x* = *ν f*
  by (*metis greatest-fixpoint greatest-fixpoint-unique has-greatest-fixpoint-def*)

lemma *least-prefixpoint-same*: *is-least-prefixpoint f x* ⟶ *x* = *pμ f*
  by (*metis least-prefixpoint least-prefixpoint-unique has-least-prefixpoint-def*)

lemma *greatest-postfixpoint-same*: *is-greatest-postfixpoint f x* ⟶ *x* = *pν f*
  by (*metis greatest-postfixpoint greatest-postfixpoint-unique has-greatest-postfixpoint-def*)

lemma *least-fixpoint-char*: $\mathit{is\text{-}least\text{-}fixpoint}\ f\ x \leftrightarrow \mathit{has\text{-}least\text{-}fixpoint}\ f \wedge x = \mu\ f$
by (*metis least-fixpoint-same has-least-fixpoint-def*)

lemma *least-prefixpoint-char*: $\mathit{is\text{-}least\text{-}prefixpoint}\ f\ x \leftrightarrow \mathit{has\text{-}least\text{-}prefixpoint}\ f \wedge x = p\mu\ f$
by (*metis least-prefixpoint-same has-least-prefixpoint-def*)

lemma *greatest-fixpoint-char*: $\mathit{is\text{-}greatest\text{-}fixpoint}\ f\ x \leftrightarrow \mathit{has\text{-}greatest\text{-}fixpoint}\ f \wedge x = \nu\ f$
by (*metis greatest-fixpoint-same has-greatest-fixpoint-def*)

lemma *greatest-postfixpoint-char*: $\mathit{is\text{-}greatest\text{-}postfixpoint}\ f\ x \leftrightarrow \mathit{has\text{-}greatest\text{-}postfixpoint}\ f \wedge x = p\nu\ f$
by (*metis greatest-postfixpoint-same has-greatest-postfixpoint-def*)

lemma *mu-unfold*: $\mathit{has\text{-}least\text{-}fixpoint}\ f \to f\ (\mu\ f) = \mu\ f$
by (*metis is-least-fixpoint-def least-fixpoint*)

lemma *pmu-unfold*: $\mathit{has\text{-}least\text{-}prefixpoint}\ f \to f\ (p\mu\ f) \leq p\mu\ f$
by (*metis is-least-prefixpoint-def least-prefixpoint*)

lemma *nu-unfold*: $\mathit{has\text{-}greatest\text{-}fixpoint}\ f \to \nu\ f = f\ (\nu\ f)$
by (*metis is-greatest-fixpoint-def greatest-fixpoint*)

lemma *pnu-unfold*: $\mathit{has\text{-}greatest\text{-}postfixpoint}\ f \to p\nu\ f \leq f\ (p\nu\ f)$
by (*metis is-greatest-postfixpoint-def greatest-postfixpoint*)

lemma *least-prefixpoint-fixpoint*: $\mathit{has\text{-}least\text{-}prefixpoint}\ f \wedge \mathit{isotone}\ f \to \mathit{is\text{-}least\text{-}fixpoint}\ f\ (p\mu\ f)$
by (*smt eq-iff is-least-fixpoint-def is-least-prefixpoint-def isotone-def least-prefixpoint*)

lemma *pmu-mu*: $\mathit{has\text{-}least\text{-}prefixpoint}\ f \wedge \mathit{isotone}\ f \to p\mu\ f = \mu\ f$
by (*smt has-least-fixpoint-def is-least-fixpoint-def least-fixpoint-unique least-prefixpoint-fixpoint least-fixpoint*)

lemma *greatest-postfixpoint-fixpoint*: $\mathit{has\text{-}greatest\text{-}postfixpoint}\ f \wedge \mathit{isotone}\ f \to \mathit{is\text{-}greatest\text{-}fixpoint}\ f\ (p\nu\ f)$
by (*smt eq-iff is-greatest-fixpoint-def is-greatest-postfixpoint-def isotone-def greatest-postfixpoint*)

lemma *pnu-nu*: $\mathit{has\text{-}greatest\text{-}postfixpoint}\ f \wedge \mathit{isotone}\ f \to p\nu\ f = \nu\ f$
by (*smt has-greatest-fixpoint-def is-greatest-fixpoint-def greatest-fixpoint-unique greatest-postfixpoint-fixpoint greatest-fixpoint*)

definition *lifted-less-eq* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow \mathit{bool}$ $((\text{-} \leq\leq \text{-})\ [51, 51]\ 50)$
where $f \leq\leq g \leftrightarrow (\forall x\ .\ f(x) \leq g(x))$

lemma *lifted-reflexive*: $f = g \to f \leq\leq g$
by (*metis lifted-less-eq-def order-refl*)

lemma *lifted-transitive*: $f \leq\leq g \wedge g \leq\leq h \to f \leq\leq h$
by (*smt lifted-less-eq-def order-trans*)

lemma *lifted-antisymmetric*: $f \leq\leq g \wedge g \leq\leq f \to f = g$
by (*metis antisym ext lifted-less-eq-def*)

lemma *pmu-isotone*: *has-least-prefixpoint f* $\wedge$ *has-least-prefixpoint g* $\wedge$ *f* $\leq\leq$ *g* $\rightarrow$ *p*μ *f* $\leq$ *p*μ *g*
by (*smt is-least-prefixpoint-def least-prefixpoint lifted-less-eq-def order-trans*)

lemma *mu-isotone*: *has-least-prefixpoint f* $\wedge$ *has-least-prefixpoint g* $\wedge$ *isotone f* $\wedge$ *isotone g* $\wedge$ *f* $\leq\leq$ *g* $\rightarrow$ μ *f* $\leq$ μ *g*
by (*metis pmu-isotone pmu-mu*)

lemma *pnu-isotone*: *has-greatest-postfixpoint f* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *f* $\leq\leq$ *g* $\rightarrow$ *p*ν *f* $\leq$ *p*ν *g*
by (*smt is-greatest-postfixpoint-def lifted-less-eq-def order-trans greatest-postfixpoint*)

lemma *nu-isotone*: *has-greatest-postfixpoint f* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *isotone f* $\wedge$ *isotone g* $\wedge$ *f* $\leq\leq$ *g* $\rightarrow$ ν *f* $\leq$ ν *g*
by (*metis pnu-isotone pnu-nu*)

lemma *mu-square*: *isotone f* $\wedge$ *has-least-fixpoint f* $\wedge$ *has-least-fixpoint* (*f* $\circ$ *f*) $\rightarrow$ μ *f* = μ (*f* $\circ$ *f*)
by (*metis antisym is-least-fixpoint-def isotone-def least-fixpoint-char least-fixpoint-unique o-apply*)

lemma *nu-square*: *isotone f* $\wedge$ *has-greatest-fixpoint f* $\wedge$ *has-greatest-fixpoint* (*f* $\circ$ *f*) $\rightarrow$ ν *f* = ν (*f* $\circ$ *f*)
by (*metis antisym is-greatest-fixpoint-def isotone-def greatest-fixpoint-char greatest-fixpoint-unique o-apply*)

lemma *mu-roll*: *isotone g* $\wedge$ *has-least-fixpoint* (*f* $\circ$ *g*) $\wedge$ *has-least-fixpoint* (*g* $\circ$ *f*) $\rightarrow$ μ (*g* $\circ$ *f*) = *g*(μ (*f* $\circ$ *g*))
apply (*rule impI*, *rule antisym*)
apply (*smt is-least-fixpoint-def least-fixpoint o-apply*)
by (*smt is-least-fixpoint-def isotone-def least-fixpoint o-apply*)

lemma *nu-roll*: *isotone g* $\wedge$ *has-greatest-fixpoint* (*f* $\circ$ *g*) $\wedge$ *has-greatest-fixpoint* (*g* $\circ$ *f*) $\rightarrow$ ν (*g* $\circ$ *f*) = *g*(ν (*f* $\circ$ *g*))
apply (*rule impI*, *rule antisym*)
apply (*smt is-greatest-fixpoint-def greatest-fixpoint isotone-def o-apply*)
by (*smt is-greatest-fixpoint-def greatest-fixpoint o-apply*)

lemma *mu-below-nu*: *has-least-fixpoint f* $\wedge$ *has-greatest-fixpoint f* $\rightarrow$ μ *f* $\leq$ ν *f*
by (*metis is-greatest-fixpoint-def is-least-fixpoint-def least-fixpoint greatest-fixpoint*)

lemma *pmu-below-pnu-fix*: *has-fixpoint f* $\wedge$ *has-least-prefixpoint f* $\wedge$ *has-greatest-postfixpoint f* $\rightarrow$ *p*μ *f* $\leq$ *p*ν *f*
by (*smt has-fixpoint-def is-fixpoint-def is-greatest-postfixpoint-def is-least-prefixpoint-def le-less order-trans least-prefixpoint greatest-postfixpoint*)

lemma *pmu-below-pnu-iso*: *isotone f* $\wedge$ *has-least-prefixpoint f* $\wedge$ *has-greatest-postfixpoint f* $\rightarrow$ *p*μ *f* $\leq$ *p*ν *f*
by (*metis has-fixpoint-def is-fixpoint-def is-least-fixpoint-def least-prefixpoint-fixpoint pmu-below-pnu-fix*)

definition *galois* :: ('*a* $\Rightarrow$ '*a*) $\Rightarrow$ ('*a* $\Rightarrow$ '*a*) $\Rightarrow$ *bool*
where *galois l u* $\leftrightarrow$ ($\forall$ *x y* . *l*(*x*) $\leq$ *y* $\leftrightarrow$ *x* $\leq$ *u*(*y*))

lemma *galois-char*: *galois l u* $\leftrightarrow$ ($\forall$ *x* . *x* $\leq$ *u*(*l*(*x*))) $\wedge$ ($\forall$ *x* . *l*(*u*(*x*)) $\leq$ *x*) $\wedge$ *isotone l* $\wedge$ *isotone u*
apply (*rule iffI*)
apply (*metis* (*full-types*) *galois-def isotone-def order-refl order-trans*)
by (*metis galois-def isotone-def order-trans*)

lemma *galois-closure*: *galois l u* $\rightarrow$ *l*(*x*) = *l*(*u*(*l*(*x*))) $\wedge$ *u*(*x*) = *u*(*l*(*u*(*x*)))

by (*smt antisym galois-char isotone-def*)

lemma *mu-fusion-1*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-fixpoint h* $\wedge$ $l(g(u(\mu\ h))) \leq h(l(u(\mu\ h))) \rightarrow l(p\mu\ g) \leq \mu\ h$
proof
assume *1*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-fixpoint h* $\wedge$ $l(g(u(\mu\ h))) \leq h(l(u(\mu\ h)))$
hence $l(g(u(\mu\ h))) \leq \mu\ h$
by (*metis galois-char least-fixpoint-same least-fixpoint-unique is-least-fixpoint-def isotone-def order-trans*)
thus $l(p\mu\ g) \leq \mu\ h$ **using** *1*
by (*metis galois-def least-prefixpoint is-least-prefixpoint-def least-fixpoint-same least-fixpoint-unique*)
qed

lemma *mu-fusion-2*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-fixpoint h* $\wedge$ $l \circ g \leq\leq h \circ l \rightarrow l(p\mu\ g) \leq \mu\ h$
by (*metis lifted-less-eq-def mu-fusion-1 o-apply*)

lemma *mu-fusion-equal-1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-fixpoint h* $\wedge$ $l(g(u(\mu\ h))) \leq h(l(u(\mu\ h)))$ $\wedge$ $l(g(p\mu\ g)) = h(l(p\mu\ g)) \rightarrow \mu\ h = l(p\mu\ g)$ $\wedge$ $\mu\ h = l(\mu\ g)$
by (*metis antisym least-fixpoint least-prefixpoint-fixpoint is-least-fixpoint-def mu-fusion-1 pmu-mu*)

lemma *mu-fusion-equal-2*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-prefixpoint h* $\wedge$ $l(g(u(\mu\ h))) \leq h(l(u(\mu\ h)))$ $\wedge$ $l(g(p\mu\ g)) = h(l(p\mu\ g)) \rightarrow p\mu\ h = l(p\mu\ g)$ $\wedge$ $\mu\ h = l(p\mu\ g)$
by (*smt antisym galois-char least-fixpoint-char least-prefixpoint least-prefixpoint-fixpoint is-least-prefixpoint-def isotone-def mu-fusion-1*)

lemma *mu-fusion-equal-3*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-fixpoint h* $\wedge$ $l \circ g = h \circ l \rightarrow \mu\ h = l(p\mu\ g)$ $\wedge$ $\mu\ h = l(\mu\ g)$
by (*metis mu-fusion-equal-1 o-apply order-refl*)

lemma *mu-fusion-equal-4*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint g* $\wedge$ *has-least-prefixpoint h* $\wedge$ $l \circ g = h \circ l \rightarrow p\mu\ h = l(p\mu\ g)$ $\wedge$ $\mu\ h = l(p\mu\ g)$
by (*metis mu-fusion-equal-2 o-apply order-refl*)

lemma *nu-fusion-1*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-fixpoint h* $\wedge$ $h(u(l(\nu\ h))) \leq u(g(l(\nu\ h))) \rightarrow \nu\ h \leq u(p\nu\ g)$
proof
assume *1*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-fixpoint h* $\wedge$ $h(u(l(\nu\ h))) \leq u(g(l(\nu\ h)))$
hence $\nu\ h \leq u(g(l(\nu\ h)))$ **using** *1*
by (*metis galois-char greatest-fixpoint-same greatest-fixpoint-unique is-greatest-fixpoint-def isotone-def order-trans*)
thus $\nu\ h \leq u(p\nu\ g)$ **using** *1*
by (*smt galois-def greatest-postfixpoint is-greatest-postfixpoint-def greatest-fixpoint-same greatest-fixpoint-unique*)
qed

lemma *nu-fusion-2*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-fixpoint h* $\wedge$ $h \circ u \leq\leq u \circ g \rightarrow \nu\ h \leq u(p\nu\ g)$
by (*metis lifted-less-eq-def nu-fusion-1 o-apply*)

lemma *nu-fusion-equal-1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-fixpoint h* $\wedge$ $h(u(l(\nu\ h))) \leq u(g(l(\nu\ h)))$ $\wedge$ $h(u(p\nu\ g)) = u(g(p\nu\ g)) \rightarrow \nu\ h = u(p\nu\ g)$ $\wedge$ $\nu\ h = u(\nu\ g)$
by (*metis antisym greatest-fixpoint greatest-postfixpoint-fixpoint is-greatest-fixpoint-def nu-fusion-1 pnu-nu*)

lemma *nu-fusion-equal-2*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-postfixpoint h* $\wedge$ $h(u(l(\nu\ h))) \leq u(g(l(\nu\ h)))$ $\wedge$ $h(u(p\nu\ g)) = u(g(p\nu\ g)) \rightarrow p\nu\ h = u(p\nu\ g)$ $\wedge$ $\nu\ h = u(p\nu\ g)$
by (*smt antisym galois-char greatest-fixpoint-char greatest-postfixpoint greatest-postfixpoint-fixpoint is-greatest-postfixpoint-def isotone-def nu-fusion-1*)

lemma *nu-fusion-equal-3*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-fixpoint h* $\wedge$ $h \circ u = u \circ g \rightarrow \nu\ h = u(p\nu\ g) \wedge \nu\ h = u(\nu\ g)$
by (*metis nu-fusion-equal-1 o-apply order-refl*)

lemma *nu-fusion-equal-4*: *galois l u* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint g* $\wedge$ *has-greatest-postfixpoint h* $\wedge$ $h \circ u = u \circ g \rightarrow p\nu\ h = u(p\nu\ g) \wedge \nu\ h = u(p\nu\ g)$
by (*metis nu-fusion-equal-2 o-apply order-refl*)

lemma *mu-exchange-1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint* $(l \circ h)$ $\wedge$ *has-least-prefixpoint* $(h \circ g)$ $\wedge$ *has-least-fixpoint* $(g \circ h)$ $\wedge$ $l \circ h \circ g \leq\leq g \circ h \circ l \rightarrow \mu(l \circ h) \leq \mu(g \circ h)$
proof
assume *1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint* $(l \circ h)$ $\wedge$ *has-least-prefixpoint* $(h \circ g)$ $\wedge$ *has-least-fixpoint* $(g \circ h)$ $\wedge$ $l \circ h \circ g \leq\leq g \circ h \circ l$
hence $l \circ (h \circ g) \leq\leq (g \circ h) \circ l$
by (*metis o-assoc*)
thus $\mu(l \circ h) \leq \mu(g \circ h)$ **using** *1*
by (*smt galois-char is-least-prefixpoint-def isotone-def least-fixpoint-char least-prefixpoint least-prefixpoint-fixpoint mu-fusion-2 mu-roll o-apply*)
qed

lemma *mu-exchange-2*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint* $(l \circ h)$ $\wedge$ *has-least-prefixpoint* $(h \circ l)$ $\wedge$ *has-least-prefixpoint* $(h \circ g)$ $\wedge$ *has-least-fixpoint* $(g \circ h)$ $\wedge$ *has-least-fixpoint* $(h \circ g)$ $\wedge$ $l \circ h \circ g \leq\leq g \circ h \circ l \rightarrow \mu(h \circ l) \leq \mu(h \circ g)$
by (*smt galois-char isotone-def least-fixpoint-char least-prefixpoint-fixpoint mu-exchange-1 mu-roll o-apply*)

lemma *mu-exchange-equal*: *galois l u* $\wedge$ *galois k t* $\wedge$ *isotone h* $\wedge$ *has-least-prefixpoint* $(l \circ h)$ $\wedge$ *has-least-prefixpoint* $(h \circ l)$ $\wedge$ *has-least-prefixpoint* $(k \circ h)$ $\wedge$ *has-least-prefixpoint* $(h \circ k)$ $\wedge$ $l \circ h \circ k = k \circ h \circ l \rightarrow \mu(l \circ h) = \mu(k \circ h) \wedge \mu(h \circ l) = \mu(h \circ k)$
by (*smt antisym galois-char isotone-def least-fixpoint-char least-prefixpoint-fixpoint lifted-reflexive mu-exchange-1 mu-exchange-2 o-apply*)

lemma *nu-exchange-1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint* $(u \circ h)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ g)$ $\wedge$ *has-greatest-fixpoint* $(g \circ h)$ $\wedge$ $g \circ h \circ u \leq\leq u \circ h \circ g$ $\rightarrow \nu(g \circ h) \leq \nu(u \circ h)$
proof
assume *1*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint* $(u \circ h)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ g)$ $\wedge$ *has-greatest-fixpoint* $(g \circ h)$ $\wedge$ $g \circ h \circ u \leq\leq u \circ h \circ g$
hence $(g \circ h) \circ u \leq\leq u \circ (h \circ g)$
by (*metis o-assoc*)
thus $\nu(g \circ h) \leq \nu(u \circ h)$ **using** *1*
by (*smt galois-char is-greatest-postfixpoint-def isotone-def greatest-fixpoint-char greatest-postfixpoint greatest-postfixpoint-fixpoint nu-fusion-2 nu-roll o-apply*)
qed

lemma *nu-exchange-2*: *galois l u* $\wedge$ *isotone g* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint* $(u \circ h)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ u)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ g)$ $\wedge$ *has-greatest-fixpoint* $(g \circ h)$ $\wedge$ *has-greatest-fixpoint* $(h \circ g)$ $\wedge$ $g \circ h \circ u \leq\leq u \circ h \circ g \rightarrow \nu(h \circ g) \leq \nu(h \circ u)$
by (*smt galois-char isotone-def greatest-fixpoint-char greatest-postfixpoint-fixpoint nu-exchange-1 nu-roll o-apply*)

lemma *nu-exchange-equal*: *galois l u* $\wedge$ *galois k t* $\wedge$ *isotone h* $\wedge$ *has-greatest-postfixpoint* $(u \circ h)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ u)$ $\wedge$ *has-greatest-postfixpoint* $(t \circ h)$ $\wedge$ *has-greatest-postfixpoint* $(h \circ t)$ $\wedge$ $u \circ h \circ t = t \circ h \circ u \rightarrow \nu(u \circ h) = \nu(t \circ h) \wedge \nu(h \circ u) = \nu(h \circ t)$
by (*smt antisym galois-char isotone-def greatest-fixpoint-char greatest-postfixpoint-fixpoint lifted-reflexive nu-exchange-1 nu-exchange-2 o-apply*)

lemma *mu-commute-fixpoint-1*: *isotone f* $\wedge$ *has-least-fixpoint* $(f \circ g)$ $\wedge$ $f \circ g = g \circ f \rightarrow$ *is-fixpoint f* $(\mu(f \circ g))$
by (*metis is-fixpoint-def mu-roll*)

lemma *mu-commute-fixpoint-2*: *isotone g* $\wedge$ *has-least-fixpoint* $(f \circ g)$ $\wedge$ $f \circ g = g \circ f \rightarrow$ *is-fixpoint g* $(\mu(f \circ g))$

by (*metis is-fixpoint-def mu-roll*)

lemma *mu-commute-least-fixpoint*: *isotone f* ∧ *isotone g* ∧ *has-least-fixpoint f* ∧ *has-least-fixpoint g* ∧ *has-least-fixpoint* (*f* ∘ *g*) ∧ *f* ∘ *g* = *g* ∘ *f* → (μ(*f* ∘ *g*) = μ *f* → μ *g* ≤ μ *f*)
by (*metis is-least-fixpoint-def least-fixpoint-same least-fixpoint-unique mu-roll*)

lemma *nu-commute-fixpoint-1*: *isotone f* ∧ *has-greatest-fixpoint* (*f* ∘ *g*) ∧ *f* ∘ *g* = *g* ∘ *f* → *is-fixpoint f* (ν(*f* ∘ *g*))
by (*metis is-fixpoint-def nu-roll*)

lemma *nu-commute-fixpoint-2*: *isotone g* ∧ *has-greatest-fixpoint* (*f* ∘ *g*) ∧ *f* ∘ *g* = *g* ∘ *f* → *is-fixpoint g* (ν(*f* ∘ *g*))
by (*metis is-fixpoint-def nu-roll*)

lemma *nu-commute-greatest-fixpoint*: *isotone f* ∧ *isotone g* ∧ *has-greatest-fixpoint f* ∧ *has-greatest-fixpoint g* ∧ *has-greatest-fixpoint* (*f* ∘ *g*) ∧ *f* ∘ *g* = *g* ∘ *f* → (ν(*f* ∘ *g*) = ν *f* → ν *f* ≤ ν *g*)
by (*smt is-greatest-fixpoint-def greatest-fixpoint-same greatest-fixpoint-unique nu-roll*)

lemma *mu-diagonal-1*: *isotone* (λ*x* . *f x x*) ∧ (∀ *x* . *isotone* (λ*y* . *f x y*)) ∧ *isotone* (λ*x* . μ(λ*y* . *f x y*)) ∧ (∀ *x* . *has-least-fixpoint* (λ*y* . *f x y*)) ∧ *has-least-prefixpoint* (λ*x* . μ(λ*y* . *f x y*)) → μ(λ*x* . *f x x*) = μ(λ*x* . μ(λ*y* . *f x y*))
by (*smt is-least-fixpoint-def is-least-prefixpoint-def least-fixpoint-same least-fixpoint-unique least-prefixpoint least-prefixpoint-fixpoint*)

lemma *mu-diagonal-2*: (∀ *x* . *isotone* (λ*y* . *f x y*) ∧ *isotone* (λ*y* . *f y x*) ∧ *has-least-prefixpoint* (λ*y* . *f x y*)) ∧ *has-least-prefixpoint* (λ*x* . μ(λ*y* . *f x y*)) → μ(λ*x* . *f x x*) = μ(λ*x* . μ(λ*y* . *f x y*))
proof
assume *1*: (∀ *x* . *isotone* (λ*y* . *f x y*) ∧ *isotone* (λ*y* . *f y x*) ∧ *has-least-prefixpoint* (λ*y* . *f x y*)) ∧ *has-least-prefixpoint* (λ*x* . μ(λ*y* . *f x y*))
hence *isotone* (λ*x* . μ(λ*y* . *f x y*))
by (*smt isotone-def lifted-less-eq-def mu-isotone*)
thus μ(λ*x* . *f x x*) = μ(λ*x* . μ(λ*y* . *f x y*)) **using** *1*
by (*smt is-least-fixpoint-def is-least-prefixpoint-def least-fixpoint-same least-prefixpoint least-prefixpoint-fixpoint*)
qed

lemma *nu-diagonal-1*: *isotone* (λ*x* . *f x x*) ∧ (∀ *x* . *isotone* (λ*y* . *f x y*)) ∧ *isotone* (λ*x* . ν(λ*y* . *f x y*)) ∧ (∀ *x* . *has-greatest-fixpoint* (λ*y* . *f x y*)) ∧ *has-greatest-postfixpoint* (λ*x* . ν(λ*y* . *f x y*)) → ν(λ*x* . *f x x*) = ν(λ*x* . ν(λ*y* . *f x y*))
by (*smt is-greatest-fixpoint-def is-greatest-postfixpoint-def greatest-fixpoint-same greatest-fixpoint-unique greatest-postfixpoint greatest-postfixpoint-fixpoint*)

lemma *nu-diagonal-2*: (∀ *x* . *isotone* (λ*y* . *f x y*) ∧ *isotone* (λ*y* . *f y x*) ∧ *has-greatest-postfixpoint* (λ*y* . *f x y*)) ∧ *has-greatest-postfixpoint* (λ*x* . ν(λ*y* . *f x y*)) → ν(λ*x* . *f x x*) = ν(λ*x* . ν(λ*y* . *f x y*))
proof
assume *1*: (∀ *x* . *isotone* (λ*y* . *f x y*) ∧ *isotone* (λ*y* . *f y x*) ∧ *has-greatest-postfixpoint* (λ*y* . *f x y*)) ∧ *has-greatest-postfixpoint* (λ*x* . ν(λ*y* . *f x y*))
hence *isotone* (λ*x* . ν(λ*y* . *f x y*))
by (*smt isotone-def lifted-less-eq-def nu-isotone*)
thus ν(λ*x* . *f x x*) = ν(λ*x* . ν(λ*y* . *f x y*)) **using** *1*
by (*smt greatest-fixpoint-same greatest-postfixpoint greatest-postfixpoint-fixpoint is-greatest-fixpoint-def is-greatest-postfixpoint-def*)
qed

end

class *tests* = *mult* + *neg* + *one* + *ord* + *plus* + *zero* +
  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-zero-def*: $0 = (THE\ x\ .\ (\forall\ y\ .\ x = -y\ ;\ --y))$ — define without imposing uniqueness
  assumes *one-def*: $1 = -\ 0$
  assumes *plus-def*: $-x + -y = -(--x\ ;\ --y)$
  assumes *leq-def*: $-x \leq -y \leftrightarrow -x\ ;\ -y = -x$
  assumes *strict-leq-def*: $-x < -y \leftrightarrow -x \leq -y \wedge \neg\ (-y \leq -x)$

begin

— uniqueness of 0, resulting in the lemma *zero-def* to replace the assumption *the-zero-def*

lemma *unique-zero*: $-x\ ;\ --x = -y\ ;\ --y$
  by (*metis sub-assoc sub-comm sub-compl*)

definition *is-zero* :: $'a \Rightarrow$ *bool*
  where *is-zero-def*: $is\text{-}zero(x) \equiv (\forall\ y\ .\ x = -y\ ;\ --y)$

lemma *the-zero-def-p*: $0 = (THE\ x\ .\ is\text{-}zero(x))$
  by (*simp only*: *the-zero-def is-zero-def*)

lemma *zero-def*: $0 = -x\ ;\ --x$
  by (*metis unique-zero the-zero-def-p is-zero-def theI'*)

— consequences for meet and complement

lemma *double-negation*: $-x = ---x$
  by (*metis sub-mult-closed sub-compl*)

lemma *compl-1*: $--x = -(-x\ ;\ -y)\ ;\ -(-x\ ;\ --y)$
  by (*metis double-negation sub-compl*)

lemma *right-zero*: $-x\ ;\ (-y\ ;\ --y) = -z\ ;\ --z$
  by (*metis compl-1 sub-assoc sub-mult-closed zero-def*)

lemma *right-one*: $-x\ ;\ -x = -x\ ;\ -(-y\ ;\ --y)$
  by (*metis compl-1 right-zero sub-mult-closed zero-def*)

lemma *mult-idempotent*: $-x\ ;\ -x = -x$

by (*metis compl-1 double-negation sub-assoc sub-mult-closed zero-def*)

lemma *compl-2*: $-x = -(-(-x\ ;\ -y)\ ;\ -(-x\ ;\ --y))$
by (*metis compl-1 double-negation*)

— consequences for join

lemma *plus-closed*: $-x + -y = --(-x + -y)$
by (*metis plus-def double-negation*)

lemma *plus-assoc*: $-x + (-y + -z) = (-x + -y) + -z$
by (*metis plus-def sub-assoc sub-mult-closed*)

lemma *plus-comm*: $-x + -y = -y + -x$
by (*metis plus-def sub-comm*)

lemma *plus-idempotent*: $-x + -x = -x$
by (*metis double-negation mult-idempotent plus-def*)

lemma *plus-absorb*: $-x\ ;\ -y + -x = -x$
by (*smt compl-1 mult-idempotent plus-def sub-assoc sub-mult-closed*)

lemma *mult-absorb*: $-x\ ;\ (-x + -y) = -x$
by (*smt plus-absorb plus-def sub-mult-closed sub-comm*)

lemma *plus-deMorgan*: $-(-x + -y) = --x\ ;\ --y$
by (*metis plus-def sub-mult-closed*)

lemma *mult-deMorgan*: $-(-x\ ;\ -y) = --x + --y$
by (*metis double-negation plus-def*)

lemma *mult-cases*: $-x = (-x + -y)\ ;\ (-x + --y)$
by (*metis compl-1 double-negation plus-def*)

lemma *plus-cases*: $-x = -x\ ;\ -y + -x\ ;\ --y$
by (*smt mult-deMorgan double-negation mult-cases sub-mult-closed*)

lemma *plus-compl-intro*: $(-x\ ;\ -y) + --x = -y + --x$
by (*smt compl-1 mult-deMorgan plus-absorb plus-cases sub-assoc sub-comm sub-mult-closed*)

lemma *mult-compl-intro*: $-x\ ;\ -y = -x\ ;\ (--x + -y)$
by (*metis sub-mult-closed mult-cases plus-absorb plus-compl-intro plus-comm*)

lemma *mult-distr-plus-left*: $-x\ ;\ (-y + -z) = (-x\ ;\ -y) + (-x\ ;\ -z)$
by (*smt mult-cases plus-absorb plus-assoc plus-comm plus-compl-intro plus-deMorgan plus-def sub-assoc sub-mult-closed*)

lemma *plus-distr-mult-left*: $-x + (-y\ ;\ -z) = (-x + -y)\ ;\ (-x + -z)$

by (*smt mult-deMorgan mult-distr-plus-left plus-def sub-mult-closed*)

lemma *mult-distr-plus-right*: $(-y + -z)\ ;\ -x = (-y\ ;\ -x) + (-z\ ;\ -x)$
by (*metis mult-distr-plus-left plus-def sub-comm*)

lemma *plus-distr-mult-right*: $(-y\ ;\ -z) + -x = (-y + -x)\ ;\ (-z + -x)$
by (*metis plus-distr-mult-left sub-mult-closed plus-comm*)

lemma *case-duality*: $(--x + -y)\ ;\ (-x + -z) = -x\ ;\ -y + --x\ ;\ -z$
proof −
have $(--x + -y)\ ;\ (-x + -z) = --(-y\ ;\ -z\ ;\ -x) + --(-y\ ;\ -x) + (--(-y\ ;\ -z\ ;\ --x) + --(--x\ ;\ -z))$
by (*smt mult-distr-plus-left plus-closed mult-compl-intro sub-comm plus-assoc plus-cases sub-mult-closed plus-comm*)
thus *?thesis*
by (*smt sub-comm sub-assoc sub-mult-closed plus-absorb*)
qed

lemma *case-duality-2*: $(-x + -y)\ ;\ (--x + -z) = -x\ ;\ -z + --x\ ;\ -y$
by (*metis case-duality double-negation plus-comm sub-mult-closed*)

lemma *compl-cases*: $(-v + -w)\ ;\ (--v + -x) + -((-v + -y)\ ;\ (--v + -z)) = (-v + -w + --y)\ ;\ (--v + -x + --z)$
by (*smt mult-deMorgan plus-deMorgan sub-mult-closed plus-closed double-negation case-duality sub-comm plus-assoc plus-comm mult-distr-plus-left case-duality-2*)

lemma *plus-cases-2*: $--x = -(-x + -y) + -(-x + --y)$
by (*metis mult-deMorgan plus-deMorgan double-negation mult-cases sub-mult-closed plus-closed*)

— consequences for 0 and 1

lemma *mult-compl*: $-x\ ;\ --x = 0$
by (*metis zero-def*)

lemma *plus-compl*: $-x + --x = 1$
by (*metis one-def plus-def zero-def*)

lemma *one-compl*: $-\ 1 = 0$
by (*metis mult-compl one-def sub-mult-closed*)

lemma *bs-mult-right-zero*: $-x\ ;\ 0 = 0$
by (*metis right-zero zero-def*)

lemma *bs-mult-left-zero*: $0\ ;\ -x = 0$
by (*metis bs-mult-right-zero one-compl sub-comm*)

lemma *plus-right-one*: $-x + 1 = 1$
by (*metis one-compl one-def mult-deMorgan double-negation bs-mult-right-zero*)

lemma *plus-left-one*: $1 + -x = 1$
by (*metis plus-right-one one-def plus-comm*)

lemma *bs-mult-right-one*: $-x \mathbin{;} 1 = -x$
by (*metis mult-compl one-def mult-idempotent right-one*)

lemma *bs-mult-left-one*: $1 \mathbin{;} -x = -x$
by (*metis one-def bs-mult-right-one sub-comm*)

lemma *plus-right-zero*: $-x + 0 = -x$
by (*metis mult-compl mult-cases plus-distr-mult-left*)

lemma *plus-left-zero*: $0 + -x = -x$
by (*metis plus-right-zero one-compl plus-comm*)

lemma *one-double-compl*: $-- \ 1 = 1$
by (*metis one-compl one-def*)

lemma *zero-double-compl*: $-- \ 0 = 0$
by (*metis one-compl one-def*)

