Algebras for Iteration, Infinite Executions and Correctness of Sequential Computations

Walter Guttmann

March 19, 2025

Abstract

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

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

Contents

1 Base

theory *Base*

imports *Stone-Relation-Algebras.Semirings*

begin

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

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

class *n* =
  fixes *n* :: $'a \Rightarrow 'a$

class *diamond* =
  fixes *diamond* :: $'a \Rightarrow 'a \Rightarrow 'a$ (‹| - > -› [*50*,*90*] *95*)

class *box* =
  fixes *box* :: $'a \Rightarrow 'a \Rightarrow 'a$ (‹| -] -› [*50*,*90*] *95*)

context *ord*
begin

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

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

definition *directed* :: $'a\ set \Rightarrow bool$
  where *directed* $X \equiv X \neq \{\} \wedge (\forall x{\in}X . \forall y{\in}X . \exists z{\in}X . x \leq z \wedge y \leq z)$

definition *co-directed* :: $'a\ set \Rightarrow bool$
  where *co-directed* $X \equiv X \neq \{\} \wedge (\forall x{\in}X . \forall y{\in}X . \exists z{\in}X . z \leq x \wedge z \leq y)$

definition *chain* :: $'a$ *set* $\Rightarrow$ *bool*
  where *chain* $X \equiv \forall\, x{\in}X\ .\ \forall\, y{\in}X\ .\ x \le y \lor y \le x$

end

context *order*
begin

lemma *ascending-chain-k*:
  ascending-chain $f \Longrightarrow f\ m \le f\ (m + k)$
  apply (*induct k*)
  apply *simp*
  using *le-add1 lift-Suc-mono-le ord.ascending-chain-def* **by** *blast*

lemma *ascending-chain-isotone*:
  ascending-chain $f \Longrightarrow m \le k \Longrightarrow f\ m \le f\ k$
  using *lift-Suc-mono-le ord.ascending-chain-def* **by** *blast*

lemma *ascending-chain-comparable*:
  ascending-chain $f \Longrightarrow f\ k \le f\ m \lor f\ m \le f\ k$
  by (*meson ascending-chain-isotone linear*)

lemma *ascending-chain-chain*:
  ascending-chain $f \Longrightarrow$ *chain* (*range* f)
  by (*simp add*: *ascending-chain-comparable chain-def*)

lemma *chain-directed*:
  $X \ne \{\} \Longrightarrow$ *chain* $X \Longrightarrow$ *directed* X
  by (*metis chain-def directed-def*)

lemma *ascending-chain-directed*:
  ascending-chain $f \Longrightarrow$ *directed* (*range* f)
  by (*simp add*: *ascending-chain-chain chain-directed*)

lemma *descending-chain-k*:
  descending-chain $f \Longrightarrow f\ (m + k) \le f\ m$
  apply (*induct k*)
  apply *simp*
  using *le-add1 lift-Suc-antimono-le ord.descending-chain-def* **by** *blast*

lemma *descending-chain-antitone*:
  descending-chain $f \Longrightarrow m \le k \Longrightarrow f\ k \le f\ m$
  using *descending-chain-def lift-Suc-antimono-le* **by** *blast*

lemma *descending-chain-comparable*:
  descending-chain $f \Longrightarrow f\ k \le f\ m \lor f\ m \le f\ k$
  by (*meson descending-chain-antitone nat-le-linear*)

lemma *descending-chain-chain*:
  $descending\text{-}chain\ f \Longrightarrow chain\ (range\ f)$
  by (*simp add*: *descending-chain-comparable chain-def*)

lemma *chain-co-directed*:
  $X \neq \{\} \Longrightarrow chain\ X \Longrightarrow co\text{-}directed\ X$
  by (*metis chain-def co-directed-def*)

lemma *descending-chain-codirected*:
  $descending\text{-}chain\ f \Longrightarrow co\text{-}directed\ (range\ f)$
  by (*simp add*: *chain-co-directed descending-chain-chain*)

end

context *semilattice-sup*
begin

lemma *ascending-chain-left-sup*:
  $ascending\text{-}chain\ f \Longrightarrow ascending\text{-}chain\ (\lambda n\ .\ x \sqcup f\ n)$
  using *ascending-chain-def sup-right-isotone* **by** *blast*

lemma *ascending-chain-right-sup*:
  $ascending\text{-}chain\ f \Longrightarrow ascending\text{-}chain\ (\lambda n\ .\ f\ n \sqcup x)$
  using *ascending-chain-def sup-left-isotone* **by** *auto*

lemma *descending-chain-left-add*:
  $descending\text{-}chain\ f \Longrightarrow descending\text{-}chain\ (\lambda n\ .\ x \sqcup f\ n)$
  using *descending-chain-def sup-right-isotone* **by** *blast*

lemma *descending-chain-right-add*:
  $descending\text{-}chain\ f \Longrightarrow descending\text{-}chain\ (\lambda n\ .\ f\ n \sqcup x)$
  using *descending-chain-def sup-left-isotone* **by** *auto*

primrec $pSum0 :: (nat \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a$
  where $pSum0\ f\ 0 = f\ 0$
    $|\ pSum0\ f\ (Suc\ m) = pSum0\ f\ m \sqcup f\ m$

lemma *pSum0-below*:
$\forall i\ .\ f\ i \leq x \Longrightarrow pSum0\ f\ m \leq x$
  apply (*induct m*)
  by *auto*

end

context *non-associative-left-semiring*
begin

lemma *ascending-chain-left-mult*:
  $ascending\text{-}chain\ f \Longrightarrow ascending\text{-}chain\ (\lambda n\ .\ x * f\ n)$

by (*simp add*: *mult-right-isotone ord.ascending-chain-def*)

lemma *ascending-chain-right-mult*:
ascending-chain f $\Longrightarrow$ *ascending-chain* (λn . *f n* $*$ *x*)
by (*simp add*: *mult-left-isotone ord.ascending-chain-def*)

lemma *descending-chain-left-mult*:
descending-chain f $\Longrightarrow$ *descending-chain* (λn . *x* $*$ *f n*)
by (*simp add*: *descending-chain-def mult-right-isotone*)

lemma *descending-chain-right-mult*:
descending-chain f $\Longrightarrow$ *descending-chain* (λn . *f n* $*$ *x*)
by (*simp add*: *descending-chain-def mult-left-isotone*)

end

context *complete-lattice*
begin

lemma *sup-Sup*:
$A \neq \{\} \Longrightarrow$ *sup x* (*Sup A*) = *Sup* ((*sup x*) ‘ *A*)
apply (*rule order.antisym*)
apply (*meson ex-in-conv imageI SUP-upper2 Sup-mono sup.boundedI sup-left-divisibility sup-right-divisibility*)
by (*meson SUP-least Sup-upper sup-right-isotone*)

lemma *sup-SUP*:
$Y \neq \{\} \Longrightarrow$ *sup x* (*SUP y*$\in$*Y* . *f y*) = (*SUP y*$\in$*Y*. *sup x* (*f y*))
apply (*subst sup-Sup*)
by (*simp-all add*: *image-image*)

lemma *inf-Inf*:
$A \neq \{\} \Longrightarrow$ *inf x* (*Inf A*) = *Inf* ((*inf x*) ‘ *A*)
apply (*rule order.antisym*)
apply (*meson INF-greatest Inf-lower inf.sup-right-isotone*)
by (*simp add*: *INF-inf-const1*)

lemma *inf-INF*:
$Y \neq \{\} \Longrightarrow$ *inf x* (*INF y*$\in$*Y* . *f y*) = (*INF y*$\in$*Y*. *inf x* (*f y*))
apply (*subst inf-Inf*)
by (*simp-all add*: *image-image*)

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

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

end

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

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

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

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

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

instance
  by *intro-classes*

end

lemma *id-power*:
  id^*m* = *id*
  apply (*induct m*)
  apply (*simp add*: *one-fun-def*)
  by (*simp add*: *times-fun-def*)

lemma *power-zero-id*:
  f^*0* = *id*
  by (*simp add*: *one-fun-def*)

lemma *power-succ-unfold*:
  f^*Suc m* = $f \circ$ *f*^*m*
  by (*simp add*: *times-fun-def*)

lemma *power-succ-unfold-ext*:
  (*f*^*Suc m*) x = f ((*f*^*m*) x)
  by (*simp add*: *times-fun-def*)

end

2 Omega Algebras

theory *Omega-Algebras*

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

begin

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

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

class *left-omega-algebra* = *left-kleene-algebra* + *omega* +
assumes *omega-unfold*: $y^\omega = y * y^\omega$
assumes *omega-induct*: $x \le z \sqcup y * x \longrightarrow x \le y^\omega \sqcup y^\star * z$
begin

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

lemma *star-bot-below-omega*:
$x^\star * bot \le x^\omega$
using *omega-unfold star-left-induct-equal* **by** *auto*

lemma *star-bot-below-omega-bot*:
$x^\star * bot \le x^\omega * bot$
by (*metis omega-unfold star-left-induct-equal sup-monoid.add-0-left mult-assoc*)

lemma *omega-induct-mult*:
$y \le x * y \Longrightarrow y \le x^\omega$
by (*metis bot-least omega-induct sup.absorb1 sup.absorb2 star-bot-below-omega*)

lemma *omega-sub-dist*:
$x^\omega \le (x \sqcup y)^\omega$
by (*metis eq-refl mult-isotone omega-unfold sup.cobounded1 omega-induct-mult*)

lemma *omega-isotone*:
$x \le y \Longrightarrow x^\omega \le y^\omega$
using *sup-left-divisibility omega-sub-dist* **by** *fastforce*

lemma *omega-induct-equal*:
$y = z \sqcup x * y \Longrightarrow y \le x^\omega \sqcup x^\star * z$
by (*simp add*: *omega-induct*)

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

lemma *omega-one-greatest*:
$x \le 1^\omega$
by (*simp add*: *omega-induct-mult*)

lemma *star-mult-omega*:
$x^\star * x^\omega = x^\omega$
by (*metis order.antisym omega-unfold star.circ-loop-fixpoint*

star-left-induct-mult-equal sup.cobounded2)

lemma *omega-sub-vector*:
$x^\omega * y \le x^\omega$
by (*metis mult-semi-associative omega-unfold omega-induct-mult*)

lemma *omega-simulation*:
$z * x \le y * z \Longrightarrow z * x^\omega \le y^\omega$
by (*smt* (*verit*, *ccfv-threshold*) *mult-isotone omega-unfold order-lesseq-imp mult-assoc omega-induct-mult*)

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

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

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

lemma *omega-simulation-2*:
$y * x \le x * y \Longrightarrow (x * y)^\omega \le x^\omega$
by (*metis mult-right-isotone sup.absorb2 omega-induct-mult omega-slide omega-sub-dist*)

lemma *wagner*:
$(x \sqcup y)^\omega = x * (x \sqcup y)^\omega \sqcup z \Longrightarrow (x \sqcup y)^\omega = x^\omega \sqcup x^\star * z$
by (*smt* (*verit*, *ccfv-SIG*) *order.refl star-left-induct sup.absorb2 sup-assoc sup-commute omega-induct-equal omega-sub-dist*)

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

lemma *omega-sub-dist-1*:
$(x * y^\star)^\omega \le (x \sqcup y)^\omega$
by (*metis left-plus-omega mult-isotone star.circ-sub-dist sup.cobounded1 sup-monoid.add-commute omega-isotone*)

lemma *omega-sub-dist-2*:
$(x^\star * y)^\omega \le (x \sqcup y)^\omega$
by (*metis mult-isotone star.circ-sub-dist sup.cobounded2 omega-isotone right-plus-omega*)

lemma *omega-star*:
  $(x^\omega)^\star = 1 \sqcup x^\omega$
  by (*metis antisym-conv star.circ-mult-increasing star-left-unfold-equal omega-sub-vector*)

lemma *omega-mult-omega-star*:
  $x^\omega * x^{\omega\star} = x^\omega$
  by (*simp add*: *order.antisym star.circ-mult-increasing omega-sub-vector*)

lemma *omega-sum-unfold-1*:
  $(x \sqcup y)^\omega = x^\omega \sqcup x^\star * y * (x \sqcup y)^\omega$
  by (*metis mult-right-dist-sup omega-unfold mult-assoc wagner*)

lemma *omega-sum-unfold-2*:
  $(x \sqcup y)^\omega \leq (x^\star * y)^\omega \sqcup (x^\star * y)^\star * x^\omega$
  using *omega-induct-equal omega-sum-unfold-1* **by** *auto*

lemma *omega-sum-unfold-3*:
  $(x^\star * y)^\star * x^\omega \leq (x \sqcup y)^\omega$
  using *star-left-induct-equal omega-sum-unfold-1* **by** *auto*

lemma *omega-decompose*:
  $(x \sqcup y)^\omega = (x^\star * y)^\omega \sqcup (x^\star * y)^\star * x^\omega$
  by (*metis sup.absorb1 sup-same-context omega-sub-dist-2 omega-sum-unfold-2 omega-sum-unfold-3*)

lemma *omega-loop-fixpoint*:
  $y * (y^\omega \sqcup y^\star * z) \sqcup z = y^\omega \sqcup y^\star * z$
  apply (*rule order.antisym*)
  apply (*smt* (*verit*, *ccfv-threshold*) *eq-refl mult-isotone mult-left-sub-dist-sup omega-induct omega-unfold star.circ-loop-fixpoint sup-assoc sup-commute sup-right-isotone*)
  by (*smt* (*z3*) *mult-left-sub-dist-sup omega-unfold star.circ-loop-fixpoint sup.left-commute sup-commute sup-right-isotone*)

lemma *omega-loop-greatest-fixpoint*:
  $y * x \sqcup z = x \Longrightarrow x \leq y^\omega \sqcup y^\star * z$
  by (*simp add*: *sup-commute omega-induct-equal*)

lemma *omega-square*:
  $x^\omega = (x * x)^\omega$
  using *order.antisym omega-unfold order-refl mult-assoc omega-induct-mult omega-simulation-2* **by** *auto*

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

lemma *mult-bot-add-omega*:

$(x \sqcup y * bot)^\omega = x^\omega \sqcup x^\star * y * bot$
by (*metis mult-left-zero sup-commute mult-assoc mult-bot-omega omega-decompose omega-loop-fixpoint*)

lemma *omega-mult-star*:
$x^\omega * x^\star = x^\omega$
by (*meson antisym-conv star.circ-back-loop-prefixpoint sup.boundedE omega-sub-vector*)

lemma *omega-loop-is-greatest-fixpoint*:
is-greatest-fixpoint $(\lambda x . y * x \sqcup z)$ $(y^\omega \sqcup y^\star * z)$
by (*simp add*: *is-greatest-fixpoint-def omega-loop-fixpoint omega-loop-greatest-fixpoint*)

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

lemma *omega-loop-bot-is-greatest-fixpoint*:
is-greatest-fixpoint $(\lambda x . y * x)$ (y^ω)
using *is-greatest-fixpoint-def omega-unfold omega-induct-mult* **by** *auto*

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

lemma *affine-has-greatest-fixpoint*:
has-greatest-fixpoint $(\lambda x . y * x \sqcup z)$
using *has-greatest-fixpoint-def omega-loop-is-greatest-fixpoint* **by** *blast*

lemma *omega-separate-unfold*:
$(x^\star * y)^\omega = y^\omega \sqcup y^\star * x * (x^\star * y)^\omega$
by (*metis star.circ-loop-fixpoint sup-commute mult-assoc omega-slide omega-sum-unfold-1*)

lemma *omega-bot-left-slide*:
$(x * y)^\star * ((x * y)^\omega * bot \sqcup 1) * x \leq x * (y * x)^\star * ((y * x)^\omega * bot \sqcup 1)$
proof −
have $x \sqcup x * (y * x) * (y * x)^\star * ((y * x)^\omega * bot \sqcup 1) \leq x * (y * x)^\star * ((y * x)^\omega * bot \sqcup 1)$
by (*metis sup-commute mult-assoc mult-right-isotone star.circ-back-loop-prefixpoint star.mult-zero-sup-circ star.mult-zero-circ le-supE le-supI order.refl star.circ-increasing star.circ-mult-upper-bound*)
hence $((x * y)^\omega * bot \sqcup 1) * x \sqcup x * y * (x * (y * x)^\star * ((y * x)^\omega * bot \sqcup 1)) \leq x * (y * x)^\star * ((y * x)^\omega * bot \sqcup 1)$
by (*smt* (*z3*) *sup.absorb-iff2 sup-assoc mult-assoc mult-left-one mult-left-sub-dist-sup-left mult-left-zero mult-right-dist-sup omega-slide star-mult-omega*)
thus *?thesis*

by (*simp add*: *star-left-induct mult-assoc*)
qed

lemma *omega-bot-add-1*:
$(x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1) = x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
proof (*rule order.antisym*)
have *1*: $(x \sqcup y) * x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1) \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
by (*smt* (*z3*) *eq-refl star.circ-mult-upper-bound star.circ-sub-dist-1 star.mult-zero-circ star.mult-zero-sup-circ star-sup-1 sup-assoc sup-commute mult-assoc*)
have *2*: $1 \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
using *reflexive-mult-closed star.circ-reflexive sup-ge2* **by** *auto*
have $(y * x^\star)^\omega * bot \le (y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot$
by (*metis mult-1-right mult-left-isotone mult-left-sub-dist-sup-right omega-isotone*)
also have *3*: ... $\le (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
by (*metis mult-isotone mult-left-one star.circ-reflexive sup-commute sup-ge2*)
finally have *4*: $(x^\star * y)^\omega * bot \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
by (*smt mult-assoc mult-right-isotone omega-slide*)
have $y * (x^\star * y)^\star * x^\omega * bot \le y * (x^\star * (x^\omega * bot \sqcup y))^\star * x^\star * x^\omega * bot * (y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot$
using *mult-isotone mult-left-sub-dist-sup-left mult-left-zero order.refl star-isotone sup-commute mult-assoc star-mult-omega* **by** *auto*
also have ... $\le y * (x^\star * (x^\omega * bot \sqcup y))^\star * (x^\star * (x^\omega * bot \sqcup 1) * y)^\omega * bot$
by (*smt mult-assoc mult-left-isotone mult-left-sub-dist-sup-left omega-slide*)
also have ... $= y * (x^\star * (x^\omega * bot \sqcup 1) * y)^\omega * bot$
using *mult-left-one mult-left-zero mult-right-dist-sup mult-assoc star-mult-omega* **by** *auto*
finally have $x^\star * y * (x^\star * y)^\star * x^\omega * bot \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
using *3* **by** (*smt mult-assoc mult-right-isotone omega-slide order-trans*)
hence $(x^\star * y)^\star * x^\omega * bot \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
by (*smt* (*verit*, *ccfv-threshold*) *sup-assoc sup-commute le-iff-sup mult-assoc mult-isotone mult-left-one mult-1-right mult-right-sub-dist-sup-left order-trans star.circ-loop-fixpoint star.circ-reflexive star.mult-zero-circ*)
hence $(x \sqcup y)^\omega * bot \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
using *4* **by** (*smt* (*z3*) *mult-right-dist-sup sup.orderE sup-assoc sup-right-divisibility omega-decompose*)
thus $(x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1) \le x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1)$
using *1 2 star-left-induct mult-assoc* **by** *force*

next
have 5: $x^\omega * bot \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
by (*metis bot-least le-supI1 le-supI2 mult-isotone star.circ-loop-fixpoint sup.cobounded1 omega-isotone*)
have 6: $(y * x^\star)^\omega * bot \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
by (*metis sup-commute mult-left-isotone omega-sub-dist-1 mult-assoc mult-left-sub-dist-sup-left order-trans star-mult-omega*)
have 7: $(y * x^\star)^\star \le (x \sqcup y)^\star$
by (*metis mult-left-one mult-right-sub-dist-sup-left star.circ-sup-1 star.circ-plus-one*)
hence $(y * x^\star)^\star * x^\omega * bot \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
by (*smt sup-assoc le-iff-sup mult-assoc mult-isotone mult-right-dist-sup omega-sub-dist*)
hence $(x^\omega * bot \sqcup y * x^\star)^\omega * bot \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
using 6 **by** (*smt sup-commute sup.bounded-iff mult-assoc mult-right-dist-sup mult-bot-add-omega omega-unfold omega-bot*)
hence $(y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \le y * x^\star * (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
by (*smt mult-assoc mult-left-one mult-left-zero mult-right-dist-sup mult-right-isotone omega-slide*)
also have ... $\le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
using 7 **by** (*metis mult-left-isotone order-refl star.circ-mult-upper-bound star-left-induct-mult-iff*)
finally have $(y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1) \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
using 5 **by** (*smt (z3) le-supE star.circ-mult-upper-bound star.circ-sub-dist-1 star.mult-zero-circ star.mult-zero-sup-circ star-involutive star-isotone sup-commute*)
hence $(x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1) \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
using 5 **by** (*metis sup-commute mult-assoc star.circ-isotone star.circ-mult-upper-bound star.mult-zero-sup-circ star.mult-zero-circ star-involutive*)
thus $x^\star * (x^\omega * bot \sqcup 1) * (y * x^\star * (x^\omega * bot \sqcup 1))^\star * ((y * x^\star * (x^\omega * bot \sqcup 1))^\omega * bot \sqcup 1) \le (x \sqcup y)^\star * ((x \sqcup y)^\omega * bot \sqcup 1)$
by (*smt sup-assoc sup-commute mult-assoc star.circ-mult-upper-bound star.circ-sub-dist star.mult-zero-sup-circ star.mult-zero-circ*)
qed

lemma *star-omega-greatest*:
$x^{\star\omega} = 1^\omega$
by (*metis sup-commute le-iff-sup omega-one-greatest omega-sub-dist star.circ-plus-one*)

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

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

lemma *omega-mult-star-2*:
$x^\omega * y^\star = x^\omega$
by (*meson order.antisym le-supE star.circ-back-loop-prefixpoint omega-sub-vector*)

lemma *omega-import*:
assumes $p \leq p * p$
and $p * x \leq x * p$
shows $p * x^\omega = p * (p * x)^\omega$
proof −
have $p * x^\omega \leq p * (p * x) * x^\omega$
by (*metis assms(1) mult-assoc mult-left-isotone omega-unfold*)
also have ... $\leq p * x * p * x^\omega$
by (*metis assms(2) mult-assoc mult-left-isotone mult-right-isotone*)
finally have $p * x^\omega \leq (p * x)^\omega$
by (*simp add*: *mult-assoc omega-induct-mult*)
hence $p * x^\omega \leq p * (p * x)^\omega$
by (*metis assms(1) mult-assoc mult-left-isotone mult-right-isotone order-trans*)
thus $p * x^\omega = p * (p * x)^\omega$
by (*metis assms(2) sup-left-divisibility order.antisym mult-right-isotone omega-induct-mult omega-slide omega-sub-dist*)
qed

proposition *omega-circ-simulate-right-plus*: $z * x \leq y * (y^\omega * bot \sqcup y^\star) * z \sqcup w \longrightarrow z * (x^\omega * bot \sqcup x^\star) \leq (y^\omega * bot \sqcup y^\star) * (z \sqcup w * (x^\omega * bot \sqcup x^\star))$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *omega-circ-simulate-left-plus*: $x * z \leq z * (y^\omega * bot \sqcup y^\star) \sqcup w \longrightarrow (x^\omega * bot \sqcup x^\star) * z \leq (z \sqcup (x^\omega * bot \sqcup x^\star) * w) * (y^\omega * bot \sqcup y^\star)$ **nitpick** [*expect=genuine,card=5*] **oops**

end

Theorem 50.2

sublocale *left-omega-algebra* < *comb0*: *left-conway-semiring* **where** *circ* = $(\lambda x .\ x^\star * (x^\omega * bot \sqcup 1))$
apply *unfold-locales*
apply (*smt sup-assoc sup-commute le-iff-sup mult-assoc mult-left-sub-dist-sup-left omega-unfold star.circ-loop-fixpoint star-mult-omega*)
using *omega-bot-left-slide mult-assoc* **apply** *fastforce*
using *omega-bot-add-1 mult-assoc* **by** *simp*

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

lemma *star-omega-absorb*:

