Unifying Correctness Statements

Walter Guttmann, Universität Ulm

2012.04.08

theory *UCS*

imports *Main*

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 *order*

begin

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

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** $is\text{-}least\text{-}fixpoint\ f\ x \leftrightarrow f(x) = x \wedge (\forall y\ .\ f(y) = y \longrightarrow x \leq y)$
definition *is-greatest-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}greatest\text{-}fixpoint\ f\ x \leftrightarrow f(x) = x \wedge (\forall y\ .\ f(y) = y \longrightarrow x \geq y)$
definition *is-least-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}least\text{-}prefixpoint\ f\ x \leftrightarrow f(x) \leq x \wedge (\forall y\ .\ f(y) \leq y \longrightarrow x \leq y)$
definition *is-greatest-postfixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a \Rightarrow bool$ **where** $is\text{-}greatest\text{-}postfixpoint\ f\ x \leftrightarrow f(x) \geq x \wedge (\forall y\ .\ f(y) \geq y \longrightarrow x \geq y)$
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*: $has\text{-}least\text{-}fixpoint\ f \longrightarrow (\exists!x\ .\ is\text{-}least\text{-}fixpoint\ f\ x)$
  by (*smt antisym has-least-fixpoint-def is-least-fixpoint-def*)

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

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

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

lemma *least-fixpoint*: $has\text{-}least\text{-}fixpoint\ f \longrightarrow is\text{-}least\text{-}fixpoint\ f\ (\mu\ f)$
proof
  assume *has-least-fixpoint f*
  hence $is\text{-}least\text{-}fixpoint\ f\ (THE\ x\ .\ is\text{-}least\text{-}fixpoint\ f\ x)$
    by (*smt least-fixpoint-unique theI'*)
  thus $is\text{-}least\text{-}fixpoint\ f\ (\mu\ f)$
    by (*simp add*: *is-least-fixpoint-def the-least-fixpoint-def*)
qed

lemma *greatest-fixpoint*: *has-greatest-fixpoint f* $\longrightarrow$ *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* $\longrightarrow$ *is-least-prefixpoint f* ($p\mu$ *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* ($p\mu$ *f*)
    by (*simp add*: *is-least-prefixpoint-def the-least-prefixpoint-def*)
qed

lemma *greatest-postfixpoint*: *has-greatest-postfixpoint f* $\longrightarrow$ *is-greatest-postfixpoint f* ($p\nu$ *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* ($p\nu$ *f*)
    by (*simp add*: *is-greatest-postfixpoint-def the-greatest-postfixpoint-def*)
qed

lemma *least-fixpoint-same*: *is-least-fixpoint f x* $\longrightarrow$ $x = \mu$ *f*
  by (*metis least-fixpoint least-fixpoint-unique has-least-fixpoint-def*)

lemma *greatest-fixpoint-same*: *is-greatest-fixpoint f x* $\longrightarrow$ $x = \nu$ *f*
  by (*metis greatest-fixpoint greatest-fixpoint-unique has-greatest-fixpoint-def*)

lemma *least-prefixpoint-same*: *is-least-prefixpoint f x* $\longrightarrow$ $x = p\mu$ *f*
  by (*metis least-prefixpoint least-prefixpoint-unique has-least-prefixpoint-def*)

lemma *greatest-postfixpoint-same*: *is-greatest-postfixpoint f x* $\longrightarrow$ $x = p\nu$ *f*
  by (*metis greatest-postfixpoint greatest-postfixpoint-unique has-greatest-postfixpoint-def*)

lemma *least-fixpoint-char*: *is-least-fixpoint f x* $\longleftrightarrow$ *has-least-fixpoint f* $\wedge$ $x = \mu$ *f*
  by (*metis least-fixpoint-same has-least-fixpoint-def*)

lemma *least-prefixpoint-char*: *is-least-prefixpoint f x* $\longleftrightarrow$ *has-least-prefixpoint f* $\wedge$ $x = p\mu$ *f*
  by (*metis least-prefixpoint-same has-least-prefixpoint-def*)

lemma *greatest-fixpoint-char*: *is-greatest-fixpoint f x* $\longleftrightarrow$ *has-greatest-fixpoint f* $\wedge$ $x = \nu$ *f*
  by (*metis greatest-fixpoint-same has-greatest-fixpoint-def*)

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

lemma *least-prefixpoint-fixpoint*: *has-least-prefixpoint f* $\wedge$ *isotone f* $\rightarrow$ *is-least-fixpoint f* $(p\mu\ f)$
by (*smt eq-iff is-least-fixpoint-def is-least-prefixpoint-def isotone-def least-prefixpoint*)

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

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

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

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

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

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

lemma *greatest-postfixpoint-fixpoint*: *has-greatest-postfixpoint f* $\wedge$ *isotone f* $\rightarrow$ *is-greatest-fixpoint f* $(p\nu\ f)$
by (*smt eq-iff is-greatest-fixpoint-def is-greatest-postfixpoint-def isotone-def greatest-postfixpoint*)

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

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

lemma *mu-square*: *isotone f* $\wedge$ *has-least-fixpoint f* $\wedge$ *has-least-fixpoint* $(f \circ f) \rightarrow \mu\ f = \mu\ (f \circ f)$
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* $(f \circ f) \rightarrow \nu\ f = \nu\ (f \circ f)$
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* $(g \circ f) \rightarrow \mu\ (g \circ f) = g(\mu\ (f \circ g))$

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* ∧ *has-greatest-fixpoint* (*f* ∘ *g*) ∧ *has-greatest-fixpoint* (*g* ∘ *f*) → *ν* (*g* ∘ *f*) = *g*(*ν* (*f* ∘ *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* ∧ *has-greatest-fixpoint f* → *μ f* ≤ *ν f*
by (*metis is-greatest-fixpoint-def is-least-fixpoint-def least-fixpoint greatest-fixpoint*)

lemma *pmu-below-pnu-fix*: *has-fixpoint f* ∧ *has-least-prefixpoint f* ∧ *has-greatest-postfixpoint f* → *pμ f* ≤ *pν 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* ∧ *has-least-prefixpoint f* ∧ *has-greatest-postfixpoint f* → *pμ f* ≤ *pν f*
by (*metis has-fixpoint-def is-fixpoint-def is-least-fixpoint-def least-prefixpoint-fixpoint pmu-below-pnu-fix*)

definition *galois* :: (*'a* ⇒ *'a*) ⇒ (*'a* ⇒ *'a*) ⇒ *bool*
where *galois l u* ↔ (∀ *x y* . *l*(*x*) ≤ *y* ↔ *x* ≤ *u*(*y*))

lemma *galois-char*: *galois l u* ↔ (∀ *x* . *x* ≤ *u*(*l*(*x*))) ∧ (∀ *x* . *l*(*u*(*x*)) ≤ *x*) ∧ *isotone l* ∧ *isotone u*
apply (*rule iffI*)
apply (*smt galois-def isotone-def order-refl order-trans*)
by (*metis galois-def isotone-def order-trans*)

lemma *galois-closure*: *galois l u* → *l*(*x*) = *l*(*u*(*l*(*x*))) ∧ *u*(*x*) = *u*(*l*(*u*(*x*)))
by (*smt antisym galois-char isotone-def*)

lemma *mu-fusion-1*: *galois l u* ∧ *isotone h* ∧ *has-least-prefixpoint g* ∧ *has-least-fixpoint h* ∧ *l*(*g*(*u*(*μ h*))) ≤ *h*(*l*(*u*(*μ h*))) → *l*(*pμ g*) ≤ *μ h*
proof
assume *1*: *galois l u* ∧ *isotone h* ∧ *has-least-prefixpoint g* ∧ *has-least-fixpoint h* ∧ *l*(*g*(*u*(*μ h*))) ≤ *h*(*l*(*u*(*μ h*)))
hence *l*(*u*(*μ h*)) ≤ *μ h*
by (*metis galois-char*)
hence *g*(*u*(*μ h*)) ≤ *u*(*μ h*) **using** *1*
by (*smt galois-def least-fixpoint-same least-fixpoint-unique is-least-fixpoint-def isotone-def order-trans*)
thus *l*(*pμ g*) ≤ *μ h* **using** *1*
by (*metis galois-def least-prefixpoint is-least-prefixpoint-def*)
qed

lemma *mu-fusion-2*: *galois l u* ∧ *isotone h* ∧ *has-least-prefixpoint g* ∧ *has-least-fixpoint h* ∧ *l* ∘ *g* ≤≤ *h* ∘ *l* → *l*(*pμ g*) ≤ *μ h*
by (*metis lifted-less-eq-def mu-fusion-1 o-apply*)

lemma *mu-fusion-equal-1*: *galois l u* ∧ *isotone g* ∧ *isotone h* ∧ *has-least-prefixpoint g* ∧ *has-least-fixpoint h* ∧ *l*(*g*(*u*(*μ h*))) ≤ *h*(*l*(*u*(*μ h*))) ∧ *l*(*g*(*pμ g*)) = *h*(*l*(*pμ g*)) → *μ h* = *l*(*pμ g*) ∧ *μ h* = *l*(*μ 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 (*smt 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 (*smt 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 $\nu\ h \leq u(l(\nu\ h))$
by (*metis galois-char*)
hence $l(\nu\ h) \leq g(l(\nu\ h))$ **using** *1*
by (*smt galois-def 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 (*metis galois-def greatest-postfixpoint is-greatest-postfixpoint-def*)
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 (*smt 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* $(g \circ h)$ $\wedge$ $l \circ h \circ g \leq\leq g \circ h \circ l \rightarrow \mu(l \circ h) \leq \mu(g \circ h)$
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 o-assoc*)

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* $(g \circ h)$ $\wedge$ $g \circ h \circ u \leq\leq u \circ h \circ g \rightarrow \nu(g \circ h) \leq \nu(u \circ h)$
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 o-assoc*)

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* $\wedge$ *isotone g* $\wedge$ *has-least-fixpoint f* $\wedge$ *has-least-fixpoint g* $\wedge$ *has-least-fixpoint* $(f \circ g)$ $\wedge$ $f \circ g = g \circ f \rightarrow (\mu(f \circ g) = \mu\ f \rightarrow \mu\ g \leq \mu\ f)$
by (*metis is-least-fixpoint-def least-fixpoint-same least-fixpoint-unique mu-roll*)

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

lemma *nu-commute-fixpoint-2*: *isotone g* $\wedge$ *has-greatest-fixpoint* $(f \circ g)$ $\wedge$ $f \circ g = g \circ f \rightarrow$ *is-fixpoint g* $(\nu(f \circ g))$
by (*metis is-fixpoint-def nu-roll*)

lemma *nu-commute-greatest-fixpoint*: *isotone f* $\wedge$ *isotone g* $\wedge$ *has-greatest-fixpoint f* $\wedge$ *has-greatest-fixpoint g* $\wedge$ *has-greatest-fixpoint* $(f \circ g)$ $\wedge$ $f \circ g = g \circ f \rightarrow (\nu(f \circ g) = \nu\ f \rightarrow \nu\ f \leq \nu\ g)$
by (*smt is-greatest-fixpoint-def greatest-fixpoint-same greatest-fixpoint-unique nu-roll*)

lemma *mu-diagonal-1*: *isotone* $(\lambda x\ .\ f\ x\ x)$ $\wedge$ $(\forall x\ .$ *isotone* $(\lambda y\ .\ f\ x\ y))$ $\wedge$ *isotone* $(\lambda x\ .\ \mu(\lambda y\ .\ f\ x\ y))$ $\wedge$ $(\forall x\ .$ *has-least-fixpoint* $(\lambda y\ .\ f\ x\ y))$ $\wedge$ *has-least-prefixpoint* $(\lambda x\ .\ \mu(\lambda y\ .\ f\ x\ y)) \rightarrow \mu(\lambda x\ .\ f\ x\ x) = \mu(\lambda x\ .\ \mu(\lambda y\ .\ f\ x\ y))$
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*: $(\forall x\ .$ *isotone* $(\lambda y\ .\ f\ x\ y)$ $\wedge$ *isotone* $(\lambda y\ .\ f\ y\ x)$ $\wedge$ *has-least-prefixpoint* $(\lambda y\ .\ f\ x\ y))$ $\wedge$ *has-least-prefixpoint* $(\lambda x\ .\ \mu(\lambda y\ .\ f\ x\ y)) \rightarrow \mu(\lambda x\ .\ f\ x\ x) = \mu(\lambda x\ .\ \mu(\lambda y\ .\ f\ x\ y))$
by (*smt is-least-fixpoint-def is-least-prefixpoint-def isotone-def least-fixpoint-same least-prefixpoint least-prefixpoint-fixpoint*)

lemma *nu-diagonal-1*: *isotone* $(\lambda x\ .\ f\ x\ x)$ $\wedge$ $(\forall x\ .$ *isotone* $(\lambda y\ .\ f\ x\ y))$ $\wedge$ *isotone* $(\lambda x\ .\ \nu(\lambda y\ .\ f\ x\ y))$ $\wedge$ $(\forall x\ .$ *has-greatest-fixpoint* $(\lambda y\ .\ f\ x\ y))$ $\wedge$ *has-greatest-postfixpoint* $(\lambda x\ .\ \nu(\lambda y\ .\ f\ x\ y)) \rightarrow \nu(\lambda x\ .\ f\ x\ x) = \nu(\lambda x\ .\ \nu(\lambda y\ .\ f\ x\ y))$
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*: $(\forall\, x\ .\ \mathit{isotone}\ (\lambda y\ .\ f\ x\ y) \wedge \mathit{isotone}\ (\lambda y\ .\ f\ y\ x) \wedge \mathit{has\text{-}greatest\text{-}postfixpoint}\ (\lambda y\ .\ f\ x\ y)) \wedge \mathit{has\text{-}greatest\text{-}postfixpoint}\ (\lambda x\ .\ \nu(\lambda y\ .\ f\ x\ y)) \rightarrow \nu(\lambda x\ .\ f\ x\ x) = \nu(\lambda x\ .\ \nu(\lambda y\ .\ f\ x\ y))$
using [[*smt-timeout* = *120*]] **by** (*smt is-greatest-fixpoint-def is-greatest-postfixpoint-def isotone-def greatest-fixpoint-same greatest-postfixpoint greatest-postfixpoint-fixpoint*)

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*: $0 = (\mathit{THE}\ x\ .\ (\forall\, y\ .\ x = -y\ ;\ --y))$ — 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* :: $'a \Rightarrow \mathit{bool}$
where *is-zero-def*: $\mathit{is\text{-}zero}(x) \equiv (\forall\, y\ .\ x = -y\ ;\ --y)$

lemma *the-zero-def-p*: $0 = (\mathit{THE}\ x\ .\ \mathit{is\text{-}zero}(x))$
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-idempotent*: $-x + -x = -x$
by (*metis double-negation mult-idempotent plus-def*)

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

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

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

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

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

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

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

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

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

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

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 *compl-anti*: $-x \leq -y \rightarrow --y \leq --x$
by (*metis 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 *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 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 *mult-distr-plus-right*: $(-y + -z)\ ;\ -x = (-y\ ;\ -x) + (-z\ ;\ -x)$
by (*metis mult-distr-plus plus-def 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*)

end

class *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-dist-add* : $x\ ;\ (y + z) = x\ ;\ y + x\ ;\ z$
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*: $x \leq y \longrightarrow x + z \leq y + z$
  by (*smt add-associative add-commutative add-idempotent less-eq-def*)

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

lemma *add-isotone*: $w \leq y \wedge x \leq z \longrightarrow w + x \leq y + z$
  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*: $x \leq y \longrightarrow x \mathbin{;} z \leq y \mathbin{;} z$
  by (*metis less-eq-def mult-right-dist-add*)

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

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

lemma *mult-left-subdist-add-left*: $x \mathbin{;} y \leq x \mathbin{;} (y + z)$
  by (*metis add-left-upper-bound mult-left-dist-add*)

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

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

lemma *mult-right-subdist-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-least-upper-bound mult-left-isotone mult-left-one mult-right-dist-add order-trans*)

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 *case-split-right*: $1 \leq w + z \wedge x\ ;\ w \leq y \wedge x\ ;\ z \leq y \rightarrow x \leq y$
by (*smt add-least-upper-bound mult-right-isotone mult-right-one mult-left-dist-add order-trans*)

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

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

end

class *semiring-T* = *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 *itering-0* = *semiring* +
fixes *circ* :: $'a \Rightarrow 'a$ ($-^\circ$ [*100*] *100*)
assumes *circ-mult*: $(x\ ;\ y)^\circ = 1 + x\ ;\ (y\ ;\ x)^\circ\ ;\ y$
assumes *circ-add*: $(x + y)^\circ = (x^\circ\ ;\ y)^\circ\ ;\ x^\circ$

begin

lemma *circ-isotone*: $x \leq y \rightarrow x^\circ \leq y^\circ$
by (*metis add-left-divisibility circ-add circ-mult mult-left-one mult-right-subdist-add-left*)

lemma *circ-slide*: $x\ ;\ (y\ ;\ x)^\circ = (x\ ;\ y)^\circ\ ;\ x$
by (*smt circ-mult mult-associative mult-left-dist-add mult-left-one mult-right-dist-add mult-right-one*)