— consequences for the order

lemma *reflexive*: $-x \leq -x$
by (*metis leq-def mult-idempotent*)

lemma *transitive*: $-x \leq -y \wedge -y \leq -z \rightarrow -x \leq -z$
by (*metis leq-def sub-assoc*)

lemma *antisymmetric*: $-x \leq -y \wedge -y \leq -x \rightarrow -x = -y$
by (*metis leq-def sub-comm*)

lemma *zero-least-test*: $0 \leq -x$
by (*metis one-compl leq-def bs-mult-right-zero sub-comm*)

lemma *one-greatest*: $-x \leq 1$
by (*metis leq-def one-def bs-mult-right-one*)

lemma *lower-bound-left*: $-x \mathbin{;} -y \leq -x$
by (*metis leq-def mult-idempotent sub-assoc sub-mult-closed sub-comm*)

lemma *lower-bound-right*: $-x \mathbin{;} -y \leq -y$
by (*metis leq-def mult-idempotent sub-assoc sub-mult-closed*)

lemma *mult-iso-left*: $-x \leq -y \rightarrow -x \mathbin{;} -z \leq -y \mathbin{;} -z$
by (*metis leq-def lower-bound-left sub-assoc sub-comm sub-mult-closed*)

lemma *mult-iso-right*: $-x \leq -y \rightarrow -z \mathbin{;} -x \leq -z \mathbin{;} -y$
by (*metis mult-iso-left sub-comm*)

lemma *mult-iso*: $-p \leq -q \wedge -r \leq -s \rightarrow -p\ ;\ -r \leq -q\ ;\ -s$
by (*smt transitive mult-iso-left mult-iso-right sub-mult-closed*)

lemma *compl-anti*: $-x \leq -y \rightarrow --y \leq --x$
by (*smt one-compl plus-compl plus-deMorgan double-negation leq-def plus-comm plus-compl-intro plus-right-zero*)

lemma *leq-plus*: $-x \leq -y \leftrightarrow -x + -y = -y$
by (*metis double-negation leq-def mult-absorb plus-def sub-comm*)

lemma *plus-compl-iso*: $-x \leq -y \rightarrow -(-y + -z) \leq -(-x + -z)$
by (*metis plus-deMorgan leq-plus lower-bound-left mult-iso-left*)

lemma *plus-iso-left*: $-x \leq -y \rightarrow -x + -z \leq -y + -z$
by (*metis plus-compl-iso compl-anti double-negation plus-def*)

lemma *plus-iso-right*: $-x \leq -y \rightarrow -z + -x \leq -z + -y$
by (*metis plus-iso-left plus-comm*)

lemma *plus-iso*: $-p \leq -q \wedge -r \leq -s \rightarrow -p + -r \leq -q + -s$
by (*smt transitive plus-iso-left plus-iso-right plus-closed*)

lemma *greatest-lower-bound*: $-x \leq -y \wedge -x \leq -z \leftrightarrow -x \leq -y\ ;\ -z$
by (*metis leq-def plus-absorb plus-comm sub-assoc sub-mult-closed*)

lemma *upper-bound-left*: $-x \leq -x + -y$
by (*metis one-compl plus-iso-right plus-right-zero zero-least-test*)

lemma *upper-bound-right*: $-y \leq -x + -y$
by (*metis upper-bound-left plus-comm*)

lemma *least-upper-bound*: $-x \leq -z \wedge -y \leq -z \leftrightarrow -x + -y \leq -z$
by (*metis leq-plus plus-assoc plus-def upper-bound-right*)

lemma *leq-mult-zero*: $-x \leq -y \leftrightarrow -x\ ;\ --y = 0$
proof −
have $-x \leq -y \rightarrow -x\ ;\ --y = 0$
by (*metis leq-def sub-assoc mult-compl bs-mult-right-zero*)
also have $-x\ ;\ --y = 0 \rightarrow -x \leq -y$
by (*metis compl-1 one-def leq-def bs-mult-right-one sub-mult-closed*)
ultimately show *?thesis* **by** *metis*
qed

lemma *leq-plus-right-one*: $-x \leq -y \leftrightarrow --x + -y = 1$
by (*metis one-compl one-def mult-deMorgan plus-deMorgan double-negation leq-mult-zero*)

lemma *shunting*: $-x\ ;\ -y \leq -z \leftrightarrow -y \leq --x + -z$

by (*smt leq-mult-zero sub-assoc sub-mult-closed sub-comm plus-deMorgan double-negation mult-deMorgan*)

lemma *shunting-right*: $-x\ ;\ -y \leq -z \leftrightarrow -x \leq -z + --y$
by (*metis plus-comm shunting sub-comm*)

lemma *leq-cases*: $-x\ ;\ -y \leq -z \wedge --x\ ;\ -y \leq -z \rightarrow -y \leq -z$
by (*smt least-upper-bound sub-mult-closed mult-distr-plus-left sub-comm plus-compl bs-mult-right-one*)

lemma *leq-cases-2*: $-x\ ;\ -y \leq -x\ ;\ -z \wedge --x\ ;\ -y \leq --x\ ;\ -z \rightarrow -y \leq -z$
by (*metis greatest-lower-bound leq-cases sub-mult-closed*)

lemma *leq-cases-3*: $-y\ ;\ -x \leq -z\ ;\ -x \wedge -y\ ;\ --x \leq -z\ ;\ --x \rightarrow -y \leq -z$
by (*metis leq-cases-2 sub-comm*)

lemma *eq-cases*: $-x\ ;\ -y = -x\ ;\ -z \wedge --x\ ;\ -y = --x\ ;\ -z \rightarrow -y = -z$
by (*metis plus-cases sub-comm*)

lemma *eq-cases-2*: $-y\ ;\ -x = -z\ ;\ -x \wedge -y\ ;\ --x = -z\ ;\ --x \rightarrow -y = -z$
by (*metis eq-cases sub-comm*)

lemma *wnf-lemma-1*: $(-x\ ;\ -y + --x\ ;\ -z)\ ;\ -x = -x\ ;\ -y$
by (*smt mult-compl mult-distr-plus-right mult-idempotent plus-right-zero sub-assoc sub-comm sub-mult-closed*)

lemma *wnf-lemma-2*: $(-x\ ;\ -y + -z\ ;\ --y)\ ;\ -y = -x\ ;\ -y$
by (*metis sub-comm wnf-lemma-1*)

lemma *wnf-lemma-3*: $(-x\ ;\ -z + --x\ ;\ -y)\ ;\ --x = --x\ ;\ -y$
by (*smt mult-compl mult-distr-plus-right mult-idempotent plus-comm plus-right-zero sub-assoc sub-comm sub-mult-closed*)

lemma *wnf-lemma-4*: $(-z\ ;\ -y + -x\ ;\ --y)\ ;\ --y = -x\ ;\ --y$
by (*metis sub-comm wnf-lemma-3*)

— sets and sequences of tests

definition *test-set* :: $'a\ set \Rightarrow bool$
where *test-set* $A \leftrightarrow (\forall\, x{\in}A\ .\ x = --x)$

lemma *mult-left-dist-test-set*: *test-set* $A \rightarrow$ *test-set* $\{\ -p\ ;\ x \mid x\ .\ x \in A\ \}$
by (*smt mem-Collect-eq sub-mult-closed test-set-def*)

lemma *mult-right-dist-test-set*: *test-set* $A \rightarrow$ *test-set* $\{\ x\ ;\ -p \mid x\ .\ x \in A\ \}$
by (*smt mem-Collect-eq sub-mult-closed test-set-def*)

lemma *plus-left-dist-test-set*: *test-set* $A \rightarrow$ *test-set* $\{\ -p + x \mid x\ .\ x \in A\ \}$
by (*smt mem-Collect-eq plus-closed test-set-def*)

lemma *plus-right-dist-test-set*: *test-set* $A \rightarrow$ *test-set* $\{\ x + -p \mid x\ .\ x \in A\ \}$

by (*smt mem-Collect-eq plus-closed test-set-def*)

lemma *test-set-closed*: $A \subseteq B \wedge$ *test-set* $B \rightarrow$ *test-set* A
by (*smt set-rev-mp test-set-def*)

definition *test-seq* :: $(nat \Rightarrow 'a) \Rightarrow bool$
where *test-seq* $t \leftrightarrow (\forall n\ .\ t\ n = --t\ n)$

lemma *test-seq-test-set*: *test-seq* $t \rightarrow$ *test-set* $\{\ t\ n \mid n{::}nat\ .\ True\ \}$
by (*smt mem-Collect-eq test-seq-def test-set-def*)

definition *nat-test* :: $(nat \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$
where *nat-test* $t\ s \leftrightarrow (\forall n\ .\ t\ n = --t\ n) \wedge s = --s \wedge (\forall n\ .\ t\ n \leq s) \wedge (\forall x\ y\ .\ (\forall n\ .\ t\ n\ ;\ -x \leq -y) \rightarrow s\ ;\ -x \leq -y)$

lemma *nat-test-seq*: *nat-test* $t\ s \rightarrow$ *test-seq* t
by (*metis nat-test-def test-seq-def*)

primrec *pSum* :: $(nat \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a$
where *pSum* $f\ 0 = 0$
| *pSum* $f\ (Suc\ n) =$ *pSum* $f\ n + f\ n$

lemma *pSum-test*: *test-seq* $t \rightarrow$ *pSum* $t\ n = --(pSum\ t\ n)$
apply (*induct n*)
apply (*metis pSum.simps*(1) *one-compl one-def*)
apply (*smt pSum.simps*(2) *plus-closed test-seq-def*)
done

lemma *pSum-test-nat*: *nat-test* $t\ s \rightarrow$ *pSum* $t\ n = --(pSum\ t\ n)$
by (*metis nat-test-seq pSum-test*)

lemma *pSum-upper*: *test-seq* $t \wedge i{<}n \rightarrow t\ i \leq pSum\ t\ n$
proof (*induct n*)
show *test-seq* $t \wedge i{<}0 \rightarrow t\ i \leq pSum\ t\ 0$
by (*smt less-zeroE*)
next
fix *n*
assume *test-seq* $t \wedge i{<}n \rightarrow t\ i \leq pSum\ t\ n$
hence *test-seq* $t \wedge i{<}n \rightarrow t\ i \leq pSum\ t\ (Suc\ n)$
by (*smt pSum.simps*(2) *pSum-test test-seq-def transitive upper-bound-left*)
thus *test-seq* $t \wedge i{<}Suc\ n \rightarrow t\ i \leq pSum\ t\ (Suc\ n)$
by (*metis pSum.simps*(2) *pSum-test test-seq-def upper-bound-right less-Suc-eq*)
qed

lemma *pSum-below*: *test-seq* $t \wedge (\forall m{<}n\ .\ t\ m\ ;\ -p \leq -q) \rightarrow pSum\ t\ n\ ;\ -p \leq -q$
apply (*induct n*)
apply (*metis bs-mult-left-zero pSum.simps*(1) *zero-least-test*)
apply (*smt least-upper-bound mult-distr-plus-right pSum.simps*(2) *pSum-test test-seq-def sub-mult-closed*)

done

lemma *pSum-below-nat*: *nat-test t s* $\wedge$ $(\forall\, m{<}n\ .\ t\ m\ ;\ -p \leq -q) \rightarrow$ *pSum t n* $;\ -p \leq -q$
by (*metis nat-test-seq pSum-below*)

lemma *pSum-below-sum*: *nat-test t s* $\rightarrow$ *pSum t n* $\leq s$
by (*smt bs-mult-right-one nat-test-def one-def pSum-below-nat pSum-test-nat*)

lemma *ascending-chain-plus-left*: *ascending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *ascending-chain* $(\lambda n\ .\ -p + t\ n)$ $\wedge$ *test-seq* $(\lambda n\ .\ -p + t\ n)$
by (*smt ascending-chain-def plus-closed plus-iso-right test-seq-def*)

lemma *ascending-chain-plus-right*: *ascending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *ascending-chain* $(\lambda n\ .\ t\ n + -p)$ $\wedge$ *test-seq* $(\lambda n\ .\ t\ n + -p)$
by (*smt ascending-chain-def plus-closed plus-iso-left test-seq-def*)

lemma *ascending-chain-mult-left*: *ascending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *ascending-chain* $(\lambda n\ .\ -p\ ;\ t\ n)$ $\wedge$ *test-seq* $(\lambda n\ .\ -p\ ;\ t\ n)$
by (*smt ascending-chain-def sub-mult-closed mult-iso-right test-seq-def*)

lemma *ascending-chain-mult-right*: *ascending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *ascending-chain* $(\lambda n\ .\ t\ n\ ;\ -p)$ $\wedge$ *test-seq* $(\lambda n\ .\ t\ n\ ;\ -p)$
by (*smt ascending-chain-def sub-mult-closed mult-iso-left test-seq-def*)

lemma *descending-chain-plus-left*: *descending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *descending-chain* $(\lambda n\ .\ -p + t\ n)$ $\wedge$ *test-seq* $(\lambda n\ .\ -p + t\ n)$
by (*smt descending-chain-def plus-closed plus-iso-right test-seq-def*)

lemma *descending-chain-plus-right*: *descending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *descending-chain* $(\lambda n\ .\ t\ n + -p)$ $\wedge$ *test-seq* $(\lambda n\ .\ t\ n + -p)$
by (*smt descending-chain-def plus-closed plus-iso-left test-seq-def*)

lemma *descending-chain-mult-left*: *descending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *descending-chain* $(\lambda n\ .\ -p\ ;\ t\ n)$ $\wedge$ *test-seq* $(\lambda n\ .\ -p\ ;\ t\ n)$
by (*smt descending-chain-def sub-mult-closed mult-iso-right test-seq-def*)

lemma *descending-chain-mult-right*: *descending-chain t* $\wedge$ *test-seq t* $\rightarrow$ *descending-chain* $(\lambda n\ .\ t\ n\ ;\ -p)$ $\wedge$ *test-seq* $(\lambda n\ .\ t\ n\ ;\ -p)$
by (*smt descending-chain-def sub-mult-closed mult-iso-left test-seq-def*)

end

class *il-semiring* = *mult* + *one* + *ord* + *plus* + *zero* +
assumes *add-associative* : $(x + y) + z = x + (y + z)$
assumes *add-commutative* : $x + y = y + x$
assumes *add-idempotent* : $x + x = x$
assumes *add-left-zero* : $0 + x = x$
assumes *mult-associative* : $(x\ ;\ y)\ ;\ z = x\ ;\ (y\ ;\ z)$
assumes *mult-left-one* : $1\ ;\ x = x$
assumes *mult-right-one* : $x\ ;\ 1 = x$
assumes *mult-left-sub-dist-add*: $x\ ;\ y + x\ ;\ z \leq x\ ;\ (y + z)$
assumes *mult-right-dist-add* : $(x + y)\ ;\ z = x\ ;\ z + y\ ;\ z$
assumes *mult-left-zero* : $0\ ;\ x = 0$

assumes *less-eq-def* : $x \leq y \leftrightarrow x + y = y$
assumes *less-def* : $x < y \leftrightarrow x \leq y \wedge \neg\ (y \leq x)$

begin

subclass *order*
by *unfold-locales* (*metis less-def*, *metis add-idempotent less-eq-def*, *metis add-associative less-eq-def*, *metis add-commutative less-eq-def*)

lemma *add-left-isotone*: $x \leq y \rightarrow x + z \leq y + z$
by (*smt add-associative add-commutative add-idempotent less-eq-def*)

lemma *add-right-isotone*: $x \leq y \rightarrow z + x \leq z + y$
by (*metis add-commutative add-left-isotone*)

lemma *add-isotone*: $w \leq y \wedge x \leq z \rightarrow w + x \leq y + z$
by (*smt add-associative add-commutative less-eq-def*)

lemma *add-left-upper-bound*: $x \leq x + y$
by (*metis add-associative add-idempotent less-eq-def*)

lemma *add-right-upper-bound*: $y \leq x + y$
by (*metis add-commutative add-left-upper-bound*)

lemma *add-least-upper-bound*: $x \leq z \wedge y \leq z \leftrightarrow x + y \leq z$
by (*smt add-associative add-commutative add-left-upper-bound less-eq-def*)

lemma *add-left-divisibility*: $x \leq y \leftrightarrow (\exists\, z\ .\ x + z = y)$
by (*metis add-left-upper-bound less-eq-def*)

lemma *add-right-divisibility*: $x \leq y \leftrightarrow (\exists\, z\ .\ z + x = y)$
by (*metis add-commutative add-left-divisibility*)

lemma *add-right-zero*: $x + 0 = x$
by (*metis add-commutative add-left-zero*)

lemma *zero-least*: $0 \leq x$
by (*metis add-left-upper-bound add-left-zero*)

lemma *mult-left-isotone*: $x \leq y \rightarrow x\ ;\ z \leq y\ ;\ z$
by (*metis less-eq-def mult-right-dist-add*)

lemma *mult-right-isotone*: $x \leq y \rightarrow z\ ;\ x \leq z\ ;\ y$
by (*metis add-least-upper-bound less-eq-def mult-left-sub-dist-add*)

lemma *mult-isotone*: $w \leq y \wedge x \leq z \rightarrow w\ ;\ x \leq y\ ;\ z$
by (*smt mult-left-isotone mult-right-isotone order-trans*)

lemma *mult-left-sub-dist-add-left*: $x \ ; \ y \leq x \ ; \ (y + z)$
by (*metis add-left-upper-bound mult-right-isotone*)

lemma *mult-left-sub-dist-add-right*: $x \ ; \ z \leq x \ ; \ (y + z)$
by (*metis add-right-upper-bound mult-right-isotone*)

lemma *mult-right-sub-dist-add-left*: $x \ ; \ z \leq (x + y) \ ; \ z$
by (*metis add-left-upper-bound mult-right-dist-add*)

lemma *mult-right-sub-dist-add-right*: $y \ ; \ z \leq (x + y) \ ; \ z$
by (*metis add-right-upper-bound mult-right-dist-add*)

lemma *case-split-left*: $1 \leq w + z \wedge w \ ; \ x \leq y \wedge z \ ; \ x \leq y \rightarrow x \leq y$
by (*smt add-associative add-commutative less-eq-def mult-left-one mult-right-dist-add order-refl*)

lemma *case-split-left-equal*: $w + z = 1 \wedge w \ ; \ x = w \ ; \ y \wedge z \ ; \ x = z \ ; \ y \rightarrow x = y$
by (*metis mult-left-one mult-right-dist-add*)

lemma *zero-right-mult-decreasing*: $x \ ; \ 0 \leq x$
by (*metis add-right-zero mult-left-sub-dist-add-right mult-right-one*)

lemma *add-same-context*:$x \leq y + z \wedge y \leq x + z \rightarrow x + z = y + z$
by (*smt add-associative add-commutative less-eq-def*)

lemma *ascending-chain-left-add*: *ascending-chain* $f \rightarrow$ *ascending-chain* $(\lambda n \ . \ x + f \ n)$
by (*metis ascending-chain-def add-right-isotone*)

lemma *ascending-chain-right-add*: *ascending-chain* $f \rightarrow$ *ascending-chain* $(\lambda n \ . \ f \ n + x)$
by (*metis ascending-chain-def add-left-isotone*)

lemma *ascending-chain-left-mult*: *ascending-chain* $f \rightarrow$ *ascending-chain* $(\lambda n \ . \ x \ ; f \ n)$
by (*metis ascending-chain-def mult-right-isotone*)

lemma *ascending-chain-right-mult*: *ascending-chain* $f \rightarrow$ *ascending-chain* $(\lambda n \ . \ f \ n \ ; \ x)$
by (*metis ascending-chain-def mult-left-isotone*)

lemma *descending-chain-left-add*: *descending-chain* $f \rightarrow$ *descending-chain* $(\lambda n \ . \ x + f \ n)$
by (*metis descending-chain-def add-right-isotone*)

lemma *descending-chain-right-add*: *descending-chain* $f \rightarrow$ *descending-chain* $(\lambda n \ . \ f \ n + x)$
by (*metis descending-chain-def add-left-isotone*)

lemma *descending-chain-left-mult*: *descending-chain* $f \rightarrow$ *descending-chain* $(\lambda n \ . \ x \ ; f \ n)$
by (*metis descending-chain-def mult-right-isotone*)

lemma *descending-chain-right-mult*: *descending-chain* $f \rightarrow$ *descending-chain* $(\lambda n \ . \ f \ n \ ; \ x)$
by (*metis descending-chain-def mult-left-isotone*)

end

class *il-semiring-T* = *il-semiring* +
  fixes $T :: 'a$ ($\top$)
  assumes *add-left-top*: $T + x = T$

begin

lemma *add-right-top*: $x + T = T$
  by (*metis add-commutative add-left-top*)

lemma *top-greatest*: $x \leq T$
  by (*metis add-left-top add-right-upper-bound*)

lemma *top-left-mult-increasing*: $x \leq T \ ; \ x$
  by (*metis mult-left-isotone mult-left-one top-greatest*)

lemma *top-right-mult-increasing*: $x \leq x \ ; \ T$
  by (*metis mult-right-isotone mult-right-one top-greatest*)

lemma *top-mult-top*: $T \ ; \ T = T$
  by (*metis add-right-divisibility add-right-top top-right-mult-increasing*)

definition *vector* :: $'a \Rightarrow$ *bool*
  where *vector* $x \leftrightarrow x = x \ ; \ T$

lemma *vector-zero*: *vector 0*
  by (*metis mult-left-zero vector-def*)

lemma *vector-top*: *vector T*
  by (*metis top-mult-top vector-def*)

lemma *vector-add-closed*: *vector* $x \wedge$ *vector* $y \rightarrow$ *vector* $(x + y)$
  by (*metis mult-right-dist-add vector-def*)

lemma *vector-left-mult-closed*: *vector* $y \rightarrow$ *vector* $(x \ ; \ y)$
  by (*metis mult-associative vector-def*)

end

class *il-conway-semiring* = *il-semiring* +
  fixes *circ* :: $'a \Rightarrow 'a$ ($-^\circ$ [*100*] *100*)
  assumes *circ-left-unfold*: $1 + x \ ; \ x^\circ = x^\circ$
  assumes *circ-left-slide*: $(x \ ; \ y)^\circ \ ; \ x \leq x \ ; \ (y \ ; \ x)^\circ$

assumes *circ-add-1*: $(x + y)^\circ = x^\circ \mathbin{;} (y \mathbin{;} x^\circ)^\circ$

begin

lemma *circ-mult-sub*: $1 + x \mathbin{;} (y \mathbin{;} x)^\circ \mathbin{;} y \leq (x \mathbin{;} y)^\circ$
by (*metis add-right-isotone circ-left-slide circ-left-unfold mult-associative mult-right-isotone*)

lemma *circ-right-unfold-sub*: $1 + x^\circ \mathbin{;} x \leq x^\circ$
by (*metis circ-mult-sub mult-left-one mult-right-one*)

lemma *circ-zero*: $0^\circ = 1$
by (*metis add-right-zero circ-left-unfold mult-left-zero*)

lemma *circ-increasing*: $x \leq x^\circ$
by (*metis add-right-upper-bound circ-left-unfold circ-right-unfold-sub mult-left-one mult-right-sub-dist-add-left order-trans*)

lemma *circ-reflexive*: $1 \leq x^\circ$
by (*metis add-left-divisibility circ-left-unfold*)

lemma *circ-transitive-equal*: $x^\circ \mathbin{;} x^\circ = x^\circ$
by (*metis add-idempotent circ-add-1 circ-left-unfold mult-associative*)

lemma *circ-circ-circ*: $x^{\circ\circ\circ} = x^{\circ\circ}$
by (*metis add-idempotent circ-add-1 circ-increasing circ-transitive-equal less-eq-def*)

lemma *circ-one*: $1^\circ = 1^{\circ\circ}$
by (*metis circ-circ-circ circ-zero*)

lemma *circ-add-sub*: $(x^\circ \mathbin{;} y)^\circ \mathbin{;} x^\circ \leq (x + y)^\circ$
by (*metis circ-add-1 circ-left-slide*)

lemma *circ-plus-one*: $x^\circ = 1 + x^\circ$
by (*metis less-eq-def circ-reflexive*)

lemma *circ-rtc-2*: $1 + x + x^\circ \mathbin{;} x^\circ = x^\circ$
by (*metis add-associative circ-increasing circ-plus-one circ-transitive-equal less-eq-def*)

lemma *mult-zero-circ*: $(x \mathbin{;} 0)^\circ = 1 + x \mathbin{;} 0$
by (*metis circ-left-unfold mult-associative mult-left-zero*)

lemma *mult-zero-add-circ*: $(x + y \mathbin{;} 0)^\circ = x^\circ \mathbin{;} (y \mathbin{;} 0)^\circ$
by (*metis circ-add-1 mult-associative mult-left-zero*)

lemma *circ-plus-sub*: $x^\circ \mathbin{;} x \leq x \mathbin{;} x^\circ$
by (*metis circ-left-slide mult-left-one mult-right-one*)

lemma *circ-loop-fixpoint*: $y \mathbin{;} (y^\circ \mathbin{;} z) + z = y^\circ \mathbin{;} z$

by (*metis add-commutative circ-left-unfold mult-associative mult-left-one mult-right-dist-add*)

lemma *left-plus-below-circ*: $x \ ; \ x^\circ \leq x^\circ$
by (*metis add-right-upper-bound circ-left-unfold*)

lemma *right-plus-below-circ*: $x^\circ \ ; \ x \leq x^\circ$
by (*metis add-least-upper-bound circ-right-unfold-sub*)

lemma *circ-add-upper-bound*: $x \leq z^\circ \wedge y \leq z^\circ \rightarrow x + y \leq z^\circ$
by (*metis add-least-upper-bound*)

lemma *circ-mult-upper-bound*: $x \leq z^\circ \wedge y \leq z^\circ \rightarrow x \ ; \ y \leq z^\circ$
by (*metis mult-isotone circ-transitive-equal*)

lemma *circ-sub-dist*: $x^\circ \leq (x + y)^\circ$
by (*metis circ-add-sub circ-plus-one mult-left-one mult-right-sub-dist-add-left order-trans*)

lemma *circ-sub-dist-1*: $x \leq (x + y)^\circ$
by (*metis add-least-upper-bound circ-increasing*)

lemma *circ-sub-dist-2*: $x \ ; \ y \leq (x + y)^\circ$
by (*metis add-commutative circ-mult-upper-bound circ-sub-dist-1*)

lemma *circ-sub-dist-3*: $x^\circ \ ; \ y^\circ \leq (x + y)^\circ$
by (*metis add-commutative circ-mult-upper-bound circ-sub-dist*)

lemma *circ-isotone*: $x \leq y \rightarrow x^\circ \leq y^\circ$
by (*metis circ-sub-dist less-eq-def*)

lemma *circ-add-2*: $(x + y)^\circ \leq (x^\circ \ ; \ y^\circ)^\circ$
by (*metis add-least-upper-bound circ-increasing circ-isotone circ-reflexive mult-isotone mult-left-one mult-right-one*)

lemma *circ-sup-one-left-unfold*: $1 \leq x \rightarrow x \ ; \ x^\circ = x^\circ$
by (*metis antisym less-eq-def mult-left-one mult-right-sub-dist-add-left left-plus-below-circ*)

lemma *circ-sup-one-right-unfold*: $1 \leq x \rightarrow x^\circ \ ; \ x = x^\circ$
by (*metis antisym less-eq-def mult-left-sub-dist-add-left mult-right-one right-plus-below-circ*)

lemma *circ-decompose-4*: $(x^\circ \ ; \ y^\circ)^\circ = x^\circ \ ; \ (y^\circ \ ; \ x^\circ)^\circ$
by (*metis add-associative add-commutative circ-add-1 circ-loop-fixpoint circ-plus-one circ-rtc-2 circ-transitive-equal mult-associative*)

lemma *circ-decompose-5*: $(x^\circ \ ; \ y^\circ)^\circ = (y^\circ \ ; \ x^\circ)^\circ$
by (*smt add-associative add-commutative add-left-zero circ-add-1 circ-decompose-4 mult-left-zero mult-right-one*)

lemma *circ-decompose-6*: $x^\circ \ ; \ (y \ ; \ x^\circ)^\circ = y^\circ \ ; \ (x \ ; \ y^\circ)^\circ$
by (*metis add-commutative circ-add-1*)

lemma *circ-decompose-7*: $(x + y)^\circ = x^\circ \ ; \ y^\circ \ ; \ (x + y)^\circ$
by (*metis circ-add-1 circ-decompose-6 circ-transitive-equal mult-associative*)

lemma *circ-decompose-8*: $(x + y)^\circ = (x + y)^\circ \ ; \ x^\circ \ ; \ y^\circ$
by (*metis antisym eq-refl mult-associative mult-isotone mult-right-one circ-mult-upper-bound circ-reflexive circ-sub-dist-3*)

lemma *circ-decompose-9*: $(x^\circ \ ; \ y^\circ)^\circ = x^\circ \ ; \ y^\circ \ ; \ (x^\circ \ ; \ y^\circ)^\circ$
by (*metis circ-decompose-4 mult-associative*)

lemma *circ-decompose-10*: $(x^\circ \ ; \ y^\circ)^\circ = (x^\circ \ ; \ y^\circ)^\circ \ ; \ x^\circ \ ; \ y^\circ$
by (*metis add-right-upper-bound circ-loop-fixpoint circ-reflexive circ-sup-one-right-unfold mult-associative order-trans*)

lemma *circ-back-loop-prefixpoint*: $(z \ ; \ y^\circ) \ ; \ y + z \leq z \ ; \ y^\circ$
by (*metis add-least-upper-bound circ-left-unfold mult-associative mult-left-sub-dist-add-left mult-right-isotone mult-right-one right-plus-below-circ*)

lemma *circ-loop-is-fixpoint*: *is-fixpoint* $(\lambda x \ . \ y \ ; \ x + z) \ (y^\circ \ ; \ z)$
by (*metis circ-loop-fixpoint is-fixpoint-def*)

lemma *circ-back-loop-is-fixpoint*: *is-prefixpoint* $(\lambda x \ . \ x \ ; \ y + z) \ (z \ ; \ y^\circ)$
by (*metis circ-back-loop-prefixpoint is-prefixpoint-def*)

lemma *circ-circ-add*: $(1 + x)^\circ = x^{\circ\circ}$
by (*metis add-commutative circ-add-1 circ-decompose-4 circ-zero mult-right-one*)

lemma *circ-circ-mult-sub*: $x^\circ \ ; \ 1^\circ \leq x^{\circ\circ}$
by (*metis circ-increasing circ-isotone circ-mult-upper-bound circ-reflexive*)

lemma *left-plus-circ*: $(x \ ; \ x^\circ)^\circ = x^\circ$
by (*smt add-idempotent circ-add-1 circ-loop-fixpoint circ-transitive-equal mult-right-dist-add*)

lemma *right-plus-circ*: $(x^\circ \ ; \ x)^\circ = x^\circ$
by (*metis add-commutative circ-isotone circ-loop-fixpoint circ-plus-sub circ-sub-dist eq-iff left-plus-circ*)

lemma *circ-square*: $(x \ ; \ x)^\circ \leq x^\circ$
by (*metis circ-increasing circ-isotone left-plus-circ mult-right-isotone*)

lemma *circ-mult-sub-add*: $(x \ ; \ y)^\circ \leq (x + y)^\circ$
by (*metis add-left-upper-bound add-right-upper-bound circ-isotone circ-square mult-isotone order-trans*)

lemma *circ-add-mult-zero*: $x^\circ \ ; \ y = (x + y \ ; \ 0)^\circ \ ; \ y$
proof −
have $(x + y \ ; \ 0)^\circ \ ; \ y = x^\circ \ ; \ (1 + y \ ; \ 0) \ ; \ y$
by (*metis mult-zero-add-circ mult-zero-circ*)
also have $\ldots = x^\circ \ ; \ (y + y \ ; \ 0)$
by (*metis mult-associative mult-left-one mult-left-zero mult-right-dist-add*)
also have $\ldots = x^\circ \ ; \ y$
by (*metis add-commutative less-eq-def zero-right-mult-decreasing*)

finally show *?thesis*
by *metis*
qed

lemma *troeger-1*: $(x + y)^\circ = x^\circ \ ; (1 + y \ ; (x + y)^\circ)$
by (*metis circ-add-1 circ-left-unfold mult-associative*)

lemma *troeger-2*: $(x + y)^\circ \ ; z = x^\circ \ ; (y \ ; (x + y)^\circ \ ; z + z)$
by (*metis circ-add-1 circ-loop-fixpoint mult-associative*)

lemma *troeger-3*: $(x + y \ ; 0)^\circ = x^\circ \ ; (1 + y \ ; 0)$
by (*metis mult-zero-add-circ mult-zero-circ*)

lemma *circ-add-sub-add-one-1*: $x + y \leq x^\circ \ ; (1 + y)$
by (*smt add-associative add-commutative add-idempotent circ-increasing circ-loop-fixpoint less-eq-def mult-left-sub-dist-add-left mult-right-one*)

lemma *circ-add-sub-add-one-2*: $x^\circ \ ; (x + y) \leq x^\circ \ ; (1 + y)$
by (*metis circ-add-sub-add-one-1 circ-transitive-equal mult-associative mult-right-isotone*)

lemma *circ-add-sub-add-one*: $x \ ; x^\circ \ ; (x + y) \leq x \ ; x^\circ \ ; (1 + y)$
by (*metis circ-add-sub-add-one-2 mult-associative mult-right-isotone*)

lemma *circ-square-2*: $(x \ ; x)^\circ \ ; (x + 1) \leq x^\circ$
by (*metis add-least-upper-bound circ-increasing circ-mult-upper-bound circ-reflexive circ-square*)

lemma *circ-extra-circ*: $(y \ ; x^\circ)^\circ = (y \ ; y^\circ \ ; x^\circ)^\circ$
by (*smt add-commutative add-idempotent circ-add-1 circ-left-unfold mult-associative*)

lemma *circ-circ-sub-mult*: $1^\circ \ ; x^\circ \leq x^{\circ\circ}$
by (*metis circ-increasing circ-isotone circ-mult-upper-bound circ-reflexive*)

lemma *circ-decompose-11*: $(x^\circ \ ; y^\circ)^\circ = (x^\circ \ ; y^\circ)^\circ \ ; x^\circ$
by (*smt circ-add-1 circ-circ-add circ-decompose-4 circ-decompose-8 circ-rtc-2 circ-transitive-equal mult-associative*)

end

class *il-conway-semiring-T* = *il-semiring-T* + *il-conway-semiring*

begin

lemma *circ-top*: $T^\circ = T$
by (*metis add-right-top antisym circ-left-unfold mult-left-sub-dist-add-left mult-right-one top-greatest*)

lemma *circ-right-top*: $x^\circ \ ; \ T = T$
by (*metis add-right-top circ-loop-fixpoint*)

lemma *circ-left-top*: $T \ ; \ x^\circ = T$
by (*metis antisym circ-plus-one mult-left-sub-dist-add-left mult-right-one top-greatest*)

lemma *mult-top-circ*: $(x \ ; \ T)^\circ = 1 + x \ ; \ T$
by (*metis circ-left-top circ-left-unfold mult-associative*)

end

class *il-conway-semiring-L* = *il-conway-semiring* +
fixes $L :: 'a$
assumes *one-circ-mult-split*: $1^\circ \ ; \ x = L + x$
assumes *L-split-add*: $x \ ; \ (y + L) \leq x \ ; \ y + L$

begin

lemma *L-def*: $L = 1^\circ \ ; \ 0$
by (*metis add-right-zero one-circ-mult-split*)

lemma *one-circ-split*: $1^\circ = L + 1$
by (*metis mult-right-one one-circ-mult-split*)

lemma *one-circ-circ-split*: $1^{\circ\circ} = L + 1$
by (*metis circ-one one-circ-split*)

lemma *sub-mult-one-circ*: $x \ ; \ 1^\circ \leq 1^\circ \ ; \ x$
by (*metis L-split-add add-commutative mult-right-one one-circ-mult-split*)

lemma *one-circ-mult-split-2*: $1^\circ \ ; \ x = x \ ; \ 1^\circ + L$
by (*smt add-associative add-commutative add-least-upper-bound circ-back-loop-prefixpoint less-eq-def one-circ-mult-split sub-mult-one-circ*)

lemma *sub-mult-one-circ-split*: $x \ ; \ 1^\circ \leq x + L$
by (*metis add-commutative one-circ-mult-split sub-mult-one-circ*)

lemma *sub-mult-one-circ-split-2*: $x \ ; \ 1^\circ \leq x + 1^\circ$
by (*metis L-def add-right-isotone order-trans sub-mult-one-circ-split zero-right-mult-decreasing*)

lemma *L-split*: $x \ ; \ L \leq x \ ; \ 0 + L$
by (*metis L-split-add add-left-zero*)

lemma *L-left-zero*: $L \ ; \ x = L$
by (*metis L-def mult-associative mult-left-zero*)

lemma *one-circ-L*: $1^\circ \ ; \ L = L$
by (*metis L-def circ-transitive-equal mult-associative*)

lemma *mult-L-circ*: $(x\ ;\ L)^\circ = 1 + x\ ;\ L$
by (*metis L-left-zero circ-left-unfold mult-associative*)

lemma *mult-L-circ-mult*: $(x\ ;\ L)^\circ\ ;\ y = y + x\ ;\ L$
by (*metis L-left-zero mult-L-circ mult-associative mult-left-one mult-right-dist-add*)

lemma *circ-L*: $L^\circ = L + 1$
by (*metis L-left-zero add-commutative circ-left-unfold*)

lemma *L-below-one-circ*: $L \leq 1^\circ$
by (*metis L-def zero-right-mult-decreasing*)

lemma *circ-circ-mult-1*: $x^\circ\ ;\ 1^\circ = x^{\circ\circ}$
by (*metis L-left-zero add-commutative circ-add-1 circ-circ-add mult-zero-circ one-circ-split*)

lemma *circ-circ-mult*: $1^\circ\ ;\ x^\circ = x^{\circ\circ}$
by (*metis antisym circ-circ-mult-1 circ-circ-sub-mult sub-mult-one-circ*)

lemma *circ-circ-split*: $x^{\circ\circ} = L + x^\circ$
by (*metis circ-circ-mult one-circ-mult-split*)

lemma *circ-add-6*: $L + (x + y)^\circ = (x^\circ\ ;\ y^\circ)^\circ$
by (*metis add-associative add-commutative circ-add-1 circ-circ-add circ-circ-split circ-decompose-4*)

end

class *star* =
fixes *star* :: $'a \Rightarrow 'a$ ($-^*$ [*100*] *100*)

class *lka* = *il-semiring* + *star* +
assumes *star-left-unfold* : $1 + y\ ;\ y^* \leq y^*$
assumes *star-left-induct* : $z + y\ ;\ x \leq x \rightarrow y^*\ ;\ z \leq x$

begin

lemma *star-left-unfold-equal*: $1 + x\ ;\ x^* = x^*$
by (*smt add-right-isotone antisym mult-right-isotone mult-right-one star-left-induct star-left-unfold*)

lemma *star-left-slide*: $(x\ ;\ y)^*\ ;\ x \leq x\ ;\ (y\ ;\ x)^*$
by (*metis mult-associative mult-left-sub-dist-add mult-right-one star-left-induct star-left-unfold-equal*)

lemma *star-isotone*: $x \leq y \rightarrow x^* \leq y^*$

by (*metis add-right-isotone mult-left-isotone order-trans star-left-unfold mult-right-one star-left-induct*)

lemma *star-add-1-sub*: $x^* \,;\, (y \,;\, x^*)^* \leq (x + y)^*$
proof −
have $x^* \,;\, (y \,;\, x^*)^* \leq x^* \,;\, (y \,;\, (x + y)^*)^*$
by (*smt add-left-upper-bound mult-right-isotone star-isotone*)
also have $\ldots \leq x^* \,;\, ((x + y) \,;\, (x + y)^*)^*$
by (*smt add-right-upper-bound mult-left-isotone mult-right-isotone star-isotone*)
also have $\ldots \leq x^* \,;\, (x + y)^{**}$
by (*smt add-least-upper-bound mult-right-isotone star-isotone star-left-unfold*)
also have $\ldots \leq (x + y)^* \,;\, (x + y)^{**}$
by (*smt add-left-upper-bound mult-left-isotone star-isotone*)
also have $\ldots \leq (x + y)^*$
by (*smt add-least-upper-bound mult-right-one star-left-induct star-left-unfold*)
finally show $x^* \,;\, (y \,;\, x^*)^* \leq (x + y)^*$
by *smt*
qed

lemma *star-add-1*: $(x + y)^* = x^* \,;\, (y \,;\, x^*)^*$
apply (*rule antisym*)
apply (*smt add-least-upper-bound add-left-upper-bound add-right-upper-bound mult-associative mult-left-one mult-right-dist-add mult-right-one star-left-induct star-left-unfold-equal*)
apply (*smt star-add-1-sub*)
done

end

sublocale *lka* < *star*!: *il-conway-semiring* **where** *circ* = *star*
apply *unfold-locales*
apply (*metis star-left-unfold-equal*)
apply (*metis star-left-slide*)
apply (*metis star-add-1*)
done

context *lka*

begin

lemma *star-sub-one*: $x \leq 1 \rightarrow x^* = 1$
by (*metis add-right-isotone eq-iff less-eq-def mult-right-one star.circ-plus-one star-left-induct*)

lemma *star-one*: $1^* = 1$
by (*metis eq-iff star-sub-one*)

lemma *star-left-induct-mult*: $x \,;\, y \leq y \rightarrow x^* \,;\, y \leq y$
by (*metis add-commutative less-eq-def order-refl star-left-induct*)

lemma *star-left-induct-mult-iff*: $x \,;\, y \leq y \leftrightarrow x^* \,;\, y \leq y$

by (*metis mult-associative mult-left-isotone mult-left-one mult-right-isotone order-trans star-left-induct-mult star.circ-reflexive star.left-plus-below-circ*)

lemma *star-involutive*: $x^* = x^{**}$
by (*smt antisym less-eq-def mult-left-sub-dist-add-left mult-right-one star-left-induct star.circ-plus-one star.left-plus-below-circ star.circ-transitive-equal*)

lemma *star-sup-one*: $(1 + x)^* = x^*$
by (*metis star.circ-circ-add star-involutive*)

lemma *star-left-induct-equal*: $z + x \ ; \ y = y \rightarrow x^* \ ; \ z \leq y$
by (*metis order-refl star-left-induct*)

lemma *star-left-induct-mult-equal*: $x \ ; \ y = y \rightarrow x^* \ ; \ y \leq y$
by (*metis order-refl star-left-induct-mult*)

lemma *star-star-upper-bound*: $x^* \leq z^* \rightarrow x^{**} \leq z^*$
by (*metis star-involutive*)

lemma *star-simulation-left*: $x \ ; \ z \leq z \ ; \ y \rightarrow x^* \ ; \ z \leq z \ ; \ y^*$
by (*smt add-commutative add-least-upper-bound mult-right-dist-add less-eq-def mult-associative mult-right-one star.left-plus-below-circ star.circ-increasing star-left-induct star-involutive star.circ-isotone star.circ-reflexive mult-left-sub-dist-add-left*)

lemma *quasicomm-1*: $y \ ; \ x \leq x \ ; \ (x + y)^* \leftrightarrow y^* \ ; \ x \leq x \ ; \ (x + y)^*$
by (*smt mult-isotone order-refl order-trans star.circ-increasing star-involutive star-simulation-left*)

lemma *star-rtc-3*: $1 + x + y \ ; \ y = y \rightarrow x^* \leq y$
by (*metis add-least-upper-bound less-eq-def mult-left-sub-dist-add-left mult-right-one star-left-induct-mult-iff star.circ-sub-dist*)

lemma *star-decompose-1*: $(x + y)^* = (x^* \ ; \ y^*)^*$
apply (*rule antisym*)
apply (*smt add-least-upper-bound mult-isotone mult-left-one mult-right-one star.circ-increasing star.circ-isotone star.circ-reflexive*)
apply (*smt star.circ-isotone star.circ-sub-dist-3 star-involutive*)
done

lemma *star-sum*: $(x + y)^* \ = (x^* + y^*)^*$
by (*metis star-decompose-1 star-involutive*)

lemma *star-decompose-3*: $(x^* \ ; \ y^*)^* = x^* \ ; \ (y \ ; \ x^*)^*$
by (*metis star-decompose-1 star.circ-add-1*)

lemma *star-loop-least-fixpoint*: $y \ ; \ x + z = x \rightarrow y^* \ ; \ z \leq x$
by (*metis add-commutative star-left-induct-equal*)

lemma *star-loop-is-least-fixpoint*: *is-least-fixpoint* $(\lambda x \ . \ y \ ; \ x + z) \ (y^* \ ; \ z)$
by (*smt is-least-fixpoint-def star.circ-loop-fixpoint star-loop-least-fixpoint*)

lemma *star-loop-mu*: $\mu \ (\lambda x \ . \ y \ ; \ x + z) = y^* \ ; \ z$
by (*metis least-fixpoint-same star-loop-is-least-fixpoint*)

end

class *strong-lka* = *lka* +
  assumes *star-right-induct*: $z + x \ ; \ y \leq x \rightarrow z \ ; \ y^* \leq x$

begin

Most lemmas in this class are taken from Georg Struth's Isabelle theories.