$y^\star * (y^\star * x)^\star * y^\omega = (y^\star * x)^\star * y^\omega$
proof −
have $y^\star * (y^\star * x)^\star * y^\omega = y^\star * y^\star * x * (y^\star * x)^\star * y^\omega \sqcup y^\star * y^\omega$
by (*metis sup-commute mult-assoc mult-right-dist-sup star.circ-back-loop-fixpoint star.circ-plus-same*)
thus *?thesis*
by (*metis mult-assoc star.circ-loop-fixpoint star.circ-transitive-equal star-mult-omega*)
qed

lemma *omega-circ-simulate-right-plus*:
assumes $z * x \leq y * (y^\omega * bot \sqcup y^\star) * z \sqcup w$
shows $z * (x^\omega * bot \sqcup x^\star) \leq (y^\omega * bot \sqcup y^\star) * (z \sqcup w * (x^\omega * bot \sqcup x^\star))$
proof −
have *1*: $z * x \leq y^\omega * bot \sqcup y * y^\star * z \sqcup w$
by (*metis assms mult-assoc mult-left-dist-sup mult-left-zero mult-right-dist-sup omega-unfold*)
hence $(y^\omega * bot \sqcup y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star) * x \leq y^\omega * bot \sqcup y^\star * (y^\omega * bot \sqcup y * y^\star * z \sqcup w) \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star$
by (*smt sup-assoc sup-ge1 sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup mult-left-zero mult-right-dist-sup star.circ-back-loop-fixpoint*)
also have ... $= y^\omega * bot \sqcup y^\star * y * y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star$
by (*smt sup-assoc sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup star.circ-back-loop-fixpoint star-mult-omega*)
also have ... $\leq y^\omega * bot \sqcup y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star$
by (*smt sup-commute sup-left-isotone mult-left-isotone star.circ-increasing star.circ-plus-same star.circ-transitive-equal*)
finally have $z \sqcup (y^\omega * bot \sqcup y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star) * x \leq y^\omega * bot \sqcup y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star$
by (*metis* (*no-types*, *lifting*) *le-supE le-supI star.circ-loop-fixpoint sup.cobounded1*)
hence *2*: $z * x^\star \leq y^\omega * bot \sqcup y^\star * z \sqcup y^\star * w * x^\omega * bot \sqcup y^\star * w * x^\star$
by (*simp add*: *star-right-induct*)
have $z * x^\omega * bot \leq (y^\omega * bot \sqcup y * y^\star * z \sqcup w) * x^\omega * bot$
using *1* **by** (*smt sup-left-divisibility mult-assoc mult-right-sub-dist-sup-left omega-unfold*)
hence $z * x^\omega * bot \leq y^\omega \sqcup y^\star * (y^\omega * bot \sqcup w * x^\omega * bot)$
by (*smt sup-assoc sup-commute left-plus-omega mult-assoc mult-left-zero mult-right-dist-sup omega-induct star.left-plus-circ*)
thus $z * (x^\omega * bot \sqcup x^\star) \leq (y^\omega * bot \sqcup y^\star) * (z \sqcup w * (x^\omega * bot \sqcup x^\star))$
using *2* **by** (*smt sup-assoc sup-commute le-iff-sup mult-assoc mult-left-dist-sup mult-left-zero mult-right-dist-sup omega-unfold omega-bot star-mult-omega zero-right-mult-decreasing*)
qed

lemma *omega-circ-simulate-left-plus*:
assumes $x * z \leq z * (y^\omega * bot \sqcup y^\star) \sqcup w$
shows $(x^\omega * bot \sqcup x^\star) * z \leq (z \sqcup (x^\omega * bot \sqcup x^\star) * w) * (y^\omega * bot \sqcup y^\star)$
proof −

have $x * (z * y^\omega * bot \sqcup z * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star)$ $= x * z * y^\omega * bot \sqcup x * z * y^\star \sqcup x^\omega * bot \sqcup x * x^\star * w * y^\omega * bot \sqcup x * x^\star * w * y^\star$
by (*smt mult-assoc mult-left-dist-sup omega-unfold*)
also have ... $\leq x * z * y^\omega * bot \sqcup x * z * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*metis sup-mono sup-right-isotone mult-left-isotone star.left-plus-below-circ*)
also have ... $\leq (z * y^\omega * bot \sqcup z * y^\star \sqcup w) * y^\omega * bot \sqcup (z * y^\omega * bot \sqcup z * y^\star \sqcup w) * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*metis assms sup-left-isotone mult-assoc mult-left-dist-sup mult-left-isotone*)
also have ... $= z * y^\omega * bot \sqcup z * y^\star * y^\omega * bot \sqcup w * y^\omega * bot \sqcup z * y^\omega * bot \sqcup z * y^\star * y^\star \sqcup w * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*smt sup-assoc mult-assoc mult-left-zero mult-right-dist-sup*)
also have ... $= z * y^\omega * bot \sqcup z * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*smt* (*verit*, *ccfv-threshold*) *sup-assoc sup-commute sup-idem mult-assoc mult-right-dist-sup star.circ-loop-fixpoint star.circ-transitive-equal star-mult-omega*)
finally have $x^\star * z \leq z * y^\omega * bot \sqcup z * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*smt* (*z3*) *le-supE sup-least sup-ge1 star.circ-back-loop-fixpoint star-left-induct*)
hence $(x^\omega * bot \sqcup x^\star) * z \leq z * y^\omega * bot \sqcup z * y^\star \sqcup x^\omega * bot \sqcup x^\star * w * y^\omega * bot \sqcup x^\star * w * y^\star$
by (*smt* (*z3*) *sup.left-commute sup-commute sup-least sup-ge1 mult-assoc mult-left-zero mult-right-dist-sup*)
thus $(x^\omega * bot \sqcup x^\star) * z \leq (z \sqcup (x^\omega * bot \sqcup x^\star) * w) * (y^\omega * bot \sqcup y^\star)$
by (*smt sup-assoc mult-assoc mult-left-dist-sup mult-left-zero mult-right-dist-sup*)
qed

lemma *omega-translate*:
$x^\star * (x^\omega * bot \sqcup 1) = x^\omega * bot \sqcup x^\star$
by (*metis mult-assoc mult-left-dist-sup mult-1-right star-mult-omega*)

lemma *omega-circ-simulate-right*:
assumes $z * x \leq y * z \sqcup w$
shows $z * (x^\omega * bot \sqcup x^\star) \leq (y^\omega * bot \sqcup y^\star) * (z \sqcup w * (x^\omega * bot \sqcup x^\star))$
proof −
have ... $\leq y * (y^\omega * bot \sqcup y^\star) * z \sqcup w$
using *comb0.circ-mult-increasing mult-isotone sup-left-isotone omega-translate* **by** *auto*
thus $z * (x^\omega * bot \sqcup x^\star) \leq (y^\omega * bot \sqcup y^\star) * (z \sqcup w * (x^\omega * bot \sqcup x^\star))$
using *assms order-trans omega-circ-simulate-right-plus* **by** *blast*
qed

end

sublocale *left-zero-omega-algebra* $<$ *comb1*: *left-conway-semiring-1* **where** *circ*

$= (\lambda x\ .\ x^\star * (x^\omega * bot \sqcup 1))$
apply *unfold-locales*
by (*smt order.eq-iff mult-assoc mult-left-dist-sup mult-left-zero mult-right-dist-sup mult-1-right omega-slide star-slide*)

sublocale *left-zero-omega-algebra* < *comb0*: *itering* **where** $circ = (\lambda x\ .\ x^\star * (x^\omega * bot \sqcup 1))$
apply *unfold-locales*
using *comb1.circ-sup-9* **apply** *blast*
using *comb1.circ-mult-1* **apply** *blast*
apply (*metis omega-circ-simulate-right-plus omega-translate*)
using *omega-circ-simulate-left-plus omega-translate* **by** *auto*

Theorem 2.2

sublocale *left-zero-omega-algebra* < *comb2*: *itering* **where** $circ = (\lambda x\ .\ x^\omega * bot \sqcup x^\star)$
apply *unfold-locales*
using *comb1.circ-sup-9 omega-translate* **apply** *force*
apply (*metis comb1.circ-mult-1 omega-translate*)
using *omega-circ-simulate-right-plus* **apply** *blast*
by (*simp add*: *omega-circ-simulate-left-plus*)

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

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

subclass *left-kleene-conway-semiring* ..

lemma *circ-below-omega-star*:
$x^\circ \le x^\omega \sqcup x^\star$
by (*metis circ-left-unfold mult-1-right omega-induct order-refl*)

lemma *omega-mult-circ*:
$x^\omega * x^\circ = x^\omega$
by (*metis circ-star omega-mult-star-2*)

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

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

lemma *omega-circ*:
$x^{\omega\circ} = 1 \sqcup x^\omega$
by (*metis order.antisym circ-left-unfold mult-left-sub-dist-sup-left mult-1-right*

omega-sub-vector)

end

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

lemma *omega-one*:
  $1^\omega = top$
  by (*simp add*: *order.antisym omega-one-greatest*)

lemma *star-omega-top*:
  $x^{\star\omega} = top$
  by (*simp add*: *star-omega-greatest omega-one*)

lemma *omega-vector*:
  $x^\omega * top = x^\omega$
  by (*simp add*: *order.antisym omega-sub-vector top-right-mult-increasing*)

lemma *mult-top-omega*:
  $(x * top)^\omega \le x * top$
  using *mult-greatest-omega omega-one* **by** *auto*

end

sublocale *bounded-left-omega-algebra* < *comb0*: *bounded-left-conway-semiring* **where** $circ = (\lambda x\ .\ x^\star * (x^\omega * bot \sqcup 1))$ **..**

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

subclass *bounded-left-omega-algebra* **..**

end

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

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

subclass *bounded-left-zero-omega-algebra* **..**

end

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

begin

subclass *left-kleene-conway-semiring* **..**

subclass *bounded-left-conway-semiring* **..**

lemma *circ-omega*:
$x^{\circ\omega} = top$
by (*simp add*: *circ-omega-greatest omega-one*)

end

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

lemma *omega-translate-3*:
$x^\star * (x^\omega * bot \sqcup 1) = x^\star * (x^\omega \sqcup 1)$
by (*metis omega-one omega-vector-greatest top-left-bot mult-assoc*)

end

Theorem 50.2

sublocale *top-left-omega-algebra* < *comb4*: *left-conway-semiring* **where** $circ = (\lambda x\ .\ x^\star * (x^\omega \sqcup 1))$
apply *unfold-locales*
using *comb0.circ-left-unfold omega-translate-3* **apply** *force*
using *omega-bot-left-slide omega-translate-3 mult-assoc* **apply** *force*
using *comb0.circ-sup-1 omega-translate-3* **by** *auto*

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

lemma *omega-translate-2*:
$x^\omega * bot \sqcup x^\star = x^\omega \sqcup x^\star$
by (*metis mult-assoc omega-mult-star-2 star.circ-top top-left-bot*)

end

Theorem 2.3

sublocale *top-left-bot-omega-algebra* < *comb3*: *itering* **where** $circ = (\lambda x\ .\ x^\omega \sqcup x^\star)$
apply *unfold-locales*
using *comb2.circ-slide-1 comb2.circ-sup-1 omega-translate-2* **apply** *force*
apply (*metis comb2.circ-mult-1 omega-translate-2*)
using *omega-circ-simulate-right-plus omega-translate-2* **apply** *force*
using *omega-circ-simulate-left-plus omega-translate-2* **by** *auto*

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

end

3 Capped Omega Algebras

theory *Capped-Omega-Algebras*

imports *Omega-Algebras*

begin

class *capped-omega* =
fixes *capped-omega* :: *'a* ⇒ *'a* ⇒ *'a* (‹-$^{\omega}$-› [*100*,*100*] *100*)

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

AACP Theorem 6.1

notation
top (‹⊤›)

sublocale *capped-omega-algebra* < *capped*: *bounded-left-zero-omega-algebra*
where *omega* = (λy . $y^{\omega}{}_{\top}$)
apply *unfold-locales*
apply (*metis capped-omega-unfold inf-top-right*)
by (*simp add*: *capped-omega-induct sup-commute*)

context *capped-omega-algebra*
begin

AACP Theorem 6.2

lemma *capped-omega-below-omega*:
$y^{\omega}{}_{v} \le y^{\omega}{}_{\top}$
using *capped.omega-induct-mult capped-omega-unfold order.eq-iff* **by** *force*

AACP Theorem 6.3

lemma *capped-omega-below*:
$y^{\omega}{}_{v} \le v$
using *capped-omega-unfold order.eq-iff* **by** *force*

AACP Theorem 6.4

lemma *capped-omega-one*:
$1^{\omega}{}_{v} = v$
proof –
have $v \le (1 * v \sqcup bot) \sqcap v$

by *simp*
hence $v \leq 1^{\omega}{}_{v} \sqcup 1^{\star} * bot$
by (*simp add*: *capped-omega-induct*)
also have ... $= 1^{\omega}{}_{v}$
by (*simp add*: *star-one*)
finally show *?thesis*
by (*simp add*: *capped-omega-below order.antisym*)
qed

AACP Theorem 6.5

lemma *capped-omega-zero*:
$bot^{\omega}{}_{v} = bot$
by (*metis capped-omega-below-omega bot-unique capped.omega-bot*)

lemma *star-below-cap*:
$y \leq u \Longrightarrow z \leq v \Longrightarrow u * v \leq v \Longrightarrow y^{\star} * z \leq v$
by (*metis le-sup-iff order.trans mult-left-isotone star-left-induct*)

lemma *capped-fix*:
assumes $y \leq u$
and $z \leq v$
and $u * v \leq v$
shows $(y * (y^{\omega}{}_{v} \sqcup y^{\star} * z) \sqcup z) \sqcap v = y^{\omega}{}_{v} \sqcup y^{\star} * z$
proof −
have $(y * (y^{\omega}{}_{v} \sqcup y^{\star} * z) \sqcup z) \sqcap v = (y * y^{\omega}{}_{v} \sqcup y^{\star} * z) \sqcap v$
by (*simp add*: *mult-left-dist-sup star.circ-loop-fixpoint sup-assoc*)
also have ... $= (y * y^{\omega}{}_{v} \sqcap v) \sqcup (y^{\star} * z \sqcap v)$
by (*simp add*: *inf-sup-distrib2*)
also have ... $= y^{\omega}{}_{v} \sqcup y^{\star} * z$
using *assms capped-omega-unfold le-iff-inf star-below-cap* **by** *auto*
finally show *?thesis*
.
qed

lemma *capped-fixpoint*:
$y \leq u \Longrightarrow z \leq v \Longrightarrow u * v \leq v \Longrightarrow$ *is-fixpoint* $(\lambda x\ .\ (y * x \sqcup z) \sqcap v)\ (y^{\omega}{}_{v} \sqcup y^{\star} * z)$
by (*simp add*: *capped-fix is-fixpoint-def*)

lemma *capped-greatest-fixpoint*:
$y \leq u \Longrightarrow z \leq v \Longrightarrow u * v \leq v \Longrightarrow$ *is-greatest-fixpoint* $(\lambda x\ .\ (y * x \sqcup z) \sqcap v)\ (y^{\omega}{}_{v} \sqcup y^{\star} * z)$
by (*smt capped-fix order-refl capped-omega-induct is-greatest-fixpoint-def*)

lemma *capped-postfixpoint*:
$y \leq u \Longrightarrow z \leq v \Longrightarrow u * v \leq v \Longrightarrow$ *is-postfixpoint* $(\lambda x\ .\ (y * x \sqcup z) \sqcap v)\ (y^{\omega}{}_{v} \sqcup y^{\star} * z)$
using *capped-fix inf.eq-refl is-postfixpoint-def* **by** *auto*

lemma *capped-greatest-postfixpoint*:
$y \le u \Longrightarrow z \le v \Longrightarrow u * v \le v \Longrightarrow$ *is-greatest-postfixpoint* $(\lambda x\ .\ (y * x \sqcup z) \sqcap v)\ (y^{\omega}{}_{v} \sqcup y^{\star} * z)$
by (*smt capped-fix order-refl capped-omega-induct is-greatest-postfixpoint-def*)

AACP Theorem 6.6

lemma *capped-nu*:
$y \le u \Longrightarrow z \le v \Longrightarrow u * v \le v \Longrightarrow \nu(\lambda x\ .\ (y * x \sqcup z) \sqcap v) = y^{\omega}{}_{v} \sqcup y^{\star} * z$
by (*metis capped-greatest-fixpoint greatest-fixpoint-same*)

lemma *capped-pnu*:
$y \le u \Longrightarrow z \le v \Longrightarrow u * v \le v \Longrightarrow p\nu(\lambda x\ .\ (y * x \sqcup z) \sqcap v) = y^{\omega}{}_{v} \sqcup y^{\star} * z$
by (*metis capped-greatest-postfixpoint greatest-postfixpoint-same*)

AACP Theorem 6.7

lemma *unfold-capped-omega*:
$y \le u \Longrightarrow u * v \le v \Longrightarrow y * y^{\omega}{}_{v} = y^{\omega}{}_{v}$
by (*smt* (*verit*, *ccfv-SIG*) *capped-omega-below capped-omega-unfold inf.order-lesseq-imp le-iff-inf mult-isotone*)

AACP Theorem 6.8

lemma *star-mult-capped-omega*:
assumes $y \le u$
and $u * v \le v$
shows $y^{\star} * y^{\omega}{}_{v} = y^{\omega}{}_{v}$
proof −
have $y * y^{\omega}{}_{v} = y^{\omega}{}_{v}$
using *assms unfold-capped-omega* **by** *auto*
hence $y^{\star} * y^{\omega}{}_{v} \le y^{\omega}{}_{v}$
by (*simp add*: *star-left-induct-mult*)
thus *?thesis*
by (*metis sup-ge2 order.antisym star.circ-loop-fixpoint*)
qed

AACP Theorem 6.9

lemma *star-zero-below-capped-omega-zero*:
assumes $y \le u$
and $u * v \le v$
shows $y^{\star} * bot \le y^{\omega}{}_{v} * bot$
proof −
have $y * y^{\omega}{}_{v} \le v$
using *assms capped-omega-below unfold-capped-omega* **by** *auto*
hence $y * y^{\omega}{}_{v} = y^{\omega}{}_{v}$
using *assms unfold-capped-omega* **by** *auto*
thus *?thesis*
by (*metis bot-least eq-refl mult-assoc star-below-cap*)
qed

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

$y \le u \Longrightarrow u * v \le v \Longrightarrow y^\star * bot \le y^\omega{}_v$
by (*simp add*: *star-loop-least-fixpoint unfold-capped-omega*)

lemma *capped-omega-induct-meet-zero*:
$x \le y * x \sqcap v \Longrightarrow x \le y^\omega{}_v \sqcup y^\star * bot$
by (*simp add*: *capped-omega-induct*)

AACP Theorem 6.10

lemma *capped-omega-induct-meet*:
$y \le u \Longrightarrow u * v \le v \Longrightarrow x \le y * x \sqcap v \Longrightarrow x \le y^\omega{}_v$
by (*metis capped-omega-induct-meet-zero sup-commute le-iff-sup star-zero-below-capped-omega*)

lemma *capped-omega-induct-equal*:
$x = (y * x \sqcup z) \sqcap v \Longrightarrow x \le y^\omega{}_v \sqcup y^\star * z$
using *capped-omega-induct inf.le-iff-sup* **by** *auto*

AACP Theorem 6.11

lemma *capped-meet-nu*:
assumes $y \le u$
and $u * v \le v$
shows $\nu(\lambda x . y * x \sqcap v) = y^\omega{}_v$
proof −
have $y^\omega{}_v \sqcup y^\star * bot = y^\omega{}_v$
by (*smt assms star-zero-below-capped-omega le-iff-sup sup-commute*)
hence $\nu(\lambda x . (y * x \sqcup bot) \sqcap v) = y^\omega{}_v$
by (*metis assms capped-nu bot-least*)
thus *?thesis*
by *simp*
qed

lemma *capped-meet-pnu*:
assumes $y \le u$
and $u * v \le v$
shows $p\nu(\lambda x . y * x \sqcap v) = y^\omega{}_v$
proof −
have $y^\omega{}_v \sqcup y^\star * bot = y^\omega{}_v$
by (*smt assms star-zero-below-capped-omega le-iff-sup sup-commute*)
hence $p\nu(\lambda x . (y * x \sqcup bot) \sqcap v) = y^\omega{}_v$
by (*metis assms capped-pnu bot-least*)
thus *?thesis*
by *simp*
qed

AACP Theorem 6.12

lemma *capped-omega-isotone*:
$y \le u \Longrightarrow u * v \le v \Longrightarrow t \le y \Longrightarrow t^\omega{}_v \le y^\omega{}_v$
by (*metis capped-omega-induct-meet capped-omega-unfold le-iff-sup inf.sup-left-isotone mult-right-sub-dist-sup-left*)

AACP Theorem 6.13

lemma *capped-omega-simulation*:
assumes $y \le u$
and $s \le u$
and $u * v \le v$
and $s * t \le y * s$
shows $s * t^{\omega}{}_{v} \le y^{\omega}{}_{v}$
proof −
have $s * t^{\omega}{}_{v} \le s * t * t^{\omega}{}_{v} \sqcap s * v$
by (*metis capped-omega-below capped-omega-unfold inf.boundedI inf.cobounded1 mult-right-isotone mult-assoc*)
also have ... $\le s * t * t^{\omega}{}_{v} \sqcap v$
using *assms*(*2*,*3*) *inf.order-lesseq-imp inf.sup-right-isotone mult-left-isotone* **by** *blast*
also have ... $\le y * s * t^{\omega}{}_{v} \sqcap v$
using *assms*(*4*) *inf.sup-left-isotone mult-left-isotone* **by** *auto*
finally show *?thesis*
using *assms*(*1*,*3*) *capped-omega-induct-meet mult-assoc* **by** *auto*
qed

lemma *capped-omega-slide-sub*:
assumes $s \le u$
and $y \le u$
and $u * u \le u$
and $u * v \le v$
shows $s * (y * s)^{\omega}{}_{v} \le (s * y)^{\omega}{}_{v}$
proof −
have $s * y \le u$
by (*meson assms*(*1*−*3*) *mult-isotone order-trans*)
thus *?thesis*
using *assms*(*1*,*4*) *capped-omega-simulation mult-assoc* **by** *auto*
qed

AACP Theorem 6.14

lemma *capped-omega-slide*:
$s \le u \Longrightarrow y \le u \Longrightarrow u * u \le u \Longrightarrow u * v \le v \Longrightarrow s * (y * s)^{\omega}{}_{v} = (s * y)^{\omega}{}_{v}$
by (*smt* (*verit*) *order.antisym mult-assoc mult-right-isotone capped-omega-unfold capped-omega-slide-sub inf.sup-ge1 order-trans*)

lemma *capped-omega-sub-dist*:
$s \le u \Longrightarrow y \le u \Longrightarrow u * v \le v \Longrightarrow s^{\omega}{}_{v} \le (s \sqcup y)^{\omega}{}_{v}$
by (*simp add*: *capped-omega-isotone*)

AACP Theorem 6.15

lemma *capped-omega-simulation-2*:
assumes $s \le u$
and $y \le u$
and $u * u \le u$
and $u * v \le v$

and $y * s \le s * y$
shows $(s * y)^\omega{}_v \le s^\omega{}_v$
proof $-$
have *1*: $s * y \le u$
using *assms*(*1*−*3*) *inf.order-lesseq-imp mult-isotone* **by** *blast*
have *2*: $s * (s * y)^\omega{}_v \le v$
by (*meson assms*(*1*,*4*) *capped-omega-below order.trans mult-isotone*)
have $(s * y)^\omega{}_v = s * (y * s)^\omega{}_v$
using *assms*(*1*−*4*) *capped-omega-slide* **by** *auto*
also have ... $\le s * (s * y)^\omega{}_v$
using *1 assms*(*4*,*5*) *capped-omega-isotone mult-right-isotone* **by** *blast*
also have ... $= s * (s * y)^\omega{}_v \sqcap v$
using *2 inf.order-iff* **by** *auto*
finally show *?thesis*
using *assms*(*1*,*4*) *capped-omega-induct-meet* **by** *blast*
qed

AACP Theorem 6.16

lemma *left-plus-capped-omega*:
assumes $y \le u$
and $u * u \le u$
and $u * v \le v$
shows $(y * y^\star)^\omega{}_v = y^\omega{}_v$
proof $-$
have *1*: $y * y^\star \le u$
by (*metis assms*(*1*,*2*) *star-plus star-below-cap*)
hence $y * y^\star * (y * y^\star)^\omega{}_v \le v$
using *assms*(*3*) *capped-omega-below unfold-capped-omega* **by** *auto*
hence $y * y^\star * (y * y^\star)^\omega{}_v = (y * y^\star)^\omega{}_v$
using *1 assms*(*3*) *unfold-capped-omega* **by** *blast*
hence $(y * y^\star)^\omega{}_v \le y^\omega{}_v$
using *1* **by** (*smt assms*(*1*,*3*) *capped-omega-simulation mult-assoc mult-semi-associative star.circ-transitive-equal star-simulation-right-equal*)
thus *?thesis*
using *1* **by** (*meson assms*(*3*) *capped-omega-isotone order.antisym star.circ-mult-increasing*)
qed

AACP Theorem 6.17

lemma *capped-omega-sub-vector*:
assumes $z \le v$
and $y \le u$
and $u * v \le v$
shows $y^\omega{}_u * z \le y^\omega{}_v$
proof $-$
have $y^\omega{}_u * z \le y * y^\omega{}_u * z \sqcap u * z$
by (*metis capped-omega-below capped-omega-unfold eq-refl inf.boundedI inf.cobounded1 mult-isotone*)
also have ... $\le y * y^\omega{}_u * z \sqcap v$

by (*metis assms*(*1*,*3*) *inf*.*sup-left-isotone inf-commute mult-right-isotone order-trans*)
finally show *?thesis*
using *assms*(*2*,*3*) *capped-omega-induct-meet mult-assoc* **by** *auto*
qed