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

lemma *circ-left-unfold*: $1 + x\ ;\ x^\circ = x^\circ$
by (*metis circ-mult circ-slide mult-associative 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-right-unfold mult-left-one mult-right-subdist-add-left order-trans*)

lemma *circ-transitive-equal*: $x^\circ \ ; \ x^\circ = x^\circ$
by (*metis add-idempotent circ-add circ-mult mult-associative mult-left-one mult-right-one*)

lemma *circ-circ-circ*: $x^{\circ\circ\circ} = x^{\circ\circ}$
by (*metis add-idempotent circ-add 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-1*: $(x + y)^\circ = x^\circ \ ; \ (y \ ; \ x^\circ)^\circ$
by (*metis circ-add circ-slide*)

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

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

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

lemma *mult-zero-add-circ-2*: $(x + y \ ; \ 0)^\circ = x^\circ + x^\circ \ ; \ y \ ; \ 0$
by (*metis mult-associative mult-left-dist-add mult-right-one mult-zero-add-circ mult-zero-circ*)

lemma *circ-plus-same*: $x^\circ \ ; \ x = x \ ; \ x^\circ$
by (*metis circ-slide mult-left-one mult-right-one*)

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

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

lemma *circ-unfold-sum*: $(x + y)^\circ = x^\circ + x^\circ \ ; \ y \ ; \ (x + y)^\circ$
by (*smt circ-add-1 circ-mult mult-associative mult-left-dist-add mult-right-one*)

lemma *circ-loop-fixpoint*: $y \ ; \ (y^\circ \ ; \ z) + z = y^\circ \ ; \ z$
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 circ-plus-same left-plus-below-circ*)

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*: $x \leq z^\circ \wedge y \leq z^\circ \rightarrow x \mathbin{;} y \leq z^\circ$
by (*metis mult-isotone circ-transitive-equal*)

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

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

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

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

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

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

lemma *circ-decompose-4*: $(x^\circ \mathbin{;} y^\circ)^\circ = x^\circ \mathbin{;} (y^\circ \mathbin{;} x^\circ)^\circ$
by (*smt circ-add circ-increasing circ-plus-one circ-slide circ-transitive-equal less-eq-def mult-associative mult-left-subdist-add-left mult-right-one*)

lemma *circ-decompose-5*: $(x^\circ \mathbin{;} y^\circ)^\circ = (y^\circ \mathbin{;} x^\circ)^\circ$
by (*metis add-associative add-commutative circ-add circ-decompose-4 circ-slide circ-zero mult-right-one*)

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

lemma *circ-decompose-7*: $(x + y)^\circ = x^\circ \mathbin{;} y^\circ \mathbin{;} (x + y)^\circ$
by (*metis add-commutative circ-add circ-slide circ-transitive-equal mult-associative*)

lemma *circ-decompose-8*: $(x + y)^\circ = (x + y)^\circ \mathbin{;} x^\circ \mathbin{;} y^\circ$
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*: $(x^\circ \mathbin{;} y^\circ)^\circ = x^\circ \mathbin{;} y^\circ \mathbin{;} (x^\circ \mathbin{;} y^\circ)^\circ$
by (*metis circ-decompose-4 mult-associative*)

lemma *circ-decompose-10*: $(x^\circ \mathbin{;} y^\circ)^\circ = (x^\circ \mathbin{;} y^\circ)^\circ \mathbin{;} x^\circ \mathbin{;} y^\circ$
by (*metis circ-decompose-9 circ-plus-same mult-associative*)

lemma *circ-add-mult-zero*: $x^\circ \ ; \ y = (x + y \ ; \ 0)^\circ \ ; \ y$
by (*smt mult-associative mult-left-zero mult-right-one circ-add circ-slide circ-zero*)

lemma *circ-back-loop-fixpoint*: $(z \ ; \ y^\circ) \ ; \ y + z = z \ ; \ y^\circ$
by (*metis add-commutative mult-associative mult-left-dist-add mult-right-one circ-plus-same circ-left-unfold*)

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-fixpoint* $(\lambda x \ . \ x \ ; \ y + z) \ (z \ ; \ y^\circ)$
by (*metis circ-back-loop-fixpoint is-fixpoint-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 *right-plus-circ*: $(x^\circ \ ; \ x)^\circ = x^\circ$
by (*metis add-idempotent circ-add circ-mult circ-right-unfold circ-slide*)

lemma *left-plus-circ*: $(x \ ; \ x^\circ)^\circ = x^\circ$
by (*metis circ-plus-same right-plus-circ*)

lemma *circ-add-sub-add-one*: $x \ ; \ x^\circ \ ; \ (x + y) \leq x \ ; \ x^\circ \ ; \ (1 + y)$
by (*smt add-least-upper-bound add-left-upper-bound add-right-upper-bound circ-plus-same mult-associative mult-isotone mult-left-dist-add mult-right-one right-plus-below-circ*)

lemma *circ-elimination*: $x \ ; \ y = 0 \rightarrow x \ ; \ y^\circ \leq x$
by (*metis add-left-zero circ-back-loop-fixpoint circ-plus-same le-less mult-associative mult-left-zero*)

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*)

end

class *itering-1* = *itering-0* +
assumes *circ-simulate*: $z \ ; \ x \leq y \ ; \ z \rightarrow z \ ; \ x^\circ \leq y^\circ \ ; \ z$

begin

lemma *circ-circ-mult*: $1^\circ \ ; \ x^\circ = x^{\circ\circ}$
by (*metis antisym circ-circ-add circ-reflexive circ-simulate circ-sub-dist-3 circ-sup-one-left-unfold circ-transitive-equal mult-left-one order-refl*)

lemma *sub-mult-one-circ*: $x \ ; \ 1^\circ \leq 1^\circ \ ; \ x$

by (*metis circ-simulate mult-left-one mult-right-one order-refl*)

end

class *itering-2* = *itering-1* +
assumes *circ-simulate-right*: $z\ ;\ x \leq y\ ;\ z + w \rightarrow z\ ;\ x^\circ \leq y^\circ\ ;\ (z + w\ ;\ x^\circ)$
assumes *circ-simulate-left*: $x\ ;\ z \leq z\ ;\ y + w \rightarrow x^\circ\ ;\ z \leq (z + x^\circ\ ;\ w)\ ;\ y^\circ$

begin

lemma *circ-simulate-right-1*: $z\ ;\ x \leq y\ ;\ z \rightarrow z\ ;\ x^\circ \leq y^\circ\ ;\ z$
by (*metis add-right-zero circ-simulate-right mult-left-zero*)

lemma *circ-simulate-left-1*: $x\ ;\ z \leq z\ ;\ y \rightarrow x^\circ\ ;\ z \leq z\ ;\ y^\circ + x^\circ\ ;\ 0$
by (*smt add-right-zero circ-simulate-left mult-associative mult-left-zero mult-right-dist-add*)

lemma *circ-simulate-1*: $y\ ;\ x \leq x\ ;\ y \rightarrow y^\circ\ ;\ x^\circ \leq x^\circ\ ;\ y^\circ$
by (*smt add-associative add-right-zero circ-loop-fixpoint circ-simulate circ-simulate-left-1 mult-associative mult-left-zero mult-zero-add-circ-2*)

lemma *circ-separate-1*: $y\ ;\ x \leq x\ ;\ y \rightarrow (x + y)^\circ = x^\circ\ ;\ y^\circ$
proof −
have $y\ ;\ x \leq x\ ;\ y \rightarrow y^\circ\ ;\ x\ ;\ y^\circ \leq x\ ;\ y^\circ + y^\circ\ ;\ 0$
by (*smt circ-simulate-left-1 circ-transitive-equal mult-associative mult-left-isotone mult-left-zero mult-right-dist-add*)
thus *?thesis*
by (*smt add-commutative circ-add-1 circ-simulate-right circ-sub-dist-3 less-eq-def mult-associative mult-left-zero zero-right-mult-decreasing*)
qed

lemma *circ-circ-mult-1*: $x^\circ\ ;\ 1^\circ = x^{\circ\circ}$
by (*metis add-commutative circ-circ-add circ-separate-1 mult-left-one mult-right-one order-refl*)

lemma *atomicity-refinement*: $s = s\ ;\ q \wedge x = q\ ;\ x \wedge q\ ;\ b = 0 \wedge r\ ;\ b \leq b\ ;\ r \wedge r\ ;\ l \leq l\ ;\ r \wedge x\ ;\ l \leq l\ ;\ x \wedge b\ ;\ l \leq l\ ;\ b \wedge q\ ;\ l \leq l\ ;\ q \wedge r^\circ\ ;\ q \leq q\ ;\ r^\circ \wedge q \leq 1 \rightarrow s\ ;\ (x + b + r + l)^\circ\ ;\ q \leq s\ ;\ (x\ ;\ b^\circ\ ;\ q + r + l)^\circ$
proof
assume *1*: $s = s\ ;\ q \wedge x = q\ ;\ x \wedge q\ ;\ b = 0 \wedge r\ ;\ b \leq b\ ;\ r \wedge r\ ;\ l \leq l\ ;\ r \wedge x\ ;\ l \leq l\ ;\ x \wedge b\ ;\ l \leq l\ ;\ b \wedge q\ ;\ l \leq l\ ;\ q \wedge r^\circ\ ;\ q \leq q\ ;\ r^\circ \wedge q \leq 1$
hence $s\ ;\ (x + b + r + l)^\circ\ ;\ q = s\ ;\ l^\circ\ ;\ (x + b + r)^\circ\ ;\ q$
using [[*smt-timeout = 120*]] **by** (*smt add-commutative add-least-upper-bound circ-separate-1 mult-associative mult-left-subdist-add-right mult-right-dist-add order-trans*)
also have ... $= s\ ;\ l^\circ\ ;\ b^\circ\ ;\ r^\circ\ ;\ q\ ;\ (x\ ;\ b^\circ\ ;\ r^\circ\ ;\ q)^\circ$ **using** *1*
by (*smt add-associative add-commutative circ-add-1 circ-separate-1 circ-slide mult-associative*)
also have ... $\leq s\ ;\ l^\circ\ ;\ b^\circ\ ;\ r^\circ\ ;\ q\ ;\ (x\ ;\ b^\circ\ ;\ q\ ;\ r^\circ)^\circ$ **using** *1*
by (*metis circ-isotone mult-associative mult-right-isotone*)
also have ... $\leq s\ ;\ q\ ;\ l^\circ\ ;\ b^\circ\ ;\ r^\circ\ ;\ (x\ ;\ b^\circ\ ;\ q\ ;\ r^\circ)^\circ$ **using** *1*
by (*metis mult-left-isotone mult-right-isotone mult-right-one*)
also have ... $\leq s\ ;\ l^\circ\ ;\ q\ ;\ b^\circ\ ;\ r^\circ\ ;\ (x\ ;\ b^\circ\ ;\ q\ ;\ r^\circ)^\circ$ **using** *1*
by (*metis circ-simulate mult-associative mult-left-isotone mult-right-isotone*)
also have ... $\leq s\ ;\ l^\circ\ ;\ r^\circ\ ;\ (x\ ;\ b^\circ\ ;\ q\ ;\ r^\circ)^\circ$ **using** *1*
by (*metis add-left-zero circ-back-loop-fixpoint circ-plus-same mult-associative mult-left-zero mult-left-isotone mult-right-isotone mult-right-one*)
also have ... $\leq s\ ;\ (x\ ;\ b^\circ\ ;\ q + r + l)^\circ$ **using** *1*

by (*metis add-commutative circ-add-1 circ-sub-dist-3 mult-associative mult-right-isotone*)
finally show $s \ ; (x + b + r + l)^\circ \ ; q \leq s \ ; (x \ ; b^\circ \ ; q + r + l)^\circ$ **.**
qed

end

class *itering-3* = *itering-2* +
assumes *circ-simulate-right-plus*: $z \ ; x \leq y \ ; y^\circ \ ; z + w \rightarrow z \ ; x^\circ \leq y^\circ \ ; (z + w \ ; x^\circ)$
assumes *circ-simulate-left-plus*: $x \ ; z \leq z \ ; y^\circ + w \rightarrow x^\circ \ ; z \leq (z + x^\circ \ ; w) \ ; y^\circ$

begin

lemma *circ-simulate-right-plus-1*: $z \ ; x \leq y \ ; y^\circ \ ; z \rightarrow z \ ; x^\circ \leq y^\circ \ ; z$
by (*metis add-right-zero circ-simulate-right-plus mult-left-zero*)

lemma *circ-simulate-left-plus-1*: $x \ ; z \leq z \ ; y^\circ \rightarrow x^\circ \ ; z \leq z \ ; y^\circ + x^\circ \ ; 0$
by (*smt add-right-zero circ-simulate-left-plus mult-associative mult-left-zero mult-right-dist-add*)

lemma *circ-simulate-2*: $y \ ; x^\circ \leq x^\circ \ ; y^\circ \leftrightarrow y^\circ \ ; x^\circ \leq x^\circ \ ; y^\circ$
by (*rule iffI*,
smt add-associative add-right-zero circ-loop-fixpoint circ-simulate-left-plus-1 mult-associative mult-left-zero mult-zero-add-circ-2,
metis circ-increasing mult-left-isotone order-trans)

lemma *circ-simulate-absorb*: $y \ ; x \leq x \rightarrow y^\circ \ ; x \leq x + y^\circ \ ; 0$
by (*metis circ-simulate-left-plus-1 circ-zero mult-right-one*)

lemma *circ-simulate-3*: $y \ ; x^\circ \leq x^\circ \rightarrow y^\circ \ ; x^\circ \leq x^\circ \ ; y^\circ$
by (*metis add-least-upper-bound circ-reflexive circ-simulate-2 less-eq-def mult-right-isotone mult-right-one*)

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-simulate-right-plus-1 mult-left-one mult-right-isotone mult-right-one*)

lemma *circ-separate-mult-1*: $y \ ; x \leq x \ ; y \rightarrow (x \ ; y)^\circ \leq x^\circ \ ; y^\circ$
by (*metis circ-mult-sub-add circ-separate-1*)

lemma *circ-extra-circ*: $(y \ ; x^\circ)^\circ = (y \ ; y^\circ \ ; x^\circ)^\circ$
proof −
have $y \ ; y^\circ \ ; x^\circ \leq y \ ; x^\circ \ ; (y \ ; x^\circ)^\circ$
by (*metis add-commutative circ-add-1 circ-sub-dist-3 mult-associative mult-right-isotone*)
hence $(y \ ; y^\circ \ ; x^\circ)^\circ \leq (y \ ; x^\circ)^\circ$
by (*metis circ-simulate-right-plus-1 mult-left-one mult-right-one*)
thus *?thesis*
by (*metis antisym circ-back-loop-fixpoint circ-isotone mult-right-subdist-add-right*)
qed

lemma *circ-separate-unfold*: $(y \ ; x^\circ)^\circ = y^\circ + y^\circ \ ; y \ ; x \ ; x^\circ \ ; (y \ ; x^\circ)^\circ$
by (*smt add-commutative circ-add circ-left-unfold circ-loop-fixpoint mult-associative mult-left-dist-add mult-right-one*)

lemma *separation*: $y \ ; \ x \leq x \ ; \ y^\circ \rightarrow (x + y)^\circ = x^\circ \ ; \ y^\circ$
proof −
have $y \ ; \ x \leq x \ ; \ y^\circ \rightarrow y^\circ \ ; \ x \ ; \ y^\circ \leq x \ ; \ y^\circ + y^\circ \ ; \ 0$
by (*smt circ-simulate-left-plus-1 circ-transitive-equal mult-associative mult-left-isotone mult-left-zero mult-right-dist-add*)
thus *?thesis*
by (*smt add-commutative circ-add-1 circ-simulate-right circ-sub-dist-3 less-eq-def mult-associative mult-left-zero zero-right-mult-decreasing*)
qed

lemma *circ-simulate-4*: $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (1 + y) \rightarrow y^\circ \ ; \ x^\circ \leq x^\circ \ ; \ y^\circ$
proof −
have $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (1 + y) \rightarrow (1 + y) \ ; \ x \leq x \ ; \ x^\circ \ ; \ (1 + y)$
by (*smt add-associative add-commutative add-left-upper-bound circ-back-loop-fixpoint less-eq-def mult-left-dist-add mult-left-one mult-right-dist-add mult-right-one*)
thus *?thesis*
by (*smt add-least-upper-bound circ-increasing circ-reflexive circ-simulate-2 circ-simulate-right-plus-1 mult-right-isotone mult-right-subdist-add-right order-trans*)
qed

lemma *circ-simulate-5*: $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (x + y) \rightarrow y^\circ \ ; \ x^\circ \leq x^\circ \ ; \ y^\circ$
by (*metis circ-add-sub-add-one circ-simulate-4 order-trans*)

lemma *circ-simulate-6*: $y \ ; \ x \leq x \ ; \ (x + y) \rightarrow y^\circ \ ; \ x^\circ \leq x^\circ \ ; \ y^\circ$
by (*metis add-commutative circ-back-loop-fixpoint circ-simulate-5 mult-right-subdist-add-left order-trans*)