lemma *star-plus*: $y^* \ ; \ y = y \ ; \ y^*$
  by (*smt add-least-upper-bound antisym less-eq-def mult-left-one mult-right-dist-add star.circ-plus-sub star-left-unfold star-right-induct*)

lemma *star-right-unfold*: $1 + y^* \ ; \ y = y^*$
  by (*metis star-left-unfold-equal star-plus*)

lemma *star-slide*: $(x \ ; \ y)^* \ ; \ x = x \ ; \ (y \ ; \ x)^*$
  by (*smt add-least-upper-bound antisym mult-associative mult-left-isotone mult-left-one mult-right-one order-trans star-left-slide star-left-unfold star-right-induct*)

lemma *star-unfold-slide*: $(x \ ; \ y)^* = 1 + x \ ; \ (y \ ; \ x)^* \ ; \ y$
  by (*metis mult-associative star.circ-left-unfold star-slide*)

lemma *star-decompose*: $(x + y)^* = (x^* \ ; \ y)^* \ ; \ x^*$
  by (*metis star.circ-add-1 star-slide*)

lemma *star-simulation-right*: $z \ ; \ x \leq y \ ; \ z \rightarrow z \ ; \ x^* \leq y^* \ ; \ z$
  by (*smt add-commutative add-least-upper-bound add-left-upper-bound mult-associative order-trans star.circ-loop-fixpoint star-left-induct star-plus star-right-induct*)

lemma *star-right-induct-mult*: $y \ ; \ x \leq y \rightarrow y \ ; \ x^* \leq y$
  by (*metis add-commutative less-eq-def order-refl star-right-induct*)

lemma *star-right-induct-mult-iff*: $y \ ; \ x \leq y \leftrightarrow y \ ; \ x^* \leq y$
  by (*metis mult-right-isotone order-trans star.circ-increasing star-right-induct-mult*)

lemma *star-simulation-right-equal*: $z \ ; \ x = y \ ; \ z \rightarrow z \ ; \ x^* = y^* \ ; \ z$
  by (*metis eq-iff star-simulation-left star-simulation-right*)

lemma *star-simulation-star*: $x \ ; \ y \leq y \ ; \ x \rightarrow x^* \ ; \ y^* \leq y^* \ ; \ x^*$
  by (*metis star-simulation-left star-simulation-right*)

lemma *star-right-induct-equal*: $z + y \ ; \ x = y \rightarrow z \ ; \ x^* \leq y$
  by (*metis order-refl star-right-induct*)

lemma *star-right-induct-mult-equal*: $y \ ; \ x = y \rightarrow y \ ; \ x^* \leq y$
  by (*metis order-refl star-right-induct-mult*)

lemma *star-back-loop-least-fixpoint*: $x \ ; \ y + z = x \rightarrow z \ ; \ y^* \leq x$
  by (*metis add-commutative star-right-induct-equal*)

lemma *star-back-loop-is-least-fixpoint*: *is-least-fixpoint* $(\lambda x\ .\ x\ ;\ y + z)\ (z\ ;\ y^*)$
by (*smt add-commutative add-right-isotone antisym is-least-fixpoint-def mult-left-isotone star.circ-back-loop-prefixpoint star-back-loop-least-fixpoint star-right-induct*)

lemma *star-back-loop-mu*: $\mu\ (\lambda x\ .\ x\ ;\ y + z) = z\ ;\ y^*$
by (*metis least-fixpoint-same star-back-loop-is-least-fixpoint*)

lemma *star-square*: $x^* = (1 + x)\ ;\ (x\ ;\ x)^*$
proof −
let *?f* $= \lambda y\ .\ y\ ;\ x + 1$
have *1*: *isotone ?f*
by (*smt add-left-isotone isotone-def mult-left-isotone*)
have *2*: *?f* $\circ$ *?f* $= (\lambda y\ .\ y\ ;\ (x\ ;\ x) + (1 + x))$
by (*simp add*: *add-associative add-commutative mult-associative mult-left-one mult-right-dist-add o-def*)
thus *?thesis* **using** *1*
by (*metis mu-square mult-left-one star-back-loop-mu has-least-fixpoint-def star-back-loop-is-least-fixpoint*)
qed

lemma *star-square-2*: $x^* = (x\ ;\ x)^*\ ;\ (x + 1)$
by (*smt add-commutative antisym mult-left-one mult-left-sub-dist-add mult-right-dist-add mult-right-one star.circ-square-2 star-slide star-square*)

lemma *star-circ-simulate-right-plus*: $z\ ;\ x \leq y\ ;\ y^*\ ;\ z + w \rightarrow z\ ;\ x^* \leq y^*\ ;\ (z + w\ ;\ x^*)$
proof
assume *1*: $z\ ;\ x \leq y\ ;\ y^*\ ;\ z + w$
have $(z + w\ ;\ x^*)\ ;\ x \leq z\ ;\ x + w\ ;\ x^*$
by (*metis add-right-isotone mult-associative mult-right-dist-add mult-right-isotone star.circ-increasing star.circ-transitive-equal*)
also have $\ldots \leq y\ ;\ y^*\ ;\ z + w + w\ ;\ x^*$ **using** *1*
by (*metis add-left-isotone*)
also have $\ldots \leq y\ ;\ y^*\ ;\ z + w\ ;\ x^*$
by (*metis add-least-upper-bound add-right-isotone add-right-upper-bound star.circ-back-loop-prefixpoint*)
also have $\ldots \leq y^*\ ;\ (z + w\ ;\ x^*)$
by (*metis add-least-upper-bound mult-isotone mult-left-isotone mult-left-one mult-left-sub-dist-add-left star.circ-reflexive star.left-plus-below-circ*)
finally have $y^*\ ;\ (z + w\ ;\ x^*)\ ;\ x \leq y^*\ ;\ (z + w\ ;\ x^*)$
by (*metis mult-associative mult-right-isotone star.circ-transitive-equal*)
thus $z\ ;\ x^* \leq y^*\ ;\ (z + w\ ;\ x^*)$
by (*metis add-least-upper-bound star-right-induct mult-left-sub-dist-add-left star.circ-loop-fixpoint*)
qed

end

class *lk-conway-semiring* = *lka* + *il-conway-semiring*

begin

lemma *star-below-circ*: $x^* \leq x^\circ$

by (*metis circ-left-unfold mult-right-one order-refl star-left-induct*)

lemma *star-zero-below-circ-mult*: $x^* \ ; \ 0 \leq x^\circ \ ; \ y$
by (*metis mult-isotone star-below-circ zero-least*)

lemma *star-mult-circ*: $x^* \ ; \ x^\circ = x^\circ$
by (*metis add-right-divisibility antisym circ-left-unfold star-left-induct-mult star.circ-loop-fixpoint*)

lemma *circ-mult-star*: $x^\circ \ ; \ x^* = x^\circ$
by (*metis add-associative add-least-upper-bound circ-left-unfold circ-rtc-2 eq-iff left-plus-circ star.circ-add-sub star.circ-back-loop-prefixpoint star.circ-increasing star-below-circ star-mult-circ star-sup-one*)

lemma *circ-star*: $x^{\circ *} = x^\circ$
by (*metis circ-left-unfold left-plus-circ less-def less-le star.circ-increasing star-below-circ star-sup-one*)

lemma *star-circ*: $x^{* \circ} = x^{\circ \circ}$
by (*metis antisym circ-circ-add circ-sub-dist less-eq-def star.circ-rtc-2 star-below-circ*)

lemma *circ-add-3*: $(x^\circ \ ; \ y^\circ)^* \leq (x + y)^\circ$
by (*metis circ-add-1 circ-isotone circ-left-unfold circ-star mult-left-sub-dist-add-left mult-right-isotone mult-right-one star.circ-isotone*)

end

class *lka-T* = *il-semiring-T* + *lka*

sublocale *lka-T* < *star*!: *il-conway-semiring-T* **where** *circ* = *star* **..**

class *omega* =
fixes *omega* :: $'a \Rightarrow 'a$ ($-^\omega$ [100] 100)

class *loa* = *lka* + *omega* +
assumes *omega-unfold*: $y^\omega = y \ ; \ y^\omega$
assumes *omega-induct*: $x \leq z + y \ ; \ x \rightarrow x \leq y^\omega + y^* \ ; \ z$

begin

Most lemmas in this class are taken from Georg Struth's Isabelle theories.

lemma *star-zero-below-omega*: $x^* \ ; \ 0 \leq x^\omega$
by (*metis add-left-zero omega-unfold star-left-induct-equal*)

lemma *star-zero-below-omega-zero*: $x^* \ ; \ 0 \leq x^\omega \ ; \ 0$
by (*metis add-left-zero mult-associative omega-unfold star-left-induct-equal*)

lemma *omega-induct-mult*: $y \leq x \ ; \ y \rightarrow y \leq x^\omega$
by (*metis add-commutative add-left-zero less-eq-def omega-induct star-zero-below-omega*)

lemma *omega-sub-dist*: $x^\omega \leq (x+y)^\omega$

by (*metis mult-right-sub-dist-add-left omega-induct-mult omega-unfold*)

lemma *omega-isotone*: $x \leq y \rightarrow x^\omega \leq y^\omega$
by (*metis less-eq-def omega-sub-dist*)

lemma *omega-induct-equal*: $y = z + x \ ; \ y \rightarrow y \leq x^\omega + x^* \ ; \ z$
by (*metis omega-induct order-refl*)

lemma *omega-zero*: $0^\omega = 0$
by (*metis mult-left-zero omega-unfold*)

lemma *omega-one-greatest*: $x \leq 1^\omega$
by (*metis mult-left-one omega-induct-mult order-refl*)

lemma *star-mult-omega*: $x^* \ ; \ x^\omega \ = x^\omega$
by (*metis antisym-conv mult-isotone omega-unfold star.circ-increasing star-left-induct-mult-equal star-left-induct-mult-iff*)

lemma *omega-sub-vector*: $x^\omega \ ; \ y \leq x^\omega$
by (*metis mult-associative omega-induct-mult omega-unfold order-refl*)

lemma *omega-simulation*: $z \ ; \ x \leq y \ ; \ z \rightarrow z \ ; \ x^\omega \leq y^\omega$
by (*smt less-eq-def mult-associative mult-right-sub-dist-add-left omega-induct-mult omega-unfold*)

lemma *omega-omega*: $x^{\omega\omega} \leq x^\omega$
by (*metis omega-sub-vector omega-unfold*)

lemma *left-plus-omega*: $(x \ ; \ x^*)^\omega = x^\omega$
by (*metis antisym mult-associative omega-induct-mult omega-unfold order-refl star.left-plus-circ star-mult-omega*)

lemma *omega-slide*: $x \ ; \ (y \ ; \ x)^\omega = (x \ ; \ y)^\omega$
by (*metis antisym mult-associative mult-right-isotone omega-simulation omega-unfold order-refl*)

lemma *omega-simulation-2*: $y \ ; \ x \leq x \ ; \ y \rightarrow (x \ ; \ y)^\omega \leq x^\omega$
by (*metis less-eq-def mult-right-isotone omega-induct-mult omega-slide omega-sub-dist*)

lemma *wagner*: $(x + y)^\omega = x \ ; \ (x + y)^\omega + z \rightarrow (x + y)^\omega = x^\omega + x^* \ ; \ z$
by (*metis add-commutative add-least-upper-bound eq-iff omega-induct omega-sub-dist star-left-induct*)

lemma *right-plus-omega*: $(x^* \ ; \ x)^\omega = x^\omega$
by (*metis left-plus-omega omega-slide star-mult-omega*)

lemma *omega-sub-dist-1*: $(x \ ; \ y^*)^\omega \leq (x + y)^\omega$
by (*metis add-least-upper-bound left-plus-omega mult-isotone mult-left-one mult-right-dist-add omega-isotone order-refl star-decompose-1 star.circ-increasing star.circ-plus-one*)

lemma *omega-sub-dist-2*: $(x^* \ ; \ y)^\omega \leq (x + y)^\omega$
by (*metis add-commutative mult-isotone omega-slide omega-sub-dist-1 star-mult-omega star.circ-sub-dist*)

lemma *omega-star*: $(x^\omega)^* = 1 + x^\omega$
by (*metis eq-iff mult-left-sub-dist-add-left mult-right-one omega-sub-vector star.circ-left-unfold*)

lemma *omega-mult-omega-star*: $x^\omega \ ; \ x^{\omega *} = x^\omega$
by (*metis add-least-upper-bound antisym omega-sub-vector star.circ-back-loop-prefixpoint*)

lemma *omega-sum-unfold-1*: $(x + y)^\omega = x^\omega + x^* \ ; \ y \ ; \ (x + y)^\omega$
by (*metis mult-associative mult-right-dist-add omega-unfold wagner*)

lemma *omega-sum-unfold-2*: $(x + y)^\omega \leq (x^* \ ; \ y)^\omega + (x^* \ ; \ y)^* \ ; \ x^\omega$
by (*metis omega-induct-equal omega-sum-unfold-1*)

lemma *omega-sum-unfold-3*: $(x^* \ ; \ y)^* \ ; \ x^\omega \leq (x + y)^\omega$
by (*metis omega-sum-unfold-1 star-left-induct-equal*)

lemma *omega-decompose*: $(x + y)^\omega = (x^* \ ; \ y)^\omega + (x^* \ ; \ y)^* \ ; \ x^\omega$
by (*metis add-least-upper-bound antisym omega-sub-dist-2 omega-sum-unfold-2 omega-sum-unfold-3*)

lemma *omega-loop-fixpoint*: $y \ ; \ (y^\omega + y^* \ ; \ z) + z = y^\omega + y^* \ ; \ z$
apply (*rule antisym*)
apply (*smt add-commutative add-least-upper-bound add-right-isotone add-right-upper-bound mult-left-sub-dist-add-left mult-right-isotone omega-induct omega-unfold order-trans star.circ-loop-fixpoint*)
apply (*smt add-associative add-left-isotone mult-left-sub-dist-add omega-unfold star.circ-loop-fixpoint*)
done

lemma *omega-loop-greatest-fixpoint*: $y \ ; \ x + z = x \rightarrow x \leq y^\omega + y^* \ ; \ z$
by (*metis add-commutative omega-induct-equal*)

lemma *omega-square*: $x^\omega = (x \ ; \ x)^\omega$
by (*metis antisym mult-associative omega-induct-mult omega-mult-omega-star omega-slide omega-sub-vector omega-unfold*)

lemma *mult-zero-omega*: $(x \ ; \ 0)^\omega = x \ ; \ 0$
by (*metis mult-left-zero omega-slide*)

lemma *mult-zero-add-omega*: $(x + y \ ; \ 0)^\omega = x^\omega + x^* \ ; \ y \ ; \ 0$
by (*smt add-associative add-commutative add-idempotent mult-associative mult-left-one mult-left-zero mult-right-dist-add mult-zero-omega star.mult-zero-circ omega-decompose*)

lemma *omega-mult-star*: $x^\omega \ ; \ x^* = x^\omega$
by (*metis antisym mult-left-sub-dist-add-left mult-right-one omega-sub-vector star.circ-plus-one*)

lemma *omega-loop-is-greatest-fixpoint*: *is-greatest-fixpoint* $(\lambda x \ . \ y \ ; \ x + z)$ $(y^\omega + y^* \ ; \ z)$
by (*smt is-greatest-fixpoint-def omega-loop-fixpoint omega-loop-greatest-fixpoint*)

lemma *omega-loop-nu*: $\nu \ (\lambda x \ . \ y \ ; \ x + z) = y^\omega + y^* \ ; \ z$
by (*metis greatest-fixpoint-same omega-loop-is-greatest-fixpoint*)

lemma *omega-loop-zero-is-greatest-fixpoint*: *is-greatest-fixpoint* $(\lambda x \ . \ y \ ; \ x)$ (y^ω)
by (*metis is-greatest-fixpoint-def omega-induct-mult omega-unfold order-refl*)

lemma *omega-loop-zero-nu*: $\nu\ (\lambda x\ .\ y\ ;\ x) = y^\omega$
by (*metis greatest-fixpoint-same omega-loop-zero-is-greatest-fixpoint*)

lemma *omega-separate-unfold*: $(x^*\ ;\ y)^\omega = y^\omega + y^*\ ;\ x\ ;\ (x^*\ ;\ y)^\omega$
by (*metis add-commutative mult-associative omega-slide omega-sum-unfold-1 star.circ-loop-fixpoint*)

lemma *omega-zero-left-slide*: $(x\ ;\ y)^*\ ;\ ((x\ ;\ y)^\omega\ ;\ 0 + 1)\ ;\ x \leq x\ ;\ (y\ ;\ x)^*\ ;\ ((y\ ;\ x)^\omega\ ;\ 0 + 1)$
proof −
have $x + x\ ;\ (y\ ;\ x)\ ;\ (y\ ;\ x)^*\ ;\ ((y\ ;\ x)^\omega\ ;\ 0 + 1) \leq x\ ;\ (y\ ;\ x)^*\ ;\ ((y\ ;\ x)^\omega\ ;\ 0 + 1)$
by (*smt add-commutative add-least-upper-bound mult-associative mult-left-isotone mult-right-isotone star.circ-back-loop-prefixpoint star.left-plus-below-circ star.mult-zero-add-circ star.mult-zero-circ*)
hence $((x\ ;\ y)^\omega\ ;\ 0 + 1)\ ;\ x + x\ ;\ y\ ;\ (x\ ;\ (y\ ;\ x)^*\ ;\ ((y\ ;\ x)^\omega\ ;\ 0 + 1)) \leq x\ ;\ (y\ ;\ x)^*\ ;\ ((y\ ;\ x)^\omega\ ;\ 0 + 1)$
by (*smt add-associative less-eq-def mult-associative mult-left-one mult-left-sub-dist-add-left mult-left-zero mult-right-dist-add omega-slide star-mult-omega*)
thus *?thesis*
by (*metis mult-associative star-left-induct*)
qed

lemma *omega-zero-add-1*: $(x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1) = x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
proof (*rule antisym*)
have *1*: $(x + y)\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1) \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
by (*smt add-associative add-commutative less-eq-def mult-associative mult-left-isotone mult-right-dist-add star.circ-add-1 star.left-plus-below-circ star.mult-zero-add-circ star.mult-zero-circ star-decompose-1*)
have *2*: $1 \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
by (*smt add-commutative mult-associative star.circ-add-1 star.circ-reflexive star.mult-zero-add-circ star.mult-zero-circ*)
have $(y\ ;\ x^*)^\omega\ ;\ 0 \leq (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0$
by (*smt mult-left-isotone mult-left-sub-dist-add-right mult-right-one omega-isotone*)
also have *3*: $\ldots \leq (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
by (*smt add-commutative mult-associative mult-left-one mult-right-sub-dist-add-left order-trans star.circ-sub-dist-1 star.mult-zero-add-circ star.mult-zero-circ*)
finally have *4*: $(x^*\ ;\ y)^\omega\ ;\ 0 \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
by (*smt mult-associative mult-right-isotone omega-slide*)
have $y\ ;\ (x^*\ ;\ y)^*\ ;\ x^\omega\ ;\ 0 \leq y\ ;\ (x^*\ ;\ (x^\omega\ ;\ 0 + y))^*\ ;\ x^*\ ;\ x^\omega\ ;\ 0\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0$
by (*metis mult-left-isotone mult-left-sub-dist-add-right mult-right-isotone star.circ-isotone mult-associative mult-left-zero star-mult-omega*)
also have $\ldots \leq y\ ;\ (x^*\ ;\ (x^\omega\ ;\ 0 + y))^*\ ;\ (x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ y)^\omega\ ;\ 0$
by (*smt mult-associative mult-left-isotone mult-left-sub-dist-add-left omega-slide*)
also have $\ldots = y\ ;\ (x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ y)^\omega\ ;\ 0$
by (*smt mult-associative mult-left-one mult-left-zero mult-right-dist-add star-mult-omega*)
finally have $x^*\ ;\ y\ ;\ (x^*\ ;\ y)^*\ ;\ x^\omega\ ;\ 0 \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$ **using** *3*
by (*smt mult-associative mult-right-isotone omega-slide order-trans*)
hence $(x^*\ ;\ y)^*\ ;\ x^\omega\ ;\ 0 \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$
by (*smt add-associative add-commutative less-eq-def mult-associative mult-isotone mult-left-one mult-right-one mult-right-sub-dist-add-left order-trans star.circ-loop-fixpoint star.circ-reflexive star.mult-zero-circ*)
hence $(x + y)^\omega\ ;\ 0 \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$ **using** *4*
by (*metis add-least-upper-bound mult-right-dist-add omega-decompose*)
thus $(x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1) \leq x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1)$ **using** *1 2*
by (*smt add-least-upper-bound mult-associative star-left-induct*)
next
have *5*: $x^\omega\ ;\ 0 \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$

by (*metis add-commutative add-left-zero mult-associative mult-left-isotone mult-left-one mult-right-dist-add omega-sub-dist order-trans star-mult-omega zero-right-mult-decreasing*)
have *6*: $(y\ ;\ x^*)^\omega\ ;\ 0 \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$
by (*metis add-commutative mult-left-isotone omega-sub-dist-1 mult-associative mult-left-sub-dist-add-left order-trans star-mult-omega*)
have *7*: $(y\ ;\ x^*)^* \leq (x + y)^*$
by (*metis mult-left-one mult-right-sub-dist-add-left star.circ-add-1 star.circ-plus-one*)
hence $(y\ ;\ x^*)^*\ ;\ x^\omega\ ;\ 0 \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$
by (*smt add-associative less-eq-def mult-associative mult-isotone mult-right-dist-add omega-sub-dist*)
hence $(x^\omega\ ;\ 0 + y\ ;\ x^*)^\omega\ ;\ 0 \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$ **using** *6*
by (*smt add-commutative add-least-upper-bound mult-associative mult-right-dist-add mult-zero-add-omega omega-unfold omega-zero*)
hence $(y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 \leq y\ ;\ x^*\ ;\ (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$
by (*smt mult-associative mult-left-one mult-left-zero mult-right-dist-add mult-right-isotone omega-slide*)
also have $... \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$ **using** *7*
by (*metis mult-left-isotone order-refl star.circ-mult-upper-bound star-left-induct-mult-iff*)
finally have $(y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1) \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$ **using** *5*
by (*smt add-commutative add-least-upper-bound mult-associative order-refl star.circ-mult-upper-bound star.circ-reflexive star.circ-sub-dist-1 star.mult-zero-add-circ star.mult-zero-circ star-left-induct*)
hence $(x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1) \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$ **using** *5*
by (*metis add-commutative mult-associative star.circ-isotone star.circ-mult-upper-bound star.mult-zero-add-circ star.mult-zero-circ star-involutive*)
thus $x^*\ ;\ (x^\omega\ ;\ 0 + 1)\ ;\ (y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^*\ ;\ ((y\ ;\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))^\omega\ ;\ 0 + 1) \leq (x + y)^*\ ;\ ((x + y)^\omega\ ;\ 0 + 1)$
by (*smt add-associative add-commutative mult-associative star.circ-mult-upper-bound star.circ-sub-dist star.mult-zero-add-circ star.mult-zero-circ*)
qed

lemma *star-omega-greatest*: $x^{*\omega} = 1^\omega$
by (*metis add-commutative less-eq-def omega-one-greatest omega-sub-dist star.circ-plus-one*)

lemma *omega-vector-greatest*: $x^\omega\ ;\ 1^\omega = x^\omega$
by (*metis antisym mult-isotone omega-mult-omega-star omega-one-greatest omega-sub-vector*)

lemma *mult-greatest-omega*: $(x\ ;\ 1^\omega)^\omega \leq x\ ;\ 1^\omega$
by (*metis mult-right-isotone omega-slide omega-sub-vector*)

end

sublocale *loa* $<$ *comb0*!: *il-conway-semiring* **where** *circ* $= (\lambda x\ .\ x^*\ ;\ (x^\omega\ ;\ 0 + 1))$
apply *unfold-locales*
apply (*smt add-associative add-commutative less-eq-def mult-associative mult-left-sub-dist-add-left omega-unfold star.circ-loop-fixpoint star-mult-omega*)
apply (*smt mult-associative omega-zero-left-slide*)
apply (*smt mult-associative omega-zero-add-1*)
done

class *lo-conway-semiring* $=$ *loa* $+$ *il-conway-semiring*

begin

subclass *lk-conway-semiring* **..**

lemma *circ-below-omega-star*: $x^\circ \leq x^\omega + x^*$
by (*metis circ-left-unfold mult-right-one omega-induct order-refl*)

lemma *omega-mult-circ*: $x^\omega \ ; \ x^\circ = x^\omega$
by (*metis circ-star mult-associative omega-mult-star omega-vector-greatest star-omega-greatest*)

lemma *circ-mult-omega*: $x^\circ \ ; \ x^\omega = x^\omega$
by (*metis antisym add-right-divisibility circ-loop-fixpoint circ-plus-sub omega-simulation*)

lemma *circ-omega-greatest*: $x^{\circ\omega} = 1^\omega$
by (*metis circ-star star-omega-greatest*)

lemma *omega-circ*: $x^{\omega\circ} = 1 + x^\omega$
by (*metis antisym circ-left-unfold mult-left-sub-dist-add-left mult-right-one omega-sub-vector*)

end

class *loa-T* = *lka-T* + *loa*

begin

lemma *omega-one*: $1^\omega = T$
by (*smt add-left-top less-eq-def omega-one-greatest*)

lemma *star-omega-top*: $x^{*\omega} = T$
by (*metis add-left-top less-eq-def omega-one omega-sub-dist star.circ-plus-one*)

lemma *omega-vector*: $x^\omega \ ; \ T = x^\omega$
by (*metis add-commutative less-eq-def omega-sub-vector top-right-mult-increasing*)

lemma *mult-top-omega*: $(x \ ; \ T)^\omega \leq x \ ; \ T$
by (*metis mult-right-isotone omega-slide top-greatest*)

end

sublocale *loa-T* < *comb0*!: *il-conway-semiring* **where** *circ* = $(\lambda x \ . \ x^* \ ; \ (x^\omega \ ; \ 0 + 1))$ **..**

class *lo-conway-semiring-T* = *loa-T* + *lo-conway-semiring*

begin

subclass *lk-conway-semiring* **..**

subclass *il-conway-semiring-T* **..**

lemma *circ-omega*: $x^{\circ\omega} = T$

by (*metis circ-star star-omega-top*)

end

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

class *gra* = *lka* + *Omega* +
assumes *Omega-unfold* : $y^{\Omega} \leq 1 + y \mathbin{;} y^{\Omega}$
assumes *Omega-induct* : $x \leq z + y \mathbin{;} x \rightarrow x \leq y^{\Omega} \mathbin{;} z$

begin

lemma *Omega-unfold-equal*: $y^{\Omega} = 1 + y \mathbin{;} y^{\Omega}$
by (*smt Omega-induct Omega-unfold add-right-isotone antisym mult-right-isotone mult-right-one*)

lemma *Omega-add-1*: $(x + y)^{\Omega} = x^{\Omega} \mathbin{;} (y \mathbin{;} x^{\Omega})^{\Omega}$
apply (*rule antisym*)
apply (*smt Omega-induct Omega-unfold-equal add-associative add-commutative add-right-isotone mult-associative mult-right-dist-add mult-right-isotone mult-right-one order-refl*)
apply (*smt Omega-induct Omega-unfold-equal add-associative add-commutative mult-associative mult-left-one mult-right-dist-add mult-right-one order-refl*)
done

lemma *Omega-left-slide*: $(x \mathbin{;} y)^{\Omega} \mathbin{;} x \leq x \mathbin{;} (y \mathbin{;} x)^{\Omega}$
proof −
have $1 + y \mathbin{;} (x \mathbin{;} y)^{\Omega} \mathbin{;} x \leq 1 + y \mathbin{;} x \mathbin{;} (1 + (y \mathbin{;} (x \mathbin{;} y)^{\Omega}) \mathbin{;} x)$
by (*smt Omega-unfold-equal add-right-isotone mult-associative mult-left-one mult-left-sub-dist-add mult-right-dist-add mult-right-isotone mult-right-one*)
thus *?thesis*
by (*smt Omega-induct Omega-unfold-equal add-least-upper-bound mult-associative mult-left-one mult-right-dist-add mult-right-isotone mult-right-one*)
qed

end

sublocale *gra* < *Omega*!: *il-conway-semiring* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*metis Omega-unfold-equal*)
apply (*metis Omega-left-slide*)
apply (*metis Omega-add-1*)
done

context *gra*

begin

lemma *star-below-Omega*: $x^{*} \leq x^{\Omega}$
by (*metis Omega-induct mult-right-one order-refl star.circ-left-unfold*)

lemma *star-mult-Omega*: $x^{\Omega} = x^{*} \mathbin{;} x^{\Omega}$

by (*metis Omega.left-plus-below-circ add-commutative add-left-upper-bound eq-iff star.circ-loop-fixpoint star-left-induct-mult-iff*)

lemma *Omega-one-greatest*: $x \leq 1^\Omega$
by (*metis Omega-induct add-left-zero mult-left-one order-refl order-trans zero-right-mult-decreasing*)

lemma *greatest-left-zero*: $1^\Omega \ ; \ x = 1^\Omega$
by (*metis antisym Omega-one-greatest Omega-induct add-right-upper-bound mult-left-one*)

end

class *gra-T* = *gra* + *lka-T*

begin

lemma *Omega-one*: $1^\Omega = T$
by (*metis Omega.circ-transitive-equal Omega-induct add-left-top add-right-upper-bound less-eq-def mult-left-one*)

lemma *top-left-zero*: $T \ ; \ x = T$
by (*metis Omega-induct Omega-one add-left-top add-right-upper-bound less-eq-def mult-left-one*)

end

sublocale *gra-T* < *Omega*!: *il-conway-semiring-T* **where** *circ* = *Omega* **..**

class *ldra* = *gra* +
assumes *Omega-isolate*: $y^\Omega \leq y^\Omega \ ; \ 0 + y^*$

begin

lemma *Omega-isolate-equal*: $y^\Omega = y^\Omega \ ; \ 0 + y^*$
by (*metis Omega-isolate add-commutative add-same-context less-eq-def star-below-Omega zero-right-mult-decreasing*)

end

class *ldra-T* = *ldra* + *lka-T*

begin

end

sublocale *ldra-T* < *Omega*!: *il-conway-semiring-T* **where** *circ* = *Omega* **..**

class *gra-omega* = *loa* + *Omega* +

assumes *omega-left-zero*: $x^\omega \leq x^\omega \ ; \ y$
assumes *Omega-def*: $x^\Omega = x^\omega + x^*$

begin

lemma *omega-left-zero-equal*: $x^\omega \ ; \ y = x^\omega$
by (*metis antisym omega-left-zero omega-sub-vector*)

subclass *ldra*
apply *unfold-locales*
apply (*metis Omega-def add-commutative eq-refl mult-right-one omega-loop-fixpoint*)
apply (*metis Omega-def mult-right-dist-add omega-induct omega-left-zero-equal*)
apply (*smt Omega-def add-least-upper-bound antisym mult-right-dist-add mult-right-sub-dist-add-left omega-left-zero-equal order-refl star-zero-below-omega*)
done

end

class *dra-omega* = *loa-T* + *Omega* +
assumes *top-left-zero*: $T \ ; \ x = T$
assumes *Omega-def*: $x^\Omega = x^\omega + x^*$

begin

subclass *gra-omega*
apply *unfold-locales*
apply (*metis mult-associative omega-vector order-refl top-left-zero*)
apply (*rule Omega-def*)
done

end

class *dom* =
fixes $d :: 'a \Rightarrow 'a$
fixes $Z :: 'a$

class *relative-domain-semiring* = *il-semiring* + *dom* +
assumes *d-restrict* : $x \leq d(x) \ ; \ x + Z$
assumes *d-mult-d* : $d(x \ ; \ y) = d(x \ ; \ d(y))$
assumes *d-below-one*: $d(x) \leq 1$
assumes *d-Z* : $d(Z) = 0$
assumes *d-dist-add* : $d(x + y) = d(x) + d(y)$
assumes *d-export* : $d(d(x) \ ; \ y) = d(x) \ ; \ d(y)$

begin

lemma *d-plus-one*: $d(x) + 1 = 1$
by (*metis d-below-one less-eq-def*)

lemma *d-zero*: $d(0) = 0$
by (*metis d-Z d-export mult-left-zero*)

lemma *d-involutive*: $d(d(x)) = d(x)$
by (*metis d-mult-d mult-left-one*)

lemma *d-fixpoint*: $(\exists\, y \;.\; x = d(y)) \leftrightarrow x = d(x)$
by (*metis d-involutive*)