AACP Theorem 6.18

lemma *capped-omega-omega*:
$y \leq u \Longrightarrow u * v \leq v \Longrightarrow (y^{\omega}{}_{u})^{\omega}{}_{v} \leq y^{\omega}{}_{v}$
by (*metis capped-omega-below capped-omega-sub-vector unfold-capped-omega*)

end

end

4 General Refinement Algebras

theory *General-Refinement-Algebras*

imports *Omega-Algebras*

begin

class *general-refinement-algebra* = *left-kleene-algebra* + *Omega* +
assumes *Omega-unfold*: $y^{\Omega} \leq 1 \sqcup y * y^{\Omega}$
assumes *Omega-induct*: $x \leq z \sqcup y * x \longrightarrow x \leq y^{\Omega} * z$
begin

lemma *Omega-unfold-equal*:
$y^{\Omega} = 1 \sqcup y * y^{\Omega}$
by (*smt Omega-induct Omega-unfold sup-right-isotone order*.*antisym mult-right-isotone mult-1-right*)

lemma *Omega-sup-1*:
$(x \sqcup y)^{\Omega} = x^{\Omega} * (y * x^{\Omega})^{\Omega}$
apply (*rule order*.*antisym*)
apply (*smt Omega-induct Omega-unfold-equal sup-assoc sup-commute sup-right-isotone mult-assoc mult-right-dist-sup mult-right-isotone mult-1-right order-refl*)
by (*smt Omega-induct Omega-unfold-equal sup-assoc sup-commute mult-assoc mult-left-one mult-right-dist-sup mult-1-right order-refl*)

lemma *Omega-left-slide*:
$(x * y)^{\Omega} * x \leq x * (y * x)^{\Omega}$
proof −
have $1 \sqcup y * (x * y)^{\Omega} * x \leq 1 \sqcup y * x * (1 \sqcup (y * (x * y)^{\Omega}) * x)$
by (*smt Omega-unfold-equal sup-right-isotone mult-assoc mult-left-one mult-left-sub-dist-sup mult-right-dist-sup mult-right-isotone mult-1-right*)
thus *?thesis*

by (*smt Omega-induct Omega-unfold-equal le-sup-iff mult-assoc mult-left-one mult-right-dist-sup mult-right-isotone mult-1-right*)
qed

end

Theorem 50.3

sublocale *general-refinement-algebra* $<$ *Omega*: *left-conway-semiring* **where** *circ* $=$ *Omega*
apply *unfold-locales*
using *Omega-unfold-equal* **apply** *simp*
apply (*simp add*: *Omega-left-slide*)
by (*simp add*: *Omega-sup-1*)

context *general-refinement-algebra*
begin

lemma *star-below-Omega*:
$x^\star \le x^\Omega$
by (*metis Omega-induct mult-1-right order-refl star.circ-left-unfold*)

lemma *star-mult-Omega*:
$x^\Omega = x^\star * x^\Omega$
by (*metis Omega.left-plus-below-circ sup-commute sup-ge1 order.eq-iff star.circ-loop-fixpoint star-left-induct-mult-iff*)

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

lemma *greatest-left-zero*:
$1^\Omega * x = 1^\Omega$
by (*simp add*: *Omega-one-greatest Omega-induct order.antisym*)

proposition *circ-right-unfold*: $1 \sqcup x^\Omega * x = x^\Omega$ **nitpick** [*expect=genuine,card=8*] **oops**
proposition *circ-slide*: $(x * y)^\Omega * x = x * (y * x)^\Omega$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *circ-simulate*: $z * x \le y * z \Longrightarrow z * x^\Omega \le y^\Omega * z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *circ-simulate-right*: $z * x \le y * z \sqcup w \Longrightarrow z * x^\Omega \le y^\Omega * (z \sqcup w * x^\Omega)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *circ-simulate-right-1*: $z * x \le y * z \Longrightarrow z * x^\Omega \le y^\Omega * z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *circ-simulate-right-plus*: $z * x \le y * y^\Omega * z \sqcup w \Longrightarrow z * x^\Omega \le y^\Omega * (z \sqcup w * x^\Omega)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *circ-simulate-right-plus-1*: $z * x \le y * y^\Omega * z \Longrightarrow z * x^\Omega \le y^\Omega * z$ **nitpick** [*expect=genuine,card=6*] **oops**

proposition *circ-simulate-left-1*: $x * z \leq z * y \Longrightarrow x^\Omega * z \leq z * y^\Omega \sqcup x^\Omega * bot$
oops
proposition *circ-simulate-left-plus-1*: $x * z \leq z * y^\Omega \Longrightarrow x^\Omega * z \leq z * y^\Omega \sqcup x^\Omega * bot$ **oops**
proposition *circ-simulate-absorb*: $y * x \leq x \Longrightarrow y^\Omega * x \leq x \sqcup y^\Omega * bot$ **nitpick** [*expect=genuine,card=8*] **oops**

end

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

lemma *Omega-one*:
  $1^\Omega = top$
  by (*simp add*: *Omega-one-greatest order.antisym*)

lemma *top-left-zero*:
  $top * x = top$
  using *Omega-one greatest-left-zero* **by** *blast*

end

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

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

lemma *Omega-isolate-equal*:
  $y^\Omega = y^\Omega * bot \sqcup y^\star$
  using *Omega-isolate order.antisym le-sup-iff star-below-Omega zero-right-mult-decreasing* **by** *auto*

proposition *Omega-sum-unfold-1*: $(x \sqcup y)^\Omega = y^\Omega \sqcup y^\star * x * (x \sqcup y)^\Omega$ **oops**
proposition *Omega-sup-3*: $(x \sqcup y)^\Omega = (x^\star * y)^\Omega * x^\Omega$ **oops**

end

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

proposition *Omega-mult*: $(x * y)^\Omega = 1 \sqcup x * (y * x)^\Omega * y$ **oops**
proposition *Omega-sup*: $(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$ **oops**
proposition *Omega-simulate*: $z * x \leq y * z \Longrightarrow z * x^\Omega \leq y^\Omega * z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *Omega-separate-2*: $y * x \leq x * (x \sqcup y) \Longrightarrow (x \sqcup y)^\Omega = x^\Omega * y^\Omega$

oops
proposition *Omega-circ-simulate-right-plus*: $z * x \leq y * y^\Omega * z \sqcup w \Longrightarrow z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *Omega-circ-simulate-left-plus*: $x * z \leq z * y^\Omega \sqcup w \Longrightarrow x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$ **oops**

end

sublocale *bounded-left-demonic-refinement-algebra* < *Omega*: *bounded-left-conway-semiring* **where** *circ* = *Omega* **..**

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

lemma *Omega-mult*:
$(x * y)^\Omega = 1 \sqcup x * (y * x)^\Omega * y$
by (*smt* (*verit, del-insts*) *Omega.circ-left-slide Omega-induct Omega-unfold-equal order.eq-iff mult-assoc mult-left-dist-sup mult-1-right*)

lemma *Omega-sup*:
$(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$
by (*smt Omega-sup-1 Omega-mult mult-assoc mult-left-dist-sup mult-left-one mult-right-dist-sup mult-1-right*)

lemma *Omega-simulate*:
$z * x \leq y * z \Longrightarrow z * x^\Omega \leq y^\Omega * z$
by (*smt Omega-induct Omega-unfold-equal sup-right-isotone mult-assoc mult-left-dist-sup mult-left-isotone mult-1-right*)

end

Theorem 2.4

sublocale *demonic-refinement-algebra* < *Omega1*: *itering-1* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*simp add*: *Omega-simulate mult-assoc*)
by (*simp add*: *Omega-simulate*)

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

context *demonic-refinement-algebra*
begin

lemma *Omega-sum-unfold-1*:
$(x \sqcup y)^\Omega = y^\Omega \sqcup y^\star * x * (x \sqcup y)^\Omega$
by (*smt Omega1.circ-sup-9 Omega.circ-loop-fixpoint Omega-isolate-equal sup-assoc sup-commute mult-assoc mult-left-zero mult-right-dist-sup*)

lemma *Omega-sup-3*:
  $(x \sqcup y)^\Omega = (x^\star * y)^\Omega * x^\Omega$
  apply (*rule order.antisym*)
  apply (*metis Omega-sum-unfold-1 Omega-induct eq-refl sup-commute*)
  by (*simp add*: *Omega.circ-isotone Omega-sup mult-left-isotone star-below-Omega*)

lemma *Omega-separate-2*:
  $y * x \leq x * (x \sqcup y) \Longrightarrow (x \sqcup y)^\Omega = x^\Omega * y^\Omega$
  apply (*rule order.antisym*)
  apply (*smt* (*verit*, *del-insts*) *Omega-induct Omega-sum-unfold-1 sup-right-isotone mult-assoc mult-left-isotone star-mult-Omega star-simulation-left*)
  by (*simp add*: *Omega.circ-sub-dist-3*)

lemma *Omega-circ-simulate-right-plus*:
  assumes $z * x \leq y * y^\Omega * z \sqcup w$
    shows $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$
proof −
  have $z * x^\Omega = z \sqcup z * x * x^\Omega$
    using *Omega1.circ-back-loop-fixpoint Omega1.circ-plus-same sup-commute mult-assoc* **by** *auto*
  also have ... $\leq y * y^\Omega * z * x^\Omega \sqcup z \sqcup w * x^\Omega$
    by (*smt assms sup-assoc sup-commute sup-right-isotone le-iff-sup mult-right-dist-sup*)
  finally have $z * x^\Omega \leq (y * y^\Omega)^\Omega * (z \sqcup w * x^\Omega)$
    by (*smt Omega-induct sup-assoc sup-commute mult-assoc*)
  thus *?thesis*
    by (*simp add*: *Omega.left-plus-circ*)
qed

lemma *Omega-circ-simulate-left-plus*:
  assumes $x * z \leq z * y^\Omega \sqcup w$
    shows $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$
proof −
  have $x * ((z \sqcup x^\Omega * w) * y^\Omega) \leq (z * y^\Omega \sqcup w \sqcup x * x^\Omega * w) * y^\Omega$
    by (*smt assms mult-assoc mult-left-dist-sup sup-left-isotone mult-left-isotone*)
  also have ... $\leq z * y^\Omega * y^\Omega \sqcup w * y^\Omega \sqcup x^\Omega * w * y^\Omega$
    by (*smt Omega.left-plus-below-circ sup-right-isotone mult-left-isotone mult-right-dist-sup*)
  finally have *1*: $x * ((z \sqcup x^\Omega * w) * y^\Omega) \leq (z \sqcup x^\Omega * w) * y^\Omega$
    by (*metis Omega.circ-transitive-equal mult-assoc Omega.circ-reflexive sup-assoc le-iff-sup mult-left-one mult-right-dist-sup*)
  have $x^\Omega * z = x^\Omega * bot \sqcup x^\star * z$
    by (*metis Omega-isolate-equal mult-assoc mult-left-zero mult-right-dist-sup*)
  also have ... $\leq x^\Omega * w * y^\Omega \sqcup x^\star * (z \sqcup x^\Omega * w) * y^\Omega$
    by (*metis Omega1.circ-back-loop-fixpoint bot-least idempotent-bot-closed le-supI2 mult-isotone mult-left-sub-dist-sup-left semiring.add-mono*

zero-right-mult-decreasing mult-assoc)
also have ... $\leq (z \sqcup x^\Omega * w) * y^\Omega$
using *1* **by** (*metis le-supI mult-right-sub-dist-sup-right star-left-induct-mult mult-assoc*)
finally show *?thesis*
.
qed

lemma *Omega-circ-simulate-right*:
assumes $z * x \leq y * z \sqcup w$
shows $z * x^\Omega \leq y^\Omega * (z \sqcup w * x^\Omega)$
proof −
have $y * z \sqcup w \leq y * y^\Omega * z \sqcup w$
using *Omega.circ-mult-increasing mult-left-isotone sup-left-isotone* **by** *auto*
thus *?thesis*
using *Omega-circ-simulate-right-plus assms order.trans* **by** *blast*
qed

end

sublocale *demonic-refinement-algebra* < *Omega*: *itering* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*simp add*: *Omega-sup*)
using *Omega-mult* **apply** *blast*
apply (*simp add*: *Omega-circ-simulate-right-plus*)
using *Omega-circ-simulate-left-plus* **by** *auto*

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

lemma *Omega-one*:
$1^\Omega = top$
by (*simp add*: *Omega-one-greatest order.antisym*)

lemma *top-left-zero*:
$top * x = top$
using *Omega-one greatest-left-zero* **by** *auto*

end

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

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

lemma *omega-left-zero-equal*:
  $x^\omega * y = x^\omega$
  by (*simp add*: *order.antisym omega-left-zero omega-sub-vector*)

subclass *left-demonic-refinement-algebra*
  apply *unfold-locales*
  apply (*metis Omega-def sup-commute eq-refl mult-1-right omega-loop-fixpoint*)
  apply (*metis Omega-def mult-right-dist-sup omega-induct omega-left-zero-equal*)
  by (*metis Omega-def mult-right-sub-dist-sup-right sup-commute*
sup-right-isotone omega-left-zero-equal)

end

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

subclass *general-refinement-algebra-omega*
  apply *unfold-locales*
  apply (*metis mult-assoc omega-vector order-refl top-left-zero*)
  by (*rule Omega-def*)

end

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

lemma *Omega-mult*:
  $(x * y)^\Omega = 1 \sqcup x * (y * x)^\Omega * y$
  by (*metis Omega-def comb1.circ-mult-1 omega-left-zero-equal omega-translate*)

lemma *Omega-sup*:
  $(x \sqcup y)^\Omega = (x^\Omega * y)^\Omega * x^\Omega$
proof −
  have $(x^\Omega * y)^\Omega * x^\Omega = (x^\star * y)^\star * x^\omega \sqcup (x^\star * y)^\omega \sqcup (x^\star * y)^\star * x^{\omega\star} * x^\Omega$
    by (*smt sup-commute Omega-def mult-assoc mult-right-dist-sup*
mult-bot-add-omega omega-left-zero-equal star.circ-sup-1)
  thus *?thesis*
    using *Omega-def Omega-sup-1 comb2.circ-slide-1 omega-left-zero-equal* **by**
auto
qed

lemma *Omega-simulate*:
  $z * x \le y * z \Longrightarrow z * x^\Omega \le y^\Omega * z$
  using *Omega-def comb2.circ-simulate omega-left-zero-equal* **by** *auto*

subclass *demonic-refinement-algebra* **..**

end

proposition *circ-circ-mult*: $1^\Omega * x^\Omega = x^{\Omega\Omega}$ **oops**
proposition *sub-mult-one-circ*: $x * 1^\Omega \leq 1^\Omega * x$ **oops**
proposition *circ-circ-mult-1*: $x^\Omega * 1^\Omega = x^{\Omega\Omega}$ **oops**
proposition $y * x \leq x \Longrightarrow y^\circ * x \leq 1^\circ * x$ **oops**

proposition *circ-simulate-2*: $y * x^\Omega \leq x^\Omega * y^\Omega \longleftrightarrow y^\Omega * x^\Omega \leq x^\Omega * y^\Omega$ **oops**
proposition *circ-simulate-3*: $y * x^\Omega \leq x^\Omega \Longrightarrow y^\Omega * x^\Omega \leq x^\Omega * y^\Omega$ **oops**
proposition *circ-separate-mult-1*: $y * x \leq x * y \Longrightarrow (x * y)^\Omega \leq x^\Omega * y^\Omega$ **oops**
proposition $x^\circ = (x * x)^\circ * (x \sqcup 1)$ **oops**
proposition $y^\circ * x^\circ \leq x^\circ * y^\circ \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$ **oops**
proposition $y * x \leq (1 \sqcup x) * y^\circ \Longrightarrow (x \sqcup y)^\circ = x^\circ * y^\circ$ **oops**

end

5 Lattice-Ordered Semirings

theory *Lattice-Ordered-Semirings*

imports *Stone-Relation-Algebras.Semirings*

begin

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

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

M0-algebra

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

subclass *bounded-pre-left-semiring*
  apply *unfold-locales*
  by *simp*

lemma *top-mult-right-one*:
  $x * top = x * top * 1$
  by (*metis order.antisym mult-sub-right-one mult-sup-associative-one surjective-one-closed*)

lemma *mult-left-sub-dist-inf-left*:
  $x * (y \sqcap z) \leq x * y$
  by (*simp add*: *mult-right-isotone*)

lemma *mult-left-sub-dist-inf-right*:
$x * (y \sqcap z) \leq x * z$
by (*simp add*: *mult-right-isotone*)

lemma *mult-right-sub-dist-inf-left*:
$(x \sqcap y) * z \leq x * z$
by (*simp add*: *mult-left-isotone*)

lemma *mult-right-sub-dist-inf-right*:
$(x \sqcap y) * z \leq y * z$
by (*simp add*: *mult-left-isotone*)

lemma *mult-right-sub-dist-inf*:
$(x \sqcap y) * z \leq x * z \sqcap y * z$
by (*simp add*: *mult-right-sub-dist-inf-left mult-right-sub-dist-inf-right*)

Figure 1: fundamental properties

definition *co-total* :: $'a \Rightarrow bool$ **where** *co-total* $x \equiv x * bot = bot$
definition *up-closed* :: $'a \Rightarrow bool$ **where** *up-closed* $x \equiv x * 1 = x$
definition *sup-distributive* :: $'a \Rightarrow bool$ **where** *sup-distributive* $x \equiv (\forall y\ z\ .\ x * (y \sqcup z) = x * y \sqcup x * z)$
definition *inf-distributive* :: $'a \Rightarrow bool$ **where** *inf-distributive* $x \equiv (\forall y\ z\ .\ x * (y \sqcap z) = x * y \sqcap x * z)$
definition *contact* :: $'a \Rightarrow bool$ **where** *contact* $x \equiv x * x \sqcup 1 = x$
definition *kernel* :: $'a \Rightarrow bool$ **where** *kernel* $x \equiv x * x \sqcap 1 = x * 1$
definition *sup-dist-contact* :: $'a \Rightarrow bool$ **where** *sup-dist-contact* $x \equiv$ *sup-distributive* $x \wedge$ *contact* x
definition *inf-dist-kernel* :: $'a \Rightarrow bool$ **where** *inf-dist-kernel* $x \equiv$ *inf-distributive* $x \wedge$ *kernel* x
definition *test* :: $'a \Rightarrow bool$ **where** *test* $x \equiv x * top \sqcap 1 = x$
definition *co-test* :: $'a \Rightarrow bool$ **where** *co-test* $x \equiv x * bot \sqcup 1 = x$
definition *co-vector* :: $'a \Rightarrow bool$ **where** *co-vector* $x \equiv x * bot = x$

AAMP Theorem 6 / Figure 2: relations between properties

lemma *reflexive-total*:
reflexive $x \Longrightarrow$ *total* x
using *sup-left-divisibility total-sup-closed* **by** *force*

lemma *reflexive-dense*:
reflexive $x \Longrightarrow$ *dense-rel* x
using *mult-left-isotone* **by** *fastforce*

lemma *reflexive-transitive-up-closed*:
reflexive $x \Longrightarrow$ *transitive* $x \Longrightarrow$ *up-closed* x
by (*metis antisym-conv mult-isotone mult-sub-right-one reflexive-dense up-closed-def*)

lemma *coreflexive-co-total*:

coreflexive x $\Longrightarrow$ *co-total x*
by (*metis co-total-def order.eq-iff mult-left-isotone mult-left-one bot-least*)

lemma *coreflexive-transitive*:
coreflexive x $\Longrightarrow$ *transitive x*
by (*simp add*: *coreflexive-transitive*)

lemma *idempotent-transitive-dense*:
idempotent x $\longleftrightarrow$ *transitive x* $\wedge$ *dense-rel x*
by (*simp add*: *order.eq-iff*)

lemma *contact-reflexive*:
contact x $\Longrightarrow$ *reflexive x*
using *contact-def sup-right-divisibility* **by** *auto*

lemma *contact-transitive*:
contact x $\Longrightarrow$ *transitive x*
using *contact-def sup-left-divisibility* **by** *blast*

lemma *contact-dense*:
contact x $\Longrightarrow$ *dense-rel x*
by (*simp add*: *contact-reflexive reflexive-dense*)

lemma *contact-idempotent*:
contact x $\Longrightarrow$ *idempotent x*
by (*simp add*: *contact-dense contact-transitive idempotent-transitive-dense*)

lemma *contact-up-closed*:
contact x $\Longrightarrow$ *up-closed x*
by (*simp add*: *contact-reflexive contact-transitive reflexive-transitive-up-closed*)

lemma *contact-reflexive-idempotent-up-closed*:
contact x $\longleftrightarrow$ *reflexive x* $\wedge$ *idempotent x* $\wedge$ *up-closed x*
by (*metis contact-def contact-idempotent contact-reflexive contact-up-closed sup-absorb2 sup-monoid.add-commute*)

lemma *kernel-coreflexive*:
kernel x $\Longrightarrow$ *coreflexive x*
by (*metis kernel-def inf.boundedE mult-sub-right-one*)

lemma *kernel-transitive*:
kernel x $\Longrightarrow$ *transitive x*
by (*simp add*: *coreflexive-transitive kernel-coreflexive*)

lemma *kernel-dense*:
kernel x $\Longrightarrow$ *dense-rel x*
by (*metis kernel-def inf.boundedE mult-sub-right-one*)

lemma *kernel-idempotent*:

kernel x $\Longrightarrow$ *idempotent x*
by (*simp add*: *idempotent-transitive-dense kernel-dense kernel-transitive*)

lemma *kernel-up-closed*:
kernel x $\Longrightarrow$ *up-closed x*
by (*metis kernel-coreflexive kernel-def kernel-idempotent inf.absorb1 up-closed-def*)

lemma *kernel-coreflexive-idempotent-up-closed*:
kernel x $\longleftrightarrow$ *coreflexive x* $\wedge$ *idempotent x* $\wedge$ *up-closed x*
by (*metis kernel-coreflexive kernel-def kernel-idempotent inf.absorb1 up-closed-def*)

lemma *test-coreflexive*:
test x $\Longrightarrow$ *coreflexive x*
using *inf.sup-right-divisibility test-def* **by** *blast*

lemma *test-up-closed*:
test x $\Longrightarrow$ *up-closed x*
by (*metis order.eq-iff mult-left-one mult-sub-right-one mult-right-sub-dist-inf test-def top-mult-right-one up-closed-def*)

lemma *co-test-reflexive*:
co-test x $\Longrightarrow$ *reflexive x*
using *co-test-def sup-right-divisibility* **by** *blast*

lemma *co-test-transitive*:
co-test x $\Longrightarrow$ *transitive x*
by (*smt co-test-def sup-assoc le-iff-sup mult-left-one mult-left-zero mult-right-dist-sup mult-semi-associative*)

lemma *co-test-idempotent*:
co-test x $\Longrightarrow$ *idempotent x*
by (*simp add*: *co-test-reflexive co-test-transitive idempotent-transitive-dense reflexive-dense*)

lemma *co-test-up-closed*:
co-test x $\Longrightarrow$ *up-closed x*
by (*simp add*: *co-test-reflexive co-test-transitive reflexive-transitive-up-closed*)

lemma *co-test-contact*:
co-test x $\Longrightarrow$ *contact x*
by (*simp add*: *co-test-idempotent co-test-reflexive co-test-up-closed contact-reflexive-idempotent-up-closed*)

lemma *vector-transitive*:
vector x $\Longrightarrow$ *transitive x*
by (*metis mult-right-isotone top.extremum*)

lemma *vector-up-closed*:
vector $x \Longrightarrow$ *up-closed* x
by (*metis top-mult-right-one up-closed-def*)