lemma *circ-separate-4*: $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (1 + y) \rightarrow (x + y)^\circ = x^\circ \ ; \ y^\circ$
proof
assume *1*: $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (1 + y)$
hence $y \ ; \ x \ ; \ x^\circ \leq x \ ; \ x^\circ + x \ ; \ x^\circ \ ; \ y \ ; \ x^\circ$
by (*smt circ-transitive-equal less-eq-def mult-associative mult-left-dist-add mult-right-dist-add mult-right-one*)
also have $\ldots \leq x \ ; \ x^\circ + x \ ; \ x^\circ \ ; \ x^\circ \ ; \ y^\circ$ **using** *1*
by (*metis add-right-isotone circ-simulate-2 circ-simulate-4 mult-associative mult-right-isotone*)
finally have $y \ ; \ x \ ; \ x^\circ \leq x \ ; \ x^\circ \ ; \ y^\circ$
by (*metis circ-reflexive circ-transitive-equal less-eq-def mult-associative mult-right-isotone mult-right-one*)
hence $y^\circ \ ; \ (y^\circ \ ; \ x)^\circ \leq x^\circ \ ; \ (y^\circ + y^\circ \ ; \ 0 \ ; \ (y^\circ \ ; \ x)^\circ)$
by (*smt add-right-upper-bound circ-back-loop-fixpoint circ-simulate-left-plus-1 circ-simulate-right-plus circ-transitive-equal mult-associative order-trans*)
thus $(x + y)^\circ = x^\circ \ ; \ y^\circ$
by (*smt add-commutative antisym circ-add-1 circ-slide circ-sub-dist-3 circ-transitive-equal less-eq-def mult-associative mult-left-zero mult-right-subdist-add-right zero-right-mult-decreasing*)
qed

lemma *circ-separate-5*: $y \ ; \ x \leq x \ ; \ x^\circ \ ; \ (x + y) \rightarrow (x + y)^\circ = x^\circ \ ; \ y^\circ$
by (*metis circ-add-sub-add-one circ-separate-4 order-trans*)

lemma *circ-separate-6*: $y \ ; \ x \leq x \ ; \ (x + y) \rightarrow (x + y)^\circ = x^\circ \ ; \ y^\circ$
by (*metis add-commutative circ-back-loop-fixpoint circ-separate-5 mult-right-subdist-add-left order-trans*)

end

class *itering-T* = *semiring-T* + *itering-3*

begin

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

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

end

class *itering-L* = *itering-3* +
fixes $L :: 'a$
assumes *L-def*: $L = 1^\circ \ ; \ 0$

begin

lemma *one-circ-split*: $1^\circ = L + 1$
by (*metis L-def add-commutative antisym circ-add-upper-bound circ-reflexive circ-simulate-absorb mult-right-one order-refl zero-right-mult-decreasing*)

lemma *one-circ-mult-split*: $1^\circ \ ; \ x = L + x$
by (*metis L-def add-commutative circ-loop-fixpoint mult-associative mult-left-zero mult-zero-circ one-circ-split*)

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

lemma *one-circ-mult-split-2*: $1^\circ \ ; \ x = x \ ; \ 1^\circ + L$
by (*smt L-def add-associative add-commutative circ-left-unfold less-eq-def mult-left-subdist-add-left mult-right-one 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 (*smt L-def mult-associative mult-left-isotone mult-right-dist-add sub-mult-one-circ-split-2*)

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 add-idempotent one-circ-mult-split*)

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

lemma *circ-left-induct-mult-L*: $L \leq x \rightarrow x \ ; \ y \leq x \rightarrow x \ ; \ y^\circ \leq x$
by (*metis circ-one circ-simulate less-eq-def one-circ-mult-split*)

lemma *circ-left-induct-mult-iff-L*: $L \leq x \rightarrow x \ ; \ y \leq x \leftrightarrow x \ ; \ y^\circ \leq x$
by (*smt add-least-upper-bound circ-back-loop-fixpoint circ-left-induct-mult-L less-eq-def*)

lemma *circ-left-induct-L*: $L \leq x \rightarrow x \ ; \ y + z \leq x \rightarrow z \ ; \ y^\circ \leq x$
by (*metis add-least-upper-bound circ-left-induct-mult-L less-eq-def mult-right-dist-add*)

lemma *mult-L-circ*: $(x \ ; \ L)^\circ = 1 + x \ ; \ L$
by (*metis L-left-zero circ-mult 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 add-left-upper-bound one-circ-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 *kleene-algebra* = *semiring* +
fixes *star* :: $'a \Rightarrow 'a$ (-* [100] 100)
assumes *star-left-unfold* : $1 + y \ ; \ y^* \leq y^*$
assumes *star-left-induct* : $z + y \ ; \ x \leq x \rightarrow y^* \ ; \ z \leq x$
assumes *star-right-induct*: $z + x \ ; \ y \leq x \rightarrow z \ ; \ y^* \leq x$

begin

lemma *star-left-unfold-equal*: $1 + x \ ; \ x^* = x^*$
by (*smt add-least-upper-bound antisym-conv mult-right-isotone mult-right-one order-refl order-trans star-left-induct star-left-unfold*)

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

lemma *star-decompose*: $(x + y)^* = (x^*\ ;\ y)^*\ ;\ x^*$
apply (*rule antisym*)
apply (*smt add-least-upper-bound add-left-upper-bound add-right-upper-bound mult-associative mult-left-dist-add mult-left-one mult-right-dist-add mult-right-one star-left-induct star-left-unfold-equal*)
by (*smt add-least-upper-bound less-eq-def mult-left-subdist-add-left mult-right-one star-left-induct star-left-unfold star-right-induct*)

lemma *star-simulation-right*: $z\ ;\ x \leq y\ ;\ z \rightarrow z\ ;\ x^* \leq y^*\ ;\ z$
by (*smt add-least-upper-bound less-eq-def mult-associative mult-left-dist-add mult-left-one mult-right-subdist-add-left star-left-induct star-left-unfold star-right-induct*)

end

sublocale *kleene-algebra* $<$ *star*!: *itering-1* **where** *circ* $=$ *star*
by (*unfold-locales*, *metis star-unfold-slide*, *metis star-decompose*, *metis star-simulation-right*)

context *kleene-algebra*

begin

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

lemma *star-sub-one*: $x \leq 1 \rightarrow x^* = 1$
by (*metis add-least-upper-bound eq-iff mult-left-one star.circ-reflexive star-right-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 (*smt 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-subdist-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 (*smt add-commutative less-eq-def mult-left-dist-add mult-left-one mult-right-dist-add mult-right-one star-involutive star.circ-plus-one star.circ-sub-dist star.left-plus-below-circ*)

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*: $x \mathbin{;} z \leq z \mathbin{;} y \rightarrow x^* \mathbin{;} z \leq z \mathbin{;} y^*$
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-subdist-add-left*)

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *star-loop-is-least-fixpoint*: *is-least-fixpoint* $(\lambda x \,.\, y \mathbin{;} x + z)$ $(y^* \mathbin{;} z)$
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*)

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

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 mult-left-dist-add mult-right-dist-add mult-right-one star.circ-decompose-5 star-involutive star-one star.circ-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 $y^*\ ;\ (z + w\ ;\ x^*)\ ;\ x \leq y^*\ ;\ z\ ;\ x + y^*\ ;\ w\ ;\ x^*$
by (*smt add-left-upper-bound add-right-isotone mult-associative mult-left-dist-add mult-right-dist-add star.circ-back-loop-fixpoint*)
also have $\ldots \leq y^*\ ;\ y\ ;\ z + y^*\ ;\ w + y^*\ ;\ w\ ;\ x^*$ **using** *1*
by (*smt add-left-isotone less-eq-def mult-associative mult-left-dist-add star.circ-plus-same star.circ-transitive-equal*)
also have $\ldots \leq y^*\ ;\ (z + w\ ;\ x^*)$
by (*smt add-associative add-idempotent add-left-isotone mult-associative mult-left-dist-add mult-right-one mult-right-subdist-add-right star.circ-plus-one star.circ-plus-same star.circ-transitive-equal star.circ-unfold-sum*)
finally show $z\ ;\ x^* \leq y^*\ ;\ (z + w\ ;\ x^*)$
by (*smt add-least-upper-bound add-right-divisibility star.circ-loop-fixpoint star-right-induct*)
qed

lemma *star-circ-simulate-left-plus*: $x\ ;\ z \leq z\ ;\ y^* + w \rightarrow x^*\ ;\ z \leq (z + x^*\ ;\ w)\ ;\ y^*$
proof
assume *1*: $x\ ;\ z \leq z\ ;\ y^* + w$
have $x\ ;\ ((z + x^*\ ;\ w)\ ;\ y^*) \leq x\ ;\ z\ ;\ y^* + x^*\ ;\ w\ ;\ y^*$
by (*smt add-right-isotone mult-associative mult-left-dist-add mult-right-dist-add mult-right-subdist-add-left star.circ-loop-fixpoint*)
also have $\ldots \leq (z + w + x^*\ ;\ w)\ ;\ y^*$ **using** *1*
by (*smt add-left-divisibility add-left-isotone mult-associative mult-right-dist-add star.circ-transitive-equal*)
also have $\ldots = (z + x^*\ ;\ w)\ ;\ y^*$
by (*metis add-associative add-right-upper-bound less-eq-def star.circ-loop-fixpoint*)
finally show $x^*\ ;\ z \leq (z + x^*\ ;\ w)\ ;\ y^*$

by (*metis add-least-upper-bound mult-left-subdist-add-left mult-right-one star.circ-right-unfold star-left-induct*)
qed

end

sublocale *kleene-algebra* < *star*!: *itering-3* **where** *circ* = *star*
apply *unfold-locales*
apply (*smt add-associative add-commutative add-left-upper-bound mult-associative mult-left-dist-add order-trans star.circ-loop-fixpoint star-circ-simulate-right-plus*)
apply (*smt add-commutative add-right-isotone mult-right-isotone order-trans star.circ-increasing star-circ-simulate-left-plus*)
apply (*rule star-circ-simulate-right-plus*)
by (*rule star-circ-simulate-left-plus*)

class *kleene-itering* = *kleene-algebra* + *itering-3*

begin

lemma *star-below-circ*: $x^* \leq x^\circ$
by (*metis circ-right-unfold mult-left-one order-refl star-right-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 circ-slide mult-left-one mult-right-one star-mult-circ star-simulation-right-equal*)

lemma *circ-star*: $x^{\circ *} = x^\circ$
by (*metis circ-reflexive circ-transitive-equal less-eq-def mult-left-one star-one star-simulation-right-equal star-square*)

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-subdist-add-left mult-right-isotone mult-right-one star.circ-isotone*)

lemma *circ-isolate*: $x^\circ = x^\circ \ ; \ 0 + x^*$
by (*metis add-commutative antisym circ-add-upper-bound circ-mult-star circ-simulate-absorb star.left-plus-below-circ star-below-circ zero-right-mult-decreasing*)

lemma *circ-isolate-mult*: $x^\circ \ ; \ y = x^\circ \ ; \ 0 + x^* \ ; \ y$
by (*metis circ-isolate mult-associative mult-left-zero mult-right-dist-add*)

lemma *circ-isolate-mult-sub*: $x^\circ \ ; \ y \leq x^\circ + x^* \ ; \ y$
by (*metis add-left-isotone circ-isolate-mult zero-right-mult-decreasing*)

lemma *circ-sub-decompose*: $(x^\circ \ ; \ y)^\circ \leq (x^* \ ; \ y)^\circ \ ; \ x^\circ$

by (*smt add-commutative add-least-upper-bound add-right-upper-bound circ-back-loop-fixpoint circ-isolate-mult mult-zero-add-circ-2 zero-right-mult-decreasing*)

lemma *circ-add-4*: $(x + y)^\circ = (x^* \ ; \ y)^\circ \ ; \ x^\circ$
by (*smt antisym circ-add circ-isotone circ-sub-decompose circ-transitive-equal mult-associative mult-left-isotone star-below-circ*)

lemma *circ-add-5*: $(x^\circ \ ; \ y)^\circ \ ; \ x^\circ = (x^* \ ; \ y)^\circ \ ; \ x^\circ$
by (*metis circ-add circ-add-4*)

lemma *plus-circ*: $(x^* \ ; \ x)^\circ = x^\circ$
by (*smt add-idempotent circ-add-4 circ-decompose-7 circ-star star.circ-decompose-5 star.right-plus-circ*)

end

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

sublocale *kleene-algebra-T* < *star*!: *itering-T* **where** *circ* = *star* **..**

class *omega-algebra-T* = *kleene-algebra-T* +
fixes *omega* :: $'a \Rightarrow 'a$ ($-^\omega$ [100] 100)
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 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-subdist-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 *omega-one*: $1^\omega = T$
by (*metis add-left-top less-eq-def omega-one-greatest*)

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

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

lemma *omega-simulation*: $z \ ; \ x \leq y \ ; \ z \rightarrow z \ ; \ x^\omega \leq y^\omega$
by (*smt less-eq-def mult-associative mult-right-subdist-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 mult-right-isotone mult-right-one omega-induct-mult omega-unfold star-mult-omega star-one star.circ-reflexive star.circ-right-unfold star.circ-plus-same star-sum star-sup-one*)

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 star.circ-plus-same*)

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 mult-associative omega-unfold omega-vector star.circ-plus-same star-left-unfold-equal star-omega-top*)

lemma *omega-mult-omega-star*: $x^\omega \ ; \ x^{\omega *} = x^\omega$
by (*metis mult-associative omega-unfold omega-vector star.circ-plus-same star-omega-top*)

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*: $y \ ; \ (y^\omega + y^* \ ; \ z) + z = y^\omega + y^* \ ; \ z$
by (*metis add-associative mult-left-dist-add omega-unfold star.circ-loop-fixpoint*)

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 (*smt antisym mult-associative omega-induct-mult omega-slide omega-sub-vector omega-unfold omega-vector top-mult-top*)

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

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 mult-associative omega-vector star.circ-left-top*)

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-circ-simulate-right-plus*: $z\ ;\ x \leq y\ ;\ (y^\omega\ ;\ 0 + y^*)\ ;\ z + w \rightarrow z\ ;\ (x^\omega\ ;\ 0 + x^*) \leq (y^\omega\ ;\ 0 + y^*)\ ;\ (z + w\ ;\ (x^\omega\ ;\ 0 + x^*))$
proof
assume $z\ ;\ x \leq y\ ;\ (y^\omega\ ;\ 0 + y^*)\ ;\ z + w$
hence *1*: $z\ ;\ x \leq y^\omega\ ;\ 0 + y\ ;\ y^*\ ;\ z + w$
by (*metis mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add omega-unfold*)
hence $(y^\omega\ ;\ 0 + y^*\ ;\ z + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*)\ ;\ x \leq y^\omega\ ;\ 0 + y^*\ ;\ (y^\omega\ ;\ 0 + y\ ;\ y^*\ ;\ z + w) + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*$
by (*smt add-associative add-left-upper-bound add-right-upper-bound less-eq-def mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add star.circ-back-loop-fixpoint*)
also have $\ldots = y^\omega\ ;\ 0 + y^*\ ;\ y\ ;\ y^*\ ;\ z + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*$
by (*smt add-associative add-right-upper-bound less-eq-def mult-associative mult-left-dist-add star.circ-back-loop-fixpoint star-mult-omega*)
finally have $z + (y^\omega\ ;\ 0 + y^*\ ;\ z + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*)\ ;\ x \leq y^\omega\ ;\ 0 + y^*\ ;\ z + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*$
by (*smt add-least-upper-bound add-left-isotone add-left-upper-bound mult-left-isotone order-trans star.circ-loop-fixpoint star.circ-plus-same star.circ-transitive-equal star.left-plus-below-circ*)
hence *2*: $z\ ;\ x^* \leq y^\omega\ ;\ 0 + y^*\ ;\ z + y^*\ ;\ w\ ;\ x^\omega\ ;\ 0 + y^*\ ;\ w\ ;\ x^*$
by (*metis star-right-induct*)
have $z\ ;\ x^\omega\ ;\ 0 \leq (y^\omega\ ;\ 0 + y\ ;\ y^*\ ;\ z + w)\ ;\ x^\omega\ ;\ 0$ **using** *1*
by (*smt add-left-divisibility mult-associative mult-right-subdist-add-left omega-unfold*)
hence $z\ ;\ x^\omega\ ;\ 0 \leq y^\omega + y^*\ ;\ (y^\omega\ ;\ 0 + w\ ;\ x^\omega\ ;\ 0)$
by (*smt add-associative add-commutative left-plus-omega mult-associative mult-left-zero mult-right-dist-add omega-induct star.left-plus-circ*)
thus $z\ ;\ (x^\omega\ ;\ 0 + x^*) \leq (y^\omega\ ;\ 0 + y^*)\ ;\ (z + w\ ;\ (x^\omega\ ;\ 0 + x^*))$ **using** *2*
by (*smt add-associative add-commutative less-eq-def mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add omega-unfold omega-zero star-mult-omega zero-right-mult-decreasing*)
qed