lemma *d-type*: $\forall\, P \;.\; (\forall\, x \;.\; x = d(x) \rightarrow P(x)) \leftrightarrow (\forall\, x \;.\; P(d(x)))$
by (*metis d-involutive*)

lemma *d-mult-sub*: $d(x \;;\; y) \leq d(x)$
by (*metis add-commutative d-below-one d-dist-add d-mult-d less-eq-def mult-left-sub-dist-add-right mult-right-one*)

lemma *d-sub-one*: $x \leq 1 \rightarrow x \leq d(x) + Z$
by (*metis add-left-isotone d-restrict mult-right-isotone mult-right-one order-trans*)

lemma *d-one*: $d(1) + Z = 1 + Z$
by (*smt add-associative add-commutative d-plus-one d-restrict less-eq-def mult-right-one*)

lemma *d-strict*: $d(x) = 0 \leftrightarrow x \leq Z$
by (*metis add-commutative add-right-zero d-Z d-dist-add d-restrict less-eq-def mult-left-zero*)

lemma *d-isotone*: $x \leq y \rightarrow d(x) \leq d(y)$
by (*metis d-dist-add less-eq-def*)

lemma *d-plus-left-upper-bound*: $d(x) \leq d(x + y)$
by (*metis add-left-upper-bound d-isotone*)

lemma *d-idempotent*: $d(x) \;;\; d(x) = d(x)$
by (*smt add-commutative add-right-zero d-Z d-dist-add d-export d-involutive d-mult-sub d-restrict less-eq-def*)

lemma *d-least-left-preserver*: $x \leq d(y) \;;\; x + Z \leftrightarrow d(x) \leq d(y)$
apply *rule*
apply (*smt add-associative add-left-divisibility add-right-zero d-Z d-dist-add d-involutive d-mult-sub less-eq-def*)
apply (*smt add-associative add-commutative d-restrict less-eq-def mult-right-dist-add*)
done

lemma *d-weak-locality*: $x \;;\; y \leq Z \leftrightarrow x \;;\; d(y) \leq Z$
by (*metis d-mult-d d-strict*)

lemma *d-add-closed*: $d(d(x) + d(y)) = d(x) + d(y)$
by (*metis d-dist-add d-involutive*)

lemma *d-mult-closed*: $d(d(x)\ ;\ d(y)) = d(x)\ ;\ d(y)$
by (*metis d-export d-mult-d*)

lemma *d-mult-left-lower-bound*: $d(x)\ ;\ d(y) \leq d(x)$
by (*metis d-export d-involutive d-mult-sub*)

lemma *d-mult-left-absorb-add*: $d(x)\ ;\ (d(x) + d(y)) = d(x)$
by (*smt d-add-closed d-export d-idempotent d-involutive d-mult-sub eq-iff mult-left-sub-dist-add-left*)

lemma *d-add-left-absorb-mult*: $d(x) + d(x)\ ;\ d(y) = d(x)$
by (*metis add-commutative d-mult-left-lower-bound less-eq-def*)

lemma *d-commutative*: $d(x)\ ;\ d(y) = d(y)\ ;\ d(x)$
by (*metis add-commutative antisym d-add-left-absorb-mult d-below-one d-export d-mult-left-absorb-add mult-associative mult-left-isotone mult-left-one*)

lemma *d-mult-greatest-lower-bound*: $d(x) \leq d(y)\ ;\ d(z) \leftrightarrow d(x) \leq d(y) \wedge d(x) \leq d(z)$
by (*metis d-commutative d-idempotent d-mult-left-lower-bound mult-isotone order-trans*)

lemma *d-add-left-dist-mult*: $d(x) + d(y)\ ;\ d(z) = (d(x) + d(y))\ ;\ (d(x) + d(z))$
by (*metis add-associative d-commutative d-dist-add d-idempotent d-mult-left-absorb-add mult-right-dist-add*)

lemma *d-order*: $d(x) \leq d(y) \leftrightarrow d(x) = d(x)\ ;\ d(y)$
by (*metis d-mult-greatest-lower-bound d-mult-left-absorb-add less-eq-def order-refl*)

lemma *Z-mult-decreasing*: $Z\ ;\ x \leq Z$
by (*metis add-left-zero d-Z d-least-left-preserver d-mult-sub mult-left-zero*)

lemma *d-below-d-one*: $d(x) \leq d(1)$
by (*metis d-mult-sub mult-left-one*)

lemma *d-relative-Z*: $d(x)\ ;\ x + Z = x + Z$
by (*metis add-left-upper-bound add-same-context d-below-one d-restrict mult-isotone mult-left-one*)

lemma *Z-left-zero-above-one*: $1 \leq x \rightarrow Z\ ;\ x = Z$
by (*metis Z-mult-decreasing eq-iff mult-right-isotone mult-right-one*)

lemma *kat-4*: $d(x)\ ;\ y = d(x)\ ;\ y\ ;\ d(z) \rightarrow d(x)\ ;\ y \leq y\ ;\ d(z)$
by (*metis d-below-one mult-left-isotone mult-left-one*)

lemma *kat-4-equiv*: $d(x)\ ;\ y = d(x)\ ;\ y\ ;\ d(z) \leftrightarrow d(x)\ ;\ y \leq y\ ;\ d(z)$
apply *rule*
apply (*metis kat-4*)
apply (*rule antisym*)
apply (*metis d-idempotent mult-associative mult-right-isotone*)
apply (*metis d-below-one mult-right-isotone mult-right-one*)
done

lemma *kat-4-equiv-opp*: $y \ ; \ d(x) = d(z) \ ; \ y \ ; \ d(x) \leftrightarrow y \ ; \ d(x) \leq d(z) \ ; \ y$
apply *rule*
apply (*metis d-below-one mult-right-isotone mult-right-one*)
apply (*rule antisym*)
apply (*metis d-idempotent mult-associative mult-left-isotone*)
apply (*metis d-below-one mult-left-isotone mult-left-one*)
done

lemma *d-restrict-iff-1*: $(d(x) \ ; \ y \leq z) \leftrightarrow (d(x) \ ; \ y \leq d(x) \ ; \ z)$
by (*smt d-below-one d-idempotent mult-associative mult-left-isotone mult-left-one mult-right-isotone order-trans*)

end

class *relative-domain-semiring-T* = *relative-domain-semiring* + *il-semiring-T*

begin

lemma *Z-top*: $Z \ ; \ T = Z$
by (*metis Z-mult-decreasing eq-iff top-right-mult-increasing*)

lemma *d-restrict-T*: $x \leq d(x) \ ; \ T + Z$
by (*metis add-left-isotone d-restrict mult-right-isotone order-trans top-greatest*)

end

class *relative-domain-semiring-split* = *relative-domain-semiring* +
assumes *split-Z*: $x \ ; \ (y + Z) \leq x \ ; \ y + Z$

begin

lemma *d-restrict-iff*: $(x \leq y + Z) \leftrightarrow (x \leq d(x) \ ; \ y + Z)$
proof −
have $x \leq y + Z \rightarrow x \leq d(x) \ ; \ (y + Z) + Z$
by (*smt add-left-isotone d-restrict less-eq-def mult-left-sub-dist-add-left order-trans*)
hence $x \leq y + Z \rightarrow x \leq d(x) \ ; \ y + Z$
by (*metis add-isotone add-right-zero add-same-context d-strict d-zero mult-left-sub-dist-add-left split-Z*)
thus *?thesis*
by (*smt d-below-one mult-left-isotone add-left-isotone mult-left-one order-trans*)
qed

end

class *relative-antidomain-semiring* = *il-semiring* + *dom* + *neg* +
assumes *a-restrict* : $-x \ ; \ x \leq Z$
assumes *a-mult-d* : $-(x \ ; \ y) = -(x \ ; \ --y)$
assumes *a-complement*: $-x \ ; \ --x = 0$

assumes *a-Z* : $-Z = 1$
assumes *a-export* : $-(--x\ ;\ y) = -x + -y$
assumes *a-dist-add* : $-(x + y) = -x\ ;\ -y$
assumes *d-def* : $d(x) = --x$

begin

notation
$uminus\ (a)$

lemma *a-complement-one*: $--x + -x = 1$
by (*metis a-Z a-complement a-export a-mult-d mult-left-one*)

lemma *a-d-closed*: $d(a(x)) = a(x)$
by (*metis a-mult-d d-def mult-left-one*)

lemma *a-below-one*: $a(x) \leq 1$
by (*metis a-complement-one add-right-divisibility*)

lemma *a-export-a*: $a(a(x)\ ;\ y) = d(x) + a(y)$
by (*metis a-d-closed a-export d-def*)

lemma *a-add-absorb*: $(x + a(y))\ ;\ a(a(y)) = x\ ;\ a(a(y))$
by (*metis a-complement add-right-zero mult-right-dist-add*)

lemma *a-greatest-left-absorber*: $a(x)\ ;\ y \leq Z \leftrightarrow a(x) \leq a(y)$
apply *rule*
apply (*smt a-Z a-add-absorb a-dist-add a-export-a a-mult-d add-commutative d-def less-eq-def mult-left-one*)
apply (*metis a-restrict le-less-trans le-neq-trans less-eq-def less-imp-le mult-right-sub-dist-add-left*)
done

lemma *a-plus-left-lower-bound*: $a(x + y) \leq a(x)$
by (*metis a-greatest-left-absorber a-restrict add-commutative mult-left-sub-dist-add-right order-trans*)

subclass *relative-domain-semiring*
apply *unfold-locales*
apply (*smt a-Z a-complement-one a-restrict add-commutative add-left-upper-bound case-split-left d-def order-trans*)
apply (*metis a-mult-d d-def*)
apply (*metis a-below-one d-def*)
apply (*metis a-Z a-complement d-def mult-left-one*)
apply (*metis a-dist-add a-export-a d-def*)
apply (*metis a-dist-add a-export d-def*)
done

subclass *tests*
apply *unfold-locales*
apply (*metis mult-associative*)

apply (*metis a-dist-add add-commutative*)
apply (*smt a-complement a-d-closed a-export-a add-right-zero d-add-left-dist-mult*)
apply (*metis a-d-closed a-dist-add d-def*)
apply (*rule the-equality*[*THEN sym*])
apply (*metis a-complement*)
apply (*metis a-complement*)
apply (*metis a-Z a-d-closed d-Z d-def*)
apply (*metis a-d-closed a-export d-def*)
apply (*smt a-d-closed a-dist-add a-plus-left-lower-bound add-commutative d-def less-eq-def*)
apply (*metis less-def*)
done

lemma *a-plus-mult-d*: $-(x\ ;\ y) + -(x\ ;\ --y) = -(x\ ;\ --y)$
by (*metis a-mult-d add-idempotent*)

lemma *a-mult-d-2*: $a(x\ ;\ y) = a(x\ ;\ d(y))$
by (*metis a-mult-d d-def*)

lemma *a-idempotent*: $a(x)\ ;\ a(x) = a(x)$
by (*metis a-dist-add add-idempotent*)

lemma *a-3*: $a(x)\ ;\ a(y)\ ;\ d(x + y) = 0$
by (*metis a-complement a-dist-add d-def*)

lemma *a-fixpoint*: $\forall\, x\ .\ (a(x) = x \rightarrow (\forall\, y\ .\ y = 0))$
by (*metis a-idempotent mult-left-one mult-left-zero one-def zero-def*)

lemma *a-strict*: $a(x) = 1 \leftrightarrow x \leq Z$
by (*metis d-def d-strict double-negation one-compl one-def*)

lemma *d-complement-zero*: $d(x)\ ;\ a(x) = 0$
by (*metis d-def sub-comm zero-def*)

lemma *a-complement-zero*: $a(x)\ ;\ d(x) = 0$
by (*metis d-def zero-def*)

lemma *a-shunting-zero*: $a(x)\ ;\ d(y) = 0 \leftrightarrow a(x) \leq a(y)$
by (*metis d-def leq-mult-zero*)

lemma *a-antitone*: $x \leq y \rightarrow a(y) \leq a(x)$
by (*metis a-plus-left-lower-bound less-eq-def*)

lemma *a-mult-deMorgan*: $a(a(x)\ ;\ a(y)) = d(x + y)$
by (*metis a-dist-add d-def*)

lemma *a-mult-deMorgan-1*: $a(a(x)\ ;\ a(y)) = d(x) + d(y)$
by (*metis a-mult-deMorgan d-dist-add*)

lemma *a-mult-deMorgan-2*: $a(d(x)\ ;\ d(y)) = a(x) + a(y)$
by (*metis d-def plus-def*)

lemma *a-plus-deMorgan*: $a(a(x) + a(y)) = d(x)\ ;\ d(y)$
by (*metis a-dist-add d-def*)

lemma *a-plus-deMorgan-1*: $a(d(x) + d(y)) = a(x)\ ;\ a(y)$
by (*metis a-mult-deMorgan-1 sub-mult-closed*)

lemma *a-mult-left-upper-bound*: $a(x) \leq a(x\ ;\ y)$
by (*metis a-antitone d-def d-mult-sub double-negation*)

lemma *d-a-closed*: $a(d(x)) = a(x)$
by (*metis a-d-closed d-def*)

lemma *a-export-d*: $a(d(x)\ ;\ y) = a(x) + a(y)$
by (*metis a-export d-def*)

lemma *a-7*: $d(x)\ ;\ a(d(y) + d(z)) = d(x)\ ;\ a(y)\ ;\ a(z)$
by (*metis a-plus-deMorgan-1 mult-associative*)

lemma *d-a-shunting*: $d(x)\ ;\ a(y) \leq d(z) \leftrightarrow d(x) \leq d(z) + d(y)$
by (*smt a-dist-add d-def plus-closed shunting sub-comm*)

lemma *d-d-shunting*: $d(x)\ ;\ d(y) \leq d(z) \leftrightarrow d(x) \leq d(z) + a(y)$
by (*metis d-a-closed d-a-shunting d-def*)

lemma *d-cancellation-1*: $d(x) \leq d(y) + (d(x)\ ;\ a(y))$
by (*metis a-dist-add add-commutative add-left-upper-bound d-def plus-compl-intro*)

lemma *d-cancellation-2*: $(d(z) + d(y))\ ;\ a(y) \leq d(z)$
by (*metis d-a-shunting d-dist-add eq-refl*)

lemma *a-add-closed*: $d(a(x) + a(y)) = a(x) + a(y)$
by (*metis d-def plus-closed*)

lemma *a-mult-closed*: $d(a(x)\ ;\ a(y)) = a(x)\ ;\ a(y)$
by (*metis d-def sub-mult-closed*)

lemma *d-a-shunting-zero*: $d(x)\ ;\ a(y) = 0 \leftrightarrow d(x) \leq d(y)$
by (*metis d-def double-negation leq-mult-zero*)

lemma *d-d-shunting-zero*: $d(x)\ ;\ d(y) = 0 \leftrightarrow d(x) \leq a(y)$
by (*metis d-def leq-mult-zero*)

lemma *d-compl-intro*: $d(x) + d(y) = d(x) + a(x)\ ;\ d(y)$

by (*metis add-commutative d-def plus-compl-intro*)

lemma *a-compl-intro*: $a(x) + a(y) = a(x) + d(x)\ ;\ a(y)$
by (*smt a-dist-add add-commutative d-def mult-right-one plus-compl plus-distr-mult-left*)

lemma *kat-2*: $y\ ;\ a(z) \leq a(x)\ ;\ y \rightarrow d(x)\ ;\ y\ ;\ a(z) = 0$
by (*metis d-complement-zero eq-iff mult-associative mult-left-zero mult-right-isotone zero-least*)

lemma *kat-2-equiv*: $y\ ;\ a(z) \leq a(x)\ ;\ y \leftrightarrow d(x)\ ;\ y\ ;\ a(z) = 0$
apply *rule*
apply (*metis kat-2*)
apply (*metis a-Z a-below-one a-complement-one case-split-left d-def mult-associative mult-right-isotone mult-right-one zero-least*)
done

lemma *kat-3-equiv-opp*: $a(z)\ ;\ y\ ;\ d(x) = 0 \leftrightarrow y\ ;\ d(x) = d(z)\ ;\ y\ ;\ d(x)$
by (*metis a-complement-one add-left-zero d-def mult-associative mult-left-one mult-left-zero mult-right-dist-add unique-zero zero-double-compl*)

lemma *kat-equiv-6*: $d(x)\ ;\ y\ ;\ a(z) = d(x)\ ;\ y\ ;\ 0 \leftrightarrow d(x)\ ;\ y\ ;\ a(z) \leq y\ ;\ 0$
by (*metis a-d-closed antisym d-idempotent kat-4 mult-associative mult-right-isotone mult-right-one one-def zero-least-test*)

lemma *a-one*: $a(1) = 0$
by (*metis one-compl*)

lemma *d-one-one*: $d(1) = 1$
by (*metis d-def one-double-compl*)

lemma *case-split-left-add*: $-p\ ;\ x \leq y \wedge --p\ ;\ x \leq z \rightarrow x \leq y + z$
by (*metis a-complement a-dist-add add-isotone mult-left-one mult-right-dist-add one-def plus-closed*)

lemma *test-mult-left-sub-dist-shunt*: $-p\ ;\ (--p\ ;\ x + Z) \leq Z$
by (*metis a-Z a-dist-add a-export a-greatest-left-absorber add-commutative add-left-upper-bound mult-left-one*)

lemma *test-mult-left-dist-shunt*: $-p\ ;\ (--p\ ;\ x + Z) = -p\ ;\ Z$
by (*smt add-commutative antisym mult-associative mult-idempotent mult-left-sub-dist-add-left mult-right-isotone test-mult-left-sub-dist-shunt*)

end

class *relative-antidomain-semiring-T* = *relative-antidomain-semiring* + *il-semiring-T*

begin

subclass *relative-domain-semiring-T* **..**

lemma *a-T*: $a(T) = 0$
by (*metis a-dist-add a-one add-right-top mult-left-zero*)

lemma *d-T*: $d(T) = 1$

by (*metis a-dist-add add-left-top d-def one-def zero-def*)

lemma *shunting-T-1*: $-p \ ; \ x \leq y \rightarrow x \leq --p \ ; \ T + y$
by (*metis add-commutative case-split-left-add mult-right-isotone top-greatest*)

lemma *shunting-Z*: $-p \ ; \ x \leq Z \leftrightarrow x \leq --p \ ; \ T + Z$
apply *rule*
apply (*metis add-commutative case-split-left-add mult-right-isotone top-greatest*)
apply (*smt a-T a-Z a-antitone a-dist-add a-export a-greatest-left-absorber add-commutative add-right-zero mult-left-one*)
done

end

class *relative-diamond-semiring* = *relative-domain-semiring* +
fixes *diamond* :: $'a \Rightarrow 'a \Rightarrow 'a$ (| - > - [50,90] 95)
assumes *diamond-def*: $|x>y = d(x \ ; \ y)$

begin

lemma *diamond-x-1*: $|x>1 = d(x)$
by (*metis diamond-def mult-right-one*)

lemma *diamond-x-d*: $|x>d(y) = d(x \ ; \ y)$
by (*metis d-mult-d diamond-def*)

lemma *diamond-x-und*: $|x>d(y) = |x>y$
by (*metis diamond-def diamond-x-d*)

lemma *diamond-d-closed*: $|x>y = d(\ |x>y)$
by (*metis d-fixpoint diamond-def*)

lemma *diamond-0-y*: $|0>y = 0$
by (*metis d-zero diamond-def mult-left-zero*)

lemma *diamond-1-y*: $|1>y = d(y)$
by (*metis diamond-def mult-left-one*)

lemma *diamond-1-d*: $|1>d(y) = d(y)$
by (*metis diamond-1-y diamond-x-und*)

lemma *diamond-d-y*: $|d(x)>y = d(x) \ ; \ d(y)$
by (*metis d-export diamond-def*)

lemma *diamond-d-0*: $|d(x)>0 = 0$
by (*metis d-commutative diamond-0-y diamond-d-y diamond-x-1*)

lemma *diamond-d-1*: $|d(x)>1 = d(x)$
by (*metis diamond-d-closed diamond-x-1*)

lemma *diamond-d-d*: $|d(x)>d(y) = d(x)\ ;\ d(y)$
by (*metis d-mult-closed diamond-def*)

lemma *diamond-d-d-same*: $|d(x)>d(x) = d(x)$
by (*metis d-idempotent diamond-d-d*)

lemma *diamond-left-dist-add*: $|x + y>z = |x>z + |y>z$
by (*metis d-dist-add diamond-def mult-right-dist-add*)

lemma *diamond-right-sub-dist-add*: $|x>y + |x>z \leq |x>(y + z)$
by (*smt add-associative d-plus-left-upper-bound diamond-def less-eq-def mult-left-sub-dist-add-left mult-left-sub-dist-add-right*)

lemma *diamond-associative*: $|x\ ;\ y>z = |x>(y\ ;\ z)$
by (*metis diamond-def mult-associative*)

lemma *diamond-left-mult*: $|x\ ;\ y>z = |x>|y>z$
by (*metis diamond-def diamond-x-d mult-associative*)

lemma *diamond-right-mult*: $|x>(y\ ;\ z) = |x>|y>z$
by (*metis diamond-associative diamond-left-mult*)

lemma *diamond-d-export*: $|d(x)\ ;\ y>z = d(x)\ ;\ |y>z$
by (*metis diamond-associative diamond-d-closed diamond-d-y diamond-right-mult*)

lemma *diamond-diamond-export*: $||x>y>z = |x>y\ ;\ |z>1$
by (*metis diamond-d-d diamond-def diamond-x-1 diamond-x-und*)

lemma *diamond-left-isotone*: $x \leq y \rightarrow |x>z \leq |y>z$
by (*metis diamond-left-dist-add less-eq-def*)

lemma *diamond-right-isotone*: $y \leq z \rightarrow |x>y \leq |x>z$
by (*metis d-isotone diamond-def mult-right-isotone*)

lemma *diamond-isotone*: $w \leq y \wedge x \leq z \rightarrow |w>x \leq |y>z$
by (*metis diamond-left-isotone diamond-right-isotone order-trans*)

lemma *diamond-left-upper-bound*: $|x>y \leq |x+z>y$
by (*metis add-left-upper-bound diamond-left-dist-add*)

lemma *diamond-right-upper-bound*: $|x>y \leq |x>(y+z)$
by (*metis add-left-upper-bound diamond-right-isotone*)

lemma *diamond-lower-bound-right*: $|x\rangle(d(y)\ ;\ d(z)) \leq |x\rangle d(y)$
by (*metis d-mult-left-lower-bound diamond-right-isotone*)

lemma *diamond-lower-bound-left*: $|x\rangle(d(y)\ ;\ d(z)) \leq |x\rangle d(z)$
by (*metis d-commutative diamond-lower-bound-right*)

lemma *diamond-right-sub-dist-mult*: $|x\rangle(d(y)\ ;\ d(z)) \leq |x\rangle d(y)\ ;\ |x\rangle d(z)$
by (*metis d-mult-greatest-lower-bound diamond-def diamond-lower-bound-left diamond-lower-bound-right*)

lemma *diamond-demodalisation-1*: $d(x)\ ;\ |y\rangle z \leq Z \leftrightarrow d(x)\ ;\ y\ ;\ d(z) \leq Z$
by (*smt d-strict diamond-associative diamond-right-mult diamond-x-1 diamond-x-und*)

lemma *diamond-demodalisation-3*: $|x\rangle y \leq d(z) \leftrightarrow x\ ;\ d(y) \leq d(z)\ ;\ x + Z$
apply *rule*
apply (*metis add-commutative add-right-isotone d-below-one d-restrict diamond-def diamond-x-und mult-left-isotone mult-right-isotone mult-right-one order-trans*)
apply (*smt add-commutative add-left-zero d-Z d-commutative d-dist-add d-involutive d-mult-sub d-plus-left-upper-bound diamond-d-y diamond-def diamond-x-und less-eq-def order-trans*)
done

end

class *relative-box-semiring* = *relative-diamond-semiring* + *relative-antidomain-semiring* +
fixes *box* :: $'a \Rightarrow 'a \Rightarrow 'a$ ($|$ -] - [50,90] 95)
assumes *box-def*: $|x]y = a(x\ ;\ a(y))$

begin

lemma *box-diamond*: $|x]y = a(\ |x\rangle a(y))$
by (*metis box-def d-a-closed diamond-def*)

lemma *diamond-box*: $|x\rangle y = a(\ |x]a(y))$
by (*metis box-diamond d-def diamond-d-closed diamond-def diamond-x-d*)

lemma *box-x-0*: $|x]0 = a(x)$
by (*metis box-def mult-right-one one-def*)

lemma *box-x-1*: $|x]1 = a(x\ ;\ 0)$
by (*metis box-def one-compl*)

lemma *box-x-d*: $|x]d(y) = a(x\ ;\ a(y))$
by (*metis box-def d-a-closed*)

lemma *box-x-und*: $|x]d(y) = |x]y$
by (*metis box-def box-x-d*)

lemma *box-x-a*: $|x]a(y) = a(x\ ;\ y)$
by (*metis a-mult-d box-def*)

lemma *box-0-y*: $|0]y = 1$
by (*metis box-def mult-left-zero one-def*)

lemma *box-1-y*: $|1]y = d(y)$
by (*metis box-def d-def mult-left-one*)

lemma *box-1-d*: $|1]d(y) = d(y)$
by (*metis box-1-y d-involutive*)

lemma *box-1-a*: $|1]a(y) = a(y)$
by (*metis a-d-closed box-1-y*)

lemma *box-d-y*: $|d(x)]y = a(x) + d(y)$
by (*metis a-dist-add box-def box-x-a diamond-box diamond-x-1 mult-right-one plus-closed*)

lemma *box-a-y*: $|a(x)]y = d(x) + d(y)$
by (*metis a-mult-deMorgan-1 box-def*)

lemma *box-d-0*: $|d(x)]0 = a(x)$
by (*metis box-x-0 d-a-closed*)

lemma *box-a-0*: $|a(x)]0 = d(x)$
by (*metis box-x-0 d-def*)

lemma *box-d-1*: $|d(x)]1 = 1$
by (*metis box-diamond diamond-d-0 one-compl one-def*)

lemma *box-a-1*: $|a(x)]1 = 1$
by (*metis box-x-1 bs-mult-right-zero one-def*)

lemma *box-d-d*: $|d(x)]d(y) = a(x) + d(y)$
by (*metis box-d-y box-x-und*)

lemma *box-a-d*: $|a(x)]d(y) = d(x) + d(y)$
by (*metis a-mult-deMorgan-1 box-x-d*)

lemma *box-d-a*: $|d(x)]a(y) = a(x) + a(y)$
by (*metis a-export-d box-x-a*)

lemma *box-a-a*: $|a(x)]a(y) = d(x) + a(y)$
by (*metis a-export-a box-x-a*)

lemma *box-d-d-same*: $|d(x)]d(x) = 1$
by (*metis box-d-y d-a-closed d-def plus-compl*)

lemma *box-a-a-same*: $|a(x)]a(x) = 1$

by (*metis box-def mult-compl one-def*)

lemma *box-d-closed*: $|x]y = d(\ |x]y)$
by (*metis box-1-a box-1-y box-def*)

lemma *box-deMorgan-1*: $a(\ |x]y) = |x{>}a(y)$
by (*metis box-def d-def diamond-def*)

lemma *box-deMorgan-2*: $a(\ |x{>}y) = |x]a(y)$
by (*metis box-def diamond-box double-negation*)

lemma *box-left-dist-add*: $|x + y]z = |x]z\ ;\ |y]z$
by (*metis a-dist-add box-def mult-right-dist-add*)

lemma *box-right-dist-add*: $|x](y + z) = a(x\ ;\ a(y)\ ;\ a(z))$
by (*metis a-dist-add box-def mult-associative*)

lemma *box-associative*: $|x\ ;\ y]z = a(x\ ;\ y\ ;\ a(z))$
by (*metis box-def*)

lemma *box-left-mult*: $|x\ ;\ y]z = |x]|y]z$
by (*metis box-def box-x-a mult-associative*)

lemma *box-right-mult*: $|x](y\ ;\ z) = a(x\ ;\ a(y\ ;\ z))$
by (*metis box-def*)

lemma *box-right-submult-d-d*: $|x](d(y)\ ;\ d(z)) \leq |x]d(y)\ ;\ |x]d(z)$
by (*smt a-antitone a-dist-add a-export-d box-diamond d-a-closed diamond-def mult-left-sub-dist-add*)

lemma *box-right-submult-a-d*: $|x](a(y)\ ;\ d(z)) \leq |x]a(y)\ ;\ |x]d(z)$
by (*metis box-d-closed box-right-submult-d-d box-x-0*)

lemma *box-right-submult-d-a*: $|x](d(y)\ ;\ a(z)) \leq |x]d(y)\ ;\ |x]a(z)$
by (*metis box-a-0 box-left-mult box-right-submult-d-d box-x-0 box-x-und*)

lemma *box-right-submult-a-a*: $|x](a(y)\ ;\ a(z)) \leq |x]a(y)\ ;\ |x]a(z)$
by (*metis a-d-closed box-right-submult-a-d*)

lemma *box-d-export*: $|d(x)\ ;\ y]z = a(x) + |y]z$
by (*metis a-d-closed box-d-y box-def box-left-mult*)

lemma *box-a-export*: $|a(x)\ ;\ y]z = d(x) + |y]z$
by (*metis a-d-closed box-d-a box-def box-left-mult d-def*)

lemma *box-left-antitone*: $y \leq x \rightarrow |x]z \leq |y]z$
by (*metis a-antitone box-def mult-left-isotone*)

lemma *box-right-isotone*: $y \leq z \rightarrow |x]y \leq |x]z$
by (*metis a-antitone box-def mult-right-isotone*)

lemma *box-antitone-isotone*: $y \leq w \wedge x \leq z \rightarrow |w]x \leq |y]z$
by (*metis box-left-antitone box-right-isotone order-trans*)

lemma *diamond-1-a*: $|1\rangle a(y) = a(y)$
by (*metis a-d-closed diamond-1-y*)

lemma *diamond-a-y*: $|a(x)\rangle y = a(x) \ ; \ d(y)$
by (*metis a-mult-closed d-def d-mult-d diamond-def*)

lemma *diamond-a-0*: $|a(x)\rangle 0 = 0$
by (*metis box-a-1 box-deMorgan-1 one-compl*)

lemma *diamond-a-1*: $|a(x)\rangle 1 = a(x)$
by (*metis a-d-closed diamond-x-1*)

lemma *diamond-a-d*: $|a(x)\rangle d(y) = a(x) \ ; \ d(y)$
by (*metis diamond-a-y diamond-x-und*)

lemma *diamond-d-a*: $|d(x)\rangle a(y) = d(x) \ ; \ a(y)$
by (*metis a-d-closed diamond-d-y*)

lemma *diamond-a-a*: $|a(x)\rangle a(y) = a(x) \ ; \ a(y)$
by (*metis a-mult-closed diamond-def*)

lemma *diamond-a-a-same*: $|a(x)\rangle a(x) = a(x)$
by (*metis a-idempotent diamond-a-a*)

lemma *diamond-a-export*: $|a(x) \ ; \ y\rangle z = a(x) \ ; \ |y\rangle z$
by (*metis diamond-a-a diamond-box diamond-left-mult*)

lemma *a-box-a-a*: $a(p) \ ; \ |a(p)]a(q) = a(p) \ ; \ a(q)$
by (*metis box-x-a double-negation mult-compl-intro plus-def*)

lemma *box-left-lower-bound*: $|x+y]z \leq |x]z$
by (*metis add-left-upper-bound box-left-antitone*)

lemma *box-right-upper-bound*: $|x]y \leq |x](y+z)$
by (*metis add-left-upper-bound box-right-isotone*)

lemma *box-lower-bound-right*: $|x](d(y) \ ; \ d(z)) \leq |x]d(y)$
by (*metis box-right-isotone d-mult-left-lower-bound*)

lemma *box-lower-bound-left*: $|x](d(y) \ ; \ d(z)) \leq |x]d(z)$
by (*metis box-lower-bound-right d-commutative*)

lemma *box-demodalisation-2*: $-p \leq |y](-q) \leftrightarrow -p \mathbin{;} y \mathbin{;} --q \leq Z$
by (*metis a-greatest-left-absorber box-def mult-associative*)

lemma *box-right-sub-dist-add*: $|x]d(y) + |x]d(z) \leq |x](d(y) + d(z))$
by (*metis add-commutative add-least-upper-bound box-right-upper-bound*)

lemma *box-diff-var*: $|x](d(y) + a(z)) \mathbin{;} |x]d(z) \leq |x]d(z)$
by (*metis box-def lower-bound-right*)

lemma *diamond-demodalisation-2*: $|x\rangle y \leq d(z) \leftrightarrow a(z) \mathbin{;} x \mathbin{;} d(y) \leq Z$
by (*metis a-mult-d box-def d-a-shunting-zero d-strict diamond-a-y diamond-box diamond-x-1 mult-associative mult-right-one sub-comm*)

lemma *diamond-split*: $|x\rangle y = d(z) \mathbin{;} |x\rangle y + a(z) \mathbin{;} |x\rangle y$
by (*metis a-export-d a-restrict add-commutative d-def d-strict mult-left-one mult-right-dist-add one-def*)

lemma *box-import-shunting*: $-p;-q \leq |x](-r) \leftrightarrow -q \leq |-p;x](-r)$
by (*smt box-demodalisation-2 mult-associative sub-comm sub-mult-closed*)

end

class *complete-tests* = *tests* + *Sup* + *Inf* +
assumes *sup-test*: *test-set A* $\rightarrow$ *Sup A* = $--$*Sup A*
assumes *sup-upper*: *test-set A* $\wedge$ $x \in A \rightarrow x \leq$ *Sup A*
assumes *sup-least*: *test-set A* $\wedge$ $(\forall x{\in}A \mathbin{.} x \leq -y) \rightarrow$ *Sup A* $\leq -y$

begin

lemma *Sup-isotone*: *test-set B* $\wedge$ $A \subseteq B \rightarrow$ *Sup A* $\leq$ *Sup B*
by (*smt subsetD sup-least sup-test sup-upper test-set-closed*)

lemma *mult-right-dist-sup*: *test-set A* $\rightarrow$ *Sup A* $\mathbin{;} -p =$ *Sup* $\{ x;-p \mid x \mathbin{.} x \in A \}$
proof
assume *1*: *test-set A*
hence *2*: *test-set* $\{ x;-p \mid x \mathbin{.} x \in A \}$
by (*simp add*: *mult-right-dist-test-set*)
have *3*: *Sup* $\{ x;-p \mid x \mathbin{.} x \in A \} \leq$ *Sup A* $\mathbin{;} -p$ **using** *1*
by (*smt mem-Collect-eq mult-iso-left sub-mult-closed sup-test sup-least sup-upper test-set-def*)
have $\forall x{\in}A \mathbin{.} x \leq --(--$*Sup* $\{ x;-p \mid x \mathbin{.} x \in A \} + --p)$
proof
fix x
assume *4*: $x \in A$
hence $x;-p + --p \leq$ *Sup* $\{ x;-p \mid x \mathbin{.} x \in A \} + --p$ **using** *1 2*

by (*smt mem-Collect-eq plus-iso-left sub-mult-closed sup-upper test-set-def sup-test*)
thus $x \leq --(--Sup\ \{\ x;-p \mid x\ .\ x \in A\ \} + --p)$ **using** *1 2 4*
by (*smt plus-closed plus-compl-intro sub-comm test-set-def transitive upper-bound-left sup-test*)
qed
hence $Sup\ A \leq --(--Sup\ \{\ x;-p \mid x\ .\ x \in A\ \} + --p)$ **using** *1*
by (*simp add*: *sup-least*)
hence $Sup\ A\ ;\ -p \leq Sup\ \{\ x;-p \mid x\ .\ x \in A\ \}$ **using** *1 2*
by (*smt plus-closed plus-comm shunting sub-comm sup-test*)
thus $Sup\ A\ ;\ -p = Sup\ \{\ x;-p \mid x\ .\ x \in A\ \}$ **using** *1 2 3*
by (*smt antisymmetric sub-mult-closed sup-test*)
qed

lemma *mult-left-dist-sup*: $test\text{-}set\ A \rightarrow -p\ ;\ Sup\ A = Sup\ \{\ -p;x \mid x\ .\ x \in A\ \}$
proof
assume *1*: $test\text{-}set\ A$
hence *2*: $Sup\ A\ ;\ -p = Sup\ \{\ x;-p \mid x\ .\ x \in A\ \}$
by (*simp add*: *mult-right-dist-sup*)
have *3*: $-p\ ;\ Sup\ A = Sup\ A\ ;\ -p$ **using** *1*
by (*metis sub-comm sup-test*)
have $\{\ -p;x \mid x\ .\ x \in A\ \} = \{\ x;-p \mid x\ .\ x \in A\ \}$
by (*rule set-eqI*, *simp*, *metis 1 sub-comm test-set-def*)
thus $-p\ ;\ Sup\ A = Sup\ \{\ -p;x \mid x\ .\ x \in A\ \}$ **using** *2 3*
by *simp*
qed

definition $Sum :: (nat \Rightarrow 'a) \Rightarrow 'a$
where $Sum\ f = Sup\ \{\ f\ n \mid n{::}nat\ .\ True\ \}$

lemma *Sum-test*: $test\text{-}seq\ t \rightarrow Sum\ t = --Sum\ t$
by (*metis Sum-def sup-test test-seq-test-set*)