AAMP Theorem 10 / Figure 3: closure properties

total

lemma *one-total*:
total 1
by *simp*

lemma *top-total*:
total top
by *simp*

lemma *sup-total*:
total $x \Longrightarrow$ *total* $y \Longrightarrow$ *total* $(x \sqcup y)$
by (*simp add*: *total-sup-closed*)

co-total

lemma *zero-co-total*:
co-total bot
by (*simp add*: *co-total-def*)

lemma *one-co-total*:
co-total 1
by (*simp add*: *co-total-def*)

lemma *sup-co-total*:
co-total $x \Longrightarrow$ *co-total* $y \Longrightarrow$ *co-total* $(x \sqcup y)$
by (*simp add*: *co-total-def mult-right-dist-sup*)

lemma *inf-co-total*:
co-total $x \Longrightarrow$ *co-total* $y \Longrightarrow$ *co-total* $(x \sqcap y)$
by (*metis co-total-def order.antisym bot-least mult-right-sub-dist-inf-right*)

lemma *comp-co-total*:
co-total $x \Longrightarrow$ *co-total* $y \Longrightarrow$ *co-total* $(x * y)$
by (*metis co-total-def order.eq-iff mult-semi-associative bot-least*)

sub-transitive

lemma *zero-transitive*:
transitive bot
by (*simp add*: *vector-transitive*)

lemma *one-transitive*:
transitive 1
by *simp*

lemma *top-transitive*:

transitive top
by *simp*

lemma *inf-transitive*:
transitive $x \Longrightarrow$ *transitive* $y \Longrightarrow$ *transitive* $(x \sqcap y)$
by (*meson inf-mono order-trans mult-left-sub-dist-inf-left mult-left-sub-dist-inf-right mult-right-sub-dist-inf*)

dense

lemma *zero-dense*:
dense-rel bot
by *simp*

lemma *one-dense*:
dense-rel 1
by *simp*

lemma *top-dense*:
dense-rel top
by *simp*

lemma *sup-dense*:
assumes *dense-rel* x
and *dense-rel* y
shows *dense-rel* $(x \sqcup y)$
proof −
have $x \le x * x \wedge y \le y * y$
using *assms* **by** *auto*
hence $x \le (x \sqcup y) * (x \sqcup y) \wedge y \le (x \sqcup y) * (x \sqcup y)$
by (*meson dense-sup-closed order-trans sup.cobounded1 sup.cobounded2*)
hence $x \sqcup y \le (x \sqcup y) * (x \sqcup y)$
by *simp*
thus *dense-rel* $(x \sqcup y)$
by *simp*
qed

reflexive

lemma *one-reflexive*:
reflexive 1
by *simp*

lemma *top-reflexive*:
reflexive top
by *simp*

lemma *sup-reflexive*:
reflexive $x \Longrightarrow$ *reflexive* $y \Longrightarrow$ *reflexive* $(x \sqcup y)$
by (*simp add*: *reflexive-sup-closed*)

lemma *inf-reflexive*:
reflexive $x \Longrightarrow$ *reflexive* $y \Longrightarrow$ *reflexive* $(x \sqcap y)$
by *simp*

lemma *comp-reflexive*:
reflexive $x \Longrightarrow$ *reflexive* $y \Longrightarrow$ *reflexive* $(x * y)$
using *reflexive-mult-closed* **by** *auto*

co-reflexive

lemma *zero-coreflexive*:
coreflexive bot
by *simp*

lemma *one-coreflexive*:
coreflexive 1
by *simp*

lemma *sup-coreflexive*:
coreflexive $x \Longrightarrow$ *coreflexive* $y \Longrightarrow$ *coreflexive* $(x \sqcup y)$
by *simp*

lemma *inf-coreflexive*:
coreflexive $x \Longrightarrow$ *coreflexive* $y \Longrightarrow$ *coreflexive* $(x \sqcap y)$
by (*simp add*: *le-infI1*)

lemma *comp-coreflexive*:
coreflexive $x \Longrightarrow$ *coreflexive* $y \Longrightarrow$ *coreflexive* $(x * y)$
by (*simp add*: *coreflexive-mult-closed*)

idempotent

lemma *zero-idempotent*:
idempotent bot
by *simp*

lemma *one-idempotent*:
idempotent 1
by *simp*

lemma *top-idempotent*:
idempotent top
by *simp*

up-closed

lemma *zero-up-closed*:
up-closed bot
by (*simp add*: *up-closed-def*)

lemma *one-up-closed*:
up-closed 1

by (*simp add*: *up-closed-def*)

lemma *top-up-closed*:
up-closed top
by (*simp add*: *vector-up-closed*)

lemma *sup-up-closed*:
up-closed x $\Longrightarrow$ *up-closed y* $\Longrightarrow$ *up-closed* $(x \sqcup y)$
by (*simp add*: *mult-right-dist-sup up-closed-def*)

lemma *inf-up-closed*:
up-closed x $\Longrightarrow$ *up-closed y* $\Longrightarrow$ *up-closed* $(x \sqcap y)$
by (*metis order.antisym mult-sub-right-one mult-right-sub-dist-inf up-closed-def*)

lemma *comp-up-closed*:
up-closed x $\Longrightarrow$ *up-closed y* $\Longrightarrow$ *up-closed* $(x * y)$
by (*metis order.antisym mult-semi-associative mult-sub-right-one up-closed-def*)

add-distributive

lemma *zero-sup-distributive*:
sup-distributive bot
by (*simp add*: *sup-distributive-def*)

lemma *one-sup-distributive*:
sup-distributive 1
by (*simp add*: *sup-distributive-def*)

lemma *sup-sup-distributive*:
sup-distributive x $\Longrightarrow$ *sup-distributive y* $\Longrightarrow$ *sup-distributive* $(x \sqcup y)$
using *sup-distributive-def mult-right-dist-sup sup-monoid.add-assoc sup-monoid.add-commute* **by** *auto*

inf-distributive

lemma *zero-inf-distributive*:
inf-distributive bot
by (*simp add*: *inf-distributive-def*)

lemma *one-inf-distributive*:
inf-distributive 1
by (*simp add*: *inf-distributive-def*)

contact

lemma *one-contact*:
contact 1
by (*simp add*: *contact-def*)

lemma *top-contact*:
contact top
by (*simp add*: *contact-def*)

lemma *inf-contact*:
contact $x \Longrightarrow$ *contact* $y \Longrightarrow$ *contact* $(x \sqcap y)$
by (*meson contact-reflexive-idempotent-up-closed contact-transitive inf-reflexive inf-transitive inf-up-closed preorder-idempotent*)

kernel

lemma *zero-kernel*:
kernel bot
by (*simp add*: *kernel-def*)

lemma *one-kernel*:
kernel 1
by (*simp add*: *kernel-def*)

lemma *sup-kernel*:
kernel $x \Longrightarrow$ *kernel* $y \Longrightarrow$ *kernel* $(x \sqcup y)$
using *kernel-coreflexive-idempotent-up-closed order.antisym coreflexive-transitive sup-dense sup-up-closed* **by** *force*

add-distributive contact

lemma *one-sup-dist-contact*:
sup-dist-contact 1
by (*simp add*: *sup-dist-contact-def one-sup-distributive one-contact*)

inf-distributive kernel

lemma *zero-inf-dist-kernel*:
inf-dist-kernel bot
by (*simp add*: *inf-dist-kernel-def zero-kernel zero-inf-distributive*)

lemma *one-inf-dist-kernel*:
inf-dist-kernel 1
by (*simp add*: *inf-dist-kernel-def one-kernel one-inf-distributive*)

test

lemma *zero-test*:
test bot
by (*simp add*: *test-def*)

lemma *one-test*:
test 1
by (*simp add*: *test-def*)

lemma *sup-test*:
test $x \Longrightarrow$ *test* $y \Longrightarrow$ *test* $(x \sqcup y)$
by (*simp add*: *inf-sup-distrib2 mult-right-dist-sup test-def*)

lemma *inf-test*:
test $x \Longrightarrow$ *test* $y \Longrightarrow$ *test* $(x \sqcap y)$

by (*smt* (*z3*) *inf.left-commute idempotent-one-closed inf.le-iff-sup inf-top.right-neutral mult-right-isotone mult-sub-right-one mult-right-sub-dist-inf test-def top-mult-right-one*)

co-test

lemma *one-co-test*:
co-test 1
by (*simp add*: *co-test-def*)

lemma *sup-co-test*:
co-test $x \Longrightarrow$ *co-test* $y \Longrightarrow$ *co-test* $(x \sqcup y)$
by (*smt* (*z3*) *co-test-def mult-right-dist-sup sup.left-idem sup-assoc sup-commute*)

vector

lemma *zero-vector*:
vector bot
by *simp*

lemma *top-vector*:
vector top
by *simp*

lemma *sup-vector*:
vector $x \Longrightarrow$ *vector* $y \Longrightarrow$ *vector* $(x \sqcup y)$
by (*simp add*: *vector-sup-closed*)

lemma *inf-vector*:
vector $x \Longrightarrow$ *vector* $y \Longrightarrow$ *vector* $(x \sqcap y)$
by (*metis order.antisym top-right-mult-increasing mult-right-sub-dist-inf*)

lemma *comp-vector*:
vector $y \Longrightarrow$ *vector* $(x * y)$
by (*simp add*: *vector-mult-closed*)

end

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

subclass *pre-left-semiring*
apply *unfold-locales*
by (*metis mult-associative-one mult-left-isotone mult-right-isotone mult-sub-right-one*)

subclass *lattice-ordered-pre-left-semiring* **..**

lemma *mult-zero-associative*:
$x * bot * y = x * bot$
by (*metis mult-associative-one mult-left-zero*)

lemma *mult-zero-sup-one-dist*:
$(x * bot \sqcup 1) * z = x * bot \sqcup z$
by (*simp add*: *mult-right-dist-sup mult-zero-associative*)

lemma *mult-zero-sup-dist*:
$(x * bot \sqcup y) * z = x * bot \sqcup y * z$
by (*simp add*: *mult-right-dist-sup mult-zero-associative*)

lemma *vector-zero-inf-one-comp*:
$(x * bot \sqcap 1) * y = x * bot \sqcap y$
by (*metis mult-left-one mult-right-dist-inf-one mult-zero-associative*)

AAMP Theorem 6 / Figure 2: relations between properties

lemma *co-test-inf-distributive*:
co-test $x \Longrightarrow$ *inf-distributive* x
by (*metis co-test-def distrib-imp1 inf-sup-distrib1 inf-distributive-def mult-zero-sup-one-dist*)

lemma *co-test-sup-distributive*:
co-test $x \Longrightarrow$ *sup-distributive* x
by (*metis sup-sup-distributive sup-distributive-def co-test-def one-sup-distributive sup.idem mult-zero-associative*)

lemma *co-test-sup-dist-contact*:
co-test $x \Longrightarrow$ *sup-dist-contact* x
by (*simp add*: *co-test-sup-distributive sup-dist-contact-def co-test-contact*)

AAMP Theorem 10 / Figure 3: closure properties

co-test

lemma *inf-co-test*:
co-test $x \Longrightarrow$ *co-test* $y \Longrightarrow$ *co-test* $(x \sqcap y)$
by (*smt* (*z3*) *co-test-def co-test-up-closed mult-right-dist-inf-one sup-commute sup-inf-distrib1 up-closed-def*)

lemma *comp-co-test*:
co-test $x \Longrightarrow$ *co-test* $y \Longrightarrow$ *co-test* $(x * y)$
by (*metis co-test-def mult-associative-one sup-assoc mult-zero-sup-one-dist*)

end

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

subclass *lattice-ordered-pre-left-semiring-1*
  apply *unfold-locales*
  apply (*simp add*: *order.antisym mult-sub-associative-one mult-sup-associative-one*)
  by (*metis order.eq-iff mult-one-associative mult-right-dist-inf-one-sub mult-right-sub-dist-inf*)

end

class *multirelation-algebra-1* = *lattice-ordered-pre-left-semiring* +
  assumes *mult-left-top*: *top* ∗ x = *top*
begin

AAMP Theorem 10 / Figure 3: closure properties

lemma *top-sup-distributive*:
  sup-distributive top
  by (*simp add*: *sup-distributive-def mult-left-top*)

lemma *top-inf-distributive*:
  inf-distributive top
  by (*simp add*: *inf-distributive-def mult-left-top*)

lemma *top-sup-dist-contact*:
  sup-dist-contact top
  by (*simp add*: *sup-dist-contact-def top-contact top-sup-distributive*)

lemma *top-co-test*:
  co-test top
  by (*simp add*: *co-test-def mult-left-top*)

end

M1-algebra

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

lemma *mult-top-associative*:
  x ∗ *top* ∗ y = x ∗ *top*
  by (*metis mult-left-top mult-associative-one*)

lemma *vector-inf-one-comp*:
  (x ∗ *top* ⊓ *1*) ∗ y = x ∗ *top* ⊓ y
  by (*metis vector-zero-inf-one-comp mult-top-associative*)

lemma *vector-left-annihilator*:
  vector x ⟹ x ∗ y = x
  by (*metis mult-top-associative*)

properties

lemma *test-comp-inf*:
test $x \Longrightarrow$ *test* $y \Longrightarrow x * y = x \sqcap y$
by (*metis inf.absorb1 inf.left-commute test-coreflexive test-def vector-inf-one-comp*)

AAMP Theorem 6 / Figure 2: relations between properties

lemma *test-sup-distributive*:
test $x \Longrightarrow$ *sup-distributive* x
by (*metis sup-distributive-def inf-sup-distrib1 test-def vector-inf-one-comp*)

lemma *test-inf-distributive*:
test $x \Longrightarrow$ *inf-distributive* x
by (*smt* (*verit, ccfv-SIG*) *inf.commute inf.sup-monoid.add-assoc inf-distributive-def test-def inf.idem vector-inf-one-comp*)

lemma *test-inf-dist-kernel*:
test $x \Longrightarrow$ *inf-dist-kernel* x
by (*simp add*: *kernel-def inf-dist-kernel-def one-test test-comp-inf test-inf-distributive*)

lemma *vector-idempotent*:
vector $x \Longrightarrow$ *idempotent* x
using *vector-left-annihilator* **by** *blast*

lemma *vector-sup-distributive*:
vector $x \Longrightarrow$ *sup-distributive* x
by (*simp add*: *sup-distributive-def vector-left-annihilator*)

lemma *vector-inf-distributive*:
vector $x \Longrightarrow$ *inf-distributive* x
by (*simp add*: *inf-distributive-def vector-left-annihilator*)

lemma *vector-co-vector*:
vector $x \longleftrightarrow$ *co-vector* x
by (*metis co-vector-def mult-zero-associative mult-top-associative*)

AAMP Theorem 10 / Figure 3: closure properties

test

lemma *comp-test*:
test $x \Longrightarrow$ *test* $y \Longrightarrow$ *test* $(x * y)$
by (*simp add*: *inf-test test-comp-inf*)

end

class *dual* =
fixes *dual* :: $'a \Rightarrow 'a$ (‹-^{d}› [*100*] *100*)

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

lemma *dual-dist-inf*:
  $(x \sqcap y)^d = x^d \sqcup y^d$
  by (*metis dual-dist-sup dual-involutive*)

lemma *dual-antitone*:
  $x \leq y \Longrightarrow y^d \leq x^d$
  using *dual-dist-sup sup-right-divisibility* **by** *fastforce*

lemma *dual-zero*:
  $bot^d = top$
  by (*metis dual-antitone bot-least dual-involutive top-le*)

lemma *dual-top*:
  $top^d = bot$
  using *dual-zero dual-involutive* **by** *auto*

AAMP Theorem 10 / Figure 3: closure properties

lemma *reflexive-coreflexive-dual*:
  reflexive $x \longleftrightarrow$ *coreflexive* (x^d)
  using *dual-antitone dual-involutive dual-one* **by** *fastforce*

end

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

subclass *multirelation-algebra-1*
  apply *unfold-locales*
  by (*metis order.antisym top.extremum dual-zero dual-sub-dist-comp dual-involutive mult-left-zero*)

lemma *dual-sub-dist-comp-one*:
  $(x * y)^d \leq (x * 1)^d * y^d$
  by (*metis dual-sub-dist-comp mult-one-associative*)

AAMP Theorem 10 / Figure 3: closure properties

lemma *co-total-total-dual*:
  co-total $x \Longrightarrow$ *total* (x^d)
  by (*metis co-total-def dual-sub-dist-comp dual-zero top-le*)

lemma *transitive-dense-dual*:
  transitive $x \Longrightarrow$ *dense-rel* (x^d)

using *dual-antitone dual-sub-dist-comp inf.order-lesseq-imp* **by** *blast*

end

M2-algebra

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

subclass *multirelation-algebra-4*
apply *unfold-locales*
by (*metis dual-antitone mult-sub-right-one mult-left-isotone dual-dist-comp-one*)

lemma *strong-up-closed*:
$x * 1 \le x \Longrightarrow x^d * y^d \le (x * y)^d$
by (*simp add*: *dual-dist-comp-one antisym-conv mult-sub-right-one*)

lemma *strong-up-closed-2*:
up-closed $x \Longrightarrow (x * y)^d = x^d * y^d$
by (*simp add*: *dual-dist-comp-one up-closed-def*)

subclass *lattice-ordered-pre-left-semiring-2*
apply *unfold-locales*
apply (*smt comp-up-closed dual-antitone dual-dist-comp-one dual-involutive dual-one mult-left-one mult-one-associative mult-semi-associative up-closed-def strong-up-closed-2*)
by (*smt dual-dist-comp-one dual-dist-inf dual-involutive eq-refl mult-one-associative mult-right-dist-sup*)

AAMP Theorem 8

subclass *multirelation-algebra-2* ..

AAMP Theorem 10 / Figure 3: closure properties

up-closed

lemma *dual-up-closed*:
up-closed $x \longleftrightarrow$ *up-closed* (x^d)
by (*metis dual-involutive dual-one up-closed-def strong-up-closed-2*)

contact

lemma *contact-kernel-dual*:
contact $x \longleftrightarrow$ *kernel* (x^d)
by (*metis contact-def contact-up-closed dual-dist-sup dual-involutive dual-one kernel-def kernel-up-closed up-closed-def strong-up-closed-2*)

add-distributive contact

lemma *sup-dist-contact-inf-dist-kernel-dual*:
sup-dist-contact $x \longleftrightarrow$ *inf-dist-kernel* (x^d)
proof

assume *1*: *sup-dist-contact x*
hence *2*: *up-closed x*
using *sup-dist-contact-def contact-up-closed* **by** *auto*
have *sup-distributive x*
using *1 sup-dist-contact-def* **by** *auto*
hence *inf-distributive* (x^d)
using *2* **by** (*smt sup-distributive-def dual-dist-comp-one dual-dist-inf dual-involutive inf-distributive-def up-closed-def*)
thus *inf-dist-kernel* (x^d)
using *1 contact-kernel-dual sup-dist-contact-def inf-dist-kernel-def* **by** *blast*
next
assume *3*: *inf-dist-kernel* (x^d)
hence *4*: *up-closed* (x^d)
using *kernel-up-closed inf-dist-kernel-def* **by** *auto*
have *inf-distributive* (x^d)
using *3 inf-dist-kernel-def* **by** *auto*
hence *sup-distributive* (x^{dd})
using *4* **by** (*smt inf-distributive-def sup-distributive-def dual-dist-sup dual-involutive strong-up-closed-2*)
thus *sup-dist-contact x*
using *3 contact-kernel-dual sup-dist-contact-def dual-involutive inf-dist-kernel-def* **by** *auto*
qed

test

lemma *test-co-test-dual*:
test x $\longleftrightarrow$ *co-test* (x^d)
by (*smt* (*z3*) *co-test-def co-test-up-closed dual-dist-comp-one dual-dist-inf dual-involutive dual-one dual-top test-def test-up-closed up-closed-def*)

vector

lemma *vector-dual*:
vector x $\longleftrightarrow$ *vector* (x^d)
by (*metis dual-dist-comp-one dual-involutive mult-top-associative*)

end

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

subclass *multirelation-algebra-5*
apply *unfold-locales*
by (*metis dual-sub-dist-comp dual-sub-dist-comp-one order.eq-iff mult-one-associative*)

proposition *dense-rel x* $\wedge$ *coreflexive x* $\longrightarrow$ *up-closed x* **nitpick** [*expect=genuine,card=5*] **oops**
proposition $x * top \sqcap y * z \leq (x * top \sqcap y) * z$ **nitpick** [*expect=genuine,card=8*] **oops**

end

M3-algebra

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

lemma *mult-right-dist-inf*:
$(x \sqcap y) * z = x * z \sqcap y * z$
by (*metis dual-dist-sup dual-dist-comp dual-involutive mult-right-dist-sup*)

AAMP Theorem 9

subclass *idempotent-left-semiring*
apply *unfold-locales*
apply (*metis order.antisym dual-antitone dual-dist-comp dual-involutive mult-semi-associative*)
apply *simp*
by (*metis order.antisym dual-antitone dual-dist-comp dual-involutive dual-one mult-sub-right-one*)

subclass *multirelation-algebra-6*
apply *unfold-locales*
by (*simp-all add*: *dual-dist-comp*)

lemma *vector-inf-comp*:
$(x * top \sqcap y) * z = x * top \sqcap y * z$
by (*simp add*: *vector-left-annihilator mult-right-dist-inf mult.assoc*)

lemma *vector-zero-inf-comp*:
$(x * bot \sqcap y) * z = x * bot \sqcap y * z$
by (*simp add*: *mult-right-dist-inf mult.assoc*)

AAMP Theorem 10 / Figure 3: closure properties

total

lemma *inf-total*:
$total\ x \Longrightarrow total\ y \Longrightarrow total\ (x \sqcap y)$
by (*simp add*: *mult-right-dist-inf*)

lemma *comp-total*:
$total\ x \Longrightarrow total\ y \Longrightarrow total\ (x * y)$
by (*simp add*: *mult-assoc*)

lemma *total-co-total-dual*:
$total\ x \longleftrightarrow co\text{-}total\ (x^d)$
by (*metis co-total-def dual-dist-comp dual-involutive dual-top*)

dense

lemma *transitive-iff-dense-dual*:

transitive $x \longleftrightarrow$ *dense-rel* (x^d)
by (*metis dual-antitone dual-dist-comp dual-involutive*)

idempotent

lemma *idempotent-dual*:
idempotent $x \longleftrightarrow$ *idempotent* (x^d)
using *dual-involutive idempotent-transitive-dense transitive-iff-dense-dual* **by** *auto*

add-distributive

lemma *comp-sup-distributive*:
sup-distributive $x \Longrightarrow$ *sup-distributive* $y \Longrightarrow$ *sup-distributive* $(x * y)$
by (*simp add*: *sup-distributive-def mult.assoc*)

lemma *sup-inf-distributive-dual*:
sup-distributive $x \longleftrightarrow$ *inf-distributive* (x^d)
by (*smt* (*verit*, *ccfv-threshold*) *sup-distributive-def dual-dist-sup dual-dist-comp dual-dist-inf dual-involutive inf-distributive-def*)

inf-distributive

lemma *inf-inf-distributive*:
inf-distributive $x \Longrightarrow$ *inf-distributive* $y \Longrightarrow$ *inf-distributive* $(x \sqcap y)$
by (*metis sup-inf-distributive-dual sup-sup-distributive dual-dist-inf dual-involutive*)

lemma *comp-inf-distributive*:
inf-distributive $x \Longrightarrow$ *inf-distributive* $y \Longrightarrow$ *inf-distributive* $(x * y)$
by (*simp add*: *inf-distributive-def mult.assoc*)

proposition *co-total* $x \wedge$ *transitive* $x \wedge$ *up-closed* $x \longrightarrow$ *coreflexive* x **nitpick** [*expect*=*genuine*,*card*=*5*] **oops**
proposition *total* $x \wedge$ *dense-rel* $x \wedge$ *up-closed* $x \longrightarrow$ *reflexive* x **nitpick** [*expect*=*genuine*,*card*=*5*] **oops**
proposition $x * top \sqcap x^d * bot = bot$ **nitpick** [*expect*=*genuine*,*card*=*6*] **oops**

end

class *multirelation-algebra-7* = *multirelation-algebra-4* +
assumes *vector-inf-comp*: $(x * top \sqcap y) * z = x * top \sqcap y * z$
begin

lemma *vector-zero-inf-comp*:
$(x * bot \sqcap y) * z = x * bot \sqcap y * z$
by (*metis vector-inf-comp vector-mult-closed zero-vector*)

lemma *test-sup-distributive*:
test $x \Longrightarrow$ *sup-distributive* x
by (*metis sup-distributive-def inf-sup-distrib1 mult-left-one test-def vector-inf-comp*)

lemma *test-inf-distributive*:
test x $\Longrightarrow$ *inf-distributive x*
by (*smt* (*z3*) *inf.right-idem inf.sup-monoid.add-assoc inf.sup-monoid.add-commute inf-distributive-def mult-left-one test-def vector-inf-comp*)

lemma *test-inf-dist-kernel*:
test x $\Longrightarrow$ *inf-dist-kernel x*
by (*metis inf.idem inf.sup-monoid.add-assoc kernel-def inf-dist-kernel-def mult-left-one test-def test-inf-distributive vector-inf-comp*)

lemma *co-test-inf-distributive*:
assumes *co-test x*
shows *inf-distributive x*
proof −
have $x = x * bot \sqcup 1$
using *assms co-test-def* **by** *auto*
hence $\forall y\ z\ .\ x * y \sqcap x * z = x * (y \sqcap z)$
by (*metis distrib-imp1 inf-sup-absorb inf-sup-distrib1 mult-left-one mult-left-top mult-right-dist-sup sup-top-right vector-zero-inf-comp*)
thus *inf-distributive x*
by (*simp add*: *inf-distributive-def*)
qed

lemma *co-test-sup-distributive*:
assumes *co-test x*
shows *sup-distributive x*
proof −
have $x = x * bot \sqcup 1$
using *assms co-test-def* **by** *auto*
hence $\forall y\ z\ .\ x * (y \sqcup z) = x * y \sqcup x * z$
by (*metis sup-sup-distributive sup-distributive-def inf-sup-absorb mult-left-top one-sup-distributive sup.idem sup-top-right vector-zero-inf-comp*)
thus *sup-distributive x*
by (*simp add*: *sup-distributive-def*)
qed

lemma *co-test-sup-dist-contact*:
co-test x $\Longrightarrow$ *sup-dist-contact x*
by (*simp add*: *sup-dist-contact-def co-test-sup-distributive co-test-contact*)

end

end

6 Boolean Semirings

theory *Boolean-Semirings*

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

begin

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

sublocale *boolean-algebra* **where** *minus* = $\lambda x\ y\ .\ x \sqcap (-y)$ **and** *inf* = *inf* **and** *sup* = *sup* **and** *bot* = *bot* **and** *top* = *top*
  apply *unfold-locales*
  apply (*simp add*: *inf-complement*)
  apply (*simp add*: *sup-complement*)
  by *simp*

end

M0-algebra

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

Section 7

lemma *vector-1*:
  $vector\ x \longleftrightarrow x * top \le x$
  by (*simp add*: *antisym-conv top-right-mult-increasing*)