lemma *omega-circ-simulate-left-plus*: $x\ ;\ z \leq z\ ;\ (y^\omega\ ;\ 0 + y^*) + w \rightarrow (x^\omega\ ;\ 0 + x^*)\ ;\ z \leq (z + (x^\omega\ ;\ 0 + x^*)\ ;\ w)\ ;\ (y^\omega\ ;\ 0 + y^*)$
proof
assume *1*: $x\ ;\ z \leq z\ ;\ (y^\omega\ ;\ 0 + y^*) + w$
have $x\ ;\ (z\ ;\ y^\omega\ ;\ 0 + z\ ;\ y^* + x^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^*) \leq x\ ;\ z\ ;\ y^\omega\ ;\ 0 + x\ ;\ z\ ;\ y^* + x^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^*$
by (*smt add-associative add-right-upper-bound less-eq-def mult-associative mult-left-dist-add omega-unfold star.circ-loop-fixpoint*)
also have $\ldots \leq (z\ ;\ y^\omega\ ;\ 0 + z\ ;\ y^* + w)\ ;\ y^\omega\ ;\ 0 + (z\ ;\ y^\omega\ ;\ 0 + z\ ;\ y^* + w)\ ;\ y^* + x^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^*$ **using** *1*
by (*metis add-left-isotone mult-associative mult-left-dist-add mult-left-isotone*)
also have $\ldots = z\ ;\ y^\omega\ ;\ 0 + z\ ;\ y^* + x^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^*$
by (*smt add-associative add-commutative add-idempotent mult-associative mult-left-zero mult-right-dist-add star.circ-loop-fixpoint star.circ-transitive-equal star-mult-omega*)
finally have $(x^\omega\ ;\ 0 + x^*)\ ;\ z \leq z\ ;\ y^\omega\ ;\ 0 + z\ ;\ y^* + x^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^\omega\ ;\ 0 + x^*\ ;\ w\ ;\ y^*$
by (*smt add-least-upper-bound add-left-upper-bound mult-associative mult-left-zero mult-right-dist-add star.circ-back-loop-fixpoint star-left-induct*)
thus $(x^\omega\ ;\ 0 + x^*)\ ;\ z \leq (z + (x^\omega\ ;\ 0 + x^*)\ ;\ w)\ ;\ (y^\omega\ ;\ 0 + y^*)$
by (*smt add-associative mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add*)
qed

end

sublocale *omega-algebra-T* $<$ *comb0*!: *itering-T* **where** *circ* $= (\lambda x\ .\ x^\omega\ ;\ 0 + x^*)$

apply *unfold-locales*
apply (*smt add-associative add-commutative mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add omega-slide star.circ-mult*)
apply (*smt add-associative add-commutative star.circ-add mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add star.mult-zero-add-circ-2 mult-zero-add-omega omega-decompose*)
apply (*smt add-isotone star.circ-simulate mult-associative mult-left-dist-add mult-left-isotone mult-left-zero mult-right-dist-add omega-simulation*)
apply (*smt add-commutative add-left-isotone mult-left-isotone mult-left-subdist-add-left mult-right-one omega-circ-simulate-right-plus order-trans star.circ-plus-one*)
apply (*smt add-commutative add-right-isotone mult-left-subdist-add-left mult-right-isotone omega-circ-simulate-left-plus order-trans star.circ-increasing*)
apply (*metis omega-circ-simulate-right-plus*)
by (*metis omega-circ-simulate-left-plus*)

class *omega-itering* = *omega-algebra-T* + *itering-T*

begin

subclass *kleene-itering* **..**

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 (*smt add-commutative circ-reflexive less-eq-def mult-left-dist-add mult-right-one omega-sub-vector*)

lemma *circ-mult-omega*: $x^\circ \ ; \ x^\omega = x^\omega$
by (*metis antisym circ-left-unfold circ-slide le-less mult-left-one mult-right-one mult-right-subdist-add-left omega-simulation*)

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

lemma *omega-circ*: $x^{\omega\circ} = 1 + x^\omega$
by (*metis circ-left-unfold circ-star mult-associative omega-vector star.circ-left-top*)

end

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

class *dra* = *kleene-algebra-T* + *Omega* +
assumes *Omega-unfold* : $y^\Omega = 1 + y \ ; \ y^\Omega$
assumes *Omega-isolate* : $y^\Omega = y^\Omega \ ; \ 0 + y^*$
assumes *Omega-induct* : $x \leq z + y \ ; \ x \rightarrow x \leq y^\Omega \ ; \ z$

begin

lemma *Omega-mult*: $(x \ ; \ y)^\Omega = 1 + x \ ; \ (y \ ; \ x)^\Omega \ ; \ y$
by (*smt Omega-induct Omega-unfold antisym mult-associative mult-left-dist-add mult-left-one mult-right-dist-add mult-right-isotone mult-right-one order-refl*)

lemma *Omega-add-1*: $(x + y)^\Omega = x^\Omega \ ; \ (y \ ; \ x^\Omega)^\Omega$
by (*smt Omega-induct Omega-unfold add-associative add-commutative add-right-isotone antisym mult-associative mult-left-one mult-right-dist-add mult-right-isotone mult-right-one*

order-refl)

lemma *Omega-add*: $(x + y)^\Omega = (x^\Omega \ ; \ y)^\Omega \ ; \ x^\Omega$
by (*smt Omega-add-1 Omega-mult mult-associative mult-left-dist-add mult-left-one mult-right-dist-add mult-right-one*)

lemma *Omega-simulate*: $z \ ; \ x \leq y \ ; \ z \rightarrow z \ ; \ x^\Omega \leq y^\Omega \ ; \ z$
by (*smt Omega-induct Omega-unfold add-right-isotone mult-associative mult-left-dist-add mult-left-isotone mult-right-one*)

end

sublocale *dra* $<$ *Omega*!: *itering-1* **where** *circ* $=$ *Omega*
apply *unfold-locales*
apply (*metis Omega-mult*)
apply (*metis Omega-add*)
by (*metis Omega-simulate*)

context *dra*

begin

lemma *star-below-Omega*: $x^* \leq x^\Omega$
by (*metis Omega-isolate add-commutative add-left-divisibility*)

lemma *Omega-sum-unfold-1*: $(x + y)^\Omega = y^\Omega + y^* \ ; \ x \ ; \ (x + y)^\Omega$
by (*smt Omega.circ-add Omega.circ-loop-fixpoint Omega-isolate add-associative add-commutative mult-associative mult-left-zero mult-right-dist-add*)

lemma *Omega-add-3*: $(x + y)^\Omega = (x^* \ ; \ y)^\Omega \ ; \ x^\Omega$
by (*smt Omega.circ-add Omega.circ-isotone Omega-induct Omega-sum-unfold-1 add-commutative antisym mult-left-isotone order-refl star-below-Omega*)

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*)

lemma *star-mult-Omega*: $x^\Omega = x^* \ ; \ x^\Omega$
by (*metis Omega.circ-plus-one Omega.circ-transitive-equal Omega-isolate add-commutative less-eq-def order-refl star.circ-add-mult-zero star.circ-increasing star.circ-plus-same star-involutive star-left-induct star-square*)

lemma *Omega-separate-2*: $y \ ; \ x \leq x \ ; \ (x + y) \rightarrow (x + y)^\Omega = x^\Omega \ ; \ y^\Omega$
by (*smt Omega.circ-sub-dist-3 Omega-induct Omega-sum-unfold-1 add-right-isotone antisym mult-associative mult-left-isotone star-mult-Omega star-simulation-left*)

lemma *Omega-circ-simulate-right-plus*: $z \ ; \ x \leq y \ ; \ y^\Omega \ ; \ z + w \rightarrow z \ ; \ x^\Omega \leq y^\Omega \ ; \ (z + w \ ; \ x^\Omega)$
by (*smt Omega.circ-back-loop-fixpoint Omega.circ-plus-same Omega.left-plus-circ Omega-induct add-associative add-commutative add-right-isotone less-eq-def mult-associative mult-right-dist-add*)

lemma *Omega-circ-simulate-left-plus*: $x \ ; \ z \leq z \ ; \ y^\Omega + w \rightarrow x^\Omega \ ; \ z \leq (z + x^\Omega \ ; \ w) \ ; \ y^\Omega$
proof

assume *1*: $x\ ;\ z \leq z\ ;\ y^\Omega + w$
hence $x\ ;\ ((z + x^\Omega\ ;\ w)\ ;\ y^\Omega) \leq z\ ;\ y^\Omega\ ;\ y^\Omega + w\ ;\ y^\Omega + x^\Omega\ ;\ w\ ;\ y^\Omega$
by (*smt Omega.left-plus-below-circ add-left-isotone add-right-isotone mult-associative mult-left-dist-add mult-left-isotone mult-right-dist-add order-trans*)
hence $x\ ;\ ((z + x^\Omega\ ;\ w)\ ;\ y^\Omega) \leq (z + x^\Omega\ ;\ w)\ ;\ y^\Omega$
by (*metis Omega.circ-transitive-equal mult-associative Omega.circ-reflexive add-associative less-eq-def mult-left-one mult-right-dist-add*)
thus $x^\Omega\ ;\ z \leq (z + x^\Omega\ ;\ w)\ ;\ y^\Omega$
by (*smt Omega.circ-back-loop-fixpoint Omega-isolate add-least-upper-bound mult-associative mult-left-zero mult-right-dist-add mult-right-subdist-add-left mult-right-subdist-add-right star-left-induct*)
qed

end

sublocale *dra* $<$ *Omega*!: *itering-T* **where** *circ* $=$ *Omega*
apply *unfold-locales*
apply (*smt Omega.circ-loop-fixpoint Omega-circ-simulate-right-plus add-associative add-commutative add-left-upper-bound mult-associative mult-left-dist-add order-trans*)
apply (*smt Omega.circ-increasing Omega-circ-simulate-left-plus add-commutative add-right-isotone mult-right-isotone order-trans*)
apply (*metis Omega-circ-simulate-right-plus*)
by (*metis Omega-circ-simulate-left-plus*)

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

begin

lemma *omega-left-zero*: $x^\omega\ ;\ y = x^\omega$
by (*metis mult-associative omega-vector top-left-zero*)

lemma *Omega-mult*: $(x\ ;\ y)^\Omega = 1 + x\ ;\ (y\ ;\ x)^\Omega\ ;\ y$
by (*smt add-associative add-commutative Omega-def mult-left-dist-add mult-right-dist-add omega-left-zero omega-slide star-unfold-slide*)

lemma *Omega-add*: $(x + y)^\Omega = (x^\Omega\ ;\ y)^\Omega\ ;\ x^\Omega$
proof $-$
have $(x^\Omega\ ;\ y)^\Omega\ ;\ x^\Omega = (x^*\ ;\ y)^*\ ;\ x^\omega + (x^*\ ;\ y)^\omega + (x^*\ ;\ y)^*\ ;\ x^{\omega *}\ ;\ x^\Omega$
by (*smt add-commutative Omega-def mult-associative mult-right-dist-add mult-zero-add-omega omega-left-zero star.circ-add-1*)
thus *?thesis*
by (*smt add-associative add-commutative Omega-def mult-associative mult-left-dist-add omega-decompose omega-left-zero star.circ-add-1 star.circ-loop-fixpoint star.circ-slide*)
qed

lemma *Omega-simulate*: $z\ ;\ x \leq y\ ;\ z \rightarrow z\ ;\ x^\Omega \leq y^\Omega\ ;\ z$
by (*smt add-isotone Omega-def mult-left-dist-add mult-right-dist-add omega-left-zero omega-simulation star-simulation-right*)

subclass *dra*
apply *unfold-locales*
apply (*metis Omega-def comb0.circ-left-unfold omega-left-zero*)
apply (*smt Omega-def add-associative less-eq-def mult-right-dist-add omega-left-zero zero-right-mult-decreasing*)
by (*metis Omega-def mult-right-dist-add omega-induct omega-left-zero*)

end

declare [[*smt-timeout* = *60*]]

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

class *relative-domain-semiring* = *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 d-dist-add d-mult-d d-plus-one less-eq-def mult-left-dist-add 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 (*metis 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*: $x \leq d(y)\ ;\ x + Z \leftrightarrow d(x) \leq d(y)$
by (*rule iffI*,
metis add-associative add-left-divisibility add-right-zero d-Z d-dist-add d-involutive d-mult-sub less-eq-def,
smt add-associative add-commutative d-restrict less-eq-def mult-right-dist-add)

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 (*metis add-commutative d-idempotent d-plus-one mult-left-dist-add mult-right-one*)

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 (*smt add-associative d-commutative d-idempotent d-mult-left-absorb-add mult-left-dist-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*)

end

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

begin

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

end

class *relative-antidomain-semiring* = *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-greatest-left-absorber*: $a(x)\ ;\ y \leq Z \leftrightarrow a(x) \leq a(y)$
by (*rule iffI*,
metis a-Z a-complement-one a-dist-add a-mult-d add-right-divisibility less-eq-def mult-left-dist-add mult-left-one mult-right-one mult-right-subdist-add-left,
metis a-restrict le-less-trans le-neq-trans less-eq-def less-imp-le mult-right-subdist-add-left)

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

lemma *a-plus-left-lower-bound*: $a(x + y) \leq a(x)$

by (*metis a-greatest-left-absorber a-restrict add-commutative mult-left-subdist-add-right order-trans*)

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*)

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*)
by (*metis a-dist-add a-export d-def*)

subclass *tests*
apply *unfold-locales*
apply (*metis mult-associative*)
apply (*metis a-dist-add add-commutative*)
apply (*metis a-complement-one a-d-closed a-dist-add d-def mult-left-dist-add mult-right-one*)
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*)
by (*metis less-def*)

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

lemma *d-cancellation-2*: $(d(z) + d(y)) \mathbin{;} a(y) \leq d(z)$
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*: $d(a(x) \mathbin{;} a(y)) = a(x) \mathbin{;} a(y)$
by (*metis d-def sub-mult-closed*)

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

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

lemma *d-compl-intro*: $d(x) + d(y) = d(x) + a(x) \mathbin{;} d(y)$
by (*metis add-commutative d-def plus-compl-intro*)

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

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

lemma *kat-3*: $d(x) \mathbin{;} y \mathbin{;} a(z) = 0 \rightarrow d(x) \mathbin{;} y = d(x) \mathbin{;} y \mathbin{;} d(z)$
by (*metis add-left-zero d-def mult-left-dist-add mult-right-one plus-compl*)

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

lemma *kat-2-equiv*: $y \mathbin{;} a(z) \leq a(x) \mathbin{;} y \leftrightarrow d(x) \mathbin{;} y \mathbin{;} a(z) = 0$
by (*smt kat-2 a-Z a-below-one a-complement-one case-split-left d-def mult-associative mult-right-isotone mult-right-one zero-least*)

lemma *kat-4-equiv*: $d(x) \mathbin{;} y = d(x) \mathbin{;} y \mathbin{;} d(z) \leftrightarrow d(x) \mathbin{;} y \leq y \mathbin{;} d(z)$
by (*rule*, *metis kat-4*, *rule antisym*,
metis d-idempotent less-eq-def mult-associative mult-left-dist-add,
metis d-plus-one less-eq-def mult-left-dist-add mult-right-one)

lemma *kat-3-equiv-opp*: $a(z) \mathbin{;} y \mathbin{;} d(x) = 0 \leftrightarrow y \mathbin{;} d(x) = d(z) \mathbin{;} y \mathbin{;} d(x)$
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-4-equiv-opp*: $y \mathbin{;} d(x) = d(z) \mathbin{;} y \mathbin{;} d(x) \leftrightarrow y \mathbin{;} d(x) \leq d(z) \mathbin{;} y$
by (*metis d-def double-negation kat-2-equiv kat-3-equiv-opp*)

lemma *kat-equiv-5*: $d(x)\ ;\ y \leq y\ ;\ d(z) \leftrightarrow d(x)\ ;\ y\ ;\ a(z) = d(x)\ ;\ y\ ;\ 0$
by (*rule*,
metis d-complement-zero kat-4-equiv mult-associative,
smt a-complement-one case-split-right d-def mult-isotone mult-left-one mult-right-subdist-add-left order-refl zero-least)

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 *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-subdist-add-left order-trans*)
hence $x \leq y + Z \rightarrow x \leq d(x)\ ;\ y + Z$
by (*metis add-associative d-plus-one mult-left-dist-add mult-left-one mult-right-dist-add*)
thus *?thesis*
by (*metis a-complement-one add-commutative add-right-isotone d-def mult-left-one mult-right-subdist-add-left order-trans*)
qed

lemma *d-restrict-iff-1*: $(d(x)\ ;\ y \leq z) \leftrightarrow (d(x)\ ;\ y \leq d(x)\ ;\ z)$
by (*smt a-complement-one add-left-divisibility d-def d-idempotent mult-associative mult-left-dist-add mult-left-one mult-right-dist-add order-trans*)

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

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

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

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

end

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

begin

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*: $-p\ ;\ x \leq y \leftrightarrow x \leq --p\ ;\ T + y$
by (*rule*,

metis add-commutative case-split-left-add mult-right-isotone top-greatest,
metis a-complement add-left-zero add-right-divisibility mult-associative mult-left-dist-add mult-left-one mult-left-zero mult-right-dist-add mult-right-isotone order-trans plus-left-one)

end

class *relative-diamond-semiring* = *relative-domain-semiring* +
fixes *diamond* :: *'a* ⇒ *'a* ⇒ *'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-dist-add*: $|x>(y + z) = |x>y + |x>z$
by (*metis d-dist-add diamond-def mult-left-dist-add*)