lemma *Sum-upper*: $test\text{-}seq\ t \rightarrow t\ n \leq Sum\ t$
by (*smt Sum-def mem-Collect-eq sup-upper test-seq-test-set*)

lemma *Sum-least*: $test\text{-}seq\ t \wedge (\forall\, n\ .\ t\ n \leq -p) \rightarrow Sum\ t \leq -p$
by (*smt Sum-def mem-Collect-eq sup-least test-seq-test-set*)

lemma *mult-right-dist-Sum*: $test\text{-}seq\ t \wedge (\forall\, n\ .\ t\ n;-p \leq -q) \rightarrow Sum\ t;-p \leq -q$
by (*smt Sum-def mem-Collect-eq mult-right-dist-sup sub-mult-closed sup-least test-seq-test-set test-set-def*)

lemma *mult-left-dist-Sum*: $test\text{-}seq\ t \wedge (\forall\, n\ .\ -p;t\ n \leq -q) \rightarrow -p;Sum\ t \leq -q$
by (*smt Sum-def mem-Collect-eq mult-left-dist-sup sub-mult-closed sup-least test-seq-test-set test-set-def*)

lemma *pSum-below-Sum*: $test\text{-}seq\ t \rightarrow pSum\ t\ n \leq Sum\ t$
by (*smt Sum-test Sum-upper bs-mult-right-one one-def pSum-below pSum-test test-seq-def*)

lemma *pSum-sup*: $test\text{-}seq\ t \rightarrow pSum\ t\ n = Sup\ \{\ t\ i \mid i\ .\ i \in \{..{<}n\}\ \}$

```
proof
  assume 1: test-seq t
  hence 2: test-set { t i | i . i ∈ {..<n} }
    by (smt mem-Collect-eq test-seq-def test-set-def)
  have ∀ x∈{ t i | i . i ∈ {..<n} } . x ≤ --pSum t n
    by (simp, smt 1 pSum-test pSum-upper)
  hence 3: Sup { t i | i . i ∈ {..<n} } ≤ --pSum t n using 2
    by (simp add: sup-least)
  have pSum t n ≤ Sup { t i | i . i ∈ {..<n} }
    apply (induct n)
    apply simp
    apply (smt sup-test test-set-def emptyE zero-least-test)
    proof −
      fix n
      assume 4: pSum t n ≤ Sup {t i |i. i ∈ {..<n}}
      have 5: test-set {t i |i. i ∈ {..<n}} using 1
        by (smt mem-Collect-eq test-seq-def test-set-def)
      have 6: test-set {t i |i. i < Suc n} using 1
        by (smt mem-Collect-eq test-seq-def test-set-def)
      hence 7: Sup {t i |i. i < Suc n} = --Sup {t i |i. i < Suc n}
        by (smt sup-test)
      hence ∀ x∈{t i |i. i ∈ {..<n}} . x ≤ --Sup {t i |i. i < Suc n} using 6
        apply simp
        apply rule+
        apply (rule mp)
        apply (rule sup-upper)
        apply simp
        by smt
      hence 8: Sup {t i |i. i ∈ {..<n}} ≤ --Sup {t i |i. i < Suc n} using 5
        by (simp add: sup-least)
      have t n ∈ {t i |i. i < Suc n}
        by (simp, metis lessI)
      hence t n ≤ Sup {t i |i. i < Suc n} using 6
        by (smt sup-upper)
      hence pSum t n + t n ≤ Sup {t i |i. i <Suc n} using 1 4 5 7 8
        by (smt least-upper-bound test-seq-def pSum-test transitive sup-test)
      thus pSum t (Suc n) ≤ Sup {t i |i. i ∈ {..<Suc n}}
        by simp
    qed
  thus pSum t n = Sup { t i | i . i ∈ {..<n} } using 1 2 3
    by (smt antisymmetric sup-test pSum-test)
qed

definition Prod :: (nat ⇒ 'a) ⇒ 'a
  where Prod f = Inf { f n | n::nat . True }

lemma Sum-range: Sum f = Sup (range f)
```

by (*simp add*: *Sum-def image-def*)

lemma *Prod-range*: *Prod f* = *Inf* (*range f*)
by (*simp add*: *Prod-def image-def*)

end

class *complete-antidomain-semiring* = *relative-antidomain-semiring* + *complete-tests* +
assumes *a-dist-Sum*: *ascending-chain f* → −(*Sum f*) = *Prod* (λ*n* . −*f n*)
assumes *a-dist-Prod*: *descending-chain f* → −(*Prod f*) = *Sum* (λ*n* . −*f n*)

begin

lemma *a-ascending-chain*: *ascending-chain f* → *descending-chain* (λ*n* . −*f n*)
by (*smt ascending-chain-def descending-chain-def a-antitone*)

lemma *a-descending-chain*: *descending-chain f* → *ascending-chain* (λ*n* . −*f n*)
by (*smt ascending-chain-def descending-chain-def a-antitone*)

lemma *d-dist-Sum*: *ascending-chain f* → *d*(*Sum f*) = *Sum* (λ*n* . *d*(*f n*))
unfolding *d-def*
apply (*metis a-dist-Sum a-dist-Prod a-ascending-chain*)
done

lemma *d-dist-Prod*: *descending-chain f* → *d*(*Prod f*) = *Prod* (λ*n* . *d*(*f n*))
unfolding *d-def*
apply (*metis a-dist-Sum a-dist-Prod a-descending-chain*)
done

end

class *pre* =
fixes *pre* :: *'a* ⇒ *'a* ⇒ *'a* (**infixr** « *55*)

class *precondition* = *tests* + *pre* +
assumes *pre-closed*: *x*«−*q* = −−(*x*«−*q*)
assumes *pre-seq*: *x*;*y*«−*q* = *x*«*y*«−*q*
assumes *pre-lower-bound-right*: *x*«−*p*;−*q* ≤ *x*«−*q*
assumes *pre-one-increasing*: −*q* ≤ *1*«−*q*

begin

lemma *pre-sub-distr*: *x*«−*p*;−*q* ≤ (*x*«−*p*);(*x*«−*q*)

by (*smt greatest-lower-bound pre-closed pre-lower-bound-right sub-comm sub-mult-closed*)

lemma *pre-below-one*: $x\mathbin{«}{-p} \leq 1$
by (*metis one-greatest pre-closed*)

lemma *pre-lower-bound-left*: $x\mathbin{«}{-p};{-q} \leq x\mathbin{«}{-p}$
by (*smt lower-bound-left pre-closed pre-sub-distr sub-mult-closed transitive*)

lemma *pre-iso*: $-p \leq -q \rightarrow x\mathbin{«}{-p} \leq x\mathbin{«}{-q}$
by (*metis leq-def pre-lower-bound-right*)

lemma *pre-below-pre-one*: $x\mathbin{«}{-p} \leq x\mathbin{«}1$
by (*metis one-def one-greatest pre-iso*)

lemma *pre-seq-below-pre-one*: $x;y\mathbin{«}1 \leq x\mathbin{«}1$
by (*metis one-def pre-below-pre-one pre-closed pre-seq*)

lemma *pre-compose*: $-p \leq x\mathbin{«}{-q} \wedge -q \leq y\mathbin{«}{-r} \rightarrow -p \leq x;y\mathbin{«}{-r}$
by (*metis pre-closed pre-iso transitive pre-seq*)

end

class *precondition-test-test* = *precondition* +
assumes *pre-test-test*: $-p;(-p\mathbin{«}{-q}) = -p;{-q}$

begin

lemma *pre-one*: $1\mathbin{«}{-p} = -p$
by (*metis bs-mult-left-one one-def pre-closed pre-test-test*)

lemma *pre-import*: $-p;(x\mathbin{«}{-q}) = -p;(-p;x\mathbin{«}{-q})$
by (*metis pre-closed pre-seq pre-test-test*)

lemma *pre-import-composition*: $-p;(-p;x;y\mathbin{«}{-q}) = -p;(x\mathbin{«}y\mathbin{«}{-q})$
by (*metis pre-closed pre-seq pre-import*)

lemma *pre-import-equiv*: $-p \leq x\mathbin{«}{-q} \leftrightarrow -p \leq -p;x\mathbin{«}{-q}$
by (*metis leq-def pre-closed pre-import*)

lemma *pre-import-equiv-mult*: $-p;{-q} \leq x\mathbin{«}{-s} \leftrightarrow -p;{-q} \leq -q;x\mathbin{«}{-s}$
by (*smt leq-def pre-closed sub-assoc sub-mult-closed pre-import*)

end

class *precondition-promote* = *precondition* +
  assumes *pre-test-promote*: $-p\mathbin{«}-q = -p\mathbin{«}-p;-q$

begin

lemma *pre-mult-test-promote*: $x;-p\mathbin{«}-q = x;-p\mathbin{«}-p;-q$
  by (*metis pre-seq pre-test-promote sub-mult-closed*)

end

class *precondition-test-box* = *precondition* +
  assumes *pre-test*: $-p\mathbin{«}-q = --p+-q$

begin

lemma *pre-test-neg*: $--p;(-p\mathbin{«}-q) = --p$
  by (*metis mult-absorb pre-test*)

lemma *pre-zero*: $0\mathbin{«}-q = 1$
  by (*metis one-compl one-def plus-left-one pre-test*)

lemma *pre-export*: $-p;x\mathbin{«}-q = --p+(x\mathbin{«}-q)$
  by (*metis pre-closed pre-seq pre-test*)

lemma *pre-neg-mult*: $--p \leq -p;x\mathbin{«}-q$
  by (*metis leq-def pre-closed pre-seq pre-test-neg*)

lemma *pre-test-test-same*: $-p\mathbin{«}-p = 1$
  by (*metis plus-comm plus-compl pre-test*)

lemma *test-below-pre-test-mult*: $-q \leq -p\mathbin{«}-p;-q$
  by (*metis pre-test reflexive shunting sub-mult-closed*)

lemma *test-below-pre-test*: $-q \leq -p\mathbin{«}-q$
  by (*metis pre-test upper-bound-right*)

lemma *test-below-pre-test-2*: $--p \leq -p\mathbin{«}-q$
  by (*metis pre-test upper-bound-left*)

lemma *pre-test-zero*: $-p\mathbin{«}0 = --p$
  by (*metis one-compl plus-right-zero pre-test*)

lemma *pre-test-one*: $-p\mathbin{«}1 = 1$
  by (*metis one-def plus-right-one pre-test*)

subclass *precondition-test-test*
apply *unfold-locales*
apply (*metis mult-compl-intro pre-test*)
done

subclass *precondition-promote*
apply *unfold-locales*
apply (*metis plus-comm plus-compl-intro pre-test sub-mult-closed*)
done

end

class *precondition-test-diamond* = *precondition* +
assumes *pre-test*: $-p\mathbin{«}-q = -p;-q$

begin

lemma *pre-test-neg*: $--p;(-p\mathbin{«}-q) = 0$
by (*metis bs-mult-right-zero mult-compl pre-test sub-assoc sub-comm*)

lemma *pre-zero*: $0\mathbin{«}-q = 0$
by (*metis bs-mult-left-zero one-compl pre-test*)

lemma *pre-export*: $-p;x\mathbin{«}-q = -p;(x\mathbin{«}-q)$
by (*metis pre-closed pre-seq pre-test*)

lemma *pre-neg-mult*: $-p;x\mathbin{«}-q \leq -p$
by (*metis lower-bound-left pre-closed pre-export*)

lemma *pre-test-test-same*: $-p\mathbin{«}-p = -p$
by (*metis mult-idempotent pre-test*)

lemma *test-above-pre-test-plus*: $--p\mathbin{«}-p+-q \leq -q$
by (*metis double-negation lower-bound-left mult-compl-intro plus-closed pre-test sub-comm*)

lemma *test-above-pre-test*: $-p\mathbin{«}-q \leq -q$
by (*metis lower-bound-right pre-test*)

lemma *test-above-pre-test-2*: $-p\mathbin{«}-q \leq -p$
by (*metis lower-bound-left pre-test*)

lemma *pre-test-zero*: $-p\mathbin{«}0 = 0$
by (*metis bs-mult-right-zero one-compl pre-test*)

lemma *pre-test-one*: $-p\mathbin{«}1 = -p$
  by (*metis bs-mult-right-one one-def pre-test*)

subclass *precondition-test-test*
  apply *unfold-locales*
  apply (*metis mult-idempotent pre-export pre-test*)
  done

subclass *precondition-promote*
  apply *unfold-locales*
  apply (*metis mult-idempotent pre-seq pre-test*)
  done

end

class *precondition-distr-mult* = *precondition* +
  assumes *pre-distr-mult*: $x\mathbin{«}-p;-q = (x\mathbin{«}-p);(x\mathbin{«}-q)$

begin

end

class *precondition-distr-plus* = *precondition* +
  assumes *pre-distr-plus*: $x\mathbin{«}-p+-q = (x\mathbin{«}-p)+(x\mathbin{«}-q)$

begin

end

class *ite* =
  fixes *ite* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ (- $\lhd$ - $\rhd$ - [*58*,*58*,*58*] *57*)

class *while* =
  fixes *while* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $*$ *59*)

class *hoare-triple* =
  fixes *hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow$ *bool* (- $\{\!|$ - $|\!\}$ - [*54*,*54*,*54*] *53*)

class *ifthenelse* = *precondition* + *ite* +

assumes *ite-pre*: $x \lhd -p \rhd y \ll -q = -p;(x \ll -q) + --p;(y \ll -q)$

begin

lemma *ite-pre-then*: $-p;(x \lhd -p \rhd y \ll -q) = -p;(x \ll -q)$
proof −
have $-p;(x \lhd -p \rhd y \ll -q) = -p;(x \ll -q) + 0;(y \ll -q)$
by (*smt ite-pre plus-absorb plus-distr-mult-left pre-closed sub-assoc sub-mult-closed zero-def*)
thus *?thesis*
by (*smt plus-absorb plus-right-zero pre-closed sub-mult-closed zero-def*)
qed

lemma *ite-pre-else*: $--p;(x \lhd -p \rhd y \ll -q) = --p;(y \ll -q)$
proof −
have $--p;(x \lhd -p \rhd y \ll -q) = 0;(x \ll -q) + --p;(y \ll -q)$
by (*smt ite-pre mult-distr-plus-left mult-idempotent pre-closed sub-assoc sub-mult-closed zero-def*)
thus *?thesis*
by (*smt mult-idempotent plus-left-zero pre-closed sub-assoc sub-mult-closed zero-def*)
qed

lemma *ite-import-mult-then*: $-p;-q \leq x \ll -r \rightarrow -p;-q \leq x \lhd -p \rhd y \ll -r$
by (*smt ite-pre-then leq-def pre-closed sub-assoc sub-comm sub-mult-closed*)

lemma *ite-import-mult-else*: $--p;-q \leq y \ll -r \rightarrow --p;-q \leq x \lhd -p \rhd y \ll -r$
by (*smt ite-pre-else leq-def pre-closed sub-assoc sub-comm sub-mult-closed*)

lemma *ite-import-mult*: $-p;-q \leq x \ll -r \wedge --p;-q \leq y \ll -r \rightarrow -q \leq x \lhd -p \rhd y \ll -r$
by (*metis ite-import-mult-then ite-import-mult-else leq-cases pre-closed*)

end

class *whiledo* = *ifthenelse* + *while* +
assumes *while-pre*: $-p*x \ll -q = -p;(x \ll -p*x \ll -q) + --p;-q$
assumes *while-post*: $-p*x \ll -q = -p*x \ll --p;-q$

begin

lemma *while-pre-then*: $-p;(-p*x \ll -q) = -p;(x \ll -p*x \ll -q)$
by (*smt pre-closed sub-comm while-pre wnf-lemma-1*)

lemma *while-pre-else*: $--p;(-p*x \ll -q) = --p;-q$
by (*smt pre-closed sub-comm while-pre wnf-lemma-3*)

lemma *while-pre-sub-1*: $-p*x \ll -q \leq x;(-p*x) \lhd -p \rhd 1 \ll -q$
by (*smt ite-pre-else ite-pre-then mult-iso-right plus-cases plus-iso-right pre-closed pre-one-increasing pre-seq sub-comm sub-mult-closed while-pre*)

lemma *while-pre-sub-2*: $-p*x \ll -q \leq x \lhd -p \rhd 1 \ll -p*x \ll -q$

by (*smt ite-pre-else ite-pre-then mult-iso-right plus-cases plus-iso-right pre-closed pre-one-increasing sub-comm sub-mult-closed while-pre while-pre-else*)

lemma *while-pre-compl*: $--p \leq -p*x\ll--p$
by (*metis lower-bound-right mult-idempotent pre-closed while-pre-else*)

lemma *while-pre-compl-one*: $--p \leq -p*x\ll 1$
by (*metis bs-mult-right-one lower-bound-right one-def pre-closed while-pre-else*)

lemma *while-export-equiv*: $-q \leq -p*x\ll 1 \leftrightarrow -p;-q \leq -p*x\ll 1$
by (*smt bs-mult-left-one leq-plus lower-bound-right one-def pre-closed shunting sub-comm while-pre-else*)

lemma *nat-test-pre*: *nat-test t s* $\wedge\ -q \leq s \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q) \rightarrow -q \leq -p*x\ll--p;-q$
proof
assume *1*: *nat-test t s* $\wedge\ -q \leq s \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q)$
have *2*: $-q;--p \leq -p*x\ll--p;-q$
by (*smt leq-def mult-idempotent pre-closed sub-assoc sub-comm sub-mult-closed while-pre-else*)
have $\forall n\ .\ t\ n;-p;-q \leq -p*x\ll--p;-q$
proof
fix *n*
show $t\ n;-p;-q \leq -p*x\ll--p;-q$
proof (*induct n rule*: *nat-less-induct*)
fix *n*
have *3*: $t\ n = --(t\ n)$ **using** *1*
by (*smt nat-test-def*)
assume $\forall m{<}n\ .\ t\ m;-p;-q \leq -p*x\ll--p;-q$
hence $\forall m{<}n\ .\ t\ m;-p;-q + t\ m;--p;-q \leq -p*x\ll--p;-q$ **using** *1 2*
by (*smt least-upper-bound leq-def nat-test-def pre-closed sub-assoc sub-comm sub-mult-closed*)
hence $\forall m{<}n\ .\ t\ m;-q \leq -p*x\ll--p;-q$ **using** *1*
by (*smt bs-mult-right-one mult-distr-plus-left mult-distr-plus-right nat-test-def plus-compl sub-mult-closed*)
hence $pSum\ t\ n;-q \leq -p*x\ll--p;-q$ **using** *1*
by (*smt pSum-below-nat pre-closed sub-mult-closed*)
hence $t\ n;-p;-q;(-p*x\ll--p;-q) = t\ n;-p;-q$ **using** *1 3*
by (*smt leq-def pSum-test-nat pre-closed pre-sub-distr sub-assoc sub-comm sub-mult-closed transitive while-pre-then*)
thus $t\ n;-p;-q \leq -p*x\ll--p;-q$ **using** *3*
by (*smt lower-bound-right pre-closed sub-mult-closed*)
qed
qed
hence $-q;-p \leq -p*x\ll--p;-q$ **using** *1*
by (*smt leq-def nat-test-def pre-closed sub-assoc sub-comm sub-mult-closed*)
thus $-q \leq -p*x\ll--p;-q$ **using** *2*
by (*smt bs-mult-right-one leq-def mult-distr-plus-left mult-distr-plus-right plus-compl pre-closed sub-mult-closed*)
qed

lemma *nat-test-pre-1*: *nat-test t s* $\wedge\ -r \leq s \wedge -r \leq -q \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q) \rightarrow -r \leq -p*x\ll--p;-q$
proof
let *?qs* $= -q;s$
assume *1*: *nat-test t s* $\wedge\ -r \leq s \wedge -r \leq -q \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q)$

hence *2*: $-r \leq ?qs$
by (*metis greatest-lower-bound nat-test-def*)
have $\forall\, n\ .\ t\ n;-p;?qs \leq x\ll pSum\ t\ n;?qs$ **using** *1*
by (*smt leq-def lower-bound-left nat-test-def pSum-below-sum pSum-test-nat sub-assoc sub-mult-closed transitive*)
hence $?qs \leq -p*x\ll--p;?qs$ **using** *1*
by (*smt lower-bound-left lower-bound-right nat-test-def nat-test-pre pSum-test-nat pre-closed sub-assoc sub-mult-closed transitive*)
thus $-r \leq -p*x\ll--p;-q$ **using** *1 2*
by (*smt lower-bound-left nat-test-def pre-closed pre-iso sub-assoc sub-mult-closed transitive*)
qed

lemma *nat-test-pre-2*: $nat\text{-}test\ t\ s \wedge -r \leq s \wedge (\forall\, n\ .\ t\ n;-p \leq x\ll pSum\ t\ n) \rightarrow -r \leq -p*x\ll 1$
proof
assume *1*: $nat\text{-}test\ t\ s \wedge -r \leq s \wedge (\forall\, n\ .\ t\ n;-p \leq x\ll pSum\ t\ n)$
hence $-r \leq -p*x\ll--p;s$
by (*smt leq-def nat-test-def nat-test-pre-1 pSum-below-sum pSum-test-nat sub-assoc sub-comm*)
thus $-r \leq -p*x\ll 1$ **using** *1*
by (*smt nat-test-def one-def pre-below-pre-one pre-closed sub-mult-closed transitive*)
qed

lemma *nat-test-pre-3*: $nat\text{-}test\ t\ s \wedge -q \leq s \wedge (\forall\, n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q) \rightarrow -q \leq -p*x\ll 1$
proof −
have $-p*x\ll--p;-q \leq -p*x\ll 1$
by (*metis pre-below-pre-one sub-mult-closed*)
thus *?thesis*
by (*smt nat-test-pre one-double-compl pre-closed sub-mult-closed transitive*)
qed

definition $aL :: 'a$
where $aL \equiv 1*1\ll 1$

lemma *aL-test*: $aL = --aL$
by (*metis aL-def one-def pre-closed*)

end

class *atoms* = *tests* +
fixes *Atomic-program* :: $'a\ set$
fixes *Atomic-test* :: $'a\ set$
assumes *one-atomic-program*: $1 \in Atomic\text{-}program$
assumes *zero-atomic-test*: $0 \in Atomic\text{-}test$
assumes *atomic-test-test*: $p \in Atomic\text{-}test \rightarrow p = --p$

class *while-program* = *whiledo* + *atoms* + *power*

begin

inductive-set *Test-expression* :: $'a\ set$

where *atom-test*: $p \in \mathit{Atomic\text{-}test} \Rightarrow p \in \mathit{Test\text{-}expression}$
| *neg-test*: $p \in \mathit{Test\text{-}expression} \Rightarrow -p \in \mathit{Test\text{-}expression}$
| *conj-test*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Test\text{-}expression} \Rightarrow p;q \in \mathit{Test\text{-}expression}$

lemma *test-expression-test*: $p \in \mathit{Test\text{-}expression} \rightarrow p = --p$
apply *rule*
apply (*induct rule*: *Test-expression.induct*)
apply (*metis atomic-test-test*)
apply *simp*
apply (*metis sub-mult-closed*)
done

lemma *disj-test*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Test\text{-}expression} \rightarrow p+q \in \mathit{Test\text{-}expression}$
by (*smt conj-test neg-test plus-def test-expression-test*)

lemma *zero-test-expression*: $0 \in \mathit{Test\text{-}expression}$
by (*metis atom-test zero-atomic-test*)

lemma *one-test-expression*: $1 \in \mathit{Test\text{-}expression}$
by (*metis neg-test one-def zero-test-expression*)

lemma *pSum-test-expression*: $(\forall n\ .\ t\ n \in \mathit{Test\text{-}expression}) \rightarrow \mathit{pSum}\ t\ n \in \mathit{Test\text{-}expression}$
apply *rule*
apply (*induct n*)
apply (*metis pSum.simps*(*1*) *zero-test-expression*)
apply (*metis disj-test pSum.simps*(*2*))
done

inductive-set *While-program* :: *'a set*
where *atom-prog*: $x \in \mathit{Atomic\text{-}program} \Rightarrow x \in \mathit{While\text{-}program}$
| *seq-prog*: $x \in \mathit{While\text{-}program} \wedge y \in \mathit{While\text{-}program} \Rightarrow x;y \in \mathit{While\text{-}program}$
| *cond-prog*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \wedge y \in \mathit{While\text{-}program} \Rightarrow x \lhd p \rhd y \in \mathit{While\text{-}program}$
| *while-prog*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \Rightarrow p * x \in \mathit{While\text{-}program}$

lemma *one-while-program*: $1 \in \mathit{While\text{-}program}$
by (*metis atom-prog one-atomic-program*)

lemma *power-while-program*: $x \in \mathit{While\text{-}program} \rightarrow x\hat{}n \in \mathit{While\text{-}program}$
apply *rule*
apply (*induct n*)
apply (*metis one-while-program power-0*)
apply (*metis seq-prog power-Suc*)
done

inductive-set *Pre-expression* :: *'a set*
where *test-pre*: $p \in \mathit{Test\text{-}expression} \Rightarrow p \in \mathit{Pre\text{-}expression}$
| *neg-pre*: $p \in \mathit{Pre\text{-}expression} \Rightarrow -p \in \mathit{Pre\text{-}expression}$

| *conj-pre*: $p \in \mathit{Pre\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \Rightarrow p;q \in \mathit{Pre\text{-}expression}$
| *pre-pre*: $p \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \Rightarrow x \ll p \in \mathit{Pre\text{-}expression}$

lemma *pre-expression-test*: $p \in \mathit{Pre\text{-}expression} \rightarrow p = --p$
apply *rule*
apply (*induct rule*: *Pre-expression.induct*)
apply (*metis test-expression-test*)
apply *simp*
apply (*metis sub-mult-closed*)
apply (*metis pre-closed*)
done

lemma *disj-pre*: $p \in \mathit{Pre\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \rightarrow p+q \in \mathit{Pre\text{-}expression}$
by (*smt conj-pre neg-pre plus-def pre-expression-test*)

lemma *zero-pre-expression*: $0 \in \mathit{Pre\text{-}expression}$
by (*metis test-pre zero-test-expression*)

lemma *one-pre-expression*: $1 \in \mathit{Pre\text{-}expression}$
by (*metis test-pre one-test-expression*)

lemma *pSum-pre-expression*: $(\forall\, n\ .\ t\ n \in \mathit{Pre\text{-}expression}) \rightarrow \mathit{pSum}\ t\ n \in \mathit{Pre\text{-}expression}$
apply *rule*
apply (*induct n*)
apply (*metis pSum.simps*(1) *zero-pre-expression*)
apply (*metis disj-pre pSum.simps*(2))
done

lemma *aL-pre-expression*: $aL \in \mathit{Pre\text{-}expression}$
by (*metis aL-def one-pre-expression one-test-expression one-while-program pre-pre while-prog*)

end

class *hoare-calculus* = *while-program* + *complete-tests*

begin

definition *tfun* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where $\mathit{tfun}\ p\ x\ q\ r \equiv p + (x \ll q;r)$

lemma *tfun-test*: $p = --p \wedge q = --q \wedge r = --r \rightarrow \mathit{tfun}\ p\ x\ q\ r = --\mathit{tfun}\ p\ x\ q\ r$
by (*smt tfun-def sub-mult-closed pre-closed plus-closed*)

lemma *tfun-pre-expression*: $x \in \mathit{While\text{-}program} \wedge p \in \mathit{Pre\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge r \in \mathit{Pre\text{-}expression} \rightarrow \mathit{tfun}\ p\ x\ q\ r \in \mathit{Pre\text{-}expression}$
by (*metis tfun-def conj-pre disj-pre pre-pre*)

lemma *tfun-iso*: $p = --p \wedge q = --q \wedge r = --r \wedge s = --s \wedge r \leq s \rightarrow \mathit{tfun}\ p\ x\ q\ r \leq \mathit{tfun}\ p\ x\ q\ s$

by (*smt tfun-def mult-iso-right pre-iso sub-mult-closed plus-iso-right pre-closed*)

definition *tseq* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow nat \Rightarrow 'a$
where *tseq p x q r n* ≡ (*tfun p x q* ^ *n*) *r*

lemma *tseq-test*: $p = --p \wedge q = --q \wedge r = --r \rightarrow tseq\ p\ x\ q\ r\ n = --tseq\ p\ x\ q\ r\ n$
apply (*induct n*)
apply (*smt tseq-def tfun-test power-zero-id id-def*)
apply (*smt tseq-def tfun-test power-succ-unfold-ext*)
done

lemma *tseq-test-seq*: $p = --p \wedge q = --q \wedge r = --r \rightarrow test\text{-}seq\ (tseq\ p\ x\ q\ r)$
by (*metis test-seq-def tseq-test*)

lemma *tseq-pre-expression*: $x \in While\text{-}program \wedge p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge r \in Pre\text{-}expression \rightarrow tseq\ p\ x\ q\ r\ n \in Pre\text{-}expression$
apply (*induct n*)
apply (*smt tseq-def id-def power-zero-id*)
apply (*smt tseq-def power-succ-unfold-ext tfun-pre-expression*)
done

definition *tsum* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where *tsum p x q r* ≡ *Sum* (*tseq p x q r*)

lemma *tsum-test*: $p = --p \wedge q = --q \wedge r = --r \rightarrow tsum\ p\ x\ q\ r = --tsum\ p\ x\ q\ r$
by (*metis Sum-test tseq-test-seq tsum-def*)

lemma *t-fun-test*: $q = --q \rightarrow tfun\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL)) = --tfun\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tfun-test*)

lemma *t-fun-pre-expression*: $x \in While\text{-}program \wedge p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \rightarrow tfun\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL)) \in Pre\text{-}expression$
by (*metis aL-pre-expression conj-pre disj-pre neg-pre pre-pre test-pre tfun-pre-expression while-prog*)

lemma *t-seq-test*: $q = --q \rightarrow tseq\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL))\ n = --tseq\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL))\ n$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tseq-test*)

lemma *t-seq-test-seq*: $q = --q \rightarrow test\text{-}seq\ (tseq\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL)))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tseq-test-seq*)

lemma *t-seq-pre-expression*: $x \in While\text{-}program \wedge p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \rightarrow tseq\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL))\ n \in Pre\text{-}expression$
by (*metis aL-pre-expression conj-pre disj-pre neg-pre pre-pre test-pre tseq-pre-expression while-prog*)

lemma *t-sum-test*: $q = --q \rightarrow tsum\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL)) = --tsum\ (-p)\ x\ (p{*}x{\ll}q)\ (-p{+}(x{\ll}(p{*}x{\ll}q);aL))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tsum-test*)

definition *tfun2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where *tfun2 p q x r s* ≡ $p + q;(x{\ll}r;s)$

lemma *tfun2-test*: $p = --p \wedge q = --q \wedge r = --r \wedge s = --s \rightarrow tfun2\ p\ q\ x\ r\ s = --tfun2\ p\ q\ x\ r\ s$
by (*smt tfun2-def sub-mult-closed pre-closed plus-closed*)

lemma *tfun2-pre-expression*: $x \in While\text{-}program \wedge p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge r \in Pre\text{-}expression \wedge s \in Pre\text{-}expression \rightarrow tfun2\ p\ q\ x\ r\ s \in Pre\text{-}expression$
by (*metis tfun2-def conj-pre disj-pre pre-pre*)

lemma *tfun2-iso*: $p = --p \wedge q = --q \wedge r = --r \wedge s1 = --s1 \wedge s2 = --s2 \wedge s1 \leq s2 \rightarrow tfun2\ p\ q\ x\ r\ s1 \leq tfun2\ p\ q\ x\ r\ s2$
by (*smt tfun2-def mult-iso-right pre-iso sub-mult-closed plus-iso-right pre-closed*)

definition *tseq2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow nat \Rightarrow 'a$
where $tseq2\ p\ q\ x\ r\ s\ n \equiv (tfun2\ p\ q\ x\ r\ \hat{}\ n)\ s$

lemma *tseq2-test*: $p = --p \wedge q = --q \wedge r = --r \wedge s = --s \rightarrow tseq2\ p\ q\ x\ r\ s\ n = --tseq2\ p\ q\ x\ r\ s\ n$
apply (*induct n*)
apply (*smt tseq2-def power-zero-id id-def*)
apply (*smt tseq2-def tfun2-test power-succ-unfold-ext*)
done

lemma *tseq2-test-seq*: $p = --p \wedge q = --q \wedge r = --r \wedge s = --s \rightarrow test\text{-}seq\ (tseq2\ p\ q\ x\ r\ s)$
by (*metis test-seq-def tseq2-test*)

lemma *tseq2-pre-expression*: $x \in While\text{-}program \wedge p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge r \in Pre\text{-}expression \wedge s \in Pre\text{-}expression \rightarrow tseq2\ p\ q\ x\ r\ s\ n \in Pre\text{-}expression$
apply (*induct n*)
apply (*smt tseq2-def id-def power-zero-id*)
apply (*smt tseq2-def power-succ-unfold-ext tfun2-pre-expression*)
done

definition *tsum2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where $tsum2\ p\ q\ x\ r\ s \equiv Sum\ (tseq2\ p\ q\ x\ r\ s)$

lemma *tsum2-test*: $p = --p \wedge q = --q \wedge r = --r \wedge s = --s \rightarrow tsum2\ p\ q\ x\ r\ s = --tsum2\ p\ q\ x\ r\ s$
by (*metis Sum-test tseq2-test-seq tsum2-def*)

lemma *t-fun2-test*: $p = --p \wedge q = --q \rightarrow tfun2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) = --tfun2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tfun2-test*)

lemma *t-fun2-pre-expression*: $x \in While\text{-}program \wedge p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \rightarrow tfun2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) \in Pre\text{-}expression$
by (*metis aL-pre-expression conj-pre disj-pre neg-pre pre-pre test-pre tfun2-pre-expression while-prog*)

lemma *t-seq2-test*: $p = --p \wedge q = --q \rightarrow tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n = --tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tseq2-test*)

lemma *t-seq2-test-seq*: $p = --p \wedge q = --q \rightarrow test\text{-}seq\ (tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tseq2-test-seq*)

lemma *t-seq2-pre-expression*: $x \in While\text{-}program \wedge p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \rightarrow tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n \in Pre\text{-}expression$
by (*metis aL-pre-expression conj-pre disj-pre neg-pre pre-pre test-pre tseq2-pre-expression while-prog*)

lemma *t-sum2-test*: $p = --p \wedge q = --q \rightarrow tsum2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) = --tsum2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))$
by (*smt aL-test double-negation plus-closed pre-closed sub-mult-closed tsum2-test*)

lemma *t-seq2-below-t-seq*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n \leq tseq\ (-p)\ x\ (p*x\ll q)\ (-p+(x\ll(p*x\ll q);aL))\ n$
proof
let *?t2* $= tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))$
let *?t* $= tseq\ (-p)\ x\ (p*x\ll q)\ (-p+(x\ll(p*x\ll q);aL))$
assume *1*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program$
show *?t2* $n \leq$ *?t* n
proof (*induct* n)
show *?t2* $0 \leq$ *?t* 0 **using** *1*
by (*smt aL-test id-def lower-bound-left lower-bound-right plus-iso power-zero-id pre-closed pre-expression-test sub-mult-closed test-pre tseq2-def tseq-def*)
next
fix n
assume *?t2* $n \leq$ *?t* n
hence *2*: *?t2* $(Suc\ n) \leq tfun2\ (-\ p\ ;\ q)\ p\ x\ (p\ *\ x\ \ll\ q)$ (*?t* n) **using** *1*
by (*smt power-succ-unfold-ext pre-closed pre-expression-test sub-mult-closed t-seq2-test t-seq-test test-pre tfun2-iso tseq2-def*)
have ... $\leq$ *?t* $(Suc\ n)$ **using** *1*
by (*smt lower-bound-left lower-bound-right plus-iso power-succ-unfold-ext pre-closed pre-expression-test sub-mult-closed t-seq-test test-pre tfun2-def tfun-def tseq-def*)
thus *?t2* $(Suc\ n) \leq$ *?t* $(Suc\ n)$ **using** *1 2*
by (*smt pre-closed pre-expression-test sub-mult-closed t-seq2-test t-seq-test test-pre tfun2-test transitive*)
qed
qed

lemma *t-seq2-below-t-sum*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n \leq tsum\ (-p)\ x\ (p*x\ll q)\ (-p+(x\ll(p*x\ll q);aL))$
by (*smt Sum-upper pre-expression-test t-seq2-below-t-seq t-seq2-test t-seq-test t-sum-test test-pre test-seq-def transitive tsum-def*)

lemma *t-sum2-below-t-sum*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tsum2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) \leq tsum\ (-p)\ x\ (p*x\ll q)\ (-p+(x\ll(p*x\ll q);aL))$
by (*smt Sum-least pre-expression-test t-seq2-below-t-sum t-seq2-test t-sum-test test-pre test-seq-def tsum2-def*)

lemma *t-seq2-below-w*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tseq2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n \leq p*x\ll q$
apply (*cases* n)
apply (*smt aL-test id-def lower-bound-left mult-iso-right plus-comm plus-iso-right power-zero-id pre-closed pre-expression-test pre-iso sub-mult-closed test-pre tseq2-def while-pre*)
apply *simp*
unfolding *tseq2-def power-succ-unfold-ext*
apply (*smt lower-bound-left mult-iso-right plus-comm plus-iso-right pre-closed pre-expression-test pre-iso sub-mult-closed t-seq2-test test-pre tseq2-def while-pre tfun2-def*)
done