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

lemma *zero-vector-left-zero*:
  assumes *zero-vector x*
    shows $x * y = x * bot$
proof –
  have $x * y \le x * bot$
    by (*metis assms mult-isotone top.extremum vector-mult-closed zero-vector zero-vector-def*)
  thus *?thesis*
    by (*simp add*: *order.antisym mult-right-isotone*)
qed

lemma *zero-vector-1*:
  $zero\text{-}vector\ x \longleftrightarrow (\forall y\ .\ x * y = x * bot)$
  by (*metis top-right-mult-increasing zero-vector-def zero-vector-left-zero*)

lemma *zero-vector-2*:
  $zero\text{-}vector\ x \longleftrightarrow (\forall y\ .\ x * y \le x * bot)$
  by (*metis eq-refl order-trans top-right-mult-increasing zero-vector-def zero-vector-left-zero*)

lemma *zero-vector-3*:
zero-vector x $\longleftrightarrow$ $x * 1 = x * bot$
by (*metis mult-sub-right-one zero-vector-def zero-vector-left-zero*)

lemma *zero-vector-4*:
zero-vector x $\longleftrightarrow$ $x * 1 \le x * bot$
using *order.antisym mult-right-isotone zero-vector-3* **by** *auto*

lemma *zero-vector-5*:
zero-vector x $\longleftrightarrow$ $x * top = x * bot$
by (*metis top-right-mult-increasing zero-vector-def zero-vector-left-zero*)

lemma *zero-vector-6*:
zero-vector x $\longleftrightarrow$ $x * top \le x * bot$
by (*meson mult-right-isotone order-trans top.extremum zero-vector-2*)

lemma *zero-vector-7*:
zero-vector x $\longleftrightarrow$ $(\forall y . x * top = x * y)$
by (*metis zero-vector-1*)

lemma *zero-vector-8*:
zero-vector x $\longleftrightarrow$ $(\forall y . x * top \le x * y)$
by (*metis zero-vector-6 zero-vector-left-zero*)

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

lemma *zero-vector-0*:
zero-vector x $\longleftrightarrow$ $(\forall y\ z . x * y = x * z)$
by (*metis zero-vector-5 zero-vector-left-zero*)

Theorem 6 / Figure 2: relations between properties

lemma *co-vector-zero-vector-one-vector*:
co-vector x $\longleftrightarrow$ *zero-vector x* $\wedge$ *one-vector x*
using *co-vector-def one-vector-def zero-vector-def* **by** *auto*

lemma *up-closed-one-vector*:
up-closed x $\Longrightarrow$ *one-vector x*
by (*metis bot-least mult-right-isotone up-closed-def one-vector-def*)

lemma *zero-vector-dense*:
zero-vector x $\Longrightarrow$ *dense-rel x*
by (*metis zero-vector-0 zero-vector-def*)

lemma *zero-vector-sup-distributive*:
zero-vector x $\Longrightarrow$ *sup-distributive x*
by (*metis sup-distributive-def sup-idem zero-vector-0*)

lemma *zero-vector-inf-distributive*:
zero-vector x $\Longrightarrow$ *inf-distributive x*
by (*metis inf-idem inf-distributive-def zero-vector-0*)

lemma *up-closed-zero-vector-vector*:
up-closed x $\Longrightarrow$ *zero-vector x* $\Longrightarrow$ *vector x*
by (*metis up-closed-def zero-vector-0*)

lemma *zero-vector-one-vector-vector*:
zero-vector x $\Longrightarrow$ *one-vector x* $\Longrightarrow$ *vector x*
by (*metis one-vector-def vector-1 zero-vector-0*)

lemma *co-vector-vector*:
co-vector x $\Longrightarrow$ *vector x*
by (*simp add*: *co-vector-zero-vector-one-vector zero-vector-one-vector-vector*)

Theorem 10 / Figure 3: closure properties

zero-vector

lemma *zero-zero-vector*:
zero-vector bot
by (*simp add*: *zero-vector-def*)

lemma *sup-zero-vector*:
zero-vector x $\Longrightarrow$ *zero-vector y* $\Longrightarrow$ *zero-vector* $(x \sqcup y)$
by (*simp add*: *mult-right-dist-sup zero-vector-3*)

lemma *comp-zero-vector*:
zero-vector x $\Longrightarrow$ *zero-vector y* $\Longrightarrow$ *zero-vector* $(x * y)$
by (*metis mult-one-associative zero-vector-0*)

one-vector

lemma *zero-one-vector*:
one-vector bot
by (*simp add*: *one-vector-def*)

lemma *one-one-vector*:
one-vector 1
by (*simp add*: *one-up-closed up-closed-one-vector*)

lemma *top-one-vector*:
one-vector top
by (*simp add*: *one-vector-def*)

lemma *sup-one-vector*:
one-vector x $\Longrightarrow$ *one-vector y* $\Longrightarrow$ *one-vector* $(x \sqcup y)$
by (*simp add*: *mult-right-dist-sup order-trans one-vector-def*)

lemma *inf-one-vector*:

one-vector x $\Longrightarrow$ *one-vector y* $\Longrightarrow$ *one-vector* $(x \sqcap y)$
by (*meson order.trans inf.boundedI mult-right-sub-dist-inf-left mult-right-sub-dist-inf-right one-vector-def*)

lemma *comp-one-vector*:
one-vector x $\Longrightarrow$ *one-vector y* $\Longrightarrow$ *one-vector* $(x * y)$
using *mult-isotone mult-semi-associative order-lesseq-imp one-vector-def* **by** *blast*

end

context *multirelation-algebra-1*
begin

Theorem 10 / Figure 3: closure properties

zero-vector

lemma *top-zero-vector*:
zero-vector top
by (*simp add*: *mult-left-top zero-vector-def*)

end

M1-algebra

context *multirelation-algebra-2*
begin

Section 7

lemma *zero-vector-10*:
zero-vector x $\longleftrightarrow$ $x * top = x * 1$
by (*metis mult-one-associative mult-top-associative zero-vector-7*)

lemma *zero-vector-11*:
zero-vector x $\longleftrightarrow$ $x * top \leq x * 1$
using *order.antisym mult-right-isotone zero-vector-10* **by** *fastforce*

Theorem 6 / Figure 2: relations between properties

lemma *vector-zero-vector*:
vector x $\Longrightarrow$ *zero-vector x*
by (*simp add*: *zero-vector-def vector-left-annihilator*)

lemma *vector-up-closed-zero-vector*:
vector x $\longleftrightarrow$ *up-closed x* $\wedge$ *zero-vector x*
using *up-closed-zero-vector-vector vector-up-closed vector-zero-vector* **by** *blast*

lemma *vector-zero-vector-one-vector*:
vector x $\longleftrightarrow$ *zero-vector x* $\wedge$ *one-vector x*
by (*simp add*: *co-vector-zero-vector-one-vector vector-co-vector*)

proposition $(x * bot \sqcap y) * 1 = x * bot \sqcap y * 1$ **nitpick** $[expect=genuine,card=7]$ **oops**

end

M3-algebra

context *up-closed-multirelation-algebra*
begin

lemma *up-closed*:
  up-closed x
  by (*simp add*: *up-closed-def*)

lemma *dedekind-1-left*:
  $x * 1 \sqcap y \leq (x \sqcap y * 1) * 1$
  by *simp*

Theorem 10 / Figure 3: closure properties

zero-vector

lemma *zero-vector-dual*:
  zero-vector $x \longleftrightarrow$ *zero-vector* (x^d)
  using *up-closed-zero-vector-vector vector-dual vector-zero-vector up-closed* **by** *blast*

end

complemented M0-algebra

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

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

Theorem 10 / Figure 3: closure properties

down-closed

lemma *zero-down-closed*:
  down-closed bot
  by (*simp add*: *down-closed-def*)

lemma *top-down-closed*:
  down-closed top
  by (*simp add*: *down-closed-def*)

lemma *complement-down-closed-up-closed*:
  down-closed $x \longleftrightarrow$ *up-closed* $(-x)$
  using *down-closed-def order.antisym mult-sub-right-one up-closed-def* **by** *auto*

lemma *sup-down-closed*:

down-closed $x \Longrightarrow$ *down-closed* $y \Longrightarrow$ *down-closed* $(x \sqcup y)$
by (*simp add*: *complement-down-closed-up-closed inf-up-closed*)

lemma *inf-down-closed*:
down-closed $x \Longrightarrow$ *down-closed* $y \Longrightarrow$ *down-closed* $(x \sqcap y)$
by (*simp add*: *complement-down-closed-up-closed sup-up-closed*)

end

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

subclass *lattice-ordered-pre-left-semiring-b* **..**

Theorem 7.1

lemma *complement-mult-zero-sub*:
$-(x * bot) \leq -x * bot$
proof −
have $top = -x * bot \sqcup x * bot$
by (*metis compl-sup-top mult-left-top mult-right-dist-sup*)
thus *?thesis*
by (*simp add*: *heyting.implies-order sup.commute*)
qed

Theorem 7.2

lemma *transitive-zero-vector-complement*:
transitive $x \Longrightarrow$ *zero-vector* $(-x)$
by (*meson complement-mult-zero-sub compl-mono mult-right-isotone order-trans zero-vector-def bot-least*)

lemma *transitive-dense-complement*:
transitive $x \Longrightarrow$ *dense-rel* $(-x)$
by (*simp add*: *zero-vector-dense transitive-zero-vector-complement*)

lemma *transitive-sup-distributive-complement*:
transitive $x \Longrightarrow$ *sup-distributive* $(-x)$
by (*simp add*: *zero-vector-sup-distributive transitive-zero-vector-complement*)

lemma *transitive-inf-distributive-complement*:
transitive $x \Longrightarrow$ *inf-distributive* $(-x)$
by (*simp add*: *zero-vector-inf-distributive transitive-zero-vector-complement*)

lemma *up-closed-zero-vector-complement*:
up-closed $x \Longrightarrow$ *zero-vector* $(-x)$
by (*meson complement-mult-zero-sub compl-le-swap2 one-vector-def order-trans up-closed-one-vector zero-vector-def*)

lemma *up-closed-dense-complement*:

up-closed $x \Longrightarrow$ *dense-rel* $(-x)$
by (*simp add*: *zero-vector-dense up-closed-zero-vector-complement*)

lemma *up-closed-sup-distributive-complement*:
up-closed $x \Longrightarrow$ *sup-distributive* $(-x)$
by (*simp add*: *zero-vector-sup-distributive up-closed-zero-vector-complement*)

lemma *up-closed-inf-distributive-complement*:
up-closed $x \Longrightarrow$ *inf-distributive* $(-x)$
by (*simp add*: *zero-vector-inf-distributive up-closed-zero-vector-complement*)

Theorem 10 / Figure 3: closure properties

closure under complement

lemma *co-total-total*:
co-total $x \Longrightarrow$ *total* $(-x)$
by (*metis complement-mult-zero-sub co-total-def compl-bot-eq mult-left-sub-dist-sup-right sup-bot-right top-le*)

lemma *complement-one-vector-zero-vector*:
one-vector $x \Longrightarrow$ *zero-vector* $(-x)$
using *compl-mono complement-mult-zero-sub one-vector-def order-trans zero-vector-def* **by** *blast*

Theorem 6 / Figure 2: relations between properties

lemma *down-closed-zero-vector*:
down-closed $x \Longrightarrow$ *zero-vector* x
using *complement-down-closed-up-closed up-closed-zero-vector-complement* **by** *force*

lemma *down-closed-one-vector-vector*:
down-closed $x \Longrightarrow$ *one-vector* $x \Longrightarrow$ *vector* x
by (*simp add*: *down-closed-zero-vector zero-vector-one-vector-vector*)

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

end

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

Theorem 7.3

lemma *schroeder-top-sub*:
$-(x * top) * top \leq -x$
proof –
have $-(x * top) * top \sqcap x \leq bot$
by (*metis dedekind-top-left p-inf zero-vector*)

thus *?thesis*
by (*simp add*: *shunting-1*)
qed

Theorem 7.4

lemma *schroeder-top*:
$x * top \le y \longleftrightarrow -y * top \le -x$
apply (*rule iffI*)
using *compl-mono inf.order-trans mult-left-isotone schroeder-top-sub* **apply** *blast*
by (*metis compl-mono double-compl mult-left-isotone order-trans schroeder-top-sub*)

Theorem 7.5

lemma *schroeder-top-eq*:
$-(x * top) * top = -(x * top)$
using *vector-1 vector-mult-closed vector-top-closed schroeder-top* **by** *auto*

lemma *schroeder-one-eq*:
$-(x * top) * 1 = -(x * top)$
by (*metis top-mult-right-one schroeder-top-eq*)

Theorem 7.6

lemma *vector-inf-comp*:
$x * top \sqcap y * z = (x * top \sqcap y) * z$
proof (*rule order.antisym*)
have $x * top \sqcap y * z = x * top \sqcap ((x * top \sqcap y) \sqcup (-(x * top) \sqcap y)) * z$
by (*simp add*: *inf-commute*)
also have ... $= x * top \sqcap ((x * top \sqcap y) * z \sqcup (-(x * top) \sqcap y) * z)$
by (*simp add*: *inf-sup-distrib2 mult-right-dist-sup*)
also have ... $= (x * top \sqcap (x * top \sqcap y) * z) \sqcup (x * top \sqcap (-(x * top) \sqcap y) * z)$
by (*simp add*: *inf-sup-distrib1*)
also have ... $\le (x * top \sqcap y) * z \sqcup (x * top \sqcap (-(x * top) \sqcap y) * z)$
by (*simp add*: *le-infI2*)
also have ... $\le (x * top \sqcap y) * z \sqcup (x * top \sqcap -(x * top) * z)$
by (*metis inf.sup-left-isotone inf-commute mult-right-sub-dist-inf-left sup-right-isotone*)
also have ... $\le (x * top \sqcap y) * z \sqcup (x * top \sqcap -(x * top) * top)$
using *inf.sup-right-isotone mult-right-isotone sup-right-isotone* **by** *auto*
also have ... $= (x * top \sqcap y) * z$
by (*simp add*: *schroeder-top-eq*)
finally show $x * top \sqcap y * z \le (x * top \sqcap y) * z$
.
next
show $(x * top \sqcap y) * z \le x * top \sqcap y * z$
by (*metis inf.bounded-iff mult-left-top mult-right-sub-dist-inf-left mult-right-sub-dist-inf-right mult-semi-associative order-lesseq-imp*)
qed

lemma *dedekind-top-left-var*:
$x * top \sqcap y \leq (x \sqcap y * top) * top$
by (*metis inf.commute top-right-mult-increasing vector-inf-comp*)

Theorem 7.7

lemma *vector-zero-inf-comp*:
$(x * bot \sqcap y) * z = x * bot \sqcap y * z$
by (*metis vector-inf-comp vector-mult-closed zero-vector*)

lemma *vector-zero-inf-comp-2*:
$(x * bot \sqcap y) * z = (x * bot \sqcap y * 1) * z$
by (*simp add*: *vector-zero-inf-comp*)

Theorem 7.8

lemma *comp-zero-inf-2*:
$x * bot \sqcap y * bot = (x \sqcap y) * bot$
using *order.antisym mult-right-sub-dist-inf comp-zero-inf vector-zero-inf-comp*
by *auto*

lemma *comp-zero-inf-3*:
$x * bot \sqcap y * bot = (x * bot \sqcap y) * bot$
by (*simp add*: *vector-zero-inf-comp*)

lemma *comp-zero-inf-4*:
$x * bot \sqcap y * bot = (x * bot \sqcap y * bot) * bot$
by (*metis comp-zero-inf-2 inf.commute vector-zero-inf-comp*)

lemma *comp-zero-inf-5*:
$x * bot \sqcap y * bot = (x * 1 \sqcap y * 1) * bot$
by (*metis comp-zero-inf-2 mult-one-associative*)

lemma *comp-zero-inf-6*:
$x * bot \sqcap y * bot = (x * 1 \sqcap y * bot) * bot$
using *inf.sup-monoid.add-commute vector-zero-inf-comp* **by** *fastforce*

lemma *comp-zero-inf-7*:
$x * bot \sqcap y * bot = (x * 1 \sqcap y) * bot$
by (*metis comp-zero-inf-2 mult-one-associative*)

Theorem 10 / Figure 3: closure properties

zero-vector

lemma *inf-zero-vector*:
$zero\text{-}vector\ x \Longrightarrow zero\text{-}vector\ y \Longrightarrow zero\text{-}vector\ (x \sqcap y)$
by (*metis comp-zero-inf-2 inf.sup-mono zero-vector-def*)

down-closed

lemma *comp-down-closed*:
$down\text{-}closed\ x \Longrightarrow down\text{-}closed\ y \Longrightarrow down\text{-}closed\ (x * y)$

by (*metis complement-down-closed-up-closed down-closed-zero-vector up-closed-def zero-vector-0 schroeder-one-eq*)

closure under complement

lemma *complement-vector*:
vector $x \longleftrightarrow$ *vector* $(-x)$
using *vector-1 schroeder-top* **by** *blast*

lemma *complement-zero-vector-one-vector*:
zero-vector $x \Longrightarrow$ *one-vector* $(-x)$
by (*metis comp-zero-inf-2 order.antisym complement-mult-zero-sub double-compl inf.sup-monoid.add-commute mult-left-zero one-vector-def order.refl pseudo-complement top-right-mult-increasing zero-vector-0*)

lemma *complement-zero-vector-one-vector-iff*:
zero-vector $x \longleftrightarrow$ *one-vector* $(-x)$
using *complement-zero-vector-one-vector complement-one-vector-zero-vector* **by** *force*

lemma *complement-one-vector-zero-vector-iff*:
one-vector $x \longleftrightarrow$ *zero-vector* $(-x)$
using *complement-zero-vector-one-vector complement-one-vector-zero-vector* **by** *force*

Theorem 6 / Figure 2: relations between properties

lemma *vector-down-closed*:
vector $x \Longrightarrow$ *down-closed* x
using *complement-vector complement-down-closed-up-closed vector-up-closed* **by** *blast*

lemma *co-vector-down-closed*:
co-vector $x \Longrightarrow$ *down-closed* x
by (*simp add*: *co-vector-vector vector-down-closed*)

lemma *vector-down-closed-one-vector*:
vector $x \longleftrightarrow$ *down-closed* $x \wedge$ *one-vector* x
using *down-closed-one-vector-vector up-closed-one-vector vector-up-closed vector-down-closed* **by** *blast*

lemma *vector-up-closed-down-closed*:
vector $x \longleftrightarrow$ *up-closed* $x \wedge$ *down-closed* x
using *down-closed-zero-vector up-closed-zero-vector-vector vector-up-closed vector-down-closed* **by** *blast*

Section 7

lemma *vector-b1*:
vector $x \longleftrightarrow -x * top = -x$
using *complement-vector* **by** *auto*

lemma *vector-b2*:
  vector x $\longleftrightarrow$ $-x * bot = -x$
  by (*metis down-closed-zero-vector vector-mult-closed zero-vector zero-vector-left-zero vector-b1 vector-down-closed*)

lemma *covector-b1*:
  co-vector x $\longleftrightarrow$ $-x * top = -x$
  using *co-vector-def co-vector-vector vector-b1 vector-b2* **by** *force*

lemma *covector-b2*:
  co-vector x $\longleftrightarrow$ $-x * bot = -x$
  using *covector-b1 vector-b1 vector-b2* **by** *auto*

lemma *vector-co-vector-iff*:
  vector x $\longleftrightarrow$ *co-vector x*
  by (*simp add*: *covector-b2 vector-b2*)

lemma *zero-vector-b*:
  zero-vector x $\longleftrightarrow$ $-x * bot \leq -x$
  by (*simp add*: *complement-zero-vector-one-vector-iff one-vector-def*)

lemma *one-vector-b1*:
  one-vector x $\longleftrightarrow$ $-x \leq -x * bot$
  by (*simp add*: *complement-one-vector-zero-vector-iff zero-vector-def*)

lemma *one-vector-b0*:
  one-vector x $\longleftrightarrow$ $(\forall\, y\ z\ .\ -x * y = -x * z)$
  by (*simp add*: *complement-one-vector-zero-vector-iff zero-vector-0*)

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

end

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

subclass *multirelation-algebra-1b* **..**

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

end

complemented M1-algebra

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

class *multirelation-algebra-3b* = *multirelation-algebra-3* +

complemented-distributive-lattice
begin

subclass *lattice-ordered-pre-left-semiring-b* **..**

lemma *dual-complement-commute*:
$-(x^d) = (-x)^d$
by (*metis compl-unique dual-dist-sup dual-dist-inf dual-top dual-zero inf-complement sup-compl-top*)

end

complemented M2-algebra

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

subclass *multirelation-algebra-2b* **..**

subclass *multirelation-algebra-3b* **..**

lemma *dual-down-closed*:
down-closed $x \longleftrightarrow$ *down-closed* (x^d)
using *complement-down-closed-up-closed dual-complement-commute dual-up-closed* **by** *auto*

end

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

lemma *complement-mult-zero-below*:
$-x * bot \leq -(x * bot)$
by (*simp add*: *comp-zero-inf-2 shunting-1*)

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

end

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

subclass *multirelation-algebra-5c*
apply *unfold-locales*

apply (*metis inf.sup-monoid.add-commute top-right-mult-increasing vector-inf-comp*)
using *mult-right-dist-inf vector-zero-inf-comp* **by** *auto*

lemma *complement-zero-vector*:
zero-vector $x \longleftrightarrow$ *zero-vector* $(-x)$
by (*simp add*: *zero-right-mult-decreasing zero-vector-b*)

lemma *down-closed*:
down-closed x
by (*simp add*: *down-closed-def*)

lemma *vector*:
vector x
by (*simp add*: *down-closed up-closed-def vector-up-closed-down-closed*)

end

end

7 Binary Iterings

theory *Binary-Iterings*

imports *Base*

begin

class *binary-itering* = *idempotent-left-zero-semiring* + *while* +
assumes *while-productstar*: $(x * y) \star z = z \sqcup x * ((y * x) \star (y * z))$
assumes *while-sumstar*: $(x \sqcup y) \star z = (x \star y) \star (x \star z)$
assumes *while-left-dist-sup*: $x \star (y \sqcup z) = (x \star y) \sqcup (x \star z)$
assumes *while-sub-associative*: $(x \star y) * z \leq x \star (y * z)$
assumes *while-simulate-left-plus*: $x * z \leq z * (y \star 1) \sqcup w \longrightarrow x \star (z * v) \leq z * (y \star v) \sqcup (x \star (w * (y \star v)))$
assumes *while-simulate-right-plus*: $z * x \leq y * (y \star z) \sqcup w \longrightarrow z * (x \star v) \leq y \star (z * v \sqcup w * (x \star v))$
begin

Theorem 9.1

lemma *while-zero*:
$bot \star x = x$
by (*metis sup-bot-right mult-left-zero while-productstar*)

Theorem 9.4

lemma *while-mult-increasing*:
$x * y \leq x \star y$
by (*metis le-supI2 mult.left-neutral mult-left-sub-dist-sup-left while-productstar*)

Theorem 9.2

lemma *while-one-increasing*:
$x \leq x \star 1$
by (*metis mult.right-neutral while-mult-increasing*)

Theorem 9.3

lemma *while-increasing*:
$y \leq x \star y$
by (*metis sup-left-divisibility mult-left-one while-productstar*)

Theorem 9.42

lemma *while-right-isotone*:
$y \leq z \Longrightarrow x \star y \leq x \star z$
by (*metis le-iff-sup while-left-dist-sup*)

Theorem 9.41

lemma *while-left-isotone*:
$x \leq y \Longrightarrow x \star z \leq y \star z$
using *sup-left-divisibility while-sumstar while-increasing* **by** *auto*

lemma *while-isotone*:
$w \leq x \Longrightarrow y \leq z \Longrightarrow w \star y \leq x \star z$
by (*meson order-lesseq-imp while-left-isotone while-right-isotone*)

Theorem 9.17

lemma *while-left-unfold*:
$x \star y = y \sqcup x * (x \star y)$
by (*metis mult-1-left mult-1-right while-productstar*)

lemma *while-simulate-left-plus-1*:
$x * z \leq z * (y \star 1) \Longrightarrow x \star (z * w) \leq z * (y \star w) \sqcup (x \star bot)$
by (*metis sup-bot-right mult-left-zero while-simulate-left-plus*)