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 diamond-right-dist-add less-eq-def*)

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-dist-add*)

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

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

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

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

lemma *diamond-demodalisation-3*: $|x\!>\!y \leq d(z) \leftrightarrow x\ ;\ d(y) \leq d(z)\ ;\ x + Z$
by (*rule iffI*,
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,
smt add-commutative add-left-zero d-Z d-dist-add d-isotone d-mult-greatest-lower-bound diamond-d-y diamond-def diamond-x-und)

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

lemma *diamond-box*: $|x\!>\!y = a(\ |x]a(y))$
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-mult-d-d*: $|x](d(y) \ ; d(z)) = |x]d(y) \ ; |x]d(z)$
by (*smt a-dist-add box-x-a diamond-box diamond-x-1 mult-left-dist-add*)

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

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

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

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

lemma *box-demodalisation-3*: $d(x) \leq |y]d(z) \rightarrow d(x)\ ;\ y \leq y\ ;\ d(z) + Z$
proof −
have $d(x) \leq |y]d(z) \rightarrow d(x)\ ;\ y\ ;\ a(z) \leq Z$
by (*metis mult-left-isotone a-mult-d a-restrict box-def d-def mult-associative order-trans*)
thus *?thesis*
by (*metis add-commutative case-split-right-add d-def d-restrict-iff-1 eq-refl mult-associative*)
qed

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*: $|x](d(y) + a(z)) \ ; \ |x]d(z) \leq |x]d(z)$
by (*metis box-lower-bound-right box-right-mult-d-d box-x-und d-commutative*)

lemma *fbox-diff*: $|x](d(y) + a(z)) \leq |x]y + a(\ |x]z)$
by (*smt a-compl-intro a-dist-add a-mult-d a-plus-left-lower-bound add-commutative box-def d-def mult-left-dist-add shunting*)

lemma *diamond-demodalisation-2*: $|x\!>\!y \leq d(z) \leftrightarrow a(z) \ ; \ x \ ; \ d(y) \leq Z$
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-diff-var*: $|x\!>\!d(y) \leq |x\!>\!(d(y) \ ; \ a(z)) + |x\!>\!d(z)$
by (*smt a-dist-add add-commutative box-def box-right-mult-a-a diamond-box diamond-right-upper-bound diamond-x-1 double-negation mult-compl-intro mult-right-one one-def plus-closed sub-comm*)

lemma *diamond-diff*: $|x\!>\!y \ ; \ a(\ |x\!>\!z) \leq |x\!>\!(d(y) \ ; \ a(z))$
by (*metis d-a-shunting d-involutive diamond-def diamond-diff-var diamond-x-und*)

lemma *diamond-split*: $|x\!>\!y = d(z) \ ; \ |x\!>\!y + a(z) \ ; \ |x\!>\!y$
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

context *tests*

begin

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* $B \longrightarrow$ *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* $t \longrightarrow$ *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* $t\ s \leftrightarrow (\forall n\ .\ t\ n = --t\ n) \wedge s = --s \wedge (\forall n\ .\ t\ n \leq s) \wedge (\forall x\ y\ .\ (\forall n\ .\ t\ n\ ;\ -x \leq -y) \longrightarrow s\ ;\ -x \leq -y)$

lemma *nat-test-seq*: *nat-test* $t\ s \longrightarrow$ *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* $t \longrightarrow$ *pSum* $t\ n = --(pSum\ t\ n)$
by (*induct* n,
metis pSum.simps(*1*) *one-compl one-def*,
smt pSum.simps(*2*) *plus-closed test-seq-def*)

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

lemma *pSum-upper*: *test-seq* $t \wedge i{<}n \longrightarrow t\ i \leq$ *pSum* $t\ n$
by (*induct* n,
smt less-zeroE,
smt pSum.simps(*2*) *pSum-test test-seq-def transitive upper-bound-left upper-bound-right*)

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

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

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

end

class *complete-tests* = *tests* + *Sup* +
  assumes *sup-test*: *test-set A* → *Sup A* = −−*Sup A*
  assumes *sup-upper*: *test-set A* ∧ *x* ∈ *A* → *x* ≤ *Sup A*
  assumes *sup-least*: *test-set A* ∧ (∀ *x*∈*A* . *x* ≤ −*y*) → *Sup A* ≤ −*y*

begin

lemma *Sup-isotone*: *test-set B* ∧ *A* ⊆ *B* → *Sup A* ≤ *Sup B*
  by (*smt subsetD sup-least sup-test sup-upper test-set-closed*)

lemma *mult-right-dist-sup*: *test-set A* → *Sup A* ; −*p* = *Sup* { *x*;−*p* | *x* . *x* ∈ *A* }
proof
  assume *1*: *test-set A*
  hence *2*: *test-set* { *x*;−*p* | *x* . *x* ∈ *A* }
    by (*simp add*: *mult-right-dist-test-set*)
  have *3*: *Sup* { *x*;−*p* | *x* . *x* ∈ *A* } ≤ *Sup A* ; −*p* **using** *1*
    by (*smt mem-Collect-eq mult-iso-left sub-mult-closed sup-test sup-least sup-upper test-set-def*)
  have *Sup A* ; −*p* ≤ *Sup* { *x*;−*p* | *x* . *x* ∈ *A* }
  proof −
    have ∀ *x*∈*A* . *x* ≤ −−(−−*Sup* { *x*;−*p* | *x* . *x* ∈ *A* } + −−*p*)
    proof
      fix *x*
      assume *4*: *x* ∈ *A*
      hence *x*;−*p* + −−*p* ≤ *Sup* { *x*;−*p* | *x* . *x* ∈ *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* ≤ −−(−−*Sup* { *x*;−*p* | *x* . *x* ∈ *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* ≤ −−(−−*Sup* { *x*;−*p* | *x* . *x* ∈ *A* } + −−*p*) **using** *1*
      by (*simp add*: *sup-least*)
    thus *Sup A* ; −*p* ≤ *Sup* { *x*;−*p* | *x* . *x* ∈ *A* } **using** *1 2*
      by (*smt plus-closed plus-comm shunting sub-comm sup-test*)
  qed
  thus *Sup A* ; −*p* = *Sup* { *x*;−*p* | *x* . *x* ∈ *A* } **using** *1 2 3*
    by (*smt antisymmetric sub-mult-closed sup-test*)
qed

lemma *mult-left-dist-sup*: *test-set A* → −*p* ; *Sup A* = *Sup* { −*p*;*x* | *x* . *x* ∈ *A* }
proof
  assume *1*: *test-set A*
  hence *2*: *Sup A* ; −*p* = *Sup* { *x*;−*p* | *x* . *x* ∈ *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* | *x* . *x* ∈ *A* } = { *x*;−*p* | *x* . *x* ∈ *A* }
    by (*rule set-eqI*, *simp*, *metis 1 sub-comm test-set-def*)
  thus −*p* ; *Sup A* = *Sup* { −*p*;*x* | *x* . *x* ∈ *A* } **using** *2 3*

```
    by simp
qed

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

lemma Sum-test: test-seq t ⟶ Sum t = --Sum t
  by (metis Sum-def sup-test test-seq-test-set)

lemma Sum-upper: test-seq t ⟶ t n ≤ Sum t
proof -
  have t n ∈ { t n | n::nat . True }
    by (smt mem-Collect-eq)
  thus ?thesis
    by (metis Sum-def sup-upper test-seq-test-set)
qed

lemma Sum-least: test-seq t ∧ (∀ n . t n ≤ -p) ⟶ Sum t ≤ -p
proof
  assume 1: test-seq t ∧ (∀ n . t n ≤ -p)
  hence ∀ x∈{ t n | n::nat . True } . x ≤ -p
    by (smt mem-Collect-eq)
  thus Sum t ≤ -p using 1
    by (metis Sum-def test-seq-test-set sup-least)
qed

lemma mult-right-dist-Sum: test-seq t ∧ (∀ n . t n;-p ≤ -q) ⟶ Sum t;-p ≤ -q
proof
  assume 1: test-seq t ∧ (∀ n . t n;-p ≤ -q)
  hence Sup { x;-p | x . x ∈ { t n | n::nat . True } } ≤ -q
    by (smt mem-Collect-eq sub-mult-closed sup-least test-seq-def test-set-def)
  thus Sum t;-p ≤ -q using 1 test-seq-test-set Sum-def mult-right-dist-sup
    by simp
qed

lemma pSum-below-Sum: test-seq t ⟶ pSum t n ≤ 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-seq t ⟶ pSum t n = Sup { t i | i . i ∈ {..<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

end

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

class precondition = tests + pre +
  assumes pre-closed: x«−q = −−(x«−q)
  assumes pre-distrib: x«−p;−q = (x«−p);(x«−q)
  assumes pre-seq: x;y«−q = x«y«−q
  assumes pre-test: −p«−q = −−p + −q

begin
```

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

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

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

lemma *pre-lower-bound-right*: $x\mathbin{«}{-p};{-q} \leq x\mathbin{«}{-q}$
by (*metis lower-bound-right pre-distrib pre-closed*)

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

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

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-one*: $1\mathbin{«}{-p} = -p$
by (*metis one-compl one-def plus-left-zero 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-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*: $-p;-q \leq x\ll -s \leftrightarrow -p;-q \leq -q;x\ll -s$
by (*smt leq-def pre-closed sub-assoc sub-mult-closed pre-import*)

lemma *pre-test-promote*: $-p\ll -q = -p\ll -p;-q$
by (*metis bs-mult-right-one leq-plus-right-one one-def plus-right-one pre-distrib pre-test reflexive sub-comm*)

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

lemma *pre-test-test-same*: $-p\ll -p = 1$
by (*metis plus-comm plus-compl pre-test*)

lemma *test-below-pre-test-mult*: $-q \leq -p\ll -p;-q$
by (*metis pre-test pre-test-promote upper-bound-right*)

lemma *test-below-pre-test*: $-q \leq -p\ll -q$
by (*metis pre-test upper-bound-right*)

end

class *complete-pre* = *complete-tests* + *precondition* + *power*

begin

definition *bnd* :: $'a \Rightarrow 'a$
where *bnd x* = *Sup* { $x\hat{}n\ll 0 \mid n{::}nat$. *True* }

lemma *bnd-test-set*: *test-set* { $x\hat{}n\ll 0 \mid n{::}nat$. *True* }
by (*smt mem-Collect-eq one-compl pre-closed test-set-def*)

lemma *bnd-test*: *bnd x* = $--$*bnd x*
by (*metis bnd-def bnd-test-set sup-test*)

lemma *bnd-upper*: $x\hat{}n\ll 0 \leq$ *bnd x*
proof −
have $x\hat{}n\ll 0 \in$ { $x\hat{}n\ll 0 \mid n{::}nat$. *True* }
by (*smt mem-Collect-eq*)
thus *?thesis*
by (*metis bnd-def bnd-test-set sup-upper*)
qed

lemma *bnd-least*: $(\forall\, n$. $x\hat{}n\ll 0 \leq -p) \rightarrow$ *bnd x* $\leq -p$
proof
assume $\forall\, n$. $x\hat{}n\ll 0 \leq -p$
hence $\forall\, y{\in}\{$ $x\hat{}n\ll 0 \mid n{::}nat$. *True* $\}$. $y \leq -p$

by (*smt mem-Collect-eq*)
thus *bnd* $x \leq -p$
by (*metis bnd-def bnd-test-set sup-least*)
qed

lemma *mult-right-dist-bnd*: $(\forall\, n\ .\ (x\hat{}n\mathbin{«}0);{-p} \leq -q) \longrightarrow bnd\ x;{-p} \leq -q$
proof
assume $\forall\, n\ .\ (x\hat{}n\mathbin{«}0);{-p} \leq -q$
hence $Sup\ \{\ y;{-p} \mid y\ .\ y \in \{\ x\hat{}n\mathbin{«}0 \mid n{::}nat\ .\ True\ \}\ \} \leq -q$
by (*smt mem-Collect-eq one-compl pre-closed sub-mult-closed sup-least test-set-def*)
thus $bnd\ x;{-p} \leq -q$ **using** *bnd-test-set bnd-def mult-right-dist-sup*
by *simp*
qed

lemma *tests-complete*: *nat-test* $(\lambda n\ .\ (-p;x)\hat{}n\mathbin{«}0)\ (bnd(-p;x))$
by (*smt bnd-test bnd-upper mult-right-dist-bnd nat-test-def one-compl pre-closed*)

end

class *modal-precondition* = *relative-box-semiring* + *pre* +
assumes *pre-def*: $x\mathbin{«}p = |x]p$

begin

subclass *precondition*
by (*unfold-locales*,
metis box-def double-negation pre-def,
metis box-right-mult-a-a pre-def,
metis box-left-mult pre-def,
metis box-a-a d-def pre-def)

lemma *pre-Z*: $-p \leq x\mathbin{«}{-q} \leftrightarrow -p\ ;\ x\ ;\ {-}{-q} \leq Z$
by (*metis box-demodalisation-2 pre-def*)

lemma *pre-left-dist-add*: $x{+}y\mathbin{«}{-q} = (x\mathbin{«}{-q})\ ;\ (y\mathbin{«}{-q})$
by (*metis box-left-dist-add pre-def*)

lemma *pre-left-antitone*: $x \leq y \longrightarrow y\mathbin{«}{-q} \leq x\mathbin{«}{-q}$
by (*metis box-left-antitone pre-def*)

lemma *pre-sub-promote*: $(x\mathbin{«}{-q})\ ;\ x \leq (x\mathbin{«}{-q})\ ;\ x\ ;\ {-q} + Z$
by (*metis case-split-right-add order-refl pre-Z pre-closed*)

lemma *pre-promote*: $(x\mathbin{«}{-q})\ ;\ x + Z = (x\mathbin{«}{-q})\ ;\ x\ ;\ {-q} + Z$
by (*smt a-below-one add-left-upper-bound add-same-context mult-right-isotone mult-right-one order-trans pre-sub-promote*)

lemma *pre-mult-sub-promote*: $(x;y\mathbin{«}{-q})\ ;\ x \leq (x;y\mathbin{«}{-q})\ ;\ x\ ;\ (y\mathbin{«}{-q}) + Z$

by (*metis pre-closed pre-seq pre-sub-promote*)

lemma *pre-mult-promote*: $(x;y\llcorner -q)\ ;\ x\ ;\ (y\llcorner -q) + Z = (x;y\llcorner -q)\ ;\ x + Z$
by (*smt pre-below-one add-left-upper-bound add-same-context mult-right-isotone mult-right-one order-trans pre-mult-sub-promote*)

lemma *pre-promote-neg*: $(x\llcorner -q)\ ;\ x\ ;\ --q \leq Z$
by (*metis order-refl pre-Z pre-closed*)

lemma *pre-pc-Z*: $x\llcorner 1 = 1 \leftrightarrow x\ ;\ 0 \leq Z$
by (*metis a-strict box-x-1 pre-def*)

end

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

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

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

class *ifthenelse* = *precondition* + *ite* +
assumes *ite-pre*: $x\lhd -p\rhd y\llcorner -q = -p;(x\llcorner -q) + --p;(y\llcorner -q)$

begin

lemma *ite-pre-then*: $-p;(x\lhd -p\rhd y\llcorner -q) = -p;(x\llcorner -q)$
by (*smt ite-pre plus-absorb plus-distr-mult plus-right-zero pre-closed sub-assoc sub-mult-closed zero-def*)

lemma *ite-pre-else*: $--p;(x\lhd -p\rhd y\llcorner -q) = --p;(y\llcorner -q)$
by (*smt ite-pre mult-distr-plus mult-idempotent plus-left-zero pre-closed sub-assoc sub-mult-closed zero-def*)

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

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

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

end

class *whiledo* = *ifthenelse* + *while* +

assumes *while-pre*: $-p*x\ll -q = x;(-p*x) \lhd -p \rhd 1\ll -q$

begin

lemma *while-pre-2*: $-p*x\ll -q = -p;(x\ll -p*x\ll -q) + --p;-q$
by (*metis ite-pre pre-one pre-seq while-pre*)

lemma *while-pre-then*: $-p;(-p*x\ll -q) = -p;(x\ll -p*x\ll -q)$
by (*metis ite-pre-then pre-seq while-pre*)

lemma *while-pre-else*: $--p;(-p*x\ll -q) = --p;-q$
by (*metis ite-pre-else pre-one while-pre*)