lemma *t-sum2-below-w*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tsum2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) \leq p*x\ll q$
by (*smt Sum-least pre-closed pre-expression-test t-seq2-below-w t-seq2-test-seq test-pre tsum2-def*)

lemma *t-sum2-w*: $aL = 1 \wedge p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow tsum2\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)) = p*x\ll q$
proof
let *?w* $= p*x\ll q$

let *?s* = $-p;q+p;(x\ll ?w;aL)$
assume *1*: $aL = 1 \wedge p \in$ *Test-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program*
have *?w* = *tseq2* $(-p;q)\ p\ x\ ?w\ ?s\ 0$ **using** *1*
by (*smt bs-mult-right-one id-def plus-comm power-zero-id pre-closed pre-expression-test sub-mult-closed test-expression-test tseq2-def while-pre*)
hence *?w* $\leq$ *tsum2* $(-p;q)\ p\ x\ ?w\ ?s$ **using** *1*
by (*smt Sum-upper pre-expression-test t-seq2-test-seq test-pre tsum2-def*)
thus *tsum2* $(-p;q)\ p\ x\ ?w\ ?s = ?w$ **using** *1*
by (*smt antisymmetric pre-closed pre-expression-test t-sum2-test t-sum2-below-w test-pre*)
qed

inductive *derived-hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow$ *bool* (- ⦇ - ⦈ - [*54,54,54*] *53*)
where *atom-trip*: $p \in$ *Pre-expression* $\wedge\ x \in$ *Atomic-program* $\Rightarrow x\ll p⦇x⦈p$
| *seq-trip*: $p⦇x⦈q \wedge q⦇y⦈r \Rightarrow p⦇x;y⦈r$
| *cond-trip*: $p \in$ *Test-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ p;q⦇x⦈r \wedge -p;q⦇y⦈r \Rightarrow q⦇x\lhd p\rhd y⦈r$
| *while-trip*: $p \in$ *Test-expression* $\wedge\ q \in$ *Pre-expression* $\wedge$ *test-seq* $t \wedge q \leq$ *Sum* $t \wedge t\ 0;p;q⦇x⦈aL;q \wedge (\forall\, n{>}0\ .\ t\ n;p;q⦇x⦈pSum\ t\ n;q) \Rightarrow q⦇p{*}x⦈{-}p;q$
| *cons-trip*: $p \in$ *Pre-expression* $\wedge\ s \in$ *Pre-expression* $\wedge\ p \leq q \wedge q⦇x⦈r \wedge r \leq s \Rightarrow p⦇x⦈s$

lemma *derived-type*: $p⦇x⦈q \Rightarrow p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program*
apply (*induct rule*: *derived-hoare-triple.induct*)
apply (*metis atom-prog pre-pre*)
apply (*metis seq-prog*)
apply (*metis cond-prog*)
apply (*metis conj-pre neg-pre test-pre while-prog*)
apply *metis*
done

lemma *cons-pre-trip*: $p \in$ *Pre-expression* $\wedge\ q⦇y⦈r \rightarrow p;q⦇y⦈r$
by (*smt conj-pre cons-trip derived-type lower-bound-right reflexive pre-expression-test*)

lemma *cons-post-trip*: $q \in$ *Pre-expression* $\wedge\ r \in$ *Pre-expression* $\wedge\ p⦇y⦈q;r \rightarrow p⦇y⦈r$
by (*smt cons-trip derived-type lower-bound-right reflexive pre-expression-test*)

definition *valid-hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow$ *bool* (- ⟨ - ⟩ - [*54,54,54*] *53*)
where $p\langle x\rangle q \equiv (p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program* $\wedge\ p \leq x\ll q)$

end

class *hoare-calculus-sound* = *hoare-calculus* +
assumes *while-soundness*: $-p;-q \leq x\ll -q \rightarrow aL;-q \leq -p{*}x\ll -q$

begin

lemma *while-soundness-0*: $-p;-q \leq x\ll -q \rightarrow -q;aL \leq -p{*}x\ll --p;-q$
by (*smt while-soundness aL-test sub-comm while-post*)

lemma *while-soundness-1*: *test-seq* $t \wedge -q \leq$ *Sum* $t \wedge t\ 0;-p;-q \leq x\ll aL;-q \wedge (\forall\, n{>}0\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n;-q) \rightarrow -q \leq -p{*}x\ll --p;-q$
proof

assume *1*: $\mathit{test\text{-}seq}\ t \wedge -q \leq \mathit{Sum}\ t \wedge t\ 0;-p;-q \leq x \ll aL;-q \wedge (\forall\, n{>}0\ .\ t\ n;-p;-q \leq x \ll \mathit{pSum}\ t\ n;-q)$
hence $\forall\, n\ .\ t\ n;-p;-q \leq x \ll -q$
by (*smt aL-test pSum-test pre-closed pre-lower-bound-right sub-mult-closed test-seq-def transitive*)
hence *2*: $-p;-q \leq x \ll -q$ **using** *1*
by (*smt Sum-test leq-def mult-right-dist-Sum pre-closed sub-assoc sub-comm sub-mult-closed test-seq-def*)
have $\forall\, n\ .\ t\ n;-q \leq -p{*}x \ll --p;-q \wedge \mathit{pSum}\ t\ n;-q \leq -p{*}x \ll --p;-q$
proof
fix n
show $t\ n;-q \leq -p{*}x \ll --p;-q \wedge \mathit{pSum}\ t\ n;-q \leq -p{*}x \ll --p;-q$
proof (*induct n rule*: *nat-less-induct*)
fix n
assume $\forall\, m{<}n\ .\ t\ m;-q \leq -p{*}x \ll --p;-q \wedge \mathit{pSum}\ t\ m;-q \leq -p{*}x \ll --p;-q$
hence *3*: $\mathit{pSum}\ t\ n;-q \leq -p{*}x \ll --p;-q$ **using** *1*
apply (*cases n*)
apply (*smt bs-mult-left-zero pSum.simps*(*1*) *pre-closed sub-mult-closed zero-least-test*)
apply (*smt least-upper-bound mult-distr-plus-right pSum.simps*(*2*) *pSum-test pre-closed sub-mult-closed test-seq-def*)
done
hence $x \ll \mathit{pSum}\ t\ n;-q \leq x \ll -p{*}x \ll --p;-q$ **using** *1*
by (*smt pSum-test pre-closed pre-iso sub-mult-closed*)
hence *4*: $-p;(t\ n;-q) \leq -p;(-p{*}x \ll --p;-q)$ **using** *1 2*
apply (*cases n*)
apply (*smt aL-test leq-def mult-idempotent mult-iso-right pre-closed pre-lower-bound-left sub-assoc sub-comm sub-mult-closed test-seq-def transitive while-pre-then while-soundness-0*)
apply (*smt greatest-lower-bound lower-bound-left pSum-test pre-closed sub-assoc sub-comm sub-mult-closed test-seq-def transitive while-pre-then*)
done
have $--p;(t\ n;-q) \leq --p;(-p{*}x \ll --p;-q)$ **using** *1*
by (*smt leq-def lower-bound-right sub-assoc sub-comm sub-mult-closed test-seq-def while-pre-else*)
thus $t\ n;-q \leq -p{*}x \ll --p;-q \wedge \mathit{pSum}\ t\ n;-q \leq -p{*}x \ll --p;-q$ **using** *1 3 4*
by (*smt leq-cases-2 pre-closed sub-mult-closed test-seq-def*)
qed
qed
thus $-q \leq -p{*}x \ll --p;-q$ **using** *1*
by (*smt Sum-test leq-def mult-right-dist-Sum pre-closed sub-comm sub-mult-closed*)
qed

lemma *while-soundness-2*: $\mathit{test\text{-}seq}\ t \wedge -r \leq \mathit{Sum}\ t \wedge (\forall\, n\ .\ t\ n;-p \leq x \ll \mathit{pSum}\ t\ n) \rightarrow -r \leq -p{*}x \ll 1$
proof
assume *1*: $\mathit{test\text{-}seq}\ t \wedge -r \leq \mathit{Sum}\ t \wedge (\forall\, n\ .\ t\ n;-p \leq x \ll \mathit{pSum}\ t\ n)$
hence *2*: $\forall\, n{>}0\ .\ t\ n;-p;\mathit{Sum}\ t \leq x \ll \mathit{pSum}\ t\ n;\mathit{Sum}\ t$
by (*smt Sum-test leq-def lower-bound-left pSum-below-Sum pSum-test pre-closed sub-mult-closed test-seq-def transitive*)
have *3*: $t\ 0;-p;\mathit{Sum}\ t \leq x \ll 0$ **using** *1*
by (*smt Sum-test Sum-upper leq-def sub-assoc sub-comm test-seq-def pSum.simps*(*1*))
have $x \ll 0 \leq x \ll aL;\mathit{Sum}\ t$ **using** *1*
by (*smt Sum-test aL-test pre-iso sub-mult-closed zero-double-compl zero-least-test*)
hence $t\ 0;-p;\mathit{Sum}\ t \leq x \ll aL;\mathit{Sum}\ t$ **using** *1 3*
by (*smt Sum-test aL-test pre-closed sub-mult-closed test-seq-def transitive zero-double-compl*)
hence $\mathit{Sum}\ t \leq -p{*}x \ll --p;\mathit{Sum}\ t$ **using** *1 2*
by (*smt Sum-test reflexive while-soundness-1*)

thus $-r \leq -p*x\ll 1$ **using** *1*
by (*smt Sum-test one-def pre-below-pre-one pre-closed sub-mult-closed transitive*)
qed

theorem *soundness*: $p(\!|x|\!)\,q \Rightarrow p\langle x\rangle q$
apply (*induct rule*: *derived-hoare-triple.induct*)
apply (*metis atom-prog pre-pre valid-hoare-triple-def pre-closed reflexive pre-expression-test*)
apply (*metis valid-hoare-triple-def pre-expression-test pre-compose seq-prog*)
apply (*metis valid-hoare-triple-def ite-import-mult pre-expression-test cond-prog test-pre*)
apply (*smt valid-hoare-triple-def pre-expression-test conj-pre neg-pre test-pre while-prog while-soundness-1*)
apply (*metis pre-expression-test pre-iso pre-pre transitive valid-hoare-triple-def*)
done

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*: $x \in \mathit{While\text{-}program} \wedge \mathit{ascending\text{-}chain}\ t \wedge \mathit{test\text{-}seq}\ t \rightarrow x\ll \mathit{Sum}\ t = \mathit{Sum}\ (\lambda n\ .\ x\ll t\ n)$

begin

lemma *aL-pre-import-equal*: $(x\ll -q);aL = (x\ll -q;aL);aL$
proof −
have $(x\ll -q);aL \leq (x\ll -q;aL);aL$
by (*smt aL-pre-import aL-test greatest-lower-bound lower-bound-right pre-closed sub-mult-closed*)
thus *?thesis*
by (*smt aL-test antisymmetric lower-bound-left mult-iso-left pre-closed pre-iso sub-mult-closed*)
qed

lemma *aL-pre-below-t-seq2*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow (p*x\ll q);aL \leq \mathit{tseq2}\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ 0$
by (*smt aL-pre-import aL-test id-def lower-bound-left mult-distr-plus-right mult-iso-right plus-comm plus-iso power-zero-id pre-closed pre-expression-test sub-assoc sub-mult-closed test-pre tseq2-def while-pre*)

lemma *t-seq2-ascending*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow \mathit{tseq2}\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ n \leq \mathit{tseq2}\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL))\ (\mathit{Suc}\ n)$
apply (*induct n*)
apply (*smt aL-pre-below-t-seq2 aL-test greatest-lower-bound id-def lower-bound-left mult-iso-right plus-closed plus-iso-right power-succ-unfold-ext power-zero-id pre-closed pre-expression-test pre-iso sub-mult-closed test-pre tfun2-def tseq2-def*)
apply (*smt mult-iso-right plus-iso-right power-succ-unfold-ext pre-closed pre-expression-test pre-iso sub-mult-closed t-seq2-test test-pre tfun2-def tseq2-def*)
done

lemma *t-seq2-ascending-chain*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow \mathit{ascending\text{-}chain}\ (\mathit{tseq2}\ (-p;q)\ p\ x\ (p*x\ll q)\ (-p;q+p;(x\ll(p*x\ll q);aL)))$
by (*metis t-seq2-ascending ascending-chain-def*)

end

class *hoare-calculus-complete* = *hoare-calculus-pre-complete* +

assumes *while-completeness*: $-p;(x\langle\langle -q) \leq -q \rightarrow -p*x\langle\langle -q \leq -q+aL$

begin

lemma *while-completeness-var*: $-p;(x\langle\langle -q)+-r \leq -q \rightarrow -p*x\langle\langle -r \leq -q+aL$
proof
assume *1*: $-p;(x\langle\langle -q)+-r \leq -q$
hence *2*: $-p*x\langle\langle -q \leq -q+aL$
by (*smt least-upper-bound pre-closed sub-mult-closed while-completeness*)
have $-p*x\langle\langle -r \leq -p*x\langle\langle -q$ **using** *1*
by (*smt least-upper-bound pre-closed pre-iso sub-mult-closed*)
thus $-p*x\langle\langle -r \leq -q+aL$ **using** *2*
by (*smt transitive pre-closed aL-test plus-closed*)
qed

lemma *while-completeness-sum*: $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program* $\rightarrow$ $p*x\langle\langle q \leq tsum\ (-p)\ x\ (p*x\langle\langle q)\ (-p+(x\langle\langle (p*x\langle\langle q);aL))$
proof
let *?w* = $p*x\langle\langle q$
let *?r* = $-p;q+p;(x\langle\langle ?w;aL)$
let *?t* = *tseq2* $(-p;q)\ p\ x\ ?w\ ?r$
let *?ts* = *tsum2* $(-p;q)\ p\ x\ ?w\ ?r$
assume *1*: $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program*
hence *2*: *?w* = $--?w$
by (*metis pre-expression-test pre-closed*)
have *3*: *?r* = $--?r$ **using** *1*
by (*smt aL-test sub-mult-closed pre-closed plus-closed pre-expression-test test-pre*)
have *4*: *?ts* = $--?ts$ **using** *1*
by (*metis t-sum2-test pre-expression-test test-pre*)
have *5*: *test-seq ?t* **using** *1*
by (*metis pre-expression-test t-seq2-test-seq test-expression-test*)
have $-p;q \leq ?r$ **using** *1*
by (*smt aL-test pre-closed pre-expression-test sub-mult-closed test-pre upper-bound-left*)
hence *6*: $-p;q \leq ?ts$ **using** *1 2 3 4*
by (*smt Sum-upper id-def transitive power-zero-id pre-expression-test sub-mult-closed test-pre tseq2-def tseq2-test-seq tsum2-def*)
have $\forall\, n\ .\ p;(x\langle\langle ?t\ n) \leq ?ts$ **using** *1 4 5*
by (*smt Sum-upper leq-def power-succ-unfold-ext pre-closed pre-expression-test sub-comm sub-mult-closed t-seq2-below-w test-pre test-seq-def tfun2-def transitive tseq2-def tsum2-def upper-bound-right*)
hence $p;(x\langle\langle ?ts) \leq ?ts$ **using** *1 4 5*
by (*smt mult-left-dist-Sum pre-closed pre-right-dist-Sum t-seq2-ascending-chain test-expression-test test-seq-def tsum2-def*)
hence $p;(x\langle\langle ?ts)+-p;q \leq ?ts$ **using** *1 4 6*
by (*smt least-upper-bound pre-closed pre-expression-test sub-mult-closed test-pre*)
hence *?w* $\leq$ *?ts*$+aL$ **using** *1 2 4*
by (*smt pre-expression-test while-post sub-mult-closed t-sum2-below-t-sum t-sum-test test-pre transitive while-completeness-var*)
hence *?w* $\leq$ *?ts* **using** *1 2 3 4*
by (*smt Sum-upper aL-pre-below-t-seq2 aL-test id-def leq-def leq-plus lower-bound-right mult-distr-plus-left plus-closed plus-comm plus-iso-left power-zero-id pre-expression-test sub-mult-closed test-pre transitive tseq2-def tseq2-test-seq tsum2-def*)
thus *?w* $\leq$ *tsum* $(-p)\ x$ *?w* $(-p+(x\langle\langle ?w;aL))$ **using** *1 2 4*

by (*smt pre-expression-test t-sum2-below-t-sum t-sum-test transitive*)
qed

lemma *while-complete*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \wedge (\forall\, r{\in}\mathit{Pre\text{-}expression}\ .\ x\mathrel{«} r\langle\!|x|\!\rangle r) \rightarrow p{*}x\mathrel{«} q\langle\!|p{*}x|\!\rangle q$
proof
let *?w* = $p{*}x\mathrel{«} q$
let *?t* = *tseq* $(-p)$ *x ?w* $(-p{+}(x\mathrel{«} ?w;aL))$
assume *1*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \wedge (\forall\, r{\in}\mathit{Pre\text{-}expression}\ .\ x\mathrel{«} r\langle\!|x|\!\rangle r)$
hence *2*: *?w* $\in$ *Pre-expression*
by (*metis pre-pre while-prog*)
have *3*: *test-seq ?t* **using** *1*
by (*metis t-seq-test-seq pre-expression-test*)
hence *4*: *?w* $\leq$ *Sum ?t* **using** *1*
by (*metis tsum-def while-completeness-sum*)
have *5*: *?t 0;p;?w*$\langle\!|x|\!\rangle$*aL;?w* **using** *1 2*
by (*smt aL-pre-expression conj-pre cons-pre-trip id-def mult-compl-intro plus-closed power-zero-id pre-closed pre-expression-test sub-comm sub-mult-closed test-pre tseq-def*)
have $\forall\, n{>}0$. *?t n;p;?w*$\langle\!|x|\!\rangle$*pSum ?t n;?w*
proof (*rule*, *rule*)
fix *n*
assume $0{<}(n{::}\mathit{nat})$ **then obtain** *m* **where** *6*: $n = \mathit{Suc}\ m$
by (*auto dest*: *less-imp-Suc-add*)
hence *?t m;?w* $\leq$ *pSum ?t n;?w* **using** *2 3*
by (*metis* (*lifting*, *full-types*) *mult-iso-left pSum.simps*(*2*) *pSum-test pre-expression-test test-seq-def upper-bound-right*)
thus *?t n;p;?w*$\langle\!|x|\!\rangle$*pSum ?t n;?w* **using** *1 2 6*
by (*smt conj-pre cons-trip lower-bound-left mult-compl-intro pSum-pre-expression power-succ-unfold-ext pre-closed pre-expression-test sub-assoc sub-comm t-seq-pre-expression test-pre tfun-def tseq-def*)
qed
hence *?w*$\langle\!|p{*}x|\!\rangle -p;?w$ **using** *1 2 3 4 5*
by (*smt while-trip*)
thus *?w*$\langle\!|p{*}x|\!\rangle q$ **using** *1*
by (*smt cons-post-trip neg-pre pre-expression-test test-pre while-pre-else*)
qed

lemma *pre-completeness*: $x \in \mathit{While\text{-}program} \Rightarrow q \in \mathit{Pre\text{-}expression} \Rightarrow x\mathrel{«} q\langle\!|x|\!\rangle q$
apply (*induct arbitrary*: *q rule*: *While-program.induct*)
apply (*metis atom-trip*)
apply (*metis pre-pre pre-seq seq-trip pre-expression-test*)
apply (*smt cond-prog cond-trip cons-pre-trip ite-pre-else ite-pre-then neg-pre pre-pre pre-expression-test test-pre*)
apply (*metis while-complete*)
done

theorem *completeness*: $p\langle x\rangle q \rightarrow p\langle\!|x|\!\rangle q$
by (*metis valid-hoare-triple-def pre-completeness reflexive pre-expression-test cons-trip*)

end

class *hoare-calculus-sound-complete* = *hoare-calculus-sound* + *hoare-calculus-complete*

begin

theorem *soundness-completeness*: $p(\!|x|\!)\,q \leftrightarrow p\langle x\rangle q$
  by (*smt soundness completeness*)

end

class *hoare-rules* = *whiledo* + *complete-tests* + *hoare-triple* +
  assumes *rule-pre*: $x \mathbin{«} -q\{\!|x|\!\}-q$
  assumes *rule-seq*: $-p\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-r \rightarrow -p\{\!|x;y|\!\}-r$
  assumes *rule-cond*: $-p;-q\{\!|x|\!\}-r \wedge --p;-q\{\!|y|\!\}-r \rightarrow -q\{\!|x \lhd -p \rhd y|\!\}-r$
  assumes *rule-while*: *test-seq* $t \wedge -q \leq Sum\ t \wedge t\ 0;-p;-q\{\!|x|\!\}aL;-q \wedge (\forall\, n{>}0\ .\ t\ n;-p;-q\{\!|x|\!\}pSum\ t\ n;-q) \rightarrow -q\{\!|-p{*}x|\!\}--p;-q$
  assumes *rule-cons*: $-p \leq -q \wedge -q\{\!|x|\!\}-r \wedge -r \leq -s \rightarrow -p\{\!|x|\!\}-s$
  assumes *rule-disj*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-s \rightarrow -p+-q\{\!|x|\!\}-r+-s$

begin

lemma *rule-cons-pre*: $-p \leq -q \wedge -q\{\!|x|\!\}-r \rightarrow -p\{\!|x|\!\}-r$
  by (*metis rule-cons reflexive*)

lemma *rule-cons-pre-mult*: $-q\{\!|x|\!\}-r \rightarrow -p;-q\{\!|x|\!\}-r$
  by (*metis rule-cons-pre lower-bound-left sub-comm sub-mult-closed*)

lemma *rule-cons-pre-plus*: $-p+-q\{\!|x|\!\}-r \rightarrow -p\{\!|x|\!\}-r$
  by (*metis rule-cons-pre upper-bound-left plus-closed*)

lemma *rule-cons-post*: $-q\{\!|x|\!\}-r \wedge -r \leq -s \rightarrow -q\{\!|x|\!\}-s$
  by (*metis rule-cons reflexive*)

lemma *rule-cons-post-mult*: $-q\{\!|x|\!\}-r;-s \rightarrow -q\{\!|x|\!\}-s$
  by (*metis rule-cons-post lower-bound-left sub-comm sub-mult-closed*)

lemma *rule-cons-post-plus*: $-q\{\!|x|\!\}-r \rightarrow -q\{\!|x|\!\}-r+-s$
  by (*metis rule-cons-post upper-bound-left plus-closed*)

lemma *rule-disj-pre*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-r \rightarrow -p+-q\{\!|x|\!\}-r$
  by (*metis rule-disj plus-idempotent*)

end

class *hoare-calculus-valid* = *hoare-calculus-sound-complete* + *hoare-triple* +
  assumes *hoare-triple-valid*: $-p\{\!|x|\!\}-q \leftrightarrow -p \leq x \mathbin{«} -q$

begin

lemma *valid-hoare-triple-same*: $p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \rightarrow p\{\!|x|\!\}q = p\langle x\rangle q$

by (*metis valid-hoare-triple-def hoare-triple-valid pre-expression-test*)

lemma *derived-hoare-triple-same*: $p \in \mathit{Pre\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow p\{\!|x|\!\}q = p(\!|x|\!)q$
by (*metis valid-hoare-triple-same soundness-completeness*)

lemma *valid-rule-disj*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-s \rightarrow -p+-q\{\!|x|\!\}-r+-s$
proof
assume *1*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-s$
have $x\mathrel{«}-r \leq x\mathrel{«}-r+-s \wedge x\mathrel{«}-s \leq x\mathrel{«}-r+-s$
by (*smt plus-closed pre-iso upper-bound-left upper-bound-right*)
thus $-p+-q\{\!|x|\!\}-r+-s$ **using** *1*
by (*smt hoare-triple-valid least-upper-bound plus-closed pre-closed transitive*)
qed

subclass *hoare-rules*
apply *unfold-locales*
apply (*smt hoare-triple-valid pre-closed reflexive*)
apply (*smt hoare-triple-valid pre-compose*)
apply (*smt hoare-triple-valid ite-import-mult sub-mult-closed*)
apply (*smt hoare-triple-valid aL-test pSum-test plus-closed sub-mult-closed test-seq-def while-soundness-1*)
apply (*smt hoare-triple-valid pre-iso transitive pre-closed*)
apply (*smt valid-rule-disj*)
done

lemma *nat-test-rule-while*: $\mathit{nat\text{-}test}\ t\ s \wedge -q \leq s \wedge (\forall\, n\ .\ t\ n;-p;-q\{\!|x|\!\}\mathit{pSum}\ t\ n;-q) \rightarrow -q\{\!|-p*x|\!\}--p;-q$
by (*smt hoare-triple-valid nat-test-def nat-test-pre pSum-test-nat sub-mult-closed*)

lemma *test-seq-rule-while*: $\mathit{test\text{-}seq}\ t \wedge -q \leq \mathit{Sum}\ t \wedge t\ 0;-p;-q\{\!|x|\!\}aL;-q \wedge (\forall\, n{>}0\ .\ t\ n;-p;-q\{\!|x|\!\}\mathit{pSum}\ t\ n;-q) \rightarrow -q\{\!|-p*x|\!\}--p;-q$
by (*smt hoare-triple-valid aL-test pSum-test sub-mult-closed test-seq-def while-soundness-1*)

lemma *rule-zero*: $0\{\!|x|\!\}-p$
by (*metis hoare-triple-valid one-compl pre-closed zero-least-test*)

lemma *rule-skip*: $-p\{\!|1|\!\}-p$
by (*metis pre-closed pre-one-increasing rule-cons-pre rule-pre*)

lemma *rule-example-4*: $\mathit{test\text{-}seq}\ t \wedge \mathit{Sum}\ t = 1 \wedge t\ 0;-p1;-p3 = 0 \wedge -p1\{\!|z1|\!\}-p1;-p2 \wedge (\forall\, n{>}0\ .\ t\ n;-p1;-p2;-p3\{\!|z2|\!\}\mathit{pSum}\ t\ n;-p1;-p2) \rightarrow -p1\{\!|z1;(-p3*z2)|\!\}-p2;--p3$
proof
assume *1*: $\mathit{test\text{-}seq}\ t \wedge \mathit{Sum}\ t = 1 \wedge t\ 0;-p1;-p3 = 0 \wedge -p1\{\!|z1|\!\}-p1;-p2 \wedge (\forall\, n{>}0\ .\ t\ n;-p1;-p2;-p3\{\!|z2|\!\}\mathit{pSum}\ t\ n;-p1;-p2)$
hence *2*: $t\ 0;-p3;(-p1;-p2)\{\!|z2|\!\}aL;(-p1;-p2)$
by (*smt aL-test bs-mult-left-zero rule-zero sub-assoc sub-comm sub-mult-closed test-seq-def*)
have $\forall\, n{>}0\ .\ t\ n;-p3;(-p1;-p2)\{\!|z2|\!\}\mathit{pSum}\ t\ n;(-p1;-p2)$ **using** *1*
by (*smt lower-bound-left pSum-test rule-cons-pre sub-assoc sub-comm sub-mult-closed test-seq-def*)
hence $-p1;-p2\{\!|-p3*z2|\!\}--p3;(-p1;-p2)$ **using** *1 2*
by (*smt one-greatest rule-while sub-mult-closed*)
thus $-p1\{\!|z1;(-p3*z2)|\!\}-p2;--p3$ **using** *1*
by (*smt lower-bound-left rule-cons-post rule-seq sub-assoc sub-comm sub-mult-closed*)

qed

end

class *hoare-calculus-pc* = *hoare-calculus-sound* + *hoare-calculus-pre-complete* +
  assumes *pre-one-one*: $x\langle\langle 1 = 1$

begin

lemma *aL-one*: $aL = 1$
  by (*metis aL-def pre-one-one*)

subclass *hoare-calculus-sound-complete*
  apply *unfold-locales*
  apply (*smt aL-one plus-right-one one-greatest pre-closed*)
  done

lemma *while-soundness-pc*: $-p;-q \leq x\langle\langle -q \rightarrow -q \leq -p*x\langle\langle --p;-q$
proof
  assume *1*: $-p;-q \leq x\langle\langle -q$
  let *?t* = λx . *1*
  have *2*: *test-seq ?t*
    by (*metis test-seq-def one-double-compl*)
  hence *3*: $-q \leq Sum\ ?t$
    by (*metis Sum-test Sum-upper antisymmetric one-double-compl one-greatest*)
  have *4*: *?t 0*$;-p;-q \leq x\langle\langle aL;-q$ **using** *1 2*
    by (*metis aL-one bs-mult-left-one*)
  have $\forall\, n{>}0$. *?t n*$;-p;-q \leq x\langle\langle pSum\ ?t\ n;-q$ **using** *1 2*
    by (*metis bs-mult-left-one gr0-implies-Suc pSum.simps*(*2*) *pSum-test plus-right-one*)
  thus $-q \leq -p*x\langle\langle --p;-q$ **using** *2 3 4*
    by (*smt while-soundness-1*)
qed

end

class *hoare-calculus-pc-valid* = *hoare-calculus-pc* + *hoare-calculus-valid*

begin

lemma *rule-while-pc*: $-p;-q\{\!|x|\!\}-q \rightarrow -q\{\!|-p*x|\!\}--p;-q$
  by (*metis hoare-triple-valid sub-mult-closed while-soundness-pc*)

lemma *rule-alternation*: $-p\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-p \rightarrow -p\{\!|-r*x;y|\!\}--r;-p$
  by (*metis rule-seq rule-cons-pre-mult rule-while-pc*)

lemma *rule-alternation-context*: $-p\{\!|v|\!\}-p \wedge -p\{\!|w|\!\}-q \wedge -q\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-p \wedge -p\{\!|z|\!\}-p \rightarrow -p\{\!|-r*v;w;x;y;z|\!\}--r;-p$
  by (*metis rule-seq rule-cons-pre-mult rule-while-pc*)

lemma *rule-example-3*: $-p;-q\{\!|x|\!\}--p;-q \wedge --p;-r\{\!|x|\!\}-p;-r \wedge -p;-r\{\!|y|\!\}-p;-q \wedge --p;-q\{\!|z|\!\}--p;-r \rightarrow -p;-q+--p;-r\{\!|-s*x;(y\lhd -p\rhd z)|\!\}--s;(-p;-q+--p;-r)$
proof (*rule*, (*erule conjE*)+)
  assume $-p;-q\{\!|x|\!\}--p;-q$ **and** $--p;-r\{\!|x|\!\}-p;-r$
  hence *t1*: $-p;-q+--p;-r\{\!|x|\!\}--p;-q+-p;-r$
    by (*smt rule-disj sub-mult-closed*)
  assume $-p;-r\{\!|y|\!\}-p;-q$
  hence $-p;-r\{\!|y|\!\}-p;-q+--p;-r$
    by (*smt rule-cons-post-plus sub-mult-closed*)
  hence *t2*: $-p;(--p;-q+-p;-r)\{\!|y|\!\}-p;-q+--p;-r$
    by (*metis mult-compl mult-distr-plus-left mult-idempotent plus-left-zero sub-assoc sub-mult-closed*)
  assume $--p;-q\{\!|z|\!\}--p;-r$
  hence $--p;-q\{\!|z|\!\}-p;-q+--p;-r$
    by (*smt plus-comm rule-cons-post-plus sub-mult-closed*)
  hence $--p;(--p;-q+-p;-r)\{\!|z|\!\}-p;-q+--p;-r$
    by (*metis mult-compl mult-distr-plus-left mult-idempotent plus-right-zero sub-assoc sub-mult-closed*)
  hence $--p;-q+-p;-r\{\!|y\lhd -p\rhd z|\!\}-p;-q+--p;-r$ **using** *t2*
    by (*smt plus-closed rule-cond sub-mult-closed*)
  hence $-s;(-p;-q+--p;-r)\{\!|x;(y\lhd -p\rhd z)|\!\}-p;-q+--p;-r$ **using** *t1*
    by (*smt plus-closed rule-cons-pre-mult rule-seq sub-mult-closed*)
  thus $-p;-q+--p;-r\{\!|-s*x;(y\lhd -p\rhd z)|\!\}--s;(-p;-q+--p;-r)$
    by (*smt plus-closed rule-while-pc sub-mult-closed*)
qed

end

class *hoare-calculus-tc* = *hoare-calculus* + *precondition-test-test* + *precondition-distr-mult* +
  assumes *while-bnd*: $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program* $\rightarrow$ $p*x\llbracket q \leq$ *Sum* $(\lambda n\ .\ (p;x)\hat{}n\llbracket 0)$

begin

lemma $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program* $\rightarrow$ $p*x\llbracket q \leq$ *tsum* $(-p)$ x $(p*x\llbracket q)$ $(-p+(x\llbracket(p*x\llbracket q);aL))$
proof
  let *?w* = $p*x\llbracket q$
  let *?s* = $-p+(x\llbracket ?w;aL)$
  let *?t* = *tseq* $(-p)$ x *?w* *?s*
  let *?b* = $\lambda n\ .\ (p;x)\hat{}n\llbracket 0$
  assume *1*: $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program*
  hence *2*: *test-seq ?t*
    by (*metis t-seq-test-seq pre-expression-test*)
  have *3*: *test-seq ?b*
    by (*smt zero-double-compl pre-closed test-seq-def*)
  have *4*: *?w* = $--$*?w* **using** *1*
    by (*metis pre-expression-test pre-closed*)
  have *?w* $\leq$ *Sum ?b* **using** *1*
    by (*metis while-bnd*)
  hence *5*: *?w* = *Sum ?b*;*?w* **using** *3 4*

by (*smt Sum-test leq-def sub-comm*)
have $\forall\, n\ .\ ?b\ n;?w \leq ?t\ n$
proof
fix *n*
show $?b\ n;?w \leq ?t\ n$
proof (*induct n*)
show $?b\ 0;?w \leq ?t\ 0$ **using** *2 4*
by (*smt bs-mult-left-zero power-0 pre-one test-seq-def zero-double-compl zero-least-test*)
next
fix *n*
assume *6*: $?b\ n;?w \leq ?t\ n$
have $-p \leq ?t\ (Suc\ n)$
apply (*simp only*: *power-succ-unfold-ext tseq-def*) **using** *1*
by (*smt pre-expression-test t-seq-test pre-closed sub-mult-closed tfun-def tseq-def upper-bound-left*)
hence *7*: $-p;?b\ (Suc\ n);?w \leq ?t\ (Suc\ n)$ **using** *2 3 4*
by (*smt lower-bound-left sub-mult-closed test-seq-def transitive*)
have *8*: $p;?b\ (Suc\ n);?w \leq x \mathbin{«} ?w;(?b\ n;?w)$ **using** *1*
by (*smt lower-bound-right mult-idempotent power-Suc pre-closed pre-distr-mult pre-expression-test pre-import-composition sub-assoc sub-comm sub-mult-closed test-expression-test while-pre-then zero-double-compl*)
have *9*: $\ldots \leq x \mathbin{«} ?w;?t\ n$ **using** *2 3 4 6*
by (*smt mult-iso-right pre-iso sub-mult-closed test-seq-def*)
have $\ldots \leq ?t\ (Suc\ n)$ **using** *2 4*
by (*smt power-succ-unfold-ext pre-closed sub-mult-closed test-seq-def tfun-def tseq-def upper-bound-right*)
hence $p;?b\ (Suc\ n);?w \leq ?t\ (Suc\ n)$ **using** *1 2 3 4 8 9*
by (*smt pre-closed sub-mult-closed test-expression-test test-seq-def transitive*)
thus $?b\ (Suc\ n);?w \leq ?t\ (Suc\ n)$ **using** *1 2 3 4 7*
by (*smt leq-cases sub-assoc sub-mult-closed test-expression-test test-seq-def*)
qed
qed
hence $Sum\ ?b;?w \leq tsum\ (-p)\ x\ ?w\ ?s$ **using** *1 3 4*
by (*smt Sum-upper mult-right-dist-Sum pre-expression-test sub-mult-closed t-seq-test t-sum-test test-seq-def transitive tsum-def*)
thus $?w \leq tsum\ (-p)\ x\ ?w\ ?s$ **using** *5*
by *metis*
qed

end

class *box-precondition* = *relative-box-semiring* + *pre* +
assumes *pre-def*: $x \mathbin{«} p = |x]p$

begin

subclass *precondition*
apply *unfold-locales*
apply (*metis box-def double-negation pre-def*)

apply (*metis box-left-mult pre-def*)
apply (*metis a-dist-add box-deMorgan-2 box-right-submult-a-a greatest-lower-bound pre-def*)
apply (*metis box-1-a pre-def reflexive*)
done

subclass *precondition-test-test*
apply *unfold-locales*
apply (*metis a-box-a-a pre-def*)
done

subclass *precondition-promote*
apply *unfold-locales*
apply (*metis box-def box-x-a pre-def pre-test-test*)
done

subclass *precondition-test-box*
apply *unfold-locales*
apply (*metis box-a-a d-def pre-def*)
done

lemma *pre-Z*: $-p \leq x\mathbin{«}-q \leftrightarrow -p \mathbin{;} x \mathbin{;} --q \leq Z$
by (*metis box-demodalisation-2 pre-def*)

lemma *pre-left-dist-add*: $x+y\mathbin{«}-q = (x\mathbin{«}-q) \mathbin{;} (y\mathbin{«}-q)$
by (*metis box-left-dist-add pre-def*)

lemma *pre-left-antitone*: $x \leq y \rightarrow y\mathbin{«}-q \leq x\mathbin{«}-q$
by (*metis box-left-antitone pre-def*)