Theorem 11.1

lemma *while-simulate-absorb*:
$y * x \leq x \Longrightarrow y \star x \leq x \sqcup (y \star bot)$
by (*metis while-simulate-left-plus-1 while-zero mult-1-right*)

Theorem 9.10

lemma *while-transitive*:
$x \star (x \star y) = x \star y$
by (*metis order.eq-iff sup-bot-right sup-ge2 while-left-dist-sup while-increasing while-left-unfold while-simulate-absorb*)

Theorem 9.25

lemma *while-slide*:
$(x * y) \star (x * z) = x * ((y * x) \star z)$
by (*metis mult-left-dist-sup while-productstar mult-assoc while-left-unfold*)

Theorem 9.21

lemma *while-zero-2*:
$(x * bot) \star y = x * bot \sqcup y$
by (*metis mult-left-zero sup-commute mult-assoc while-left-unfold*)

Theorem 9.5

lemma *while-mult-star-exchange*:
$x * (x \star y) = x \star (x * y)$
by (*metis mult-left-one while-slide*)

Theorem 9.18

lemma *while-right-unfold*:
$x \star y = y \sqcup (x \star (x * y))$
by (*metis while-left-unfold while-mult-star-exchange*)

Theorem 9.7

lemma *while-one-mult-below*:
$(x \star 1) * y \leq x \star y$
by (*metis mult-left-one while-sub-associative*)

lemma *while-plus-one*:
$x \star y = y \sqcup (x \star y)$
by (*simp add*: *sup.absorb2 while-increasing*)

Theorem 9.19

lemma *while-rtc-2*:
$y \sqcup x * y \sqcup (x \star (x \star y)) = x \star y$
by (*simp add*: *sup-absorb2 while-increasing while-mult-increasing while-transitive*)

Theorem 9.6

lemma *while-left-plus-below*:
$x * (x \star y) \leq x \star y$
by (*metis sup-right-divisibility while-left-unfold*)

lemma *while-right-plus-below*:
$x \star (x * y) \leq x \star y$
using *while-left-plus-below while-mult-star-exchange* **by** *auto*

lemma *while-right-plus-below-2*:
$(x \star x) * y \leq x \star y$
by (*smt order-trans while-right-plus-below while-sub-associative*)

Theorem 9.47

lemma *while-mult-transitive*:
$x \leq z \star y \Longrightarrow y \leq z \star w \Longrightarrow x \leq z \star w$
by (*smt order-trans while-right-isotone while-transitive*)

Theorem 9.48

lemma *while-mult-upper-bound*:
$x \leq z \star 1 \Longrightarrow y \leq z \star w \Longrightarrow x * y \leq z \star w$
by (*metis order.trans mult-isotone while-one-mult-below while-transitive*)

lemma *while-one-mult-while-below*:
$(y \star 1) * (y \star v) \leq y \star v$
by (*simp add*: *while-mult-upper-bound*)

Theorem 9.34

lemma *while-sub-dist*:
$x \star z \leq (x \sqcup y) \star z$
by (*simp add*: *while-left-isotone*)

lemma *while-sub-dist-1*:
$x * z \leq (x \sqcup y) \star z$
using *order.trans while-mult-increasing while-sub-dist* **by** *blast*

lemma *while-sub-dist-2*:
$x * y * z \leq (x \sqcup y) \star z$
by (*metis sup-commute mult-assoc while-mult-transitive while-sub-dist-1*)

Theorem 9.36

lemma *while-sub-dist-3*:
$x \star (y \star z) \leq (x \sqcup y) \star z$
by (*metis sup-commute while-mult-transitive while-sub-dist*)

Theorem 9.44

lemma *while-absorb-2*:
$x \leq y \Longrightarrow y \star (x \star z) = y \star z$
using *sup-left-divisibility while-sumstar while-transitive* **by** *auto*

lemma *while-simulate-right-plus-1*:
$z * x \leq y * (y \star z) \Longrightarrow z * (x \star w) \leq y \star (z * w)$
by (*metis sup-bot-right mult-left-zero while-simulate-right-plus*)

Theorem 9.39

lemma *while-sumstar-1-below*:
$x \star ((y * (x \star 1)) \star z) \leq ((x \star 1) * y) \star (x \star z)$
proof −
have *1*: $x * (((x \star 1) * y) \star (x \star z)) \leq ((x \star 1) * y) \star (x \star z)$
by (*smt sup-mono sup-ge2 mult-assoc mult-left-dist-sup mult-right-sub-dist-sup-right while-left-unfold*)
have $x \star ((y * (x \star 1)) \star z) \leq (x \star z) \sqcup (x \star (y * (((x \star 1) * y) \star ((x \star 1) * z))))$
by (*metis eq-refl while-left-dist-sup while-productstar*)
also have ... $\leq (x \star z) \sqcup (x \star ((x \star 1) * y * (((x \star 1) * y) \star ((x \star 1) * z))))$
by (*metis sup-right-isotone mult-assoc mult-left-one mult-right-sub-dist-sup-left while-left-unfold while-right-isotone*)
also have ... $\leq (x \star z) \sqcup (x \star (((x \star 1) * y) \star ((x \star 1) * z)))$

using *semiring.add-left-mono while-left-plus-below while-right-isotone* **by** *blast*
also have ... $\leq x \star (((x \star 1) * y) \star (x \star z))$
by (*meson order.trans le-supI while-increasing while-one-mult-below while-right-isotone*)
also have ... $\leq (((x \star 1) * y) \star (x \star z)) \sqcup (x \star bot)$
using *1 while-simulate-absorb* **by** *auto*
also have ... $= ((x \star 1) * y) \star (x \star z)$
by (*smt sup-assoc sup-commute sup-bot-left while-left-dist-sup while-left-unfold*)
finally show *?thesis*
.
qed

lemma *while-sumstar-2-below*:
$((x \star 1) * y) \star (x \star z) \leq (x \star y) \star (x \star z)$
by (*simp add*: *while-left-isotone while-one-mult-below*)

Theorem 9.38

lemma *while-sup-1-below*:
$x \star ((y * (x \star 1)) \star z) \leq (x \sqcup y) \star z$
proof –
have $((x \star 1) * y) \star ((x \star 1) * z) \leq (x \sqcup y) \star z$
using *while-sumstar while-isotone while-one-mult-below* **by** *auto*
hence $(y * (x \star 1)) \star z \leq z \sqcup y * ((x \sqcup y) \star z)$
by (*metis sup-right-isotone mult-right-isotone while-productstar*)
also have ... $\leq (x \sqcup y) \star z$
by (*metis sup-right-isotone sup-ge2 mult-left-isotone while-left-unfold*)
finally show *?thesis*
using *while-mult-transitive while-sub-dist* **by** *blast*
qed

Theorem 9.16

lemma *while-while-while*:
$((x \star 1) \star 1) \star y = (x \star 1) \star y$
by (*smt (z3) sup.absorb1 while-sumstar while-absorb-2 while-increasing while-one-increasing*)

lemma *while-one*:
$(1 \star 1) \star y = 1 \star y$
by (*metis while-while-while while-zero*)

Theorem 9.22

lemma *while-sup-below*:
$x \sqcup y \leq x \star (y \star 1)$
by (*metis le-supI le-supI1 while-left-dist-sup while-left-unfold while-one-increasing*)

Theorem 9.32

lemma *while-sup-2*:
$(x \sqcup y) \star z \leq (x \star (y \star 1)) \star z$

using *while-left-isotone while-sup-below* **by** *auto*

Theorem 9.45

lemma *while-sup-one-left-unfold*:
$1 \leq x \Longrightarrow x * (x \star y) = x \star y$
by (*metis order.antisym mult-1-left mult-left-isotone while-left-plus-below*)

lemma *while-sup-one-right-unfold*:
$1 \leq x \Longrightarrow x \star (x * y) = x \star y$
using *while-mult-star-exchange while-sup-one-left-unfold* **by** *auto*

Theorem 9.30

lemma *while-decompose-7*:
$(x \sqcup y) \star z = x \star (y \star ((x \sqcup y) \star z))$
by (*metis order.eq-iff order-trans while-increasing while-sub-dist-3 while-transitive*)

Theorem 9.31

lemma *while-decompose-8*:
$(x \sqcup y) \star z = (x \sqcup y) \star (x \star (y \star z))$
using *while-absorb-2* **by** *auto*

Theorem 9.27

lemma *while-decompose-9*:
$(x \star (y \star 1)) \star z = x \star (y \star ((x \star (y \star 1)) \star z))$
by (*smt sup-commute le-iff-sup order-trans while-sup-below while-increasing while-sub-dist-3*)

lemma *while-decompose-10*:
$(x \star (y \star 1)) \star z = (x \star (y \star 1)) \star (x \star (y \star z))$
proof −
have *1*: $(x \star (y \star 1)) \star z \leq (x \star (y \star 1)) \star (x \star (y \star z))$
by (*meson order.trans while-increasing while-right-isotone*)
have $x \sqcup (y \star 1) \leq x \star (y \star 1)$
using *while-increasing while-sup-below* **by** *auto*
hence $(x \star (y \star 1)) \star (x \star (y \star z)) \leq (x \star (y \star 1)) \star z$
using *while-absorb-2 while-sup-below* **by** *force*
thus *?thesis*
using *1 order.antisym* **by** *blast*
qed

lemma *while-back-loop-fixpoint*:
$z * (y \star (y * x)) \sqcup z * x = z * (y \star x)$
by (*metis sup-commute mult-left-dist-sup while-right-unfold*)

lemma *while-back-loop-prefixpoint*:
$z * (y \star 1) * y \sqcup z \leq z * (y \star 1)$
by (*metis le-supI le-supI2 mult-1-right mult-right-isotone mult-assoc while-increasing while-one-mult-below while-right-unfold*)

Theorem 9

lemma *while-loop-is-fixpoint*:
  is-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (y \star z)$
  using *is-fixpoint-def sup-commute while-left-unfold* **by** *auto*

Theorem 9

lemma *while-back-loop-is-prefixpoint*:
  is-prefixpoint $(\lambda x\ .\ x * y \sqcup z)\ (z * (y \star 1))$
  using *is-prefixpoint-def while-back-loop-prefixpoint* **by** *auto*

Theorem 9.20

lemma *while-while-sup*:
  $(1 \sqcup x) \star y = (x \star 1) \star y$
  by (*metis sup-commute while-decompose-10 while-sumstar while-zero*)

lemma *while-while-mult-sub*:
  $x \star (1 \star y) \leq (x \star 1) \star y$
  by (*metis sup-commute while-sub-dist-3 while-while-sup*)

Theorem 9.11

lemma *while-right-plus*:
  $(x \star x) \star y = x \star y$
  by (*metis sup-idem while-plus-one while-sumstar while-transitive*)

Theorem 9.12

lemma *while-left-plus*:
  $(x * (x \star 1)) \star y = x \star y$
  by (*simp add*: *while-mult-star-exchange while-right-plus*)

Theorem 9.9

lemma *while-below-while-one*:
  $x \star x \leq x \star 1$
  by (*meson eq-refl while-mult-transitive while-one-increasing*)

lemma *while-below-while-one-mult*:
  $x * (x \star x) \leq x * (x \star 1)$
  by (*simp add*: *mult-right-isotone while-below-while-one*)

Theorem 9.23

lemma *while-sup-sub-sup-one*:
  $x \star (x \sqcup y) \leq x \star (1 \sqcup y)$
  using *semiring.add-right-mono while-left-dist-sup while-below-while-one* **by** *auto*

lemma *while-sup-sub-sup-one-mult*:
  $x * (x \star (x \sqcup y)) \leq x * (x \star (1 \sqcup y))$
  by (*simp add*: *mult-right-isotone while-sup-sub-sup-one*)

lemma *while-elimination*:

$x * y = bot \Longrightarrow x * (y \star z) = x * z$
by (*metis sup-bot-right mult-assoc mult-left-dist-sup mult-left-zero while-left-unfold*)

Theorem 9.8

lemma *while-square*:
$(x * x) \star y \leq x \star y$
by (*metis while-left-isotone while-mult-increasing while-right-plus*)

Theorem 9.35

lemma *while-mult-sub-sup*:
$(x * y) \star z \leq (x \sqcup y) \star z$
by (*metis while-increasing while-isotone while-mult-increasing while-sumstar*)

Theorem 9.43

lemma *while-absorb-1*:
$x \leq y \Longrightarrow x \star (y \star z) = y \star z$
by (*metis order.antisym le-iff-sup while-increasing while-sub-dist-3*)

lemma *while-absorb-3*:
$x \leq y \Longrightarrow x \star (y \star z) = y \star (x \star z)$
by (*simp add*: *while-absorb-1 while-absorb-2*)

Theorem 9.24

lemma *while-square-2*:
$(x * x) \star ((x \sqcup 1) * y) \leq x \star y$
by (*metis le-supI while-increasing while-mult-transitive while-mult-upper-bound while-one-increasing while-square*)

lemma *while-separate-unfold-below*:
$(y * (x \star 1)) \star z \leq (y \star z) \sqcup (y \star (y * x * (x \star ((y * (x \star 1)) \star z))))$
proof −
have $(y * (x \star 1)) \star z = (y \star (y * x * (x \star 1))) \star (y \star z)$
by (*metis mult-assoc mult-left-dist-sup mult-1-right while-left-unfold while-sumstar*)
hence $(y * (x \star 1)) \star z = (y \star z) \sqcup (y \star (y * x * (x \star 1))) * ((y * (x \star 1)) \star z)$
using *while-left-unfold* **by** *auto*
also have ... $\leq (y \star z) \sqcup (y \star (y * x * (x \star 1)) * ((y * (x \star 1)) \star z))$
using *sup-right-isotone while-sub-associative* **by** *auto*
also have ... $\leq (y \star z) \sqcup (y \star (y * x * (x \star ((y * (x \star 1)) \star z))))$
by (*smt sup-right-isotone mult-assoc mult-right-isotone while-one-mult-below while-right-isotone*)
finally show *?thesis*
.
qed

Theorem 9.33

lemma *while-mult-zero-sup*:
$(x \sqcup y * bot) \star z = x \star ((y * bot) \star z)$

proof −
  have $(x \sqcup y * bot) \star z = (x \star (y * bot)) \star (x \star z)$
    by (*simp add*: *while-sumstar*)
  also have ... $= (x \star z) \sqcup (x \star (y * bot)) * ((x \star (y * bot)) \star (x \star z))$
    using *while-left-unfold* **by** *auto*
  also have ... $\leq (x \star z) \sqcup (x \star (y * bot))$
    by (*metis sup-right-isotone mult-assoc mult-left-zero while-sub-associative*)
  also have ... $= x \star ((y * bot) \star z)$
    by (*simp add*: *sup-commute while-left-dist-sup while-zero-2*)
  finally show *?thesis*
    by (*simp add*: *order.antisym while-sub-dist-3*)
qed

lemma *while-sup-mult-zero*:
  $(x \sqcup y * bot) \star y = x \star y$
  by (*simp add*: *sup-absorb2 zero-right-mult-decreasing while-mult-zero-sup while-zero-2*)

lemma *while-mult-zero-sup-2*:
  $(x \sqcup y * bot) \star z = (x \star z) \sqcup (x \star (y * bot))$
  by (*simp add*: *sup-commute while-left-dist-sup while-mult-zero-sup while-zero-2*)

lemma *while-sup-zero-star*:
  $(x \sqcup y * bot) \star z = x \star (y * bot \sqcup z)$
  by (*simp add*: *while-mult-zero-sup while-zero-2*)

lemma *while-unfold-sum*:
  $(x \sqcup y) \star z = (x \star z) \sqcup (x \star (y * ((x \sqcup y) \star z)))$
  apply (*rule order.antisym*)
  apply (*metis semiring.add-left-mono while-sub-associative while-sumstar while-left-unfold*)
  by (*metis le-supI while-decompose-7 while-mult-increasing while-right-isotone while-sub-dist*)

lemma *while-simulate-left*:
  $x * z \leq z * y \sqcup w \Longrightarrow x \star (z * v) \leq z * (y \star v) \sqcup (x \star (w * (y \star v)))$
  by (*metis sup-left-isotone mult-right-isotone order-trans while-one-increasing while-simulate-left-plus*)

lemma *while-simulate-right*:
  assumes $z * x \leq y * z \sqcup w$
    shows $z * (x \star v) \leq y \star (z * v \sqcup w * (x \star v))$
proof −
  have $y * z \sqcup w \leq y * (y \star z) \sqcup w$
    using *sup-left-isotone while-increasing while-mult-star-exchange* **by** *force*
  thus *?thesis*
    by (*meson assms order.trans while-simulate-right-plus*)
qed

lemma *while-simulate*:
$z * x \leq y * z \Longrightarrow z * (x \star v) \leq y \star (z * v)$
by (*metis sup-bot-right mult-left-zero while-simulate-right*)

Theorem 9.14

lemma *while-while-mult*:
$1 \star (x \star y) = (x \star 1) \star y$
proof –
have $(x \star 1) \star y \leq (x \star 1) * ((x \star 1) \star y)$
by (*simp add*: *while-increasing while-sup-one-left-unfold*)
also have ... $\leq 1 \star ((x \star 1) * y)$
by (*simp add*: *while-one-mult-while-below while-simulate*)
also have ... $\leq 1 \star (x \star y)$
by (*simp add*: *while-isotone while-one-mult-below*)
finally show *?thesis*
by (*metis order.antisym while-sub-dist-3 while-while-sup*)
qed

lemma *while-simulate-left-1*:
$x * z \leq z * y \Longrightarrow x \star (z * v) \leq z * (y \star v) \sqcup (x \star bot)$
by (*meson order.trans mult-right-isotone while-one-increasing while-simulate-left-plus-1*)

Theorem 9.46

lemma *while-associative-1*:
assumes $1 \leq z$
shows $x \star (y * z) = (x \star y) * z$
proof –
have $x \star (y * z) \leq x \star ((x \star y) * z)$
by (*simp add*: *mult-isotone while-increasing while-right-isotone*)
also have ... $\leq (x \star y) * (bot \star z) \sqcup (x \star bot)$
by (*metis mult-assoc mult-right-sub-dist-sup-right while-left-unfold while-simulate-absorb while-zero*)
also have ... $\leq (x \star y) * z \sqcup (x \star bot) * z$
by (*metis assms le-supI sup-ge1 sup-ge2 case-split-right while-plus-one while-zero*)
also have ... $= (x \star y) * z$
by (*metis sup-bot-right mult-right-dist-sup while-left-dist-sup*)
finally show *?thesis*
by (*simp add*: *order.antisym while-sub-associative*)
qed

Theorem 9.29

lemma *while-associative-while-1*:
$x \star (y * (z \star 1)) = (x \star y) * (z \star 1)$
by (*simp add*: *while-associative-1 while-increasing*)

Theorem 9.13

lemma *while-one-while*:

$(x \star 1) * (y \star 1) = x \star (y \star 1)$
by (*metis mult-left-one while-associative-while-1*)

lemma *while-decompose-5-below*:
$(x \star (y \star 1)) \star z \leq (y \star (x \star 1)) \star z$
by (*smt* (*z3*) *while-left-dist-sup while-sumstar while-absorb-2*
while-one-increasing while-plus-one while-sub-dist)

Theorem 9.26

lemma *while-decompose-5*:
$(x \star (y \star 1)) \star z = (y \star (x \star 1)) \star z$
by (*simp add*: *order.antisym while-decompose-5-below*)

lemma *while-decompose-4*:
$(x \star (y \star 1)) \star z = x \star ((y \star (x \star 1)) \star z)$
using *while-absorb-1 while-decompose-5 while-sup-below* **by** *auto*

Theorem 11.7

lemma *while-simulate-2*:
$y * (x \star 1) \leq x \star (y \star 1) \longleftrightarrow y \star (x \star 1) \leq x \star (y \star 1)$
proof
assume $y * (x \star 1) \leq x \star (y \star 1)$
hence $y * (x \star 1) \leq (x \star 1) * (y \star 1)$
by (*simp add*: *while-one-while*)
hence $y \star ((x \star 1) * 1) \leq (x \star 1) * (y \star 1) \sqcup (y \star bot)$
using *while-simulate-left-plus-1* **by** *blast*
hence $y \star (x \star 1) \leq (x \star (y \star 1)) \sqcup (y \star bot)$
by (*simp add*: *while-one-while*)
also have ... $= x \star (y \star 1)$
by (*metis sup-commute le-iff-sup order-trans while-increasing*
while-right-isotone bot-least)
finally show $y \star (x \star 1) \leq x \star (y \star 1)$
.
next
assume $y \star (x \star 1) \leq x \star (y \star 1)$
thus $y * (x \star 1) \leq x \star (y \star 1)$
using *order-trans while-mult-increasing* **by** *blast*
qed

lemma *while-simulate-1*:
$y * x \leq x * y \Longrightarrow y \star (x \star 1) \leq x \star (y \star 1)$
by (*metis order-trans while-mult-increasing while-right-isotone while-simulate*
while-simulate-2)

lemma *while-simulate-3*:
$y * (x \star 1) \leq x \star 1 \Longrightarrow y \star (x \star 1) \leq x \star (y \star 1)$
by (*meson order.trans while-increasing while-right-isotone while-simulate-2*)

Theorem 9.28

lemma *while-extra-while*:
$(y * (x \star 1)) \star z = (y * (y \star (x \star 1))) \star z$
proof −
have $y * (y \star (x \star 1)) \le y * (x \star 1) * (y * (x \star 1) \star 1)$
using *while-back-loop-prefixpoint while-left-isotone while-mult-star-exchange* **by** *auto*
hence *1*: $(y * (y \star (x \star 1))) \star z \le (y * (x \star 1)) \star z$
by (*metis while-simulate-right-plus-1 mult-left-one*)
have $(y * (x \star 1)) \star z \le (y * (y \star (x \star 1))) \star z$
by (*simp add*: *while-increasing while-left-isotone while-mult-star-exchange*)
thus *?thesis*
using *1 order.antisym* **by** *auto*
qed

Theorem 11.6

lemma *while-separate-4*:
assumes $y * x \le x * (x \star (1 \sqcup y))$
shows $(x \sqcup y) \star z = x \star (y \star z)$
proof −
have $(1 \sqcup y) * x \le x * (x \star (1 \sqcup y))$
by (*smt assms sup-assoc le-supI mult-left-one mult-left-sub-dist-sup-left mult-right-dist-sup mult-1-right while-left-unfold*)
hence *1*: $(1 \sqcup y) * (x \star 1) \le x \star (1 \sqcup y)$
by (*metis mult-1-right while-simulate-right-plus-1*)
have $y * x * (x \star 1) \le x * (x \star ((1 \sqcup y) * (x \star 1)))$
by (*smt assms le-iff-sup mult-assoc mult-right-dist-sup while-associative-1 while-increasing*)
also have ... $\le x * (x \star (1 \sqcup y))$
using *1 mult-right-isotone while-mult-transitive* **by** *blast*
also have ... $\le x * (x \star 1) * (y \star 1)$
by (*simp add*: *mult-right-isotone mult-assoc while-increasing while-one-increasing while-one-while while-right-isotone*)
finally have $y \star (x * (x \star 1)) \le x * (x \star 1) * (y \star 1) \sqcup (y \star bot)$
by (*metis mult-assoc mult-1-right while-simulate-left-plus-1*)
hence $(y \star 1) * (y \star x) \le x * (x \star y \star 1) \sqcup (y \star bot)$
by (*smt le-iff-sup mult-assoc mult-1-right order-refl order-trans while-absorb-2 while-left-dist-sup while-mult-star-exchange while-one-mult-below while-one-while while-plus-one*)
hence $(y \star 1) * ((y \star x) \star (y \star z)) \le x \star ((y \star 1) * (y \star z) \sqcup (y \star bot) * ((y \star x) \star (y \star z)))$
by (*simp add*: *while-simulate-right-plus*)
also have ... $\le x \star ((y \star z) \sqcup (y \star bot))$
by (*metis sup-mono mult-left-zero order-refl while-absorb-2 while-one-mult-below while-right-isotone while-sub-associative*)
also have ... $= x \star y \star z$
using *sup.absorb-iff1 while-right-isotone* **by** *auto*
finally show *?thesis*
by (*smt sup-commute le-iff-sup mult-left-one mult-right-dist-sup while-plus-one while-sub-associative while-sumstar*)

qed

lemma *while-separate-5*:
$y * x \leq x * (x \star (x \sqcup y)) \Longrightarrow (x \sqcup y) \star z = x \star (y \star z)$
using *order-lesseq-imp while-separate-4 while-sup-sub-sup-one-mult* **by** *blast*

lemma *while-separate-6*:
$y * x \leq x * (x \sqcup y) \Longrightarrow (x \sqcup y) \star z = x \star (y \star z)$
by (*smt order-trans while-increasing while-mult-star-exchange while-separate-5*)