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* $\wedge$ $-p;-q \leq x\ll -q \wedge -q \leq s \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n) \rightarrow -q \leq -p*x\ll --p;-q$
proof
assume *1*: *nat-test t s* $\wedge$ $-p;-q \leq x\ll -q \wedge -q \leq s \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n)$
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 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-distrib sub-assoc sub-comm sub-mult-closed 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 mult-distr-plus-right plus-compl pre-closed sub-mult-closed*)
qed

lemma *nat-test-pre-1*: *nat-test t s* $\wedge$ $-p;-q \leq x\ll-q \wedge -r \leq s \wedge -r \leq -q \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n) \rightarrow -r \leq -p*x\ll--p;-q$
proof
let *?qs* = $-q;s$
assume *1*: *nat-test t s* $\wedge$ $-p;-q \leq x\ll-q \wedge -r \leq s \wedge -r \leq -q \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n)$
hence *2*: $-r \leq$ *?qs*
by (*metis greatest-lower-bound nat-test-def*)
have $-p;$*?qs* $\leq x\ll$ *?qs* **using** *1*
by (*smt greatest-lower-bound lower-bound-left nat-test-def pSum-below-sum pSum-test-nat pre-closed pre-distrib pre-iso sub-assoc sub-comm 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-test t s* $\wedge$ $-r \leq s \wedge (\forall n\ .\ t\ n;-p \leq x\ll pSum\ t\ n) \rightarrow -r \leq -p*x\ll1$
proof
assume *1*: *nat-test t s* $\wedge$ $-r \leq s \wedge (\forall n\ .\ t\ n;-p \leq x\ll pSum\ t\ n)$
hence $-p;s \leq x\ll s$
by (*smt nat-test-def pSum-below-sum pSum-test-nat pre-closed pre-iso sub-comm sub-mult-closed transitive*)
hence $-r \leq -p*x\ll--p;s$ **using** *1*
by (*smt leq-def nat-test-def nat-test-pre-1 sub-assoc sub-comm*)
thus $-r \leq -p*x\ll1$ **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-test t s* $\wedge$ $-p;-q \leq x\ll-q \wedge -q \leq s \wedge (\forall n\ .\ t\ n;-p;-q \leq x\ll pSum\ t\ n) \rightarrow -q \leq -p*x\ll1$
by (*smt nat-test-pre one-double-compl pre-below-pre-one pre-closed sub-mult-closed transitive*)

abbreviation *aL* :: *'a*
where $aL \equiv 1*1\ll1$

end

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

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

begin

inductive-set *Test-expression* :: *'a set*
  where *atom-test*: $p \in$ *Atomic-test* $\Rightarrow p \in$ *Test-expression*
    | *neg-test*: $p \in$ *Test-expression* $\Rightarrow -p \in$ *Test-expression*
    | *conj-test*: $p \in$ *Test-expression* $\wedge\ q \in$ *Test-expression* $\Rightarrow p;q \in$ *Test-expression*

lemma *test-expression-test*: $p \in$ *Test-expression* $\rightarrow p = --p$
  by (*rule*, *induct rule*: *Test-expression.induct*,
    metis atomic-test-test, *simp*, *metis sub-mult-closed*)

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

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

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

lemma *pSum-test-expression*: $(\forall\, n\ .\ t\ n \in$ *Test-expression*$) \rightarrow$ *pSum* $t\ n \in$ *Test-expression*
  by (*rule*, *induct n*,
    metis pSum.simps(*1*) *zero-test-expression*,
    metis disj-test pSum.simps(*2*))

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

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

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

inductive-set *Pre-expression* :: *'a set*
  where *test-pre*: $p \in$ *Test-expression* $\Rightarrow p \in$ *Pre-expression*
    | *neg-pre*: $p \in$ *Pre-expression* $\Rightarrow -p \in$ *Pre-expression*
    | *conj-pre*: $p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\Rightarrow p;q \in$ *Pre-expression*
    | *pre-pre*: $p \in$ *Pre-expression* $\wedge\ x \in$ *While-program* $\Rightarrow x \ll p \in$ *Pre-expression*

lemma *pre-expression-test*: $p \in$ *Pre-expression* $\rightarrow p = --p$

by (*rule*, *induct rule*: *Pre-expression.induct*,
metis test-expression-test, *simp*, *metis sub-mult-closed*, *metis pre-closed*)

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

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

lemma *bnd-pre-expression*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow (p;x)\,\hat{}\,n\mathrel{«}0 \in \mathit{Pre\text{-}expression}$
by (*rule*, *induct n*,
metis pre-pre one-while-program power-0 zero-pre-expression,
simp, *smt disj-pre neg-pre one-compl pre-closed pre-export pre-pre pre-seq test-expression-test test-pre*)

end

class *hoare-calculus* = *while-program* + *complete-pre* +
assumes *while-soundness*: $-p;-q \leq x\mathrel{«}-q \rightarrow -q;aL \leq -p*x\mathrel{«}--p;-q$

begin

inductive *derived-hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow \mathit{bool}$ (- (| - |) - [*54*,*54*,*54*] *53*)
where *atom-trip*: $p \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{Atomic\text{-}program} \Rightarrow x\mathrel{«}p(|x|)p$
| *seq-trip*: $p(|x|)q \wedge q(|y|)r \Rightarrow p(|x;y|)r$
| *cond-trip*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge p;q(|x|)r \wedge -p;q(|y|)r \Rightarrow q(|x \lhd p \rhd y|)r$
| *while-trip*: $p \in \mathit{Test\text{-}expression} \wedge \mathit{test\text{-}seq}\ t \wedge p;q(|x|)q \wedge q \leq aL + \mathit{Sum}\ t \wedge (\forall\, n\ .\ t\ n;p;q(|x|)aL+\mathit{pSum}\ t\ n) \Rightarrow q(|p*x|)-p;q$
| *cons-trip*: $p \in \mathit{Pre\text{-}expression} \wedge s \in \mathit{Pre\text{-}expression} \wedge p \leq q \wedge q(|x|)r \wedge r \leq s \Rightarrow p(|x|)s$

lemma *derived-type*: $p(|x|)q \Rightarrow p \in \mathit{Pre\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program}$
by (*induct rule*: *derived-hoare-triple.induct*,
(*smt atom-prog seq-prog cond-prog while-prog test-pre neg-pre conj-pre pre-pre*)+)

lemma *cons-pre-trip*: $p \in \mathit{Pre\text{-}expression} \wedge q(|y|)r \rightarrow p;q(|y|)r$
by (*smt conj-pre cons-trip derived-type lower-bound-right reflexive pre-expression-test*)

lemma *cons-post-trip*: $q \in \mathit{Pre\text{-}expression} \wedge r \in \mathit{Pre\text{-}expression} \wedge p(|y|)q;r \rightarrow p(|y|)r$

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 $p\langle x\rangle q \equiv (p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \wedge p \leq x \ll q)$

lemma *while-soundness-1*: *test-seq* $t \wedge -p;-q \leq x \ll -q \wedge -q \leq aL + Sum\ t \wedge (\forall n\ .\ t\ n;-p;-q \leq x \ll aL+pSum\ t\ n) \rightarrow -q \leq -p*x \ll --p;-q$
proof
assume *1*: *test-seq* $t \wedge -p;-q \leq x \ll -q \wedge -q \leq aL + Sum\ t \wedge (\forall n\ .\ t\ n;-p;-q \leq x \ll aL+pSum\ t\ n)$
have $\forall n\ .\ t\ n;-q \leq -p*x \ll --p;-q \wedge pSum\ t\ n;-q \leq -p*x \ll --p;-q$
proof
fix *n*
show $t\ n;-q \leq -p*x \ll --p;-q \wedge pSum\ t\ n;-q \leq -p*x \ll --p;-q$
proof (*induct n rule*: *nat-less-induct*)
fix *n*
assume $\forall m{<}n\ .\ t\ m;-q \leq -p*x \ll --p;-q \wedge pSum\ t\ m;-q \leq -p*x \ll --p;-q$
hence *2*: $pSum\ t\ n;-q \leq -p*x \ll --p;-q$ **using** *1*
by (*cases n*,
smt bs-mult-left-zero pSum.simps(1) pre-closed sub-mult-closed zero-least-test,
smt least-upper-bound mult-distr-plus-right pSum.simps(2) pSum-test pre-closed sub-mult-closed test-seq-def)
hence *3*: $x \ll (aL+pSum\ t\ n);-q \leq x \ll -p*x \ll --p;-q$ **using** *1*
by (*smt least-upper-bound mult-distr-plus-right one-def pSum-test plus-closed pre-closed pre-iso sub-comm sub-mult-closed while-soundness*)
have $t\ n;-p;-q \leq x \ll (aL+pSum\ t\ n);-q$ **using** *1*
by (*smt greatest-lower-bound lower-bound-right one-def pSum-test plus-closed pre-closed pre-distrib sub-assoc sub-mult-closed test-seq-def transitive*)
hence *4*: $-p;(t\ n;-q) \leq -p;(-p*x \ll --p;-q)$ **using** *1 3*
by (*smt greatest-lower-bound lower-bound-left one-def pSum-test plus-closed pre-closed sub-assoc sub-comm sub-mult-closed test-seq-def transitive while-pre-then*)
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 $t\ n;-q \leq -p*x \ll --p;-q \wedge pSum\ t\ n;-q \leq -p*x \ll --p;-q$ **using** *1 2 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-distr-plus-right mult-right-dist-Sum one-def plus-closed pre-closed sub-comm sub-mult-closed while-soundness*)
qed

lemma *while-soundness-2*: *test-seq* $t \wedge -r \leq Sum\ t \wedge (\forall n\ .\ t\ n;-p \leq x \ll pSum\ t\ n) \rightarrow -r \leq -p*x \ll 1$
proof
assume *1*: *test-seq* $t \wedge -r \leq Sum\ t \wedge (\forall n\ .\ t\ n;-p \leq x \ll pSum\ t\ n)$
hence *2*: $\forall n\ .\ t\ n;-p;--Sum\ t \leq x \ll aL+pSum\ t\ n$
by (*smt leq-def one-def pSum-test plus-closed pre-closed pre-iso sub-assoc sub-comm sub-mult-closed test-seq-def upper-bound-right*)
have $-p;--Sum\ t \leq x \ll --Sum\ t$ **using** *1*
by (*smt Sum-test mult-right-dist-Sum pSum-below-Sum pSum-test pre-closed pre-iso sub-comm sub-mult-closed test-seq-def transitive*)
hence $--Sum\ t \leq -p*x \ll --p;--Sum\ t$ **using** *1 2*
by (*smt Sum-test one-def pre-closed upper-bound-right 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*: $p(\!|x|\!)\,q \Rightarrow p\langle x\rangle q$
by (*induct rule*: *derived-hoare-triple.induct*,
metis atom-prog pre-pre valid-hoare-triple-def pre-closed reflexive pre-expression-test,
metis valid-hoare-triple-def pre-expression-test pre-compose seq-prog,
metis valid-hoare-triple-def ite-import-mult pre-expression-test cond-prog test-pre,
smt valid-hoare-triple-def pre-expression-test conj-pre neg-pre test-pre while-prog while-soundness-1,
smt valid-hoare-triple-def pre-expression-test pre-pre pre-iso transitive)

end

class *hoare-calculus-completeness* = *hoare-calculus* +
assumes *while-bnd*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow p{*}x{\ll}1 \leq aL + \mathit{bnd}(p;x)$

begin

lemma *while-complete*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \wedge (\forall\, r{\in}\mathit{Pre\text{-}expression}\ .\ x{\ll}r(\!|x|\!)r) \rightarrow p{*}x{\ll}q(\!|p{*}x|\!)q$
proof
assume *1*: $p \in \mathit{Test\text{-}expression} \wedge q \in \mathit{Pre\text{-}expression} \wedge x \in \mathit{While\text{-}program} \wedge (\forall\, r{\in}\mathit{Pre\text{-}expression}\ .\ x{\ll}r(\!|x|\!)r)$
hence *2*: $p;(p{*}x{\ll}q)(\!|x|\!)p{*}x{\ll}q$
by (*smt cons-pre-trip pre-pre pre-expression-test while-pre-then while-prog test-pre*)
let $?t = \lambda n\ .\ (p;x)\hat{}n{\ll}0$
have *3*: *test-seq ?t*
by (*smt one-compl pre-closed test-seq-def*)
have *4*: $p{*}x{\ll}q \leq aL + \mathit{bnd}(p;x)$ **using** *1*
by (*smt bnd-test one-def plus-closed pre-below-pre-one pre-closed pre-expression-test transitive while-bnd*)
have $\forall\, n\ .\ ?t\ n;p;(p{*}x{\ll}q)(\!|x|\!)aL{+}pSum\ ?t\ n$
proof
fix *n*
show $?t\ n;p;(p{*}x{\ll}q)(\!|x|\!)aL{+}pSum\ ?t\ n$
proof (*cases n*)
show $n = 0 \Rightarrow ?t\ n;p;(p{*}x{\ll}q)(\!|x|\!)aL{+}pSum\ ?t\ n$ **using** *1*
by (*smt aL-pre-expression bnd-pre-expression bs-mult-left-zero cons-trip disj-pre one-compl pSum-pre-expression power-0 pre-closed pre-expression-test pre-one test-pre zero-least-test*)
next
fix *m*
have $?t\ m \leq aL + pSum\ ?t\ (\mathit{Suc}\ m)$ **using** *3*
by (*smt one-compl one-def pSum.simps*(*2*) *pSum-test plus-closed pre-closed transitive upper-bound-right*)
thus $n = \mathit{Suc}\ m \Rightarrow ?t\ n;p;(p{*}x{\ll}q)(\!|x|\!)aL{+}pSum\ ?t\ n$ **using** *1*
by (*smt aL-pre-expression bnd-pre-expression conj-pre cons-trip disj-pre lower-bound-right one-compl pSum-pre-expression power-Suc pre-expression-test pre-import-composition pre-pre sub-comm test-pre transitive while-prog*)
qed
qed
hence $p \in \mathit{Test\text{-}expression} \wedge \mathit{test\text{-}seq}\ ?t \wedge p;(p{*}x{\ll}q)(\!|x|\!)p{*}x{\ll}q \wedge p{*}x{\ll}q \leq aL + \mathit{Sum}\ ?t \wedge (\forall\, n\ .\ ?t\ n;p;(p{*}x{\ll}q)(\!|x|\!)aL{+}pSum\ ?t\ n)$ **using** *1 2 3 4*
by (*metis Sum-def bnd-def*)
thus $p{*}x{\ll}q(\!|p{*}x|\!)q$ **using** *1*
by (*smt cons-post-trip neg-pre pre-expression-test test-pre while-pre-else while-trip*)
qed

lemma *pre-completeness*: $[\![\ x \in \mathit{While\text{-}program}\ ;\ q \in \mathit{Pre\text{-}expression}\ ]\!] \Rightarrow x \mathrel{«} q⦇x⦈q$
by (*induct arbitrary*: *q rule*: *While-program.induct*,
metis atom-trip,
metis pre-pre pre-seq seq-trip pre-expression-test,
smt cond-prog cond-trip cons-pre-trip ite-pre-else ite-pre-then neg-pre pre-pre pre-expression-test test-pre,
metis while-complete)

theorem *completeness*: $p⟨x⟩q \longrightarrow p⦇x⦈q$
by (*metis valid-hoare-triple-def pre-completeness reflexive pre-expression-test cons-trip*)

theorem *soundness-completeness*: $p⦇x⦈q \longleftrightarrow p⟨x⟩q$
by (*smt soundness completeness*)

end

class *hoare-rules* = *whiledo* + *complete-pre* + *hoare-triple* +
assumes *rule-pre*: $x \mathrel{«} -q\{\!|x|\!\}-q$
assumes *rule-seq*: $-p\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-r \longrightarrow -p\{\!|x;y|\!\}-r$
assumes *rule-cond*: $-p;-q\{\!|x|\!\}-r \wedge --p;-q\{\!|y|\!\}-r \longrightarrow -q\{\!|x \triangleleft -p \triangleright y|\!\}-r$
assumes *rule-while*: *test-seq t* $\wedge -p;-q\{\!|x|\!\}-q \wedge -q \leq aL + \mathit{Sum}\ t \wedge (\forall\, n\ .\ t\ n;-p;-q\{\!|x|\!\}aL+\mathit{pSum}\ t\ n) \longrightarrow -q\{\!|-p*x|\!\}--p;-q$
assumes *rule-cons*: $-p \leq -q \wedge -q\{\!|x|\!\}-r \wedge -r \leq -s \longrightarrow -p\{\!|x|\!\}-s$
assumes *rule-disj*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-s \longrightarrow -p+-q\{\!|x|\!\}-r+-s$
assumes *rule-conj*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-s \longrightarrow -p;-q\{\!|x|\!\}-r;-s$

begin

lemma *rule-cons-pre*: $-p \leq -q \wedge -q\{\!|x|\!\}-r \longrightarrow -p\{\!|x|\!\}-r$
by (*metis rule-cons reflexive*)

lemma *rule-cons-pre-mult*: $-q\{\!|x|\!\}-r \longrightarrow -p;-q\{\!|x|\!\}-r$
by (*metis rule-cons-pre lower-bound-left sub-comm sub-mult-closed*)

lemma *rule-cons-pre-plus*: $-p+-q\{\!|x|\!\}-r \longrightarrow -p\{\!|x|\!\}-r$
by (*metis rule-cons-pre upper-bound-left plus-closed*)

lemma *rule-cons-post*: $-q\{\!|x|\!\}-r \wedge -r \leq -s \longrightarrow -q\{\!|x|\!\}-s$
by (*metis rule-cons reflexive*)

lemma *rule-cons-post-mult*: $-q\{\!|x|\!\}-r;-s \longrightarrow -q\{\!|x|\!\}-s$
by (*metis rule-cons-post lower-bound-left sub-comm sub-mult-closed*)

lemma *rule-cons-post-plus*: $-q\{\!|x|\!\}-r \longrightarrow -q\{\!|x|\!\}-r+-s$
by (*metis rule-cons-post upper-bound-left plus-closed*)

lemma *rule-disj-pre*: $-p\{\!|x|\!\}-r \wedge -q\{\!|x|\!\}-r \longrightarrow -p+-q\{\!|x|\!\}-r$
by (*metis rule-disj plus-idempotent*)

lemma *rule-conj-post*: $-p\{\!|x|\!\}-r \wedge -p\{\!|x|\!\}-s \longrightarrow -p\{\!|x|\!\}-r;-s$
by (*metis rule-conj mult-idempotent*)

end

class *hoare-calculus-valid* = *hoare-calculus-completeness* + *hoare-triple* +
assumes *hoare-triple-valid*: $-p\{\!|x|\!\}-q \leftrightarrow -p \leq x\mathbin{«}-q$

begin

lemma *valid-hoare-triple-same*: $p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program* $\longrightarrow p\{\!|x|\!\}q = p\langle x\rangle q$
by (*metis valid-hoare-triple-def hoare-triple-valid pre-expression-test*)

lemma *derived-hoare-triple-same*: $p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program* $\longrightarrow p\{\!|x|\!\}q = p(\!|x|\!)q$
by (*metis valid-hoare-triple-same soundness-completeness*)