lemma *pre-promote-neg*: $(x\mathbin{«}-q) \mathbin{;} x \mathbin{;} --q \leq Z$
by (*metis order-refl pre-Z pre-closed*)

lemma *pre-pc-Z*: $x\mathbin{«}1 = 1 \leftrightarrow x \mathbin{;} 0 \leq Z$
by (*metis a-strict box-x-1 pre-def*)

end

class *diamond-precondition* = *relative-box-semiring* + *pre* +
assumes *pre-def*: $x\mathbin{«}p = |x\mathbin{>}p$

begin

subclass *precondition*
apply *unfold-locales*
apply (*metis d-def diamond-d-closed pre-def*)

apply (*metis diamond-left-mult pre-def*)
apply (*smt diamond-right-isotone lower-bound-right pre-def*)
apply (*metis diamond-1-a pre-def reflexive*)
done

subclass *precondition-test-test*
apply *unfold-locales*
apply (*metis diamond-a-a-same diamond-a-export diamond-associative diamond-right-mult pre-def*)
done

subclass *precondition-promote*
apply *unfold-locales*
apply (*metis box-deMorgan-1 diamond-a-a pre-def pre-test-test*)
done

subclass *precondition-test-diamond*
apply *unfold-locales*
apply (*metis diamond-a-a pre-def*)
done

lemma *pre-left-dist-add*: $x+y\mathbin{«}-q = (x\mathbin{«}-q) + (y\mathbin{«}-q)$
by (*metis d-dist-add diamond-def mult-right-dist-add pre-def*)

lemma *pre-left-isotone*: $x \leq y \rightarrow x\mathbin{«}-q \leq y\mathbin{«}-q$
by (*metis diamond-left-isotone pre-def*)

end

class *box-while* = *box-precondition* + *il-conway-semiring-T* + *ite* + *while* +
assumes *ite-def*: $x \lhd p \rhd y = p \ ; \ x + -p \ ; \ y$
assumes *while-def*: $p \ast x = (p \ ; \ x)^\circ \ ; \ -p$

begin

subclass *relative-antidomain-semiring-T* **..**

lemma *Z-circ-left-zero*: $Z \ ; \ x^\circ = Z$
by (*metis Z-top circ-left-top mult-associative*)

subclass *ifthenelse*
apply *unfold-locales*
apply (*smt a-d-closed box-a-export box-left-dist-add box-x-a case-duality d-def ite-def pre-def*)
done

subclass *whiledo*
apply *unfold-locales*
apply (*smt circ-loop-fixpoint ite-def ite-pre mult-associative mult-right-one pre-one pre-seq while-def*)

apply (*metis pre-mult-test-promote while-def*)
done

lemma *pre-while-1*: $-p;(-p*x)\mathrel{«}1 = -p*x\mathrel{«}1$
proof −
have $--p;(-p;(-p*x)\mathrel{«}1) = --p;(-p*x\mathrel{«}1)$
by (*metis a-mult-left-upper-bound box-def bs-mult-right-one leq-def mult-associative one-def pre-def while-pre-else*)
thus *?thesis*
by (*smt eq-cases one-def pre-closed pre-import*)
qed

lemma *aL-one-circ*: $aL = a(1^\circ;0)$
by (*metis a-one box-0-y box-left-mult box-x-1 mult-left-one pre-def while-def aL-def*)

end

class *diamond-while* = *diamond-precondition* + *il-conway-semiring-T* + *ite* + *while* +
assumes *ite-def*: $x\mathrel{◁}p\mathrel{▷}y = p\ ;\ x + -p\ ;\ y$
assumes *while-def*: $p*x = (p\ ;\ x)^\circ\ ;\ -p$

begin

subclass *relative-antidomain-semiring-T* **..**

lemma *Z-circ-left-zero*: $Z\ ;\ x^\circ = Z$
by (*metis Z-top circ-left-top mult-associative*)

subclass *ifthenelse*
apply *unfold-locales*
apply (*metis ite-def pre-def diamond-left-dist-add diamond-a-export*)
done

subclass *whiledo*
apply *unfold-locales*
apply (*smt circ-loop-fixpoint ite-def ite-pre mult-associative mult-right-one pre-one pre-seq while-def*)
apply (*metis pre-mult-test-promote while-def*)
done

lemma *aL-one-circ*: $aL = d(1^\circ;0)$
by (*metis aL-def a-one diamond-x-1 mult-left-one pre-def while-def*)

end

class *box-while-program* = *box-while* + *atoms*

begin

subclass *while-program* **..**

end

class *diamond-while-program* = *diamond-while* + *atoms*

begin

subclass *while-program* **..**

end

class *box-hoare-calculus* = *box-while-program* + *complete-antidomain-semiring*

begin

subclass *hoare-calculus* **..**

end

class *diamond-hoare-calculus* = *diamond-while-program* + *complete-antidomain-semiring*

begin

subclass *hoare-calculus* **..**

end

class *box-hoare-sound* = *box-hoare-calculus* + *relative-domain-semiring-split* + *lk-conway-semiring* +
  assumes *aL-circ*: $aL\ ;\ x^\circ \leq x^*$

begin

lemma *aL-circ-ext*: $|x^*]y \leq |aL\ ;\ x^\circ]y$
  by (*metis aL-circ box-left-antitone*)

lemma *box-star-induct*: $-p \leq |x](-p) \rightarrow -p \leq |x^*](-p)$
proof
  assume $-p \leq |x](-p)$
  hence *1*: $x;--p;T \leq Z + --p;T$
    by (*metis Z-top add-commutative box-demodalisation-2 mult-associative mult-left-isotone shunting-Z*)
  have $x;(Z + --p;T) \leq x;--p;T + Z$
    by (*smt add-commutative mult-associative split-Z*)
  also have $... \leq Z + --p;T$ **using** *1*
    by (*smt add-commutative add-least-upper-bound add-right-upper-bound*)
  finally have $x;(Z + --p;T) + --p \leq Z + --p;T$
    by (*smt add-commutative add-least-upper-bound mult-left-sub-dist-add order-trans split-Z top-right-mult-increasing*)

thus $-p \leq |x^*](-p)$
by (*metis add-commutative box-demodalisation-2 mult-associative shunting-Z star-left-induct*)
qed

lemma *box-circ-induct*: $-p \leq |x](-p) \rightarrow -p;aL \leq |x^\circ](-p)$
by (*smt aL-circ-ext aL-test box-left-mult box-star-induct order-trans plus-comm pre-closed pre-def pre-test shunting-right*)

lemma *a-while-soundness*: $-p;-q \leq |x](-q) \rightarrow aL;-q \leq |(-p;x)^\circ;--p](-q)$
proof −
have $|(-p;x)^\circ](-q) \leq |(-p;x)^\circ;--p](-q)$
by (*smt add-right-upper-bound box-def box-right-dist-add box-right-isotone*)
thus *?thesis*
by (*smt box-import-shunting box-circ-induct order-trans sub-comm aL-test*)
qed

subclass *hoare-calculus-sound*
apply *unfold-locales*
apply (*metis a-while-soundness while-def pre-def*)
done

end

class *diamond-hoare-sound* = *diamond-hoare-calculus* + *lk-conway-semiring* +
assumes *aL-circ*: $aL\ ;\ x^\circ \leq x^*$

begin

lemma *aL-circ-equal*: $aL\ ;\ x^\circ = aL\ ;\ x^*$
by (*smt aL-circ aL-one-circ antisym d-restrict-iff-1 mult-right-isotone star-below-circ*)

lemma *aL-zero*: $aL = 0$
by (*smt aL-circ-equal aL-one-circ d-export d-idempotent diamond-d-0 diamond-def mult-associative mult-right-one star-one*)

subclass *hoare-calculus-sound*
apply *unfold-locales*
apply (*metis aL-zero bs-mult-left-zero zero-least*)
done

end

class *box-hoare-complete* = *box-hoare-calculus* + *lk-conway-semiring* +
assumes *box-circ-induct-2*: $-p;|x](-q) \leq -q \rightarrow |x^\circ](-p) \leq -q+aL$
assumes *aL-zero-or-one*: $aL = 0 \vee aL = 1$
assumes *while-mult-left-dist-Prod*: $x \in \mathit{While\text{-}program} \wedge \mathit{descending\text{-}chain}\ t \wedge \mathit{test\text{-}seq}\ t \rightarrow x;\mathit{Prod}\ t = \mathit{Prod}\ (\lambda n\ .\ x;t\ n)$

begin

subclass *hoare-calculus-complete*
apply *unfold-locales*
prefer *3*
apply (*smt box-circ-induct-2 double-negation least-upper-bound lower-bound-left mult-distr-plus-right pre-closed pre-def pre-import pre-seq pre-test sub-mult-closed while-def*)
apply (*metis aL-zero-or-one bs-mult-right-zero mult-right-one order-refl pre-closed zero-least*)
unfolding *pre-def box-def*
apply (*metis a-ascending-chain a-dist-Prod a-dist-Sum descending-chain-left-mult while-mult-left-dist-Prod test-seq-def*)
done

end

class *diamond-hoare-complete* = *diamond-hoare-calculus* + *relative-domain-semiring-split* + *lk-conway-semiring* +
assumes *dL-circ*: $-aL;x^\circ \leq x^*$
assumes *aL-zero-or-one*: $aL = 0 \vee aL = 1$
assumes *while-mult-left-dist-Sum*: $x \in \mathit{While\text{-}program} \wedge \mathit{ascending\text{-}chain}\ t \wedge \mathit{test\text{-}seq}\ t \rightarrow x;\mathit{Sum}\ t = \mathit{Sum}\ (\lambda n\ .\ x;t\ n)$

begin

lemma *diamond-star-induct-var*: $|x\rangle(d\ p) \leq d\ p \rightarrow |x^*\rangle(d\ p) \leq d\ p$
proof
assume $|x\rangle(d\ p) \leq d\ p$
hence $x\ ;\ (d\ p\ ;\ x^* + Z) \leq d\ p\ ;\ x\ ;\ x^* + Z\ ;\ x^* + Z$
by (*metis add-left-isotone d-mult-d diamond-def diamond-demodalisation-3 mult-associative mult-left-isotone mult-right-dist-add order-trans split-Z*)
also have $\ldots \leq d\ p\ ;\ x^* + Z$
by (*smt Z-mult-decreasing add-associative add-left-isotone less-eq-def mult-associative mult-right-isotone star.left-plus-below-circ*)
finally show $|x^*\rangle(d\ p) \leq d\ p$
by (*smt add-commutative add-least-upper-bound add-right-upper-bound d-mult-d diamond-def diamond-demodalisation-3 order-trans star.circ-back-loop-prefixpoint star-left-induct*)
qed

lemma *diamond-star-induct*: $d\ q + |x\rangle(d\ p) \leq d\ p \rightarrow |x^*\rangle(d\ q) \leq d\ p$
by (*metis add-least-upper-bound diamond-star-induct-var diamond-right-isotone order-trans*)

lemma *while-completeness-1*: $-p;(x\langle -q) \leq -q \rightarrow -p{*}x\langle -q \leq -q{+}aL$
proof
assume $-p;(x\langle -q) \leq -q$
hence $--p;-q + |-p;x\rangle(-q) \leq -q$
by (*metis add-least-upper-bound diamond-a-export lower-bound-right pre-def*)
hence $|(-p;x)^*\rangle(--p;-q) \leq -q$
by (*smt diamond-star-induct d-def sub-mult-closed double-negation*)
hence $|-aL;(-p;x)^\circ\rangle(--p;-q) \leq -q$
by (*smt dL-circ diamond-left-isotone order-trans*)
thus $-p{*}x\langle -q \leq -q{+}aL$
by (*smt aL-test diamond-a-export diamond-def mult-associative plus-comm pre-closed pre-def shunting while-def*)
qed

subclass *hoare-calculus-complete*
apply *unfold-locales*

prefer *3*
apply (*rule while-completeness-1*)
apply (*metis aL-zero-or-one bs-mult-right-zero mult-right-one order-refl pre-closed zero-least*)
unfolding *pre-def diamond-def*
apply (*metis while-mult-left-dist-Sum d-dist-Sum ascending-chain-left-mult*)
done

end

class *box-hoare-valid* = *box-hoare-sound* + *box-hoare-complete* + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \leftrightarrow p \leq |x]q$

begin

subclass *hoare-calculus-valid*
apply *unfold-locales*
apply (*metis hoare-triple-def pre-def*)
done

lemma *rule-skip-valid*: $-p\{\!|1|\!\}-p$
by (*metis box-1-a hoare-triple-def reflexive*)

end

class *diamond-hoare-valid* = *diamond-hoare-sound* + *diamond-hoare-complete* + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \leftrightarrow p \leq |x>q$

begin

lemma *circ-star-equal*: $x^\circ = x^*$
by (*metis aL-zero antisym dL-circ mult-left-one one-def star-below-circ*)

subclass *hoare-calculus-valid*
apply *unfold-locales*
apply (*metis hoare-triple-def pre-def*)
done

end

class *diamond-hoare-sound-2* = *diamond-hoare-calculus* + *lk-conway-semiring* +
assumes *diamond-circ-induct-2*: $--p;-q \leq |x>(-q) \rightarrow\ \ aL;-q \leq |x^\circ>(-p)$

begin

subclass *hoare-calculus-sound*
apply *unfold-locales*
apply (*smt a-export diamond-associative diamond-circ-induct-2 double-negation mult-compl-intro pre-def pre-import-equiv-mult sub-comm sub-mult-closed while-def*)

done

end

class *diamond-hoare-valid-2* = *diamond-hoare-sound-2* + *diamond-hoare-complete* + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \leftrightarrow p \leq |x>q$

begin

subclass *hoare-calculus-valid*
apply *unfold-locales*
apply (*metis hoare-triple-def pre-def*)
done

end

context *mbt-algebra*

begin

lemma *directed-left-mult*: *directed Y* $\rightarrow$ *directed* (*op* ; *x* ‘ *Y*)
unfolding *directed-def*
apply *simp*
apply (*metis le-comp*)
done

lemma *neg-assertion*: *neg-assert x* $\in$ *assertion*
unfolding *assertion-def*
apply *rule*
apply (*smt dual-comp dual-dual dual-neg dual-one dual-sup dual-top inf-commute inf-le2 inf-sup-distrib1 mult.assoc mult.left-neutral neg-assert-def sup-bot-left sup-comp top-comp*)
done

lemma *assertion-neg-assert*: $x \in$ *assertion* $\leftrightarrow x =$ *neg-assert* (*neg-assert x*)
by (*metis neg-assertion uminus-uminus*)

definition
assumption = $\{x \,.\, 1 \leq x \wedge (x * \bot) \sqcup (x \hat{} o) = x\}$

definition
neg-assume ($x::'a$) = $(x \hat{} o * top) \sqcup 1$

lemma *neg-assume-assert*: *neg-assume x* = (*neg-assert* (*x* ^ *o*)) ^ *o*
by (*metis dual-bot dual-comp dual-dual dual-inf dual-one neg-assert-def neg-assume-def*)

lemma *assert-iff-assume*: $x \in assertion \leftrightarrow x \hat{} o \in assumption$
by (*smt assertion-def assumption-def dual-bot dual-comp dual-dual dual-inf dual-le dual-one mem-Collect-eq*)

lemma *assertion-iff-assumption-subseteq*: $X \subseteq assertion \leftrightarrow dual \mathbin{`} X \subseteq assumption$
unfolding *subset-eq*
apply *simp*
by (*metis assert-iff-assume*)

lemma *assumption-iff-assertion-subseteq*: $X \subseteq assumption \leftrightarrow dual \mathbin{`} X \subseteq assertion$
unfolding *subset-eq*
apply *simp*
by (*metis dual-dual assert-iff-assume*)

lemma *assumption-prop*: $x \in assumption \Rightarrow (x * bot) \sqcup 1 = x$
by (*smt assert-iff-assume assertion-prop dual-comp dual-dual dual-neg-top dual-one dual-sup dual-top*)

lemma *neg-assumption*: $neg\text{-}assume\ x \in assumption$
unfolding *assumption-def*
apply *rule*
by (*smt dual-comp dual-dual dual-neg-top dual-one dual-sup dual-top inf-commute inf-sup-distrib1 le-iff-inf mult.assoc mult.left-neutral neg-assume-def sup-bot-right sup-comp sup-inf-absorb sup-inf-distrib1 sup-left-commute top-comp*)

lemma *assumption-neg-assume*: $x \in assumption \leftrightarrow x = neg\text{-}assume\ (neg\text{-}assume\ x)$
by (*smt assert-iff-assume assertion-neg-assert dual-dual neg-assume-assert*)

lemma *assumption-sup-comp-eq*: $x \in assumption \Rightarrow y \in assumption \Rightarrow x \sqcup y = x * y$
by (*smt assert-iff-assume assertion-inf-comp-eq dual-comp dual-dual dual-sup*)

lemma *sup-uminus-assume*[*simp*]: $x \in assumption \Rightarrow x \sqcap neg\text{-}assume\ x = 1$
by (*smt assert-iff-assume dual-dual dual-one dual-sup neg-assume-assert sup-uminus*)

lemma *inf-uminus-assume*[*simp*]: $x \in assumption \Rightarrow x \sqcup neg\text{-}assume\ x = top$
by (*smt assert-iff-assume dual-dual dual-sup dual-top inf-uminus neg-assume-assert sup-bot-right*)

lemma *uminus-assumption*[*simp*]: $x \in assumption \Rightarrow neg\text{-}assume\ x \in assumption$
by (*smt assert-iff-assume dual-dual neg-assume-assert uminus-assertion*)

lemma *uminus-uminus-assume*[*simp*]: $x \in assumption \Rightarrow neg\text{-}assume\ (neg\text{-}assume\ x) = x$
by (*smt assert-iff-assume dual-dual neg-assume-assert uminus-uminus*)

lemma *sup-assumption*[*simp*]: $x \in assumption \Rightarrow y \in assumption \Rightarrow x \sqcup y \in assumption$
by (*smt assert-iff-assume dual-dual dual-sup inf-assertion*)

lemma *comp-assumption*[*simp*]: $x \in assumption \Rightarrow y \in assumption \Rightarrow x * y \in assumption$
by (*smt assert-iff-assume comp-assertion dual-comp dual-dual*)

lemma *inf-assumption*[*simp*]: $x \in \mathit{assumption} \Rightarrow y \in \mathit{assumption} \Rightarrow x \sqcap y \in \mathit{assumption}$
by (*smt assert-iff-assume dual-dual dual-inf sup-assertion*)

lemma [*simp*]: $x \in \mathit{assumption} \Rightarrow x * x = x$
by (*simp add*: *assumption-sup-comp-eq* [*THEN sym*])

lemma [*simp*]: $x \in \mathit{assumption} \Rightarrow (x \hat{} o) * (x \hat{} o) = x \hat{} o$
apply (*rule dual-eq*)
by (*simp add*: *dual-comp assumption-sup-comp-eq* [*THEN sym*])

lemma [*simp*]: $\mathit{top} \in \mathit{assumption}$
by (*unfold assumption-def*, *simp*)

lemma [*simp*]: $1 \in \mathit{assumption}$
by (*unfold assumption-def*, *simp*)

lemma *assert-top*: $\mathit{neg\text{-}assert}\ (\mathit{neg\text{-}assert}\ p) \hat{}\ o\ ;\ \mathit{bot} = \mathit{neg\text{-}assert}\ p\ ;\ \mathit{top}$
by (*smt bot-comp dual-comp dual-dual dual-top inf-comp inf-top-right mult.assoc mult.left-neutral neg-assert-def*)

lemma *assume-bot*: $\mathit{neg\text{-}assume}\ (\mathit{neg\text{-}assume}\ p) \hat{}\ o\ ;\ \mathit{top} = \mathit{neg\text{-}assume}\ p\ ;\ \mathit{bot}$
by (*smt dual-bot dual-comp dual-one dual-sup dual-top mult.assoc mult.left-neutral neg-assert-def neg-assume-assert neg-assume-def sup-bot-right sup-comp top-comp*)

definition
$\mathit{wpb}\ x = (x * \mathit{bot}) \sqcup 1$

lemma *wpt-iff-wpb*: $\mathit{wpb}\ x = \mathit{wpt}\ (x \hat{} o) \hat{} o$
by (*smt dual-comp dual-dual dual-one dual-sup dual-top wpb-def wpt-def*)

lemma *wpb-is-assumption*[*simp*]: $\mathit{wpb}\ x \in \mathit{assumption}$
by (*smt assert-iff-assume wpt-iff-wpb wpt-is-assertion*)

lemma *wpb-comp*: $(\mathit{wpb}\ x) * x = x$
by (*smt dual-comp dual-dual dual-neg-top dual-sup wpt-comp wpt-iff-wpb*)

lemma *wpb-comp-2*: $\mathit{wpb}\ (x * y) = \mathit{wpb}\ (x * (\mathit{wpb}\ y))$
by (*smt dual-comp dual-dual wpt-comp-2 wpt-iff-wpb*)

lemma *wpb-assumption*[*simp*]: $x \in \mathit{assumption} \Rightarrow \mathit{wpb}\ x = x$
by (*smt assert-iff-assume dual-dual wpt-assertion wpt-iff-wpb*)

lemma *wpb-choice*: $\mathit{wpb}\ (x \sqcup y) = \mathit{wpb}\ x \sqcup \mathit{wpb}\ y$
by (*smt dual-inf dual-sup wpt-choice wpt-iff-wpb*)

lemma *wpb-dual-assumption*: $x \in \mathit{assumption} \Rightarrow \mathit{wpb}\ (x \hat{} o) = 1$
by (*smt assert-iff-assume dual-dual dual-one wpt-dual-assertion wpt-iff-wpb*)

lemma *wpb-mono*: $x \leq y \Rightarrow \mathit{wpb}\ x \leq \mathit{wpb}\ y$

by (*metis le-iff-sup wpb-choice*)

lemma *assumption-disjunctive*: $x \in assumption \Rightarrow x \in disjunctive$
by (*smt assert-iff-assume assertion-conjunctive dual-comp dual-conjunctive dual-dual*)

lemma *assumption-conjunctive*: $x \in assumption \Rightarrow x \in conjunctive$
by (*smt assert-iff-assume assertion-disjunctive dual-comp dual-disjunctive dual-dual*)

lemma *wpb-le-assumption*: $x \in assumption \Rightarrow x * y = y \Rightarrow x \leq wpb\ y$
by (*metis comp-assumption le-comp mult.right-neutral sup.commute sup-ge1 wpb-assumption wpb-comp-2 wpb-def wpb-is-assumption*)

definition *dual-omega* :: $'a \Rightarrow 'a$ ((- ^ $\mho$) [*81*] *80*)
where $(x\ \hat{}\ \mho) = (((x\ \hat{}\ o)\ \hat{}\ \omega)\ \hat{}\ o)$

lemma *dual-omega-fix*: $x\hat{}\mho = (x * (x\hat{}\mho)) \sqcup 1$
by (*smt dual-comp dual-dual dual-omega-def dual-one dual-sup omega-fix*)

lemma *dual-omega-comp-fix*: $x\hat{}\mho * y = (x * (x\hat{}\mho) * y) \sqcup y$
apply (*subst dual-omega-fix*)
by (*simp add*: *sup-comp*)

lemma *dual-omega-greatest*: $z \leq (x * z) \sqcup y \Rightarrow z \leq (x\hat{}\mho) * y$
by (*smt dual-comp dual-dual dual-le dual-neg-top dual-omega-def dual-sup omega-least*)

end

context *complete-mbt-algebra*

begin

lemma *Inf-assumption*[*simp*]: $X \subseteq assumption \Rightarrow Inf\ X \in assumption$
by (*metis SUP-def Sup-assertion assert-iff-assume assumption-iff-assertion-subseteq dual-Inf dual-dual*)

definition *continuous* $x \leftrightarrow (\forall\ Y\ .\ directed\ Y \rightarrow x\ ;\ (SUP\ y{:}Y\ .\ y) = (SUP\ y{:}Y\ .\ x\ ;\ y))$

definition *Continuous* $= \{\ x\ .\ continuous\ x\ \}$

lemma *continuous-Continuous*: $continuous\ x \leftrightarrow x \in Continuous$
by (*simp add*: *Continuous-def*)

lemma *one-continuous*: $1 \in Continuous$
by (*simp add*: *Continuous-def continuous-def SUP-def image-def*)

lemma *continuous-dist-ascending-chain*: $x \in Continuous \wedge ascending\text{-}chain\ f \rightarrow x\ ;\ (SUP\ n{::}nat\ .\ f\ n) = (SUP\ n{::}nat\ .\ x\ ;\ f\ n)$
proof
assume *1*: $x \in Continuous \wedge ascending\text{-}chain\ f$
hence *directed* (*range f*)

```
    by (metis ascending-chain-directed)
  hence x ; (SUP n::nat . f n) = (SUP y:range f . x ; y) using 1
    by (simp add: Continuous-def continuous-def SUP-def)
  thus x ; (SUP n::nat . f n) = (SUP n::nat . x ; f n)
    by simp
qed

lemma assertion-continuous: x ∈ assertion → x ∈ Continuous
proof
  assume x ∈ assertion
  hence 1: x = (x ; top) ⊓ 1
    by (metis assertion-prop)
  have ∀ Y . directed Y → x ; (SUP y:Y . y) = (SUP y:Y . x ; y)
  proof (rule,rule)
    fix Y
    assume directed Y
    have x ; (SUP y:Y . y) = (x ; top) ⊓ (SUP y:Y . y) using 1
      by (smt inf-comp mult.assoc mult.left-neutral top-comp)
    also have ... = (SUP y:Y . (x ; top) ⊓ y)
      by (smt inf-SUP SUP-cong)
    finally show x ; (SUP y:Y . y) = (SUP y:Y . x ; y) using 1
      by (smt inf-comp mult.left-neutral mult.assoc top-comp SUP-cong)
  qed
  thus x ∈ Continuous
    by (simp add: continuous-def Continuous-def)
qed

lemma assumption-continuous: x ∈ assumption → x ∈ Continuous
proof
  assume x ∈ assumption
  hence 1: x = (x ; bot) ⊔ 1
    by (metis assumption-prop)
  have ∀ Y . directed Y → x ; (SUP y:Y . y) = (SUP y:Y . x ; y)
  proof (rule,rule)
    fix Y
    assume 2: directed Y
    have x ; (SUP y:Y . y) = (x ; bot) ⊔ (SUP y:Y . y) using 1
      by (smt sup-comp mult.assoc mult.left-neutral bot-comp)
    also have ... = (SUP y:Y . (x ; bot) ⊔ y) using 2
      by (smt sup-SUP SUP-cong directed-def)
    finally show x ; (SUP y:Y . y) = (SUP y:Y . x ; y) using 1
      by (smt sup-comp mult.left-neutral mult.assoc bot-comp SUP-cong)
  qed
  thus x ∈ Continuous
    by (simp add: continuous-def Continuous-def)
qed
```

lemma *mult-continuous*: $x \in \mathit{Continuous} \wedge y \in \mathit{Continuous} \to x \mathbin{;} y \in \mathit{Continuous}$
proof
  assume *1*: $x \in \mathit{Continuous} \wedge y \in \mathit{Continuous}$
  have $\forall\, Y.\ \mathit{directed}\ Y \to x \mathbin{;} y \mathbin{;} (\mathit{SUP}\ y{:}Y\ .\ y) = (\mathit{SUP}\ z{:}Y\ .\ x \mathbin{;} y \mathbin{;} z)$
  proof (*rule*,*rule*)
    fix *Y*
    assume *directed Y*
    hence $x \mathbin{;} y \mathbin{;} (\mathit{SUP}\ w{:}Y\ .\ w) = (\mathit{SUP}\ z{:}Y\ .\ x \mathbin{;} (y \mathbin{;} z))$ **using** *1*
      by (*metis SUP-def SUP-image continuous-Continuous continuous-def directed-left-mult image-ident mult-assoc*)
    thus $x \mathbin{;} y \mathbin{;} (\mathit{SUP}\ y{:}Y\ .\ y) = (\mathit{SUP}\ z{:}Y\ .\ x \mathbin{;} y \mathbin{;} z)$
      by (*smt SUP-cong mult.assoc*)
  qed
  thus $x \mathbin{;} y \in \mathit{Continuous}$
    by (*metis continuous-Continuous continuous-def*)
qed

lemma *sup-continuous*: $x \in \mathit{Continuous} \wedge y \in \mathit{Continuous} \to x \sqcup y \in \mathit{Continuous}$
  by (*smt SUP-cong SUP-sup-distrib continuous-Continuous continuous-def sup-comp*)

lemma *inf-continuous*: $x \in \mathit{Continuous} \wedge y \in \mathit{Continuous} \to x \sqcap y \in \mathit{Continuous}$
proof
  assume *1*: $x \in \mathit{Continuous} \wedge y \in \mathit{Continuous}$
  have $\forall\, Y.\ \mathit{directed}\ Y \to (x \sqcap y) \mathbin{;} (\mathit{SUP}\ y{:}Y\ .\ y) = (\mathit{SUP}\ z{:}Y\ .\ (x \sqcap y) \mathbin{;} z)$
  proof (*rule*,*rule*)
    fix *Y*
    assume *2*: *directed Y*
    have *3*: $(\mathit{SUP}\ w{:}Y\ .\ \mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} w) \sqcap (y \mathbin{;} z)) \leq (\mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} z) \sqcap (y \mathbin{;} z))$
      apply (*rule SUP-least*)
      apply (*rule SUP-least*)
    proof −
      fix *w z*
      assume $w \in Y$ **and** $z \in Y$
      with *2* **obtain** *v* **where** *4*: $v{\in}Y \wedge w \leq v \wedge z \leq v$
        by (*metis directed-def*)
      hence $x \mathbin{;} w \sqcap (y \mathbin{;} z) \leq (x \mathbin{;} v) \sqcap (y \mathbin{;} v)$
        by (*metis inf-mono le-comp-left-right order-refl*)
      thus $x \mathbin{;} w \sqcap (y \mathbin{;} z) \leq (\mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} z) \sqcap (y \mathbin{;} z))$ **using** *4*
        by (*smt SUP-upper imageI order-trans*)
    qed
    have $(\mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} z) \sqcap (y \mathbin{;} z)) \leq (\mathit{SUP}\ w{:}Y\ .\ \mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} w) \sqcap (y \mathbin{;} z))$
      apply (*rule SUP-least*)
      apply (*smt SUP-upper order-trans*)
      done
    hence $(\mathit{SUP}\ w{:}Y\ .\ \mathit{SUP}\ z{:}Y\ .\ (x \mathbin{;} w) \sqcap (y \mathbin{;} z)) = (\mathit{SUP}\ z{:}Y\ .\ (x \sqcap y) \mathbin{;} z)$ **using** *3*
      by (*smt antisym SUP-cong inf-comp*)
    thus $(x \sqcap y) \mathbin{;} (\mathit{SUP}\ y{:}Y\ .\ y) = (\mathit{SUP}\ z{:}Y\ .\ (x \sqcap y) \mathbin{;} z)$ **using** *1 2*
      by (*metis inf-comp continuous-Continuous continuous-def SUP-inf-distrib2*)

```
  qed
  thus x ⊓ y ∈ Continuous
    by (metis continuous-Continuous continuous-def)
qed

lemma dual-star-continuous: x ∈ Continuous → x ^ ⊗ ∈ Continuous
proof
  assume 1: x ∈ Continuous
  have ∀ Y. directed Y → (x ^ ⊗) ; (SUP y:Y . y) = (SUP z:Y . (x ^ ⊗) ; z)
  proof (rule,rule)
    fix Y
    assume directed Y
    hence directed (op ; (x ^ ⊗) ‘ Y)
      by (metis directed-left-mult)
    hence x ; (SUP y:Y . (x ^ ⊗) ; y) = (SUP y:Y . x ; (x ^ ⊗) ; y) using 1
      by (smt SUP-cong SUP-image continuous-Continuous continuous-def mult-assoc)
    also have ... ≤ (SUP y:Y . (x ^ ⊗) ; y)
      apply (rule SUP-least)
      apply (metis SUP-upper dual-star-comp-fix order-trans sup-ge1)
      done
    finally have x ; (SUP y:Y . (x ^ ⊗) ; y) ⊔ (SUP y:Y . y) ≤ (SUP y:Y . (x ^ ⊗) ; y)
      apply (rule sup-least)
      apply (smt SUP-least SUP-upper dual-star-comp-fix order-trans sup-ge2)
      done
    thus (x ^ ⊗) ; (SUP y:Y . y) = (SUP z:Y . (x ^ ⊗) ; z)
      by (smt SUP-least SUP-upper antisym dual-star-least le-comp)
  qed
  thus x ^ ⊗ ∈ Continuous
    by (metis continuous-Continuous continuous-def)
qed

lemma omega-continuous: x ∈ Continuous → x ^ ω ∈ Continuous
proof
  assume 1: x ∈ Continuous
  have ∀ Y. directed Y → (x ^ ω) ; (SUP y:Y . y) = (SUP z:Y . (x ^ ω) ; z)
  proof (rule,rule)
    fix Y
    assume 2: directed Y
    hence directed (op ; (x ^ ω) ‘ Y)
      by (metis directed-left-mult)
    hence x ; (SUP y:Y . (x ^ ω) ; y) = (SUP y:Y . x ; (x ^ ω) ; y) using 1
      by (smt SUP-cong SUP-image continuous-Continuous continuous-def mult-assoc)
    hence x ; (SUP y:Y . (x ^ ω) ; y) ⊓ (SUP y:Y . y) = (SUP w:Y . SUP z:Y . (x ; (x ^ ω) ; w) ⊓ z)
      by (simp add: SUP-inf-distrib2)
    also have ... ≤ (SUP y:Y . (x ^ ω) ; y)
      apply (rule SUP-least)
      apply (rule SUP-least)
```

```
    proof −
      fix w z
      assume w ∈ Y and z ∈ Y
      with 2 obtain v where 3: v∈Y ∧ w ≤ v ∧ z ≤ v
        by (metis directed-def)
      hence x ; x ^ ω ; w ⊓ z ≤ x ^ ω ; v
        by (metis inf-mono le-comp-left-right order-refl omega-comp-fix)
      thus x ; x ^ ω ; w ⊓ z ≤ (SUP y:Y . (x ^ ω) ; y) using 3
        by (smt SUP-upper imageI order-trans)
    qed
    finally show (x ^ ω) ; (SUP y:Y . y) = (SUP z:Y . (x ^ ω) ; z)
      by (smt SUP-least SUP-upper antisym omega-least le-comp)
  qed
  thus x ^ ω ∈ Continuous
    by (metis continuous-Continuous continuous-def)
qed

definition cocontinuous x ↔ (∀ Y . codirected Y → x ; (INF y:Y . y) = (INF y:Y . x ; y))

definition Cocontinuous = { x . cocontinuous x }

lemma directed-dual: directed X ↔ codirected (dual ‘ X)
  by (simp add: directed-def codirected-def dual-le[THEN sym])

lemma dual-INF: (INF x:X . f x) ^ o = (SUP x:X . f x ^ o)
  by (simp add: INF-def dual-Inf)

lemma dual-SUP: (SUP x:X . f x) ^ o = (INF x:X . f x ^ o)
  by (simp add: SUP-def dual-Sup)

lemma dual-dual-image: dual ‘ dual ‘ X = X
  unfolding image-def
  apply (auto intro: dual-dual)
  done

lemma continuous-dual: continuous x ↔ cocontinuous (x ^ o)
  unfolding continuous-def cocontinuous-def
  proof
    assume ∀ Y. directed Y → x ; (SUP y:Y . y) = (SUP y:Y . x ; y)
    thus ∀ Y. codirected Y → x ^ o ; (INF y:Y . y) = (INF y:Y . x ^ o ; y)
      by (smt INF-cong INF-image directed-dual dual-SUP dual-comp dual-dual-image)
  next
    assume ∀ Y. codirected Y → x ^ o ; (INF y:Y . y) = (INF y:Y . x ^ o ; y)
    thus ∀ Y. directed Y → x ; (SUP y:Y . y) = (SUP y:Y . x ; y)
      by (smt INF-cong INF-image directed-dual dual-INF dual-SUP dual-comp dual-dual)
  qed
```

lemma *cocontinuous-Cocontinuous*: $\mathit{cocontinuous}\ x \leftrightarrow x \in \mathit{Cocontinuous}$
by (*simp add*: *Cocontinuous-def*)

lemma *Continuous-dual*: $x \in \mathit{Continuous} \leftrightarrow x\ \hat{}\ o \in \mathit{Cocontinuous}$
by (*metis cocontinuous-Cocontinuous continuous-Continuous continuous-dual*)

lemma *one-cocontinuous*: $1 \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-one one-continuous*)

lemma *ascending-chain-dual*: $\mathit{ascending\text{-}chain}\ f \leftrightarrow \mathit{descending\text{-}chain}\ (\mathit{dual}\ o\ f)$
by (*metis ascending-chain-def descending-chain-def o-def dual-le*)