Theorem 11.4

lemma *while-separate-1*:
$y * x \leq x * y \Longrightarrow (x \sqcup y) \star z = x \star (y \star z)$
using *mult-left-sub-dist-sup-right order-lesseq-imp while-separate-6* **by** *blast*

Theorem 11.2

lemma *while-separate-mult-1*:
$y * x \leq x * y \Longrightarrow (x * y) \star z \leq x \star (y \star z)$
by (*metis while-mult-sub-sup while-separate-1*)

Theorem 11.5

lemma *separation*:
assumes $y * x \leq x * (y \star 1)$
shows $(x \sqcup y) \star z = x \star (y \star z)$
proof −
have $y \star x \leq x * (y \star 1) \sqcup (y \star bot)$
by (*metis assms mult-1-right while-simulate-left-plus-1*)
also have ... $\leq x * (x \star y \star 1) \sqcup (y \star bot)$
using *sup-left-isotone while-increasing while-mult-star-exchange* **by** *force*
finally have $(y \star 1) * (y \star x) \leq x * (x \star y \star 1) \sqcup (y \star bot)$
using *order.trans while-one-mult-while-below* **by** *blast*
hence $(y \star 1) * ((y \star x) \star (y \star z)) \leq x \star ((y \star 1) * (y \star z) \sqcup (y \star bot) * ((y \star x) \star (y \star z)))$
by (*simp add*: *while-simulate-right-plus*)
also have ... $\leq x \star ((y \star z) \sqcup (y \star bot))$
by (*metis sup-mono mult-left-zero order-refl while-absorb-2 while-one-mult-below while-right-isotone while-sub-associative*)
also have ... $= x \star y \star z$
using *sup.absorb-iff1 while-right-isotone* **by** *auto*
finally show *?thesis*
by (*smt sup-commute le-iff-sup mult-left-one mult-right-dist-sup while-plus-one while-sub-associative while-sumstar*)
qed

Theorem 11.5

lemma *while-separate-left*:
$y * x \leq x * (y \star 1) \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$
by (*metis sup-commute separation while-sub-dist-3*)

Theorem 11.6

lemma *while-simulate-4*:
$y * x \leq x * (x \star (1 \sqcup y)) \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$
by (*metis sup-commute while-separate-4 while-sub-dist-3*)

lemma *while-simulate-5*:
$y * x \leq x * (x \star (x \sqcup y)) \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$
by (*smt order-trans while-sup-sub-sup-one-mult while-simulate-4*)

lemma *while-simulate-6*:
$y * x \leq x * (x \sqcup y) \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$
by (*smt order-trans while-increasing while-mult-star-exchange while-simulate-5*)

Theorem 11.3

lemma *while-simulate-7*:
$y * x \leq x * y \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$
using *mult-left-sub-dist-sup-right order-lesseq-imp while-simulate-6* **by** *blast*

lemma *while-while-mult-1*:
$x \star (1 \star y) = 1 \star (x \star y)$
by (*metis sup-commute mult-left-one mult-1-right order-refl while-separate-1*)

Theorem 9.15

lemma *while-while-mult-2*:
$x \star (1 \star y) = (x \star 1) \star y$
by (*simp add: while-while-mult while-while-mult-1*)

Theorem 11.8

lemma *while-import*:
assumes $p \leq p * p \wedge p \leq 1 \wedge p * x \leq x * p$
shows $p * (x \star y) = p * ((p * x) \star y)$
proof –
have $p * (x \star y) \leq (p * x) \star (p * y)$
using *assms test-preserves-equation while-simulate* **by** *auto*
also have ... $\leq (p * x) \star y$
by (*metis assms le-iff-sup mult-left-one mult-right-dist-sup while-right-isotone*)
finally have *2*: $p * (x \star y) \leq p * ((p * x) \star y)$
by (*smt assms sup-commute le-iff-sup mult-assoc mult-left-dist-sup mult-1-right*)
have $p * ((p * x) \star y) \leq p * (x \star y)$
by (*metis assms mult-left-isotone mult-left-one mult-right-isotone while-left-isotone*)
thus *?thesis*
using *2 order.antisym* **by** *auto*
qed

Theorem 11.8

lemma *while-preserve*:
assumes $p \leq p * p$
and $p \leq 1$

and $p * x \leq x * p$
shows $p * (x \star y) = p * (x \star (p * y))$
proof (*rule order.antisym*)
show $p * (x \star y) \leq p * (x \star (p * y))$
by (*metis assms order.antisym coreflexive-transitive mult-right-isotone mult-assoc while-simulate*)
show $p * (x \star (p * y)) \leq p * (x \star y)$
by (*metis assms(2) mult-left-isotone mult-left-one mult-right-isotone while-right-isotone*)
qed

lemma *while-plus-below-while*:
$(x \star 1) * x \leq x \star 1$
by (*simp add: while-mult-upper-bound while-one-increasing*)

Theorem 9.40

lemma *while-01*:
$(w * (x \star 1)) \star (y * z) \leq (x \star w) \star ((x \star y) * z)$
proof –
have $(w * (x \star 1)) \star (y * z) = y * z \sqcup w * (((x \star 1) * w) \star ((x \star 1) * y * z))$
by (*metis mult-assoc while-productstar*)
also have ... $\leq y * z \sqcup w * ((x \star w) \star ((x \star y) * z))$
by (*metis sup-right-isotone mult-left-isotone mult-right-isotone while-isotone while-one-mult-below*)
also have ... $\leq (x \star y) * z \sqcup (x \star w) * ((x \star w) \star ((x \star y) * z))$
by (*meson mult-left-isotone semiring.add-mono while-increasing*)
finally show *?thesis*
using *while-left-unfold* **by** *auto*
qed

Theorem 9.37

lemma *while-while-sub-associative*:
$x \star (y \star z) \leq ((x \star y) \star z) \sqcup (x \star z)$
proof –
have *1*: $x * (x \star 1) \leq (x \star 1) * ((x \star y) \star 1)$
by (*metis le-supE while-back-loop-prefixpoint while-mult-increasing while-mult-transitive while-one-while*)
have $x \star (y \star z) \leq x \star ((x \star 1) * (y \star z))$
by (*metis mult-left-isotone mult-left-one while-increasing while-right-isotone*)
also have ... $\leq (x \star 1) * ((x \star y) \star (y \star z)) \sqcup (x \star bot)$
using *1 while-simulate-left-plus-1* **by** *auto*
also have ... $\leq (x \star 1) * ((x \star y) \star z) \sqcup (x \star z)$
by (*simp add: le-supI1 sup-commute while-absorb-2 while-increasing while-right-isotone*)
also have ... $= (x \star 1) * z \sqcup (x \star 1) * (x \star y) * ((x \star y) \star z) \sqcup (x \star z)$
by (*metis mult-assoc mult-left-dist-sup while-left-unfold*)
also have ... $= (x \star y) * ((x \star y) \star z) \sqcup (x \star z)$
by (*smt sup-assoc sup-commute le-iff-sup mult-left-one mult-right-dist-sup order-refl while-absorb-1 while-plus-one while-sub-associative*)

also have ... $\leq ((x \star y) \star z) \sqcup (x \star z)$
using *sup-left-isotone while-left-plus-below* **by** *auto*
finally show *?thesis*
.
qed

lemma *while-induct*:
$x * z \leq z \wedge y \leq z \wedge x \star 1 \leq z \Longrightarrow x \star y \leq z$
by (*metis le-supI1 sup-commute sup-bot-left le-iff-sup while-right-isotone while-simulate-absorb*)

proposition *while-sumstar-4-below*: $(x \star y) \star ((x \star 1) * z) \leq x \star ((y * (x \star 1)) \star z)$ **oops**
proposition *while-sumstar-2*: $(x \sqcup y) \star z = x \star ((y * (x \star 1)) \star z)$ **oops**
proposition *while-sumstar-3*: $(x \sqcup y) \star z = ((x \star 1) * y) \star (x \star z)$ **oops**
proposition *while-decompose-6*: $x \star ((y * (x \star 1)) \star z) = y \star ((x * (y \star 1)) \star z)$ **oops**
proposition *while-finite-associative*: $x \star bot = bot \Longrightarrow (x \star y) * z = x \star (y * z)$ **oops**
proposition *atomicity-refinement*: $s = s * q \Longrightarrow x = q * x \Longrightarrow q * b = bot \Longrightarrow r * b \leq b * r \Longrightarrow r * l \leq l * r \Longrightarrow x * l \leq l * x \Longrightarrow b * l \leq l * b \Longrightarrow q * l \leq l * q \Longrightarrow r \star q \leq q * (r \star 1) \Longrightarrow q \leq 1 \Longrightarrow s * ((x \sqcup b \sqcup r \sqcup l) \star (q * z)) \leq s * ((x * (b \star q) \sqcup r \sqcup l) \star z)$ **oops**

proposition *while-separate-right-plus*: $y * x \leq x * (x \star (1 \sqcup y)) \sqcup 1 \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$ **oops**
proposition *while-square-1*: $x \star 1 = (x * x) \star (x \sqcup 1)$ **oops**
proposition *while-absorb-below-one*: $y * x \leq x \Longrightarrow y \star x \leq 1 \star x$ **oops**
proposition $y \star (x \star 1) \leq x \star (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**
proposition $y * x \leq (1 \sqcup x) * (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**

end

class *bounded-binary-itering* = *bounded-idempotent-left-zero-semiring* + *binary-itering*
begin

Theorem 9

lemma *while-right-top*:
$x \star top = top$
by (*metis sup-left-top while-left-unfold*)

Theorem 9

lemma *while-left-top*:
$top * (x \star 1) = top$
by (*meson order.antisym le-supE top-greatest while-back-loop-prefixpoint*)

end

class *extended-binary-itering* = *binary-itering* +
assumes *while-denest-0*: $w * (x \star (y * z)) \leq (w * (x \star y)) \star (w * (x \star y) * z)$
begin

Theorem 10.2

lemma *while-denest-1*:
$w * (x \star (y * z)) \leq (w * (x \star y)) \star z$
using *while-denest-0 while-mult-increasing while-mult-transitive* **by** *blast*

lemma *while-mult-sub-while-while*:
$x \star (y * z) \leq (x \star y) \star z$
by (*metis mult-left-one while-denest-1*)

lemma *while-zero-zero*:
$(x \star bot) \star bot = x \star bot$
by (*metis order.antisym mult-left-zero sup-bot-left while-left-unfold while-sub-associative while-mult-sub-while-while*)

Theorem 10.11

lemma *while-mult-zero-zero*:
$(x * (y \star bot)) \star bot = x * (y \star bot)$
apply (*rule order.antisym*)
apply (*metis sup-bot-left while-left-unfold mult-assoc le-supI1 mult-left-zero mult-right-isotone while-left-dist-sup while-sub-associative*)
by (*metis mult-left-zero while-denest-1*)

Theorem 10.3

lemma *while-denest-2*:
$w * ((x \star (y * w)) \star z) = w * (((x \star y) * w) \star z)$
apply (*rule order.antisym*)
apply (*metis mult-assoc while-denest-0 while-simulate-right-plus-1 while-slide*)
by (*simp add*: *mult-isotone while-left-isotone while-sub-associative*)

Theorem 10.12

lemma *while-denest-3*:
$(x \star w) \star (x \star bot) = (x \star w) \star bot$
by (*metis while-absorb-2 while-right-isotone while-zero-zero bot-least*)

Theorem 10.15

lemma *while-02*:
$x \star ((x \star w) \star ((x \star y) * z)) = (x \star w) \star ((x \star y) * z)$
proof −
have $x * ((x \star w) \star ((x \star y) * z)) = x * (x \star y) * z \sqcup x * (x \star w) * ((x \star w) \star ((x \star y) * z))$
by (*metis mult-assoc mult-left-dist-sup while-left-unfold*)
also have ... $\leq (x \star w) \star ((x \star y) * z)$
by (*metis sup-mono mult-right-sub-dist-sup-right while-left-unfold*)
finally have $x \star ((x \star w) \star ((x \star y) * z)) \leq ((x \star w) \star ((x \star y) * z)) \sqcup (x \star bot)$
using *while-simulate-absorb* **by** *auto*

also have ... $= (x \star w) \star ((x \star y) * z)$
by (*metis sup-commute le-iff-sup order-trans while-mult-sub-while-while while-right-isotone bot-least*)
finally show *?thesis*
by (*simp add*: *order.antisym while-increasing*)
qed

lemma *while-sumstar-3-below*:
$(x \star y) \star (x \star z) \leq (x \star y) \star ((x \star 1) * z)$
proof –
have $(x \star y) \star (x \star z) = (x \star z) \sqcup ((x \star y) \star ((x \star y) * (x \star z)))$
using *while-right-unfold* **by** *blast*
also have ... $\leq (x \star z) \sqcup ((x \star y) \star (x \star (y * (x \star z))))$
by (*meson sup-right-isotone while-right-isotone while-sub-associative*)
also have ... $\leq (x \star z) \sqcup ((x \star y) \star (x \star ((x \star y) \star (x \star z))))$
by (*smt sup-right-isotone order-trans while-increasing while-mult-upper-bound while-one-increasing while-right-isotone*)
also have ... $\leq (x \star z) \sqcup ((x \star y) \star (x \star ((x \star y) \star ((x \star 1) * z))))$
by (*metis sup-right-isotone mult-left-isotone mult-left-one order-trans while-increasing while-right-isotone while-sumstar while-transitive*)
also have ... $= (x \star z) \sqcup ((x \star y) \star ((x \star 1) * z))$
by (*simp add*: *while-transitive while-02*)
also have ... $= (x \star y) \star ((x \star 1) * z)$
by (*smt sup-assoc mult-left-one mult-right-dist-sup while-02 while-left-dist-sup while-plus-one*)
finally show *?thesis*
.
qed

lemma *while-sumstar-4-below*:
$(x \star y) \star ((x \star 1) * z) \leq x \star ((y * (x \star 1)) \star z)$
proof –
have $(x \star y) \star ((x \star 1) * z) = (x \star 1) * z \sqcup (x \star y) * ((x \star y) \star ((x \star 1) * z))$
using *while-left-unfold* **by** *auto*
also have ... $\leq (x \star z) \sqcup (x \star (y * ((x \star y) \star ((x \star 1) * z))))$
by (*meson sup-mono while-one-mult-below while-sub-associative*)
also have ... $= (x \star z) \sqcup (x \star (y * (((x \star 1) * y) \star ((x \star 1) * z))))$
by (*metis mult-left-one while-denest-2*)
also have ... $= x \star ((y * (x \star 1)) \star z)$
by (*metis while-left-dist-sup while-productstar*)
finally show *?thesis*
.
qed

Theorem 10.10

lemma *while-sumstar-1*:
$(x \sqcup y) \star z = (x \star y) \star ((x \star 1) * z)$
by (*smt order.eq-iff order-trans while-sup-1-below while-sumstar while-sumstar-3-below while-sumstar-4-below*)

Theorem 10.8

lemma *while-sumstar-2*:
$(x \sqcup y) \star z = x \star ((y * (x \star 1)) \star z)$
using *order.antisym while-sup-1-below while-sumstar-1 while-sumstar-4-below*
by *auto*

Theorem 10.9

lemma *while-sumstar-3*:
$(x \sqcup y) \star z = ((x \star 1) * y) \star (x \star z)$
using *order.antisym while-sumstar while-sumstar-1-below while-sumstar-2-below while-sumstar-2* **by** *force*

Theorem 10.6

lemma *while-decompose-6*:
$x \star ((y * (x \star 1)) \star z) = y \star ((x * (y \star 1)) \star z)$
by (*metis sup-commute while-sumstar-2*)

Theorem 10.4

lemma *while-denest-4*:
$(x \star w) \star (x \star (y * z)) = (x \star w) \star ((x \star y) * z)$
proof –
have $(x \star w) \star (x \star (y * z)) = x \star ((w * (x \star 1)) \star (y * z))$
using *while-sumstar while-sumstar-2* **by** *force*
also have ... $\leq (x \star w) \star ((x \star y) * z)$
by (*metis while-01 while-right-isotone while-02*)
finally show *?thesis*
using *order.antisym while-right-isotone while-sub-associative* **by** *auto*
qed

Theorem 10.13

lemma *while-denest-5*:
$w * ((x \star (y * w)) \star (x \star (y * z))) = w * (((x \star y) * w) \star ((x \star y) * z))$
by (*simp add*: *while-denest-2 while-denest-4*)

Theorem 10.5

lemma *while-denest-6*:
$(w * (x \star y)) \star z = z \sqcup w * ((x \sqcup y * w) \star (y * z))$
by (*metis while-denest-5 while-productstar while-sumstar*)

Theorem 10.1

lemma *while-sum-below-one*:
$y * ((x \sqcup y) \star z) \leq (y * (x \star 1)) \star z$
by (*simp add*: *while-denest-6*)

Theorem 10.14

lemma *while-separate-unfold*:
$(y * (x \star 1)) \star z = (y \star z) \sqcup (y \star (y * x * (x \star ((y * (x \star 1)) \star z))))$
proof –

have $y \star (y * x * (x \star ((y * (x \star 1)) \star z))) \leq y \star (y * ((x \sqcup y) \star z))$
using *mult-right-isotone while-left-plus-below while-right-isotone mult-assoc while-sumstar-2* **by** *auto*
also have ... $\leq (y * (x \star 1)) \star z$
by (*metis sup-commute sup-ge1 while-absorb-1 while-mult-star-exchange while-sum-below-one*)
finally have $(y \star z) \sqcup (y \star (y * x * (x \star ((y * (x \star 1)) \star z)))) \leq (y * (x \star 1)) \star z$
using *sup.bounded-iff while-back-loop-prefixpoint while-left-isotone* **by** *auto*
thus *?thesis*
by (*simp add*: *order.antisym while-separate-unfold-below*)
qed

Theorem 10.7

lemma *while-finite-associative*:
$x \star bot = bot \Longrightarrow (x \star y) * z = x \star (y * z)$
by (*metis while-denest-4 while-zero*)

Theorem 12

lemma *atomicity-refinement*:
assumes $s = s * q$
and $x = q * x$
and $q * b = bot$
and $r * b \leq b * r$
and $r * l \leq l * r$
and $x * l \leq l * x$
and $b * l \leq l * b$
and $q * l \leq l * q$
and $r \star q \leq q * (r \star 1) \wedge q \leq 1$
shows $s * ((x \sqcup b \sqcup r \sqcup l) \star (q * z)) \leq s * ((x * (b \star q) \sqcup r \sqcup l) \star z)$
proof −
have *1*: $(x \sqcup b \sqcup r) * l \leq l * (x \sqcup b \sqcup r)$
by (*smt assms*(*5*−*7*) *mult-left-dist-sup semiring.add-mono semiring.distrib-right*)
have $q * ((x * (b \star r \star 1) * q) \star z) \leq (x * (b \star r \star 1) * q) \star z$
using *assms*(*9*) *order-lesseq-imp while-increasing while-mult-upper-bound* **by** *blast*
also have ... $\leq (x * (b \star ((r \star 1) * q))) \star z$
by (*simp add*: *mult-right-isotone while-left-isotone while-sub-associative mult-assoc*)
also have ... $\leq (x * (b \star r \star q)) \star z$
by (*simp add*: *mult-right-isotone while-left-isotone while-one-mult-below while-right-isotone*)
also have ... $\leq (x * (b \star (q * (r \star 1)))) \star z$
by (*simp add*: *assms*(*9*) *mult-right-isotone while-left-isotone while-right-isotone*)
finally have *2*: $q * ((x * (b \star r \star 1) * q) \star z) \leq (x * (b \star q) * (r \star 1)) \star z$
using *while-associative-while-1 mult-assoc* **by** *auto*
have $s * ((x \sqcup b \sqcup r \sqcup l) \star (q * z)) = s * (l \star (x \sqcup b \sqcup r) \star (q * z))$

using *1 sup-commute while-separate-1* **by** *fastforce*
also have ... $= s * q * (l \star b \star r \star (q * x * (b \star r \star 1)) \star (q * z))$
by (*smt assms*(*1*,*2*,*4*) *sup-assoc sup-commute while-sumstar-2 while-separate-1*)
also have ... $= s * q * (l \star b \star r \star (q * ((x * (b \star r \star 1) * q) \star z)))$
by (*simp add*: *while-slide mult-assoc*)
also have ... $\leq s * q * (l \star b \star r \star (x * (b \star q) * (r \star 1)) \star z)$
using *2* **by** (*meson mult-right-isotone while-right-isotone*)
also have ... $\leq s * (l \star q * (b \star r \star (x * (b \star q) * (r \star 1)) \star z))$
by (*simp add*: *assms*(*8*) *mult-right-isotone while-simulate mult-assoc*)
also have ... $= s * (l \star q * (r \star (x * (b \star q) * (r \star 1)) \star z))$
using *assms*(*3*) *while-elimination* **by** *auto*
also have ... $\leq s * (l \star r \star (x * (b \star q) * (r \star 1)) \star z)$
by (*meson assms*(*9*) *order.trans mult-right-isotone order.refl while-increasing while-mult-upper-bound while-right-isotone*)
also have ... $= s * (l \star (r \sqcup x * (b \star q)) \star z)$
by (*simp add*: *while-sumstar-2*)
also have ... $\leq s * ((x * (b \star q) \sqcup r \sqcup l) \star z)$
using *mult-right-isotone sup-commute while-sub-dist-3* **by** *auto*
finally show *?thesis*
.
qed

end

class *bounded-extended-binary-itering* = *bounded-binary-itering* + *extended-binary-itering*

end

8 Strict Binary Iterings

theory *Binary-Iterings-Strict*

imports *Stone-Kleene-Relation-Algebras.Iterings Binary-Iterings*

begin

class *strict-itering* = *itering* + *while* +
assumes *while-def*: $x \star y = x^\circ * y$
begin

Theorem 8.1

subclass *extended-binary-itering*
apply *unfold-locales*
apply (*metis circ-loop-fixpoint circ-slide-1 sup-commute while-def mult-assoc*)
apply (*metis circ-sup mult-assoc while-def*)
apply (*simp add*: *mult-left-dist-sup while-def*)
apply (*simp add*: *while-def mult-assoc*)

apply (*metis circ-simulate-left-plus mult-assoc mult-left-isotone mult-right-dist-sup mult-1-right while-def*)
apply (*metis circ-simulate-right-plus mult-assoc mult-left-isotone mult-right-dist-sup while-def*)
by (*metis circ-loop-fixpoint mult-right-sub-dist-sup-right while-def mult-assoc*)

Theorem 13.1

lemma *while-associative*:
$(x \star y) * z = x \star (y * z)$
by (*simp add*: *while-def mult-assoc*)

Theorem 13.3

lemma *while-one-mult*:
$(x \star 1) * x = x \star x$
by (*simp add*: *while-def*)

lemma *while-back-loop-is-fixpoint*:
is-fixpoint $(\lambda x . x * y \sqcup z)$ $(z * (y \star 1))$
by (*simp add*: *circ-back-loop-is-fixpoint while-def*)

Theorem 13.4

lemma *while-sumstar-var*:
$(x \sqcup y) \star z = ((x \star 1) * y) \star ((x \star 1) * z)$
by (*simp add*: *while-sumstar-3 while-associative*)

Theorem 13.2

lemma *while-mult-1-assoc*:
$(x \star 1) * y = x \star y$
by (*simp add*: *while-def*)

proposition $y \star (x \star 1) \leq x \star (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**
proposition $y * x \leq (1 \sqcup x) * (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**
proposition *while-square-1*: $x \star 1 = (x * x) \star (x \sqcup 1)$ **oops**
proposition *while-absorb-below-one*: $y * x \leq x \Longrightarrow y \star x \leq 1 \star x$ **oops**

end

class *bounded-strict-itering* = *bounded-itering* + *strict-itering*
begin

subclass *bounded-extended-binary-itering* **..**

Theorem 13.6

lemma *while-top-2*:
$top \star z = top * z$
by (*simp add*: *circ-top while-def*)

Theorem 13.5

lemma *while-mult-top-2*:

$(x * top) \star z = z \sqcup x * top * z$
by (*metis circ-left-top mult-assoc while-def while-left-unfold*)