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 one-def pSum-test plus-closed pre-closed sub-mult-closed test-seq-def while-soundness-1*)
apply (*smt hoare-triple-valid pre-iso transitive pre-closed*)
apply (*smt hoare-triple-valid pre-closed least-upper-bound plus-closed upper-bound-left upper-bound-right transitive pre-iso*)
apply (*smt hoare-triple-valid pre-closed greatest-lower-bound sub-mult-closed lower-bound-left lower-bound-right transitive pre-distrib*)
done

lemma *nat-test-rule-while*: *nat-test* $t\ s \wedge -q \leq s \wedge (\forall\, n\ .\ t\ n;-p;-q\{\!|x|\!\}pSum\ t\ n) \wedge -p;-q\{\!|x|\!\}-q \longrightarrow -q\{\!|-p{*}x|\!\}--p;-q$
by (*smt hoare-triple-valid nat-test-def nat-test-pre pSum-test-nat sub-mult-closed*)

lemma *test-seq-rule-while*: *test-seq* $t \wedge -p;-q\{\!|x|\!\}-q \wedge -q \leq aL + Sum\ t \wedge (\forall\, n\ .\ t\ n;-p;-q\{\!|x|\!\}aL{+}pSum\ t\ n) \longrightarrow -q\{\!|-p{*}x|\!\}--p;-q$
by (*smt hoare-triple-valid one-def pSum-test plus-closed pre-closed sub-mult-closed test-seq-def while-soundness-1*)

lemma *rule-skip*: $-p\{\!|1|\!\}-p$
by (*metis pre-one rule-pre*)

lemma *rule-example-4*: *test-seq* $t \wedge Sum\ t = 1 \wedge -p1\{\!|z1|\!\}-p1;-p2 \wedge -p1;-p2;-p3\{\!|z2|\!\}-p1;-p2 \wedge (\forall\, n\ .\ t\ n;-p1;-p3\{\!|z2|\!\}pSum\ t\ n) \longrightarrow -p1\{\!|z1;(-p3{*}z2)|\!\}-p2;--p3$
proof
assume *1*: *test-seq* $t \wedge Sum\ t = 1 \wedge -p1\{\!|z1|\!\}-p1;-p2 \wedge -p1;-p2;-p3\{\!|z2|\!\}-p1;-p2 \wedge (\forall\, n\ .\ t\ n;-p1;-p3\{\!|z2|\!\}pSum\ t\ n)$
hence $\forall\, n\ .\ t\ n;-p3;(-p1;-p2)\{\!|z2|\!\}aL{+}pSum\ t\ n$
by (*smt lower-bound-left one-def pSum-test plus-closed pre-closed rule-cons sub-assoc sub-comm sub-mult-closed test-seq-def upper-bound-right*)
hence $-p1;-p2\{\!|-p3{*}z2|\!\}--p3;(-p1;-p2)$ **using** *1*
by (*smt one-def one-greatest plus-right-one pre-closed rule-while sub-comm 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-completeness* +
  assumes *pre-one-one*: $x\mathbin{«}1 = 1$

begin

lemma *aL-one*: $aL = 1$
  by (*metis pre-one-one*)

lemma *while-soundness-pc*: $-p;-q \leq x\mathbin{«}-q \longrightarrow -q \leq -p{*}x\mathbin{«}--p;-q$
proof −
  have *test-seq* $(\lambda x\ .\ 0)$
    by (*metis test-seq-def zero-double-compl*)
  thus *?thesis*
    by (*metis Sum-test bs-mult-left-zero one-def one-greatest pSum-test plus-left-one pre-one-one while-soundness-1 zero-least-test*)
qed

inductive $dp :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
  where *atom-trip*: $p \in Pre\text{-}expression \wedge x \in Atomic\text{-}program \Rightarrow dp\ (x\mathbin{«}p)\ x\ p$
    | *seq-trip*: $dp\ p\ x\ q \wedge dp\ q\ y\ r \Rightarrow dp\ p\ (x;y)\ r$
    | *cond-trip*: $p \in Test\text{-}expression \wedge q \in Pre\text{-}expression \wedge dp\ (p;q)\ x\ r \wedge dp\ (-p;q)\ y\ r \Rightarrow dp\ q\ (x\lhd p\rhd y)\ r$
    | *while-trip*: $p \in Test\text{-}expression \wedge dp\ (p;q)\ x\ q \Rightarrow dp\ q\ (p{*}x)\ (-p;q)$
    | *cons-trip*: $p \in Pre\text{-}expression \wedge s \in Pre\text{-}expression \wedge p \leq q \wedge dp\ q\ x\ r \wedge r \leq s \Rightarrow dp\ p\ x\ s$

lemma *derived-dp*: $p(\!|x|\!)q \leftrightarrow dp\ p\ x\ q$
  by (*rule iffI*,
    induct rule: *derived-hoare-triple.induct*,
    metis dp.atom-trip, *metis dp.seq-trip*, *metis dp.cond-trip*, *metis dp.while-trip*, *metis dp.cons-trip*,
    induct rule: *dp.induct*,
    metis derived-hoare-triple.atom-trip, *metis derived-hoare-triple.seq-trip*, *metis derived-hoare-triple.cond-trip*,
    metis conj-pre neg-pre test-pre soundness-completeness pre-expression-test valid-hoare-triple-def while-prog while-soundness-pc,
    metis derived-hoare-triple.cons-trip)

end

class *hoare-calculus-pc-valid* = *hoare-calculus-pc* + *hoare-calculus-valid*

begin

lemma *rule-while-pc*: $-p;-q\{\!|x|\!\}-q \longrightarrow -q\{\!|-p{*}x|\!\}--p;-q$
  by (*metis hoare-triple-valid sub-mult-closed while-soundness-pc*)

lemma *rule-alternation*: $-p\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-p \longrightarrow -p\{\!|-r{*}x;y|\!\}--r;-p$
  by (*metis rule-seq rule-cons-pre-mult rule-while-pc*)

lemma *rule-alternation-context*: $-p\{\!|v|\!\}-p \wedge -p\{\!|w|\!\}-q \wedge -q\{\!|x|\!\}-q \wedge -q\{\!|y|\!\}-p \wedge -p\{\!|z|\!\}-p \longrightarrow -p\{\!|-r{*}v;w;x;y;z|\!\}--r;-p$
  by (*metis rule-seq rule-cons-pre-mult rule-while-pc*)

lemma *rule-example-3*: $-p\{\!|x|\!\}--p \wedge --p\{\!|x|\!\}-p \wedge -q\{\!|x|\!\}-q \wedge -r\{\!|x|\!\}-r \wedge -p\{\!|y|\!\}-p \wedge -r\{\!|y|\!\}-q \wedge --p\{\!|z|\!\}--p \wedge -q\{\!|z|\!\}-r \longrightarrow -p;-q+--p;-r\{\!|-s{*}x;(y\lhd -p\rhd z)|\!\}--s;(-p;-q+--p;-r)$
proof (*rule*, (*erule conjE*)+)
  assume $-p\{\!|x|\!\}--p$ **and** $-q\{\!|x|\!\}-q$ **and** $--p\{\!|x|\!\}-p$ **and** $-r\{\!|x|\!\}-r$
  hence *t1*: $-p;-q+--p;-r\{\!|x|\!\}--p;-q+-p;-r$
    by (*smt rule-conj rule-disj sub-mult-closed*)
  assume $-p\{\!|y|\!\}-p$ **and** $-r\{\!|y|\!\}-q$
  hence $-p;-r\{\!|y|\!\}-p;-q+--p;-r$
    by (*smt rule-conj 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 mult-idempotent plus-left-zero sub-assoc sub-mult-closed*)
  assume $--p\{\!|z|\!\}--p$ **and** $-q\{\!|z|\!\}-r$
  hence $--p;-q\{\!|z|\!\}-p;-q+--p;-r$
    by (*smt plus-comm rule-conj 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 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 *modal-itering-0 = modal-precondition + itering-T + atoms + ite + while +*
  assumes *ite-def*: $x\lhd p\rhd y = p\ ;\ x + -p\ ;\ y$
  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*
  by (*unfold-locales*,
      smt a-d-closed box-a-export box-left-dist-add box-x-a case-duality d-def ite-def pre-def)

subclass *whiledo*
  by (*unfold-locales*,
      metis circ-loop-fixpoint ite-def mult-associative mult-right-one while-def)

subclass *while-program* **..**

lemma *pre-while-1*: $-p;(-p*x)\text{«}1 = -p*x\text{«}1$
proof −
  have $--p;(-p;(-p*x)\text{«}1) = --p;(-p*x\text{«}1)$
    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 *a-while-soundness-1*: $-p;-q \leq |(-p\ ;\ x)^\circ\ ;\ --p]1\ ;\ |x](-q) \rightarrow -q \leq |(-p\ ;\ x)^\circ\ ;\ --p](--p;-q)$
proof
  assume $-p;-q \leq |(-p\ ;\ x)^\circ\ ;\ --p]1\ ;\ |x](-q)$
  hence *1*: $-p;-q \leq |(-p\ ;\ x)^\circ\ ;\ --p]1 \wedge -p;-q \leq |x](-q)$
    by (*smt a-dist-add box-x-a greatest-lower-bound one-double-compl*)
  hence *2*: $-p;(-q;(-p\ ;\ x)^\circ;0) \leq Z$
    by (*metis a-dist-add a-greatest-left-absorber box-x-1 bs-mult-right-zero mult-associative*)
  have $--p;(-q;(-p\ ;\ x)^\circ;0) = 0$
    by (*smt a-dist-add add-commutative add-right-zero mult-associative mult-left-dist-add mult-left-zero mult-right-dist-add mult-right-one circ-left-unfold zero-def*)
  hence *3*: $-q;(-p\ ;\ x)^\circ;0 \leq Z$ **using** *2*
    by (*metis a-export-a a-greatest-left-absorber a-strict d-def less-eq-def zero-least*)
  have $-q\ ;\ -p\ ;\ x\ ;\ Z \leq Z$ **using** *1*
    by (*metis a-Z a-below-one a-dist-add a-greatest-left-absorber add-commutative box-left-mult box-right-isotone box-x-0 mult-associative order-trans*)
  hence $-p\ ;\ x\ ;\ (--q\ ;\ T + Z) \leq --q\ ;\ T + Z$ **using** *1*
    by (*smt Z-top a-dist-add a-greatest-left-absorber add-associative box-def circ-increasing circ-top less-eq-def mult-associative mult-isotone mult-left-dist-add shunting-T sub-comm*)
  hence $(-p\ ;\ x)^\circ\ ;\ --q \leq --q\ ;\ T + Z + (-p\ ;\ x)^\circ\ ;\ 0$
    by (*metis add-commutative circ-simulate-absorb mult-associative mult-left-subdist-add-right order-trans top-right-mult-increasing*)
  hence $-q\ ;\ (-p\ ;\ x)^\circ\ ;\ --q \leq -q\ ;\ Z + -q\ ;\ (-p\ ;\ x)^\circ\ ;\ 0$
    by (*smt a-complement add-left-zero add-right-divisibility mult-associative mult-left-dist-add mult-left-zero*)
  also have $... \leq Z$ **using** *3*
    by (*metis add-least-upper-bound add-right-upper-bound shunting-T*)
  finally have $-q;(-p;x)^\circ \leq -q;(-p;x)^\circ;-q + Z$
    by (*metis case-split-right-add mult-associative order-refl*)
  hence $-q;(-p;x)^\circ;--p;-(--p;-q) \leq (-q;(-p;x)^\circ;-q + Z);--p;--q$
    by (*metis a-export-a d-def mult-associative mult-compl-intro mult-left-isotone*)
  also have $... \leq Z$ **using** *3*
    by (*smt Z-mult-decreasing a-dist-add add-commutative add-least-upper-bound mult-associative mult-left-zero mult-right-dist-add zero-def*)
  finally show $-q \leq |(-p\ ;\ x)^\circ\ ;\ --p](--p;-q)$
    by (*metis a-greatest-left-absorber box-def mult-associative*)
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*)

end

class *modal-itering-1* = *modal-itering-0* + *kleene-itering* + *complete-pre* + *hoare-triple* +
  assumes *split-Z*: $x\ ;\ Z \leq x\ ;\ 0 + Z$
  assumes *aL-circ*: $|x^*]y \leq |aL\ ;\ x^\circ]y$
  assumes *hoare-triple-def*: $p\{\!|x|\!\}q \leftrightarrow p \leq |x]q$

begin

lemma *box-star-induct*: $-p \leq |x](-p) \rightarrow -p \leq |x^*](-p)$
proof
  assume $-p \leq |x](-p)$
  hence $x;--p;T \leq Z + --p;T$
    by (*metis Z-top add-commutative box-demodalisation-2 mult-associative mult-left-isotone shunting-T*)
  hence $x;(Z + --p;T) + --p \leq Z + --p;T$
    by (*smt add-least-upper-bound add-left-upper-bound add-right-upper-bound mult-associative mult-left-dist-add mult-right-isotone order-trans split-Z top-right-mult-increasing zero-least*)
  thus $-p \leq |x^*](-p)$
    by (*metis add-commutative box-demodalisation-2 mult-associative shunting-T star-left-induct*)
qed

lemma *box-circ-induct*: $-p \leq |x](-p) \rightarrow -p;aL \leq |x^\circ](-p)$
  by (*smt aL-circ box-left-mult box-star-induct one-def order-trans plus-comm pre-closed pre-def pre-test shunting-right*)

lemma *a-while-soundness*: $-p;-q \leq |x](-q) \rightarrow -q;aL \leq |(-p;x)^\circ;--p](--p;-q)$
  by (*metis add-right-upper-bound box-circ-induct box-def box-import-shunting box-right-dist-add box-right-isotone mult-associative mult-compl-intro mult-deMorgan order-trans*)

subclass *hoare-calculus*
  by (*unfold-locales*,
    metis a-while-soundness while-def pre-def)

lemma *rule-skip-valid*: $-p\{\!|1|\!\}-p$
  by (*metis box-1-a hoare-triple-def reflexive*)

end

class *modal-itering-2* = *modal-itering-1* +
  assumes *bnd-circ*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow |(p;x)^\circ]1 \leq aL + bnd(p;x)$

begin

lemma *a-while-bnd*: $p \in \mathit{Test\text{-}expression} \wedge x \in \mathit{While\text{-}program} \rightarrow |(p;x)^\circ;-p]1 \leq aL + bnd(p;x)$
  by (*metis bnd-circ box-a-1 box-left-mult*)

subclass *hoare-calculus-completeness*
  by (*unfold-locales*,
    metis a-while-bnd while-def pre-def)

subclass *hoare-calculus-valid*

by (*unfold-locales*,
metis hoare-triple-def pre-def)

end

class *modal-itering-pc* = *modal-itering-2* +
assumes *mult-zero-below-Z*: $x\ ;\ 0 \leq Z$

begin

subclass *hoare-calculus-pc*
by (*unfold-locales*,
metis mult-zero-below-Z pre-pc-Z)

subclass *hoare-calculus-pc-valid* **..**

end

class *pre-post* =
fixes *pre-post* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infix** $\dashv$ 55)

class *pre-post-spec* = *semiring-T* + *precondition* + *pre-post* +
assumes *pre-post-galois*: $-p \leq x\mathbin{«}-q \leftrightarrow x \leq -p\dashv -q$

begin

lemma *pre-left-antitone*: $x \leq y \rightarrow y\mathbin{«}-q \leq x\mathbin{«}-q$
by (*smt order-refl order-trans pre-closed pre-post-galois*)

lemma *pre-left-subdist*: $x+y\mathbin{«}-q \leq x\mathbin{«}-q$
by (*metis add-left-upper-bound pre-left-antitone*)