lemma *cocontinuous-dist-descending-chain*: $x \in \mathit{Cocontinuous} \wedge \mathit{descending\text{-}chain}\ f \rightarrow x\ ;\ (\mathit{INF}\ n{::}\mathit{nat}\ .\ f\ n) = (\mathit{INF}\ n{::}\mathit{nat}\ .\ x\ ;\ f\ n)$
proof
assume $x \in \mathit{Cocontinuous} \wedge \mathit{descending\text{-}chain}\ f$
hence $x\ \hat{}\ o\ ;\ (\mathit{SUP}\ n{::}\mathit{nat}\ .\ (\mathit{dual}\ o\ f)\ n) = (\mathit{SUP}\ n{::}\mathit{nat}\ .\ x\ \hat{}\ o\ ;\ (\mathit{dual}\ o\ f)\ n)$
by (*smt Continuous-dual SUP-cong ascending-chain-dual continuous-dist-ascending-chain descending-chain-def dual-dual o-def*)
thus $x\ ;\ (\mathit{INF}\ n{::}\mathit{nat}\ .\ f\ n) = (\mathit{INF}\ n{::}\mathit{nat}\ .\ x\ ;\ f\ n)$
by (*smt INF-cong dual-SUP dual-comp dual-dual o-def*)
qed

lemma *assertion-cocontinuous*: $x \in \mathit{assertion} \rightarrow x \in \mathit{Cocontinuous}$
by (*smt Continuous-dual assert-iff-assume assumption-continuous dual-dual*)

lemma *assumption-cocontinuous*: $x \in \mathit{assumption} \rightarrow x \in \mathit{Cocontinuous}$
by (*smt Continuous-dual assert-iff-assume assertion-continuous dual-dual*)

lemma *mult-cocontinuous*: $x \in \mathit{Cocontinuous} \wedge y \in \mathit{Cocontinuous} \rightarrow x\ ;\ y \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-comp dual-dual mult-continuous*)

lemma *sup-cocontinuous*: $x \in \mathit{Cocontinuous} \wedge y \in \mathit{Cocontinuous} \rightarrow x \sqcup y \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-sup dual-dual inf-continuous*)

lemma *inf-cocontinuous*: $x \in \mathit{Cocontinuous} \wedge y \in \mathit{Cocontinuous} \rightarrow x \sqcap y \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-inf dual-dual sup-continuous*)

lemma *dual-omega-cocontinuous*: $x \in \mathit{Cocontinuous} \rightarrow x\ \hat{}\ \mho \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-omega-def dual-dual omega-continuous*)

lemma *star-cocontinuous*: $x \in \mathit{Cocontinuous} \rightarrow x\ \hat{}\ * \in \mathit{Cocontinuous}$
by (*smt Continuous-dual dual-star-def dual-dual dual-star-continuous*)

lemma *dual-omega-iterate*: $y \in \mathit{Cocontinuous} \rightarrow y\ \hat{}\ \mho * z = (\mathit{INF}\ n{::}\mathit{nat}\ .\ ((\lambda x\ .\ y * x \sqcup z)\ \hat{}\ n)\ \mathit{top})$
apply *rule*
apply (*rule antisym*)
apply (*rule INF-greatest*)
apply *simp*

```
  proof −
    fix n
    show y ^ 𝔘 ; z ≤ ((λx. y ; x ⊔ z) ^ n) top
      apply (induct n)
      apply (metis power-zero-id id-def top-greatest)
      apply (smt dual-omega-comp-fix le-comp mult-assoc order-refl sup-mono power-succ-unfold-ext)
      done
  next
    assume 1: y ∈ Cocontinuous
    have 2: descending-chain (λn . ((λx. y ; x ⊔ z) ^ n) top)
    proof (subst descending-chain-def, rule)
      fix n
      show ((λx. y ; x ⊔ z) ^ Suc n) top ≤ ((λx. y ; x ⊔ z) ^ n) top
        apply (induct n)
        apply (metis power-zero-id id-def top-greatest)
        apply (smt power-succ-unfold-ext sup-mono order-refl le-comp)
        done
    qed
    have (INF n. ((λx. y ; x ⊔ z) ^ n) top) ≤ (INF n. ((λx. y ; x ⊔ z) ^ Suc n) top)
      apply (rule INF-greatest)
      unfolding power-succ-unfold-ext
      apply (smt power-succ-unfold-ext INF-lower UNIV-I)
      done
    thus (INF n. ((λx. y ; x ⊔ z) ^ n) top) ≤ y ^ 𝔘 ; z using 1 2
      by (smt INF-cong cocontinuous-dist-descending-chain power-succ-unfold-ext sup-INF sup-commute dual-omega-greatest)
  qed

lemma dual-omega-iterate-one: y ∈ Cocontinuous → y ^ 𝔘 = (INF n::nat . ((λx . y ∗ x ⊔ 1) ^ n) top)
  by (metis dual-omega-iterate mult.right-neutral)

subclass ccpo
  apply unfold-locales
  apply (metis Sup-upper)
  apply (metis Sup-least)
  done

end



sublocale mbt-algebra < mbta!: il-semiring-T where plus = sup and zero = bot and T = top
  apply unfold-locales
  apply (metis sup-assoc)
  apply (metis sup-commute)
  apply (metis sup-idem)
  apply (metis sup-bot-left)
  apply (metis mult.assoc)
```

apply (*metis mult.left-neutral*)
apply (*metis mult.right-neutral*)
apply (*metis le-comp le-supI sup-ge1 sup-ge2*)
apply (*metis sup-comp*)
apply (*metis bot-comp*)
apply (*metis le-iff-sup*)
apply (*metis less-le-not-le*)
apply (*metis sup-top-left*)
done

sublocale *mbt-algebra* < *mbta-dual*!: *il-semiring-T* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **and** *T* = *bot*
apply *unfold-locales*
apply (*metis inf-assoc*)
apply (*metis inf-commute*)
apply (*metis inf-idem*)
apply (*metis inf-top-left*)
apply (*metis mult.assoc*)
apply (*metis mult.left-neutral*)
apply (*metis mult.right-neutral*)
apply (*metis le-comp le-infI inf-le1 inf-le2*)
apply (*metis inf-comp*)
apply (*metis top-comp*)
apply (*metis inf.commute le-iff-inf*)
apply (*metis less-le-not-le*)
apply (*metis inf-bot-left*)
done

sublocale *mbt-algebra* < *mbta*!: *gra-T* **where** *plus* = *sup* **and** *star* = *dual-star* **and** *zero* = *bot* **and** *Omega* = *dual-omega* **and** *T* = *top*
apply *unfold-locales*
apply (*metis dual-star-fix eq-refl sup.commute*)
apply (*metis dual-star-least sup.commute*)
apply (*metis dual-omega-fix eq-refl sup.commute*)
apply (*metis dual-omega-greatest sup.commute*)
done

sublocale *mbt-algebra* < *mbta-dual*!: *gra-T* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **and** *Omega* = *omega* **and** *T* = *bot*
apply *unfold-locales*
apply (*metis eq-refl inf.commute star-fix*)
apply (*metis inf.commute star-greatest*)
apply (*metis eq-refl inf.commute omega-fix*)
apply (*metis inf.commute omega-least*)
done

sublocale *mbt-algebra* < *mbta*!: *il-conway-semiring-L* **where** *circ* = *dual-star* **and** *plus* = *sup* **and** *zero* = *bot* **and** *L* = *bot*
apply *unfold-locales*
apply (*metis mbta.add-left-zero mbta.star-one mult.left-neutral*)
apply (*metis eq-refl mbta.add-right-zero*)

done

sublocale *mbt-algebra* < *mbta-dual*!: *il-conway-semiring-L* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **and** *L* = *bot*
apply *unfold-locales*
apply (*metis bot-comp inf-bot-left mbta-dual.Omega-one*)
apply (*metis bot-least inf-bot-right*)
done

sublocale *mbt-algebra* < *mbta-fix*!: *il-conway-semiring-L* **where** *circ* = *dual-omega* **and** *plus* = *sup* **and** *zero* = *bot* **and** *L* = *top*
apply *unfold-locales*
apply (*metis mbta.Omega-one mbta.add-left-top mbta.top-left-zero*)
apply (*metis mbta.add-right-top top-greatest*)
done

sublocale *mbt-algebra* < *mbta-fix-dual*!: *il-conway-semiring-L* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **and** *L* = *top*
apply *unfold-locales*
apply (*metis inf-top-left mbta-dual.star-one mult.left-neutral*)
apply (*metis eq-refl inf-top-right*)
done

sublocale *mbt-algebra* < *mbta*!: *lk-conway-semiring* **where** *circ* = *dual-star* **and** *plus* = *sup* **and** *star* = *dual-star* **and** *zero* = *bot* **..**

sublocale *mbt-algebra* < *mbta-dual*!: *lk-conway-semiring* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **..**

sublocale *mbt-algebra* < *mbta-fix*!: *lk-conway-semiring* **where** *circ* = *dual-omega* **and** *plus* = *sup* **and** *star* = *dual-star* **and** *zero* = *bot* **..**

sublocale *mbt-algebra* < *mbta-fix-dual*!: *lk-conway-semiring* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *zero* = *top* **..**

sublocale *mbt-algebra* < *mbta*!: *tests* **where** *plus* = *sup* **and** *uminus* = *neg-assert* **and** *zero* = *bot*
apply *unfold-locales*
apply (*metis mult.assoc*)
apply (*metis neg-assertion assertion-inf-comp-eq inf-commute*)
apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assert-def*)
apply (*metis inf-assoc dual-neg sup-bot-right sup-inf-distrib1*)
apply (*metis comp-assertion neg-assertion uminus-uminus*)
apply (*rule the-equality*[*THEN sym*])
apply (*metis assertion-inf-comp-eq inf-uminus neg-assertion*)
apply (*metis bot-comp dual-top inf-bot-left neg-assert-def*)
apply (*metis dual-bot inf-top-left neg-assert-def top-comp*)
apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assert-def*)
apply (*metis inf-sup-distrib2*)
apply (*metis assertion-inf-comp-eq le-iff-inf neg-assertion*)
apply (*metis less-le-not-le*)
done

sublocale *mbt-algebra* < *mbta-dual*!: *tests* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *uminus* = *neg-assume* **and** *zero* = *top*
apply *unfold-locales*

apply (*metis mult.assoc*)
apply (*metis neg-assumption assumption-sup-comp-eq sup-commute*)
apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assume-def*)
apply (*smt dual-comp dual-dual dual-inf dual-neg dual-top inf-sup-distrib1 inf-top-right sup.commute sup.left-commute*)
apply (*metis comp-assumption neg-assumption uminus-uminus-assume*)
apply (*rule the-equality*[*THEN sym*])
apply (*metis assumption-sup-comp-eq inf-uminus-assume neg-assumption*)
apply (*metis top-comp dual-bot sup-top-left neg-assume-def*)
apply (*metis dual-top sup-bot-left neg-assume-def bot-comp*)
apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assume-def*)
apply (*metis sup-inf-distrib2*)
apply (*metis assumption-sup-comp-eq le-iff-sup neg-assumption sup.commute*)
apply (*metis less-le-not-le*)
done

sublocale *mbt-algebra* < *mbta*!: *relative-antidomain-semiring-T* **where** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** *plus* = *sup* **and** *uminus* = *neg-assert* **and** *zero* = *bot* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply (*simp-all add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assert-def*)
apply (*metis dual-neg eq-refl inf.commute inf-mono mbta.top-right-mult-increasing*)
apply (*metis mbta.add-left-zero mbta.mult-right-dist-add mult.assoc mult.left-neutral sup.commute*)
apply (*metis dual-neg inf.commute inf.left-commute inf-bot-left*)
apply (*metis inf.commute inf-sup-distrib1*)
apply (*metis inf.assoc*)
done

sublocale *mbt-algebra* < *mbta-dual*!: *relative-antidomain-semiring-T* **where** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *uminus* = *neg-assume* **and** *zero* = *top* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply (*simp-all add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assume-def*)
apply (*metis dual-dual dual-neg-top mbta.add-isotone mbta.zero-right-mult-decreasing mbta-dual.order-refl sup.commute*)
apply (*smt bot-comp dual-bot dual-comp dual-one dual-sup mbta.add-right-zero mbta.mult-right-dist-add mult.assoc mult.left-neutral*)
apply (*metis dual-dual dual-neg-top mbta.add-right-top sup.commute sup.left-commute*)
apply (*metis sup.commute sup-inf-distrib1*)
apply (*smt sup.assoc*)
done

sublocale *mbt-algebra* < *mbta*!: *relative-domain-semiring-split* **where** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** *plus* = *sup* **and** *zero* = *bot* **and** *Z* = *bot*
apply *unfold-locales*
apply (*metis eq-refl mbta.add-right-zero*)
done

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** *plus* = *inf* **and** *zero* = *top* **and** *Z* = *top*
apply *unfold-locales*
apply (*metis eq-refl inf-top-right*)
done

sublocale *mbt-algebra* < *mbta*!: *diamond-while* **where** *box* = $\lambda x\ y$. *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-star* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *plus* = *sup* **and** *pre* = $\lambda x\ y$. *wpt* (*x* ∗ *y*) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ ⊗) ∗ *neg-assert p* **and** *zero* = *bot* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply *simp-all*
apply (*metis wpt-def*)
done

sublocale *mbt-algebra* < *mbta-dual*!: *box-while* **where** *box* = $\lambda x\ y$. *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *omega* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* (*x* ^ *o* ; *y*) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ *ω*) ∗ *neg-assume p* **and** *zero* = *top* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply *simp-all*
apply (*metis bot-comp dual-comp dual-dual dual-top mbta.add-left-zero mbta.mult-right-dist-add mult.assoc mult.left-neutral neg-assume-def sup.commute wpb-def*)
done

sublocale *mbt-algebra* < *mbta-fix*!: *diamond-while* **where** *box* = $\lambda x\ y$. *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-omega* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *plus* = *sup* **and** *pre* = $\lambda x\ y$. *wpt* (*x* ∗ *y*) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ ℧) ∗ *neg-assert p* **and** *zero* = *bot* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply *simp-all*
done

sublocale *mbt-algebra* < *mbta-fix-dual*!: *box-while* **where** *box* = $\lambda x\ y$. *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* (*x* ^ *o* ; *y*) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ ∗) ∗ *neg-assume p* **and** *zero* = *top* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply *simp-all*
done

sublocale *mbt-algebra* < *mbta-pre*!: *box-while* **where** *box* = $\lambda x\ y$. *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-star* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *plus* = *sup* **and** *pre* = $\lambda x\ y$. *wpt* (*x* ^ *o* ∗ *y*) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ ⊗) ∗ *neg-assert p* **and** *zero* = *bot* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply *simp-all*
apply (*smt dual-bot dual-comp dual-dual dual-inf mbta.add-left-zero mbta.mult-associative mbta.mult-right-dist-add mbta-dual.Z-top mult.left-neutral neg-assert-def sup.commute wpt-def*)
done

sublocale *mbt-algebra* < *mbta-pre-dual*!: *diamond-while* **where** *box* = $\lambda x\ y$. *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *omega* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = $\lambda x\ p\ y$. (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* (*x* ; *y*) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. ((*p* ∗ *x*) ^ *ω*) ∗ *neg-assume p* **and** *zero* = *top* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply *simp-all*
apply (*metis wpb-def*)
done

sublocale *mbt-algebra* < *mbta-pre-fix*!: *box-while* **where** *box* = $\lambda x\ y$. *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-omega* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = $\lambda x\ y$. (*x* ∗ *y* ∗

top) ⊓ *1* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *plus* = *sup* **and** *pre* = λ*x y* . *wpt* (*x* ˆ *o* ∗ *y*) **and** *uminus* = *neg-assert* **and** *while* = λ*p x* . ((*p* ∗ *x*) ˆ ℧) ∗ *neg-assert p* **and** *zero* = *bot* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply *simp-all*
done

sublocale *mbt-algebra* < *mbta-pre-fix-dual*!: *diamond-while* **where** *box* = λ*x y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λ*x* . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = λ*x y* . *wpb* (*x* ; *y*) **and** *uminus* = *neg-assume* **and** *while* = λ*p x* . ((*p* ∗ *x*) ˆ ∗) ∗ *neg-assume p* **and** *zero* = *top* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply *simp-all*
done

sublocale *complete-mbt-algebra* < *mbta*!: *complete-tests* **where** *plus* = *sup* **and** *uminus* = *neg-assert* **and** *zero* = *bot*
apply *unfold-locales*
apply (*smt mbta.test-set-def neg-assertion subset-eq Sup-assertion assertion-neg-assert*)
apply (*metis Sup-upper*)
apply (*metis Sup-least*)
done

sublocale *complete-mbt-algebra* < *mbta-dual*!: *complete-tests* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *uminus* = *neg-assume* **and** *zero* = *top* **and** *Inf* = *Sup* **and** *Sup* = *Inf*
apply *unfold-locales*
apply (*smt mbta-dual.test-set-def neg-assumption subset-eq Inf-assumption assumption-neg-assume*)
apply (*metis Inf-lower*)
apply (*metis Inf-greatest*)
done

sublocale *complete-mbt-algebra* < *mbta*!: *complete-antidomain-semiring* **where** *d* = λ*x* . (*x* ∗ *top*) ⊓ *1* **and** *plus* = *sup* **and** *uminus* = *neg-assert* **and** *zero* = *bot* **and** *Z* = *bot*
apply *unfold-locales*
apply *rule*
unfolding *mbta.Sum-def mbta.Prod-def neg-assert-def dual-Inf dual-Sup INF-def SUP-def Inf-comp Sup-comp*
unfolding *inf-commute*
apply (*subst inf-Inf*)
apply (*metis* (*mono-tags*) *empty-Collect-eq image-is-empty*)
apply (*rule arg-cong*[**where** *f*=*Inf*])
apply *auto*
unfolding *inf-Sup SUP-def*
apply (*rule arg-cong*[**where** *f*=*Sup*])
apply *auto*
done

sublocale *complete-mbt-algebra* < *mbta-dual*!: *complete-antidomain-semiring* **where** *d* = λ*x* . (*x* ∗ *bot*) ⊔ *1* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *uminus* = *neg-assume* **and** *zero* = *top* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *Z* = *top*
apply *unfold-locales*
apply *rule*
unfolding *mbta-dual.Sum-def mbta-dual.Prod-def neg-assume-def dual-Inf dual-Sup INF-def SUP-def Inf-comp Sup-comp*

unfolding *sup-commute*
apply (*subst sup-Sup*)
apply (*metis* (*mono-tags*) *empty-Collect-eq image-is-empty*)
apply (*rule arg-cong*[**where** *f*=*Sup*])
apply *auto*
unfolding *sup-Inf INF-def*
apply (*rule arg-cong*[**where** *f*=*Inf*])
apply *auto*
done

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** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .$ *wpt* ($x * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply (*metis one-continuous*)
apply *simp-all*
done

sublocale *complete-mbt-algebra* < *mbta-dual*!: *box-while-program* **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** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .$ *wpb* ($x\ \hat{}\ o\ ;\ y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply (*metis one-continuous*)
apply *simp-all*
done

sublocale *complete-mbt-algebra* < *mbta-fix*!: *diamond-while-program* **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** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .$ *wpt* ($x * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assertion* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply (*metis one-cocontinuous*)
apply *simp-all*
done

sublocale *complete-mbt-algebra* < *mbta-fix-dual*!: *box-while-program* **where** *box* = $\lambda x\ y\ .$ *neg-assume* (x ∗ *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* = $\lambda x\ p\ y\ .\ (p * x) \sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .$ *wpb* ($x\ \hat{}\ o\ ;\ y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * neg\text{-}assume\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assumption* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply (*metis one-cocontinuous*)
apply *simp-all*
done

sublocale *complete-mbt-algebra* < *mbta-pre*!: *box-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** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .$ *wpt* ($x\ \hat{}\ o * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and** *T* = *top* **and** *Z* = *bot* **..**

sublocale *complete-mbt-algebra* < *mbta-pre-dual*!: *diamond-while-program* **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* = $\lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ wpb\ (x\ ;\ y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** T = *bot* **and** Z = *top* **..**

sublocale *complete-mbt-algebra* < *mbta-pre-fix*!: *box-while-program* **where** *box* = $\lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** *circ* = *dual-omega* **and** *d* = $\lambda x\ .\ (x * top) \sqcap 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .\ wpt\ (x\ \hat{}\ o * y)$ **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assertion* **and** T = *top* **and** Z = *bot* **..**

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-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ wpb\ (x\ ;\ y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * neg\text{-}assume\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assumption* **and** T = *bot* **and** Z = *top* **..**

sublocale *complete-mbt-algebra* < *mbta*!: *diamond-hoare-sound* **where** *box* = $\lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** *circ* = *dual-star* **and** *d* = $\lambda x\ .\ (x * top) \sqcap 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .\ wpt\ (x * y)$ **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and** T = *top* **and** Z = *bot*
apply *unfold-locales*
apply (*metis bot-comp bot-least mbta.aL-one-circ mbta.d-Z mbta.one-circ-L*)
done

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-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ wpb\ (x\ \hat{}\ o\ ;\ y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** T = *bot* **and** Z = *top*
apply *unfold-locales*
apply (*metis bot-comp mbta.top-left-zero mbta-dual.Omega-one mbta-dual.aL-one-circ mbta-dual.a-T top-greatest*)
done

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-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** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .\ wpt\ (x * y)$ **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * neg\text{-}assert\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assertion* **and** T = *top* **and** Z = *bot*
apply *unfold-locales*
proof
fix *p q x*
let *?pt* = *neg-assert p*
let *?qt* = *neg-assert q*
have *d-mult-top*: $\forall y\ .\ ((y * top) \sqcap 1) * top = y * top$
by (*metis inf.commute inf-comp inf-top-left mult.assoc mult.left-neutral*)
assume *neg-assert ?pt* ; *?qt* $\leq x$; *?qt* ; *top* $\sqcap$ *1*
hence *?qt* ; *top* $\leq x\ \hat{}\ \mho$; *?pt* ; *top*
by (*smt mbta.Omega-induct mbta.d-def d-mult-top mbta.mult-left-isotone mbta.shunting-T-1 mult.assoc*)
thus *mbta-fix.aL* ; *?qt* $\leq x\ \hat{}\ \mho$; *?pt* ; *top* $\sqcap$ *1*
by (*smt inf.commute inf-top-left mbta.Omega-one mbta.d-T mbta-dual.Omega.circ-isotone mbta-dual.Omega.mult-top-circ mbta-dual.Z-top mbta-dual.mult-L-circ-mult mbta-fix.aL-one-circ mult.assoc mult.left-neutral neg-assert-def*)
qed

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-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ wpb\ (x\ \hat{}\ o\ ;\ y)$ **and** *uminus* =

neg-assume **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * \mathit{neg\text{-}assume}\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** T = *bot* **and** Z = *top*
apply *unfold-locales*
apply (*metis eq-refl mbta-dual.a-Z mbta-fix-dual.aL-one-circ mbta-fix-dual.one-circ-L mult.left-neutral*)
done

sublocale *complete-mbt-algebra* < *mbta-pre*!: *box-hoare-sound* **where** *box* = $\lambda x\ y\ .\ \mathit{neg\text{-}assert}\ (x * \mathit{neg\text{-}assert}\ y)$ **and** *circ* = *dual-star* **and** *d* = $\lambda x\ .\ (x * \mathit{top}) \sqcap 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * \mathit{top}) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (\mathit{neg\text{-}assert}\ p * y)$ **and** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .\ \mathit{wpt}\ (x\ \hat{}\ o * y)$ **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * \mathit{neg\text{-}assert}\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and** T = *top* **and** Z = *bot*
apply *unfold-locales*
apply (*metis eq-refl mbta.a-Z mbta.one-circ-L mbta-pre.aL-one-circ mult.left-neutral*)
done

sublocale *complete-mbt-algebra* < *mbta-pre-dual*!: *diamond-hoare-sound-2* **where** *box* = $\lambda x\ y\ .\ \mathit{neg\text{-}assume}\ (x * \mathit{neg\text{-}assume}\ y)$ **and** *circ* = *omega* **and** *d* = $\lambda x\ .\ (x * \mathit{bot}) \sqcup 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * \mathit{bot}) \sqcup 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcap (\mathit{neg\text{-}assume}\ p * y)$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ \mathit{wpb}\ (x\ ;\ y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * \mathit{neg\text{-}assume}\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** T = *bot* **and** Z = *top*
apply *unfold-locales*
proof
fix *p q x*
let *?pt* = *neg-assume p*
let *?qt* = *neg-assume q*
assume $x\ ;\ ?qt\ ;\ \mathit{bot} \sqcup 1 \leq \mathit{neg\text{-}assume}\ ?pt\ ;\ ?qt$
hence $x\ ;\ ?qt\ ;\ \mathit{bot} \sqcap ?pt \leq ?qt$
by (*smt inf-commute inf-le1 le-supE mbta-dual.a-compl-intro mbta-dual.add-right-isotone mbta-dual.d-def order-trans*)
hence $(x\ ;\ ?qt\ ;\ \mathit{bot} \sqcap ?pt)\ ;\ \mathit{bot} \leq ?qt\ ;\ \mathit{bot}$
by (*smt mbta.mult-left-isotone*)
hence $x\ \hat{}\ \omega\ ;\ ?pt\ ;\ \mathit{bot} \sqcup 1 \leq ?qt$
by (*smt bot-comp inf-comp mbta.add-left-isotone mbta-dual.a-d-closed mult-assoc omega-least*)
thus $x\ \hat{}\ \omega\ ;\ ?pt\ ;\ \mathit{bot} \sqcup 1 \leq \mathit{mbta\text{-}pre\text{-}dual.aL}\ ;\ ?qt$
by (*metis bot-comp mbta.add-right-zero mbta-dual.Omega-one mbta-pre-dual.aL-one-circ mult.left-neutral sup.commute*)
qed

sublocale *complete-mbt-algebra* < *mbta-pre-fix*!: *box-hoare-sound* **where** *box* = $\lambda x\ y\ .\ \mathit{neg\text{-}assert}\ (x * \mathit{neg\text{-}assert}\ y)$ **and** *circ* = *dual-omega* **and** *d* = $\lambda x\ .\ (x * \mathit{top}) \sqcap 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * \mathit{top}) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (\mathit{neg\text{-}assert}\ p * y)$ **and** *plus* = *sup* **and** *pre* = $\lambda x\ y\ .\ \mathit{wpt}\ (x\ \hat{}\ o * y)$ **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * \mathit{neg\text{-}assert}\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assertion* **and** T = *top* **and** Z = *bot*
apply *unfold-locales*
apply (*metis bot-comp bot-least mbta.Omega-one mbta.a-T mbta-dual.Z-top mbta-pre-fix.aL-one-circ*)
done

sublocale *complete-mbt-algebra* < *mbta-pre-fix-dual*!: *diamond-hoare-sound* **where** *box* = $\lambda x\ y\ .\ \mathit{neg\text{-}assume}\ (x * \mathit{neg\text{-}assume}\ y)$ **and** *circ* = *star* **and** *d* = $\lambda x\ .\ (x * \mathit{bot}) \sqcup 1$ **and** *diamond* = $\lambda x\ y\ .\ (x * y * \mathit{bot}) \sqcup 1$ **and** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcap (\mathit{neg\text{-}assume}\ p * y)$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = $\lambda x\ y\ .\ \mathit{wpb}\ (x\ ;\ y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * \mathit{neg\text{-}assume}\ p$ **and** *zero* = *top* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** T = *bot* **and** Z = *top*
apply *unfold-locales*
apply (*metis mbta.add-left-top mbta.top-left-zero mbta-fix-dual.L-def mbta-pre-fix-dual.aL-one-circ top-greatest*)
done

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** $plus = sup$ **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** $zero = bot$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assertion$ **and** $T = top$ **and** $Z = bot$
apply *unfold-locales*
apply (*smt Inf-fin.idem inf.commute inf-le1 mbta-dual.case-split-left neg-assert-def*)
apply (*metis mbta.aL-one-circ mbta.d-Z mbta.one-circ-L*)
defer
apply (*metis wpt-def*)
unfolding *mbta.Sum-range SUP-def*[*THEN sym*]
proof
fix *x t*
assume *1*: $x \in$ *while-program.While-program op* ; *neg-assert Continuous assertion* $(\lambda x\ p\ y\ .\ p\ ;\ x \sqcup neg\text{-}assert\ p\ ;\ y)$ $(\lambda p\ x\ .\ (p\ ;\ x)\ \hat{}\ \otimes\ ;\ neg\text{-}assert\ p) \wedge$ *ascending-chain t* $\wedge$ *tests.test-seq neg-assert t*
have $x \in$ *Continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** $pre = \lambda x\ y\ .\ wpt\ (x * y)$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$])
apply *unfold-locales*
apply (*metis 1*)
apply *metis*
apply (*metis mult-continuous*)
apply (*metis assertion-continuous mbta.test-expression-test mult-continuous neg-assertion sup-continuous*)
apply (*metis assertion-continuous dual-star-continuous mbta.test-expression-test mult-continuous neg-assertion*)
done
thus $x\ ;\ (SUP\ n{::}nat\ .\ t\ n) = (SUP\ n{::}nat\ .\ x\ ;\ t\ n)$ **using** *1*
by (*smt continuous-dist-ascending-chain SUP-cong*)
qed

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** $plus = inf$ **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** $zero = top$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assumption$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $T = bot$ **and** $Z = top$
apply *unfold-locales*
apply *rule*
defer
apply (*metis bot-comp mbta-dual.Omega-one mbta-dual.aL-one-circ mbta-dual.a-T*)
prefer *2*
apply (*metis mbta-dual.pre-def*)
unfolding *mbta-dual.Prod-range SUP-def*[*THEN sym*]
proof
fix *x t*
assume *1*: $x \in$ *while-program.While-program op* ; *neg-assume Continuous assumption* $(\lambda x\ p\ y\ .\ (p\ ;\ x) \sqcap (neg\text{-}assume\ p\ ;\ y))$ $(\lambda p\ x\ .\ (p\ ;\ x)\ \hat{}\ \omega\ ;\ neg\text{-}assume\ p) \wedge$ *ord.descending-chain greater-eq t* $\wedge$ *tests.test-seq neg-assume t*
have $x \in$ *Continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** $pre = \lambda x\ y\ .\ wpb\ (x\ \hat{}\ o\ ;\ y)$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$])
apply *unfold-locales*
apply (*metis 1*)
apply *metis*

apply (*metis mult-continuous*)
apply (*metis assumption-continuous mbta-dual.test-expression-test mult-continuous neg-assumption inf-continuous*)
apply (*metis assumption-continuous omega-continuous mbta-dual.test-expression-test mult-continuous neg-assumption*)
done
thus *x* ; (*SUP n*::*nat* . *t n*) = (*SUP n*::*nat* . *x* ; *t n*) **using** *1*
by (*smt ord.descending-chain-def ascending-chain-def continuous-dist-ascending-chain SUP-cong*)
next
fix *p x q*
let *?pt* = *neg-assume p*
let *?qt* = *neg-assume q*
assume *?qt* ≤ *?pt* ; *neg-assume* (*x* ; *neg-assume ?qt*)
also have ... ≤ *x* ^ *o* ; *?qt* ⊔ *?pt*
by (*smt assumption-sup-comp-eq mbta.add-left-isotone mbta.zero-right-mult-decreasing mbta-dual.pre-def neg-assume-def neg-assumption sup.commute sup.left-commute sup.left-idem wpb-def*)
finally show *?qt* ⊓ *mbta-dual.aL* ≤ *neg-assume* (*x* ^ ω ; *neg-assume ?pt*)
by (*smt dual-dual dual-omega-def dual-omega-greatest le-infI1 mbta-dual.a-d-closed mbta-dual.d-isotone mbta-dual.pre-def wpb-def*)
qed

sublocale *complete-mbt-algebra* < *mbta-pre-fix-dual*!: *diamond-hoare-valid* **where** *box* = λ*x y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λ*x* . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *hoare-triple* = λ*p x q* . *wpb*(*x* ∗ *q*) ≤ *p* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *plus* = *inf* **and** *pre* = λ*x y* . *wpb* (*x* ; *y*) **and** *uminus* = *neg-assume* **and** *while* = λ*p x* . ((*p* ∗ *x*) ^ ∗) ∗ *neg-assume p* **and** *zero* = *top* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *T* = *bot* **and** *Z* = *top*
apply *unfold-locales*
apply (*smt gt-one-comp mbta.add-commutative mbta.add-left-upper-bound neg-assume-def*)
apply (*metis mbta-pre-fix-dual.aL-zero*)
defer
apply (*metis wpb-def*)
unfolding *mbta-dual.Sum-range INF-def*[*THEN sym*]
proof
fix *x t*
assume *1*: *x* ∈ *while-program.While-program op* ; *neg-assume Cocontinuous assumption* (λ*x p y* . (*p* ; *x*) ⊓ (*neg-assume p* ; *y*)) (λ*p x* . (*p* ; *x*) ^ ∗ ; *neg-assume p*) ∧ *ord.ascending-chain greater-eq t* ∧ *tests.test-seq neg-assume t*
have *x* ∈ *Cocontinuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=λ*x y* . *wpb* (*x* ; *y*) **and** *while*=λ*p x* . ((*p* ∗ *x*) ^ ∗) ∗ *neg-assume p*])
apply *unfold-locales*
apply (*metis 1*)
apply *metis*
apply (*metis mult-cocontinuous*)
apply (*metis assumption-cocontinuous mbta-dual.test-expression-test mult-cocontinuous neg-assumption inf-cocontinuous*)
apply (*metis assumption-cocontinuous star-cocontinuous mbta-dual.test-expression-test mult-cocontinuous neg-assumption*)
done
thus *x* ; (*INF n*::*nat* . *t n*) = (*INF n*::*nat* . *x* ; *t n*) **using** *1*
by (*smt descending-chain-def ord.ascending-chain-def cocontinuous-dist-descending-chain INF-cong*)
qed

sublocale *complete-mbt-algebra* < *mbta-pre-fix*!: *box-hoare-valid* **where** *box* = λ*x y* . *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-omega* **and** *d* = λ*x* . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *hoare-triple* = λ*p x q* . *p* ≤ *wpt*(*x* ^ *o* ∗ *q*) **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *plus* = *sup* **and** *pre* = λ*x y* . *wpt* (*x* ^ *o* ∗ *y*) **and** *star* =

dual-star **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * \textit{neg-assert}\ p$ **and** *zero* = *bot* **and** *Atomic-program* = *Cocontinuous* **and** *Atomic-test* = *assertion* **and** *T* = *top* **and** *Z* = *bot*
apply *unfold-locales*
apply *rule*
defer
apply (*metis mbta.Omega-one mbta.a-T mbta.top-left-zero mbta-pre-fix.aL-one-circ*)
prefer *2*
apply (*metis mbta-pre.pre-def*)
unfolding *mbta.Prod-range INF-def*[*THEN sym*]
proof
fix *x t*
assume *1*: $x \in \textit{while-program.While-program}\ \textit{op}\ ;\ \textit{neg-assert Cocontinuous assertion}\ (\lambda x\ p\ y\ .\ p\ ;\ x \sqcup \textit{neg-assert}\ p\ ;\ y)\ (\lambda p\ x\ .\ (p\ ;\ x)\ \hat{}\ \mho\ ;\ \textit{neg-assert}\ p) \wedge \textit{descending-chain}\ t \wedge \textit{tests.test-seq neg-assert}\ t$
have $x \in \textit{Cocontinuous}$
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=$\lambda x\ y\ .\ \textit{wpt}\ (x\ \hat{}\ o\ ;\ y)$ **and** *while*=$\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * \textit{neg-assert}\ p$])
apply *unfold-locales*
apply (*metis 1*)
apply *metis*
apply (*metis mult-cocontinuous*)
apply (*metis assertion-cocontinuous mbta.test-expression-test mult-cocontinuous neg-assertion sup-cocontinuous*)
apply (*metis assertion-cocontinuous dual-omega-cocontinuous mbta.test-expression-test mult-cocontinuous neg-assertion*)
done
thus $x\ ;\ (\textit{INF}\ n{::}\textit{nat}\ .\ t\ n) = (\textit{INF}\ n{::}\textit{nat}\ .\ x\ ;\ t\ n)$ **using** *1*
by (*smt descending-chain-def cocontinuous-dist-descending-chain INF-cong*)
next
fix *p x q*
let *?pt* = *neg-assert p*
let *?qt* = *neg-assert q*
assume *1*: *?pt* ; *neg-assert* (*x* ; *neg-assert ?qt*) ≤ *?qt*
have $x\ \hat{}\ o\ ;\ ?qt \sqcap ?pt \leq ?pt\ ;\ \textit{neg-assert}\ (x\ ;\ \textit{neg-assert}\ ?qt)$
by (*smt inf-comp mbta.sub-comm mbta.top-right-mult-increasing mbta-dual.add-left-isotone mbta-pre.pre-def mult.left-neutral mult-assoc top-comp wpt-def*)
also have ... ≤ *?qt* **using** *1*
by *metis*
finally have $(x\ \hat{}\ o)\ \hat{}\ \omega\ ;\ ?pt\ ;\ \textit{top} \leq ?qt\ ;\ \textit{top}$
by (*metis mbta.mult-left-isotone omega-least mult-assoc*)
hence *neg-assert* $(x\ \hat{}\ \mho\ ;\ \textit{neg-assert}\ ?pt) \leq ?qt$
by (*smt dual-omega-def inf-mono mbta.d-a-closed mbta.d-def mbta-pre.pre-def order-refl wpt-def mbta.a-d-closed*)
thus *neg-assert* $(x\ \hat{}\ \mho\ ;\ \textit{neg-assert}\ ?pt) \leq ?qt \sqcup \textit{mbta-pre-fix.aL}$
by (*smt mbta.add-left-upper-bound order-trans*)
qed

end