lemma *pre-post-left-antitone*: $-p \leq -q \rightarrow -q\dashv -r \leq -p\dashv -r$
by (*metis order-refl order-trans pre-post-galois*)

lemma *pre-post-left-subdist*: $-p+-q\dashv -r \leq -p\dashv -r$
by (*metis add-left-upper-bound plus-closed pre-post-left-antitone*)

lemma *pre-post-left-supdist*: $-p\dashv -r \leq -p;-q\dashv -r$
by (*metis lower-bound-left pre-post-left-antitone sub-mult-closed*)

lemma *pre-pre-post*: $x \leq (x\mathbin{«}-p)\dashv -p$
by (*metis order-refl pre-closed pre-post-galois*)

lemma *pre-post-pre*: $-p \leq (-p\dashv -q)\mathbin{«}-q$

by (*metis eq-refl pre-post-galois*)

lemma *pre-post-zero-top*: $0 \rightarrow\!\!\!\!\leftarrow -q = T$
by (*metis eq-iff pre-post-galois top-greatest zero-double-compl zero-least*)

lemma *pre-post-pre-one*: $(1 \rightarrow\!\!\!\!\leftarrow -q) \ll -q = 1$
by (*metis add-left-zero leq-plus-right-one one-compl one-def pre-closed pre-post-pre*)

lemma *pre-post-right-isotone*: $-p \leq -q \rightarrow -r \rightarrow\!\!\!\!\leftarrow -p \leq -r \rightarrow\!\!\!\!\leftarrow -q$
by (*metis order-trans pre-iso pre-post-galois pre-post-pre*)

lemma *pre-post-right-subdist*: $-r \rightarrow\!\!\!\!\leftarrow -p \leq -r \rightarrow\!\!\!\!\leftarrow -p + -q$
by (*metis add-left-upper-bound plus-closed pre-post-right-isotone*)

lemma *pre-post-right-supdist*: $-r \rightarrow\!\!\!\!\leftarrow -p ; -q \leq -r \rightarrow\!\!\!\!\leftarrow -p$
by (*metis lower-bound-left pre-post-right-isotone sub-mult-closed*)

lemma *pre-post-seq-sub-associative*: $(-p \rightarrow\!\!\!\!\leftarrow -q) ; -r \leq -p \rightarrow\!\!\!\!\leftarrow -q ; -r$
by (*smt leq-def pre-closed pre-distrib pre-post-galois pre-post-right-isotone pre-seq pre-test-test-same sub-mult-closed test-below-pre-test*)

lemma *pre-post-right-import-mult*: $(-p \rightarrow\!\!\!\!\leftarrow -q) \ ; \ -r = (-p \rightarrow\!\!\!\!\leftarrow -q ; -r) \ ; \ -r$
by (*metis antisym mult-associative mult-idempotent mult-left-isotone pre-post-right-supdist pre-post-seq-sub-associative*)

lemma *seq-pre-post-sub-associative*: $-r ; (-p \rightarrow\!\!\!\!\leftarrow -q) \leq --r + -p \rightarrow\!\!\!\!\leftarrow -q$
by (*smt add-least-upper-bound leq-def mult-left-isotone mult-left-one mult-right-one one-def plus-closed pre-neg-mult pre-post-galois*)

lemma *pre-post-left-import-add*: $-r \ ; \ (-p \rightarrow\!\!\!\!\leftarrow -q) = -r \ ; \ (--r + -p \rightarrow\!\!\!\!\leftarrow -q)$
by (*smt add-commutative antisym mult-associative mult-idempotent mult-right-isotone pre-post-left-subdist seq-pre-post-sub-associative*)

lemma *pre-post-import-same*: $-p ; (-p \rightarrow\!\!\!\!\leftarrow -q) = -p ; (1 \rightarrow\!\!\!\!\leftarrow -q)$
by (*metis double-negation plus-compl pre-post-left-import-add*)

lemma *pre-post-import-complement*: $--p ; (-p \rightarrow\!\!\!\!\leftarrow -q) = --p ; T$
by (*metis mult-idempotent plus-cases plus-closed pre-post-left-import-add pre-post-zero-top zero-def zero-double-compl*)

lemma *pre-post-export*: $-p \rightarrow\!\!\!\!\leftarrow -q = (1 \rightarrow\!\!\!\!\leftarrow -q) + --p ; T$
proof −
have *1*: $-p ; (-p \rightarrow\!\!\!\!\leftarrow -q) = -p ; ((1 \rightarrow\!\!\!\!\leftarrow -q) + --p ; T)$
by (*metis add-right-zero mult-associative mult-left-dist-add mult-left-zero pre-post-import-same zero-def*)
have $--p ; (-p \rightarrow\!\!\!\!\leftarrow -q) = --p ; ((1 \rightarrow\!\!\!\!\leftarrow -q) + --p ; T)$
by (*metis add-right-top mult-associative mult-idempotent mult-left-dist-add pre-post-import-complement*)
thus *?thesis* **using** *1*
by (*metis case-split-left-equal plus-compl*)
qed

lemma *pre-post-left-dist-mult*: $-p ; -q \rightarrow\!\!\!\!\leftarrow -r = (-p \rightarrow\!\!\!\!\leftarrow -r) + (-q \rightarrow\!\!\!\!\leftarrow -r)$
proof −

have $\forall\, p\ q\ .\ -p\ ;\ (-p;-q \dashv -r) = -p\ ;\ (-q \dashv -r)$
by (*metis add-commutative plus-compl-intro pre-post-left-import-add sub-mult-closed*)
hence *1*: $(-p + -q)\ ;\ (-p;-q \dashv -r) \leq (-p \dashv -r) + (-q \dashv -r)$
by (*smt add-commutative add-least-upper-bound add-right-upper-bound mult-left-one mult-right-dist-add plus-left-one sub-comm*)
have $-(-p + -q)\ ;\ (-p;-q \dashv -r) = -(-p + -q)\ ;\ T$
by (*smt add-associative add-commutative one-compl plus-absorb plus-closed plus-right-zero pre-post-left-import-add pre-post-zero-top sub-mult-closed*)
hence $-(-p + -q)\ ;\ (-p;-q \dashv -r) \leq (-p \dashv -r) + (-q \dashv -r)$
by (*smt add-left-upper-bound mult-left-one mult-right-subdist-add-right order-trans plus-left-one mult-associative plus-deMorgan pre-post-import-complement sub-comm*)
thus *?thesis* **using** *1*
by (*smt add-least-upper-bound antisym case-split-left order-refl plus-closed plus-compl pre-post-left-supdist sub-comm*)
qed

lemma *pre-post-left-import-mult*: $-r\ ;\ (-p \dashv -q) = -r\ ;\ (-r;-p \dashv -q)$
by (*metis add-commutative mult-left-dist-add mult-right-one one-def pre-post-import-same pre-post-left-dist-mult*)

lemma *pre-post-right-dist-add*: $-p \dashv -q + -r = (-p \dashv -q) + (-p \dashv -r)$
proof −
have *1*: $(-p \dashv -q + -r);-q \leq (-p \dashv -q) + (-p \dashv -r)$
by (*smt add-left-upper-bound mult-absorb mult-left-subdist-add-right mult-right-one order-trans plus-closed plus-left-one pre-post-right-import-mult sub-comm*)
have $(-p \dashv -q + -r);--q = (-p \dashv -r);--q$
by (*smt plus-def pre-post-right-import-mult mult-compl-intro mult-distr-plus-right mult-left-dist-add unique-zero*)
hence $(-p \dashv -q + -r);--q \leq (-p \dashv -q) + (-p \dashv -r)$
by (*metis add-right-upper-bound mult-left-subdist-add-right mult-right-one order-trans plus-left-one*)
thus *?thesis* **using** *1*
by (*metis add-least-upper-bound antisym case-split-right one-greatest plus-comm plus-compl pre-post-right-subdist*)
qed

lemma *pre-post-right-import-add*: $(-p \dashv -q)\ ;\ -r = (-p \dashv -q + --r)\ ;\ -r$
by (*smt bs-mult-right-one case-duality plus-closed plus-comm plus-compl pre-post-right-import-mult sub-comm wnf-lemma-1*)

lemma *pre-left-dist-add*: $x + y \mathbin{«} -q = (x \mathbin{«} -q)\ ;\ (y \mathbin{«} -q)$
by (*smt add-commutative add-isotone antisym greatest-lower-bound pre-closed pre-left-subdist pre-post-galois pre-post-left-dist-mult pre-pre-post sub-mult-closed*)

lemma *pre-post-reflexive*: $1 \leq -p \dashv -p$
by (*metis pre-one pre-post-galois reflexive*)

lemma *pre-post-compose*: $-q \leq -r \rightarrow (-p \dashv -q);(-r \dashv -s) \leq -p \dashv -s$
by (*smt order-trans pre-closed pre-iso pre-post-galois pre-post-pre pre-seq*)

lemma *pre-post-compose-1*: $(-p \dashv -q);(-q \dashv -r) \leq -p \dashv -r$
by (*metis pre-post-compose reflexive*)

lemma *pre-post-compose-2*: $(-p \dashv -p);(-p \dashv -q) = -p \dashv -q$
by (*metis antisym mult-left-isotone mult-left-one pre-post-compose-1 pre-post-reflexive*)

lemma *pre-post-compose-3*: $(-p \dashv -q);(-q \dashv -q) = -p \dashv -q$
by (*metis antisym mult-right-isotone mult-right-one pre-post-compose-1 pre-post-reflexive*)

lemma *pre-post-compose-4*: $(-p \dashv -p);(-p \dashv -p) = -p \dashv -p$
by (*metis pre-post-compose-2*)

lemma *pre-post-one-one*: $x \ll 1 = 1 \leftrightarrow x \leq 1 \dashv 1$
by (*metis eq-iff one-def pre-below-one pre-post-galois*)

lemma *pre-post-shunting*: $x \leq -p;-q \dashv -r \leftrightarrow -p;x \leq -q \dashv -r$
proof −
have $--p;x \leq --p;(-p;-q \dashv -r)$
by (*smt mult-compl mult-left-zero mult-right-isotone pre-post-left-import-mult pre-post-zero-top sub-assoc sub-comm sub-mult-closed top-greatest*)
hence $-p;x \leq -q \dashv -r \rightarrow x \leq -p;-q \dashv -r$
by (*smt add-least-upper-bound case-split-left less-eq-def mult-left-isotone mult-left-one order-refl plus-compl pre-post-left-supdist sub-comm*)
thus *?thesis*
by (*smt mult-right-isotone pre-post-left-import-mult mult-left-isotone mult-left-one one-greatest order-trans*)
qed

end

class *havoc* =
fixes $H :: 'a$

class *semiring-H* = *semiring-T* + *havoc* +
assumes *H-zero* : $H\ ;\ 0 = 0$
assumes *H-split*: $x \leq x\ ;\ 0 + H$

begin

lemma *H-galois*: $x\ ;\ 0 \leq y \leftrightarrow x \leq y + H$
by (*smt H-split H-zero add-associative add-commutative add-left-zero add-right-isotone less-eq-def mult-right-dist-add mult-right-subdist-add-left zero-right-mult-decreasing*)

lemma *H-greatest-finite*: $x\ ;\ 0 = 0 \leftrightarrow x \leq H$
by (*metis H-galois add-left-zero eq-iff zero-least*)

lemma *H-reflexive*: $1 \leq H$
by (*metis H-greatest-finite mult-left-one*)

lemma *H-transitive*: $H = H\ ;\ H$
by (*metis H-greatest-finite H-reflexive H-zero antisym-conv mult-associative mult-right-isotone mult-right-one*)

lemma *T-split-H*: $T\ ;\ 0 + H = T$
by (*metis H-split add-left-top less-eq-def*)

end

class *pre-post-spec-H* = *pre-post-spec* + *modal-precondition* + *semiring-H*

begin

lemma *pre-post-def-iff*: $-p\ ;\ x\ ;\ --q \leq Z \leftrightarrow x \leq Z + --p\ ;\ T + H\ ;\ -q$
proof (*rule iffI*)
  assume *1*: $-p\ ;\ x\ ;\ --q \leq Z$
  have $-p\ ;\ x\ ;\ -q \leq -p\ ;\ x\ ;\ 0 + H\ ;\ -q$
    by (*smt H-split bs-mult-left-zero mult-associative mult-isotone mult-right-dist-add reflexive*)
  hence $-p\ ;\ x\ ;\ -q \leq -p\ ;\ x\ ;\ --q + H\ ;\ -q$
    by (*smt add-commutative add-right-isotone mult-right-isotone order-trans zero-least*)
  hence $-p\ ;\ x \leq Z + H\ ;\ -q$ **using** *1*
    by (*smt add-associative add-commutative case-split-right-add less-eq-def*)
  thus $x \leq Z + --p\ ;\ T + H\ ;\ -q$
    by (*smt add-associative add-commutative case-split-left-add mult-right-isotone top-greatest*)
next
  assume $x \leq Z + --p\ ;\ T + H\ ;\ -q$
  hence $-p\ ;\ x\ ;\ --q \leq (-p\ ;\ Z + -p\ ;\ H\ ;\ -q)\ ;\ --q$
    by (*smt a-complement add-right-zero mult-associative mult-isotone mult-left-dist-add mult-left-zero reflexive*)
  also have $... = -p\ ;\ Z\ ;\ --q$
    by (*smt H-zero a-complement add-commutative add-left-zero bs-mult-right-zero mult-associative mult-right-dist-add*)
  finally show $-p\ ;\ x\ ;\ --q \leq Z$
    by (*metis Z-mult-decreasing a-below-one a-greatest-left-absorber a-strict mult-associative order-trans*)
qed

lemma *pre-post-def*: $-p \dashv -q = Z + --p; T + H; -q$
  by (*metis eq-iff pre-Z pre-post-def-iff pre-post-galois*)

end

class *pre-post-spec-whiledo* = *pre-post-spec* + *whiledo*

begin

lemma *nat-test-pre-post*: *nat-test t s* $\wedge\ x \leq -p;-q \dashv -q \wedge -q \leq s \wedge (\forall n\ .\ x \leq t\ n;-p;-q \dashv (pSum\ t\ n)) \rightarrow -p * x \leq -q \dashv --p;-q$
  by (*smt nat-test-def nat-test-pre pSum-test-nat pre-post-galois sub-mult-closed*)

lemma *nat-test-pre-post-2*: *nat-test t s* $\wedge\ -r \leq s \wedge (\forall n\ .\ x \leq t\ n;-p \dashv (pSum\ t\ n)) \rightarrow -p * x \leq -r \dashv 1$
  by (*smt nat-test-def nat-test-pre-2 one-def pSum-test-nat pre-post-galois sub-mult-closed*)

end

class *pre-post-spec-hoare* = *pre-post-spec-whiledo* + *hoare-calculus*

begin

lemma *pre-post-while*: $x \leq -p;-q \dashv -q \rightarrow -p * x \leq -q;aL \dashv --p;-q$
  by (*smt one-def pre-closed pre-post-galois sub-mult-closed while-soundness*)

lemma *while-soundness-1*: *test-seq* $t \wedge x \leq -p;-q \dashv -q \wedge -q \leq aL + Sum\ t \wedge (\forall n\ .\ x \leq t\ n;-p;-q \dashv aL+pSum\ t\ n) \rightarrow -p*x \leq -q \dashv --p;-q$
by (*smt one-def pSum-test plus-closed pre-closed pre-post-galois sub-mult-closed test-seq-def while-soundness-1*)

lemma *while-soundness-2*: *test-seq* $t \wedge -r \leq Sum\ t \wedge (\forall n\ .\ x \leq t\ n;-p \dashv pSum\ t\ n) \rightarrow -p*x \leq -r \dashv 1$
by (*smt one-def pSum-test pre-post-galois sub-mult-closed test-seq-def while-soundness-2*)

end

class *pre-post-spec-hoare-pc* = *pre-post-spec-hoare* + *hoare-calculus-pc*

begin

lemma *pre-post-one-one-top*: $1 \dashv 1 = T$
by (*metis add-left-top less-eq-def pre-one-one pre-post-one-one*)

lemma *pre-post-while-pc*: $x \leq -p;-q \dashv -q \rightarrow -p*x \leq -q \dashv --p;-q$
by (*metis pre-post-galois sub-mult-closed while-soundness-pc*)

end

class *pre-post-L* = *pre-post-spec* + *modal-itering-1* + *itering-L* +
assumes *circ-below-L-add-star*: $x^\circ \leq L + x^*$

begin

lemma *body-abort-loop*: $Z = L \wedge x \leq -p \dashv 1 \rightarrow -p*x \leq 1 \dashv 1$
proof
assume *1*: $Z = L \wedge x \leq -p \dashv 1$
hence $-p\ ;\ x\ ;\ 0 \leq L$
by (*metis a-one one-def pre-Z pre-post-galois*)
hence $(-p\ ;\ x)^\circ\ ;\ 0 \leq L$
by (*smt L-left-zero L-split add-commutative add-left-zero circ-below-L-add-star less-eq-def mult-right-dist-add star-left-induct*)
thus $-p*x \leq 1 \dashv 1$ **using** *1*
by (*metis a-one a-strict box-def bs-mult-right-zero mult-associative pre-def pre-post-one-one while-def*)
qed

end

end