Theorem 13 counterexamples

proposition *while-one-top*: $1 \star x = top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-top*: $top \star x = top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-sub-mult-one*: $x * (1 \star y) \leq 1 \star x$ **oops**
proposition *while-unfold-below-1*: $x = y * x \Longrightarrow x \leq y \star 1$ **oops**
proposition *while-unfold-below*: $x = z \sqcup y * x \Longrightarrow x \leq y \star z$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-unfold-below*: $x \leq z \sqcup y * x \Longrightarrow x \leq y \star z$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-mult-top*: $(x * top) \star z = z \sqcup x * top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *tarski-mult-top-idempotent*: $x * top = x * top * x * top$ **oops**

proposition *while-loop-is-greatest-postfixpoint*: *is-greatest-postfixpoint* $(\lambda x . y * x \sqcup z)$ $(y \star z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-loop-is-greatest-fixpoint*: *is-greatest-fixpoint* $(\lambda x . y * x \sqcup z)$ $(y \star z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-sub-while-zero*: $x \star z \leq (x \star y) \star z$ **oops**
proposition *while-while-sub-associative*: $x \star (y \star z) \leq (x \star y) \star z$ **oops**
proposition *tarski*: $x \leq x * top * x * top$ **oops**
proposition *tarski-top-omega-below*: $x * top \leq (x * top) \star bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *tarski-top-omega*: $x * top = (x * top) \star bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *tarski-below-top-omega*: $x \leq (x * top) \star bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *tarski*: $x = bot \vee top * x * top = top$ **oops**
proposition $1 = (x * bot) \star 1$ **oops**
proposition $1 \sqcup x * bot = x \star 1$ **oops**
proposition $x = x * (x \star 1)$ **oops**
proposition $x * (x \star 1) = x \star 1$ **oops**
proposition $x \star 1 = x \star (1 \star 1)$ **oops**
proposition $(x \sqcup y) \star 1 = (x \star (y \star 1)) \star 1$ **oops**
proposition $z \sqcup y * x = x \Longrightarrow y \star z \leq x$ **oops**
proposition $y * x = x \Longrightarrow y \star x \leq x$ **oops**
proposition $z \sqcup x * y = x \Longrightarrow z * (y \star 1) \leq x$ **oops**
proposition $x * y = x \Longrightarrow x * (y \star 1) \leq x$ **oops**
proposition $x * z = z * y \Longrightarrow x \star z \leq z * (y \star 1)$ **oops**

end

class *binary-itering-unary* = *extended-binary-itering* + *circ* +
assumes *circ-def*: $x^\circ = x \star 1$
begin

Theorem 50.7

subclass *left-conway-semiring*
apply *unfold-locales*
using *circ-def while-left-unfold* **apply** *simp*
apply (*metis circ-def mult-1-right while-one-mult-below while-slide*)
using *circ-def while-one-while while-sumstar-2* **by** *auto*

end

class *strict-binary-itering* = *binary-itering* + *circ* +
assumes *while-associative*: $(x \star y) * z = x \star (y * z)$
assumes *circ-def*: $x^\circ = x \star 1$
begin

Theorem 2.8

subclass *itering*
apply *unfold-locales*
apply (*simp add*: *circ-def while-associative while-sumstar*)
apply (*metis circ-def mult-1-right while-associative while-productstar while-slide*)
apply (*metis circ-def mult-1-right while-associative mult-1-left while-slide while-simulate-right-plus*)
by (*metis circ-def mult-1-right while-associative mult-1-left while-simulate-left-plus mult-right-dist-sup*)

Theorem 8.5

subclass *extended-binary-itering*
apply *unfold-locales*
by (*simp add*: *while-associative while-increasing mult-assoc*)

end

end

9 Nonstrict Binary Iterings

theory *Binary-Iterings-Nonstrict*

imports *Omega-Algebras Binary-Iterings*

begin

class *nonstrict-itering* = *bounded-left-zero-omega-algebra* + *while* +
assumes *while-def*: $x \star y = x^\omega \sqcup x^\star * y$
begin

Theorem 8.2

subclass *bounded-binary-itering*
proof (*unfold-locales*)
fix $x\ y\ z$
show $(x * y) \star z = z \sqcup x * ((y * x) \star (y * z))$

by (*metis sup-commute mult-assoc mult-left-dist-sup omega-loop-fixpoint omega-slide star.circ-slide while-def*)
next
fix x y z
show $(x \sqcup y) \star z = (x \star y) \star (x \star z)$
proof −
have *1*: $(x \sqcup y) \star z = (x^\star * y)^\omega \sqcup (x^\star * y)^\star * (x^\omega \sqcup x^\star * z)$
using *mult-left-dist-sup omega-decompose star.circ-sup-9 sup-assoc while-def mult-assoc* **by** *auto*
hence *2*: $(x \sqcup y) \star z \leq (x \star y) \star (x \star z)$
by (*smt sup-mono sup-ge2 le-iff-sup mult-left-isotone omega-sub-dist star.circ-sub-dist while-def*)
let *?rhs* $= x^\star * y * ((x^\omega \sqcup x^\star * y)^\omega \sqcup (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z)) \sqcup (x^\omega \sqcup x^\star * z)$
have $x^\omega * (x^\omega \sqcup x^\star * y)^\omega \leq x^\omega$
by (*simp add*: *omega-sub-vector*)
hence $x^\omega * (x^\omega \sqcup x^\star * y)^\omega \sqcup x^\star * y * (x^\omega \sqcup x^\star * y)^\omega \leq$ *?rhs*
by (*smt sup-commute sup-mono sup-ge1 mult-left-dist-sup order-trans*)
hence *3*: $(x^\omega \sqcup x^\star * y)^\omega \leq$ *?rhs*
by (*metis mult-right-dist-sup omega-unfold*)
have $x^\omega * (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \leq x^\omega$
by (*simp add*: *omega-mult-star-2 omega-sub-vector*)
hence $x^\omega * (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \sqcup x^\star * y * (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \leq$ *?rhs*
by (*smt sup-commute sup-mono sup-ge2 mult-assoc mult-left-dist-sup order-trans*)
hence $(x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \leq$ *?rhs*
by (*smt sup-assoc sup-ge2 le-iff-sup mult-assoc mult-right-dist-sup star.circ-loop-fixpoint*)
hence $(x^\omega \sqcup x^\star * y)^\omega \sqcup (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \leq$ *?rhs*
using *3* **by** *simp*
hence $(x^\omega \sqcup x^\star * y)^\omega \sqcup (x^\omega \sqcup x^\star * y)^\star * (x^\omega \sqcup x^\star * z) \leq (x^\star * y)^\omega \sqcup (x^\star * y)^\star * (x^\omega \sqcup x^\star * z)$
by (*metis sup-commute omega-induct*)
thus *?thesis*
using *1 2 order.antisym while-def* **by** *force*
qed
next
fix x y z
show $x \star (y \sqcup z) = (x \star y) \sqcup (x \star z)$
using *mult-left-dist-sup sup-assoc sup-commute while-def* **by** *auto*
next
fix x y z
show $(x \star y) * z \leq x \star (y * z)$
using *mult-semi-associative omega-sub-vector semiring.add-mono semiring.distrib-right while-def* **by** *fastforce*
next
fix v w x y z
show $x * z \leq z * (y \star 1) \sqcup w \longrightarrow x \star (z * v) \leq z * (y \star v) \sqcup (x \star (w * (y \star$

$v)))$
proof
assume $x * z \le z * (y \star 1) \sqcup w$
hence *1*: $x * z \le z * y^\omega \sqcup z * y^\star \sqcup w$
by (*metis mult-left-dist-sup mult-1-right while-def*)
let *?rhs* $= z * (y^\omega \sqcup y^\star * v) \sqcup x^\omega \sqcup x^\star * w * (y^\omega \sqcup y^\star * v)$
have *2*: $z * v \le$ *?rhs*
by (*metis le-supI1 mult-left-sub-dist-sup-right omega-loop-fixpoint*)
have $x * z * (y^\omega \sqcup y^\star * v) \le$ *?rhs*
proof −
have $x * z * (y^\omega \sqcup y^\star * v) \le (z * y^\omega \sqcup z * y^\star \sqcup w) * (y^\omega \sqcup y^\star * v)$
using *1 mult-left-isotone* **by** *auto*
also have ... $= z * (y^\omega * (y^\omega \sqcup y^\star * v) \sqcup y^\star * (y^\omega \sqcup y^\star * v)) \sqcup w * (y^\omega \sqcup$
$y^\star * v)$
by (*smt mult-assoc mult-left-dist-sup mult-right-dist-sup*)
also have ... $= z * (y^\omega * (y^\omega \sqcup y^\star * v) \sqcup y^\omega \sqcup y^\star * v) \sqcup w * (y^\omega \sqcup y^\star * v)$
by (*smt sup-assoc mult-assoc mult-left-dist-sup star.circ-transitive-equal*
star-mult-omega)
also have ... $\le z * (y^\omega \sqcup y^\star * v) \sqcup x^\star * w * (y^\omega \sqcup y^\star * v)$
by (*smt sup-commute sup-mono sup-left-top mult-left-dist-sup mult-left-one*
mult-right-dist-sup mult-right-sub-dist-sup-left omega-vector order-refl
star.circ-plus-one)
finally show *?thesis*
by (*smt sup-assoc sup-commute le-iff-sup*)
qed
hence $x *$ *?rhs* $\le$ *?rhs*
by (*smt sup-assoc sup-commute sup-ge1 le-iff-sup mult-assoc*
mult-left-dist-sup mult-right-dist-sup omega-unfold star.circ-increasing
star.circ-transitive-equal)
hence $z * v \sqcup x *$ *?rhs* $\le$ *?rhs*
using *2 le-supI* **by** *blast*
hence $x^\star * z * v \le$ *?rhs*
by (*simp add*: *star-left-induct mult-assoc*)
hence $x^\omega \sqcup x^\star * z * v \le$ *?rhs*
by (*meson order-trans sup-ge1 sup-ge2 sup-least*)
thus $x \star (z * v) \le z * (y \star v) \sqcup (x \star (w * (y \star v)))$
by (*simp add*: *sup-assoc while-def mult-assoc*)
qed
next
fix $v\ w\ x\ y\ z$
show $z * x \le y * (y \star z) \sqcup w \longrightarrow z * (x \star v) \le y \star (z * v \sqcup w * (x \star v))$
proof
assume $z * x \le y * (y \star z) \sqcup w$
hence $z * x \le y * (y^\omega \sqcup y^\star * z) \sqcup w$
by (*simp add*: *while-def*)
hence *1*: $z * x \le y^\omega \sqcup y * y^\star * z \sqcup w$
using *mult-left-dist-sup omega-unfold mult-assoc* **by** *auto*
let *?rhs* $= y^\omega \sqcup y^\star * z * v \sqcup y^\star * w * (x^\omega \sqcup x^\star * v)$
have *2*: $z * x^\omega \le$ *?rhs*

proof −
have $z * x^\omega \leq y * y^\star * z * x^\omega \sqcup y^\omega * x^\omega \sqcup w * x^\omega$
using *1* **by** (*smt sup-commute le-iff-sup mult-assoc mult-right-dist-sup omega-unfold*)
also have ... $\leq y * y^\star * z * x^\omega \sqcup y^\omega \sqcup w * x^\omega$
using *omega-sub-vector semiring.add-mono* **by** *blast*
also have ... $= y * y^\star * (z * x^\omega) \sqcup (y^\omega \sqcup w * x^\omega)$
by (*simp add*: *sup-assoc mult-assoc*)
finally have $z * x^\omega \leq (y * y^\star)^\omega \sqcup (y * y^\star)^\star * (y^\omega \sqcup w * x^\omega)$
by (*simp add*: *omega-induct sup-commute*)
also have ... $= y^\omega \sqcup y^\star * w * x^\omega$
by (*simp add*: *left-plus-omega semiring.distrib-left star.left-plus-circ star-mult-omega mult-assoc*)
also have ... $\leq$ *?rhs*
using *mult-left-sub-dist-sup-left sup.mono sup-ge1* **by** *blast*
finally show *?thesis*
.
qed
let *?rhs2* $= y^\omega \sqcup y^\star * z \sqcup y^\star * w * (x^\omega \sqcup x^\star)$
have *?rhs2* $* x \leq$ *?rhs2*
proof −
have *3*: $y^\omega * x \leq$ *?rhs2*
by (*simp add*: *le-supI1 omega-sub-vector*)
have $y^\star * z * x \leq y^\star * (y^\omega \sqcup y * y^\star * z \sqcup w)$
using *1 mult-right-isotone mult-assoc* **by** *auto*
also have ... $= y^\omega \sqcup y^\star * y * y^\star * z \sqcup y^\star * w$
by (*simp add*: *semiring.distrib-left star-mult-omega mult-assoc*)
also have ... $= y^\omega \sqcup y * y^\star * z \sqcup y^\star * w$
by (*simp add*: *star.circ-plus-same star.circ-transitive-equal mult-assoc*)
also have ... $\leq y^\omega \sqcup y^\star * z \sqcup y^\star * w$
by (*metis sup-left-isotone sup-right-isotone mult-left-isotone star.left-plus-below-circ*)
also have ... $\leq y^\omega \sqcup y^\star * z \sqcup y^\star * w * x^\star$
using *semiring.add-left-mono star.circ-back-loop-prefixpoint* **by** *auto*
finally have *4*: $y^\star * z * x \leq$ *?rhs2*
using *mult-left-sub-dist-sup-right order-lesseq-imp semiring.add-left-mono* **by** *blast*
have $(x^\omega \sqcup x^\star) * x \leq x^\omega \sqcup x^\star$
using *omega-sub-vector semiring.distrib-right star.left-plus-below-circ star-plus sup-mono* **by** *fastforce*
hence $y^\star * w * (x^\omega \sqcup x^\star) * x \leq$ *?rhs2*
by (*simp add*: *le-supI2 mult-right-isotone mult-assoc*)
thus *?thesis*
using *3 4 mult-right-dist-sup* **by** *force*
qed
hence $z \sqcup$ *?rhs2* $* x \leq$ *?rhs2*
by (*metis omega-loop-fixpoint sup.boundedE sup-ge1 sup-least*)
hence *5*: $z * x^\star \leq$ *?rhs2*
using *star-right-induct* **by** *blast*

have $z * x^\star * v \leq ?rhs$
proof $-$
have $z * x^\star * v \leq ?rhs2 * v$
using *5 mult-left-isotone* **by** *auto*
also have ... $= y^\omega * v \sqcup y^\star * z * v \sqcup y^\star * w * (x^\omega * v \sqcup x^\star * v)$
using *mult-right-dist-sup mult-assoc* **by** *auto*
also have ... $\leq y^\omega \sqcup y^\star * z * v \sqcup y^\star * w * (x^\omega * v \sqcup x^\star * v)$
using *omega-sub-vector semiring.add-right-mono* **by** *blast*
also have ... $\leq ?rhs$
using *mult-right-isotone omega-sub-vector semiring.add-left-mono semiring.add-right-mono* **by** *auto*
finally show *?thesis*
.
qed
hence $z * (x^\omega \sqcup x^\star * v) \leq ?rhs$
using *2 semiring.distrib-left mult-assoc* **by** *force*
thus $z * (x \star v) \leq y \star (z * v \sqcup w * (x \star v))$
by (*simp add*: *semiring.distrib-left sup-assoc while-def mult-assoc*)
qed
qed

Theorem 13.8

lemma *while-top*:
$top \star x = top$
by (*metis sup-left-top star.circ-top star-omega-top while-def*)

Theorem 13.7

lemma *while-one-top*:
$1 \star x = top$
by (*simp add*: *omega-one while-def*)

lemma *while-finite-associative*:
$x^\omega = bot \Longrightarrow (x \star y) * z = x \star (y * z)$
by (*simp add*: *while-def mult-assoc*)

lemma *star-below-while*:
$x^\star * y \leq x \star y$
by (*simp add*: *while-def*)

Theorem 13.9

lemma *while-sub-mult-one*:
$x * (1 \star y) \leq 1 \star x$
by (*simp add*: *omega-one while-def*)

lemma *while-while-one*:
$y \star (x \star 1) = y^\omega \sqcup y^\star * x^\omega \sqcup y^\star * x^\star$
using *mult-left-dist-sup sup-assoc while-def* **by** *auto*

lemma *while-simulate-4-plus*:

assumes $y * x \leq x * (x \star (1 \sqcup y))$
shows $y * x * x^\star \leq x * (x \star (1 \sqcup y))$
proof –
have *1*: $x * (x \star (1 \sqcup y)) = x^\omega \sqcup x * x^\star \sqcup x * x^\star * y$
using *mult-left-dist-sup omega-unfold sup-assoc while-def mult-assoc* **by** *force*
hence $y * x * x^\star \leq (x^\omega \sqcup x * x^\star \sqcup x * x^\star * y) * x^\star$
using *assms mult-left-isotone* **by** *auto*
also have ... $= x^\omega * x^\star \sqcup x * x^\star * x^\star \sqcup x * x^\star * y * x^\star$
using *mult-right-dist-sup* **by** *force*
also have ... $= x * x^\star * (y * x * x^\star) \sqcup x^\omega \sqcup x * x^\star \sqcup x * x^\star * y$
by (*smt sup-assoc sup-commute mult-assoc omega-mult-star-2 star.circ-back-loop-fixpoint star.circ-plus-same star.circ-transitive-equal*)
finally have $y * x * x^\star \leq x * x^\star * (y * x * x^\star) \sqcup (x^\omega \sqcup x * x^\star \sqcup x * x^\star * y)$
using *sup-assoc* **by** *force*
hence $y * x * x^\star \leq (x * x^\star)^\omega \sqcup (x * x^\star)^\star * (x^\omega \sqcup x * x^\star \sqcup x * x^\star * y)$
by (*simp add*: *omega-induct sup-monoid.add-commute*)
also have ... $= x^\omega \sqcup x^\star * (x^\omega \sqcup x * x^\star \sqcup x * x^\star * y)$
by (*simp add*: *left-plus-omega star.left-plus-circ*)
finally show *?thesis*
using *1* **by** (*metis while-def while-mult-star-exchange while-transitive*)
qed

lemma *while-simulate-4-omega*:
assumes $y * x \leq x * (x \star (1 \sqcup y))$
shows $y * x^\omega \leq x^\omega$
proof –
have $x * (x \star (1 \sqcup y)) = x^\omega \sqcup x * x^\star \sqcup x * x^\star * y$
using *mult-1-right mult-left-dist-sup omega-unfold sup-assoc while-def mult-assoc* **by** *auto*
hence $y * x^\omega \leq (x^\omega \sqcup x * x^\star \sqcup x * x^\star * y) * x^\omega$
by (*smt assms le-iff-sup mult-assoc mult-right-dist-sup omega-unfold*)
also have ... $= x^\omega * x^\omega \sqcup x * x^\star * x^\omega \sqcup x * x^\star * y * x^\omega$
using *semiring.distrib-right* **by** *auto*
also have ... $= x * x^\star * (y * x^\omega) \sqcup x^\omega$
by (*metis sup-commute le-iff-sup mult-assoc omega-sub-vector omega-unfold star-mult-omega*)
finally have $y * x^\omega \leq x * x^\star * (y * x^\omega) \sqcup x^\omega$
.
hence $y * x^\omega \leq (x * x^\star)^\omega \sqcup (x * x^\star)^\star * x^\omega$
by (*simp add*: *omega-induct sup-commute*)
thus *?thesis*
by (*metis sup-idem left-plus-omega star-mult-omega*)
qed

Theorem 13.11

lemma *while-unfold-below*:
$x = z \sqcup y * x \Longrightarrow x \leq y \star z$
by (*simp add*: *omega-induct while-def*)

Theorem 13.12

lemma *while-unfold-below-sub*:
$x \leq z \sqcup y * x \Longrightarrow x \leq y \star z$
by (*simp add*: *omega-induct while-def*)

Theorem 13.10

lemma *while-unfold-below-1*:
$x = y * x \Longrightarrow x \leq y \star 1$
by (*simp add*: *while-unfold-below-sub*)

lemma *while-square-1*:
$x \star 1 = (x * x) \star (x \sqcup 1)$
by (*metis mult-1-right omega-square star-square-2 while-def*)

lemma *while-absorb-below-one*:
$y * x \leq x \Longrightarrow y \star x \leq 1 \star x$
by (*simp add*: *while-unfold-below-sub*)

lemma *while-loop-is-greatest-postfixpoint*:
is-greatest-postfixpoint $(\lambda x . y * x \sqcup z)$ $(y \star z)$
proof −
have $(y \star z) \leq (\lambda x . y * x \sqcup z)$ $(y \star z)$
using *sup-commute while-left-unfold* **by** *force*
thus *?thesis*
by (*simp add*: *is-greatest-postfixpoint-def sup-commute while-unfold-below-sub*)
qed

lemma *while-loop-is-greatest-fixpoint*:
is-greatest-fixpoint $(\lambda x . y * x \sqcup z)$ $(y \star z)$
by (*simp add*: *omega-loop-is-greatest-fixpoint while-def*)

proposition *while-sumstar-4-below*: $(x \star y) \star ((x \star 1) * z) \leq x \star ((y * (x \star 1)) \star z)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *while-sumstar-2*: $(x \sqcup y) \star z = x \star ((y * (x \star 1)) \star z)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *while-sumstar-3*: $(x \sqcup y) \star z = ((x \star 1) * y) \star (x \star z)$ **oops**
proposition *while-decompose-6*: $x \star ((y * (x \star 1)) \star z) = y \star ((x * (y \star 1)) \star z)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *while-finite-associative*: $x \star bot = bot \Longrightarrow (x \star y) * z = x \star (y * z)$ **oops**
proposition *atomicity-refinement*: $s = s * q \Longrightarrow x = q * x \Longrightarrow q * b = bot \Longrightarrow r * b \leq b * r \Longrightarrow r * l \leq l * r \Longrightarrow x * l \leq l * x \Longrightarrow b * l \leq l * b \Longrightarrow q * l \leq l * q \Longrightarrow r \star q \leq q * (r \star 1) \Longrightarrow q \leq 1 \Longrightarrow s * ((x \sqcup b \sqcup r \sqcup l) \star (q * z)) \leq s * ((x * (b \star q) \sqcup r \sqcup l) \star z)$ **oops**

proposition *while-separate-right-plus*: $y * x \leq x * (x \star (1 \sqcup y)) \sqcup 1 \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$ **oops**
proposition $y \star (x \star 1) \leq x \star (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**
proposition $y * x \leq (1 \sqcup x) * (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**

proposition *while-mult-sub-while-while*: $x \star (y * z) \leq (x \star y) \star z$ **oops**
proposition *while-zero-zero*: $(x \star bot) \star bot = x \star bot$ **oops**
proposition *while-denest-3*: $(x \star w) \star (x \star bot) = (x \star w) \star bot$ **oops**
proposition *while-02*: $x \star ((x \star w) \star ((x \star y) * z)) = (x \star w) \star ((x \star y) * z)$ **oops**
proposition *while-sumstar-3-below*: $(x \star y) \star (x \star z) \leq (x \star y) \star ((x \star 1) * z)$ **oops**
proposition *while-sumstar-1*: $(x \sqcup y) \star z = (x \star y) \star ((x \star 1) * z)$ **oops**
proposition *while-denest-4*: $(x \star w) \star (x \star (y * z)) = (x \star w) \star ((x \star y) * z)$ **oops**

end

class *nonstrict-itering-zero* = *nonstrict-itering* +
assumes *mult-right-zero*: $x * bot = bot$
begin

lemma *while-finite-associative-2*:
$x \star bot = bot \Longrightarrow (x \star y) * z = x \star (y * z)$
by (*metis sup-bot-left sup-bot-right mult-assoc mult-right-zero while-def*)

Theorem 13 counterexamples

proposition *while-mult-top*: $(x * top) \star z = z \sqcup x * top$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski-mult-top-idempotent*: $x * top = x * top * x * top$ **nitpick** [*expect=genuine,card=3*] **oops**

proposition *while-denest-0*: $w * (x \star (y * z)) \leq (w * (x \star y)) \star (w * (x \star y) * z)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-denest-1*: $w * (x \star (y * z)) \leq (w * (x \star y)) \star z$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-mult-zero-zero*: $(x * (y \star bot)) \star bot = x * (y \star bot)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-denest-2*: $w * ((x \star (y * w)) \star z) = w * (((x \star y) * w) \star z)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-denest-5*: $w * ((x \star (y * w)) \star (x \star (y * z))) = w * (((x \star y) * w) \star ((x \star y) * z))$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-denest-6*: $(w * (x \star y)) \star z = z \sqcup w * ((x \sqcup y * w) \star (y * z))$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-sum-below-one*: $y * ((x \sqcup y) \star z) \leq (y * (x \star 1)) \star z$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-separate-unfold*: $(y * (x \star 1)) \star z = (y \star z) \sqcup (y \star (y * x * (x \star ((y * (x \star 1)) \star z))))$ **nitpick** [*expect=genuine,card=3*] **oops**

proposition *while-sub-while-zero*: $x \star z \leq (x \star y) \star z$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *while-while-sub-associative*: $x \star (y \star z) \leq (x \star y) \star z$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *tarski*: $x \leq x * top * x * top$ **nitpick** [*expect=genuine,card=3*]

oops
proposition *tarski-top-omega-below*: $x * top \leq (x * top)^\omega$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski-top-omega*: $x * top = (x * top)^\omega$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski-below-top-omega*: $x \leq (x * top)^\omega$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski-mult-omega-omega*: $(x * y^\omega)^\omega = x * y^\omega$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski-mult-omega-omega*: $(\forall z . z^{\omega\omega} = z^\omega) \Longrightarrow (x * y^\omega)^\omega = x * y^\omega$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *tarski*: $x = bot \vee top * x * top = top$ **nitpick** [*expect=genuine,card=3*] **oops**

end

class *nonstrict-itering-tarski* = *nonstrict-itering* +
assumes *tarski*: $x \leq x * top * x * top$
begin

Theorem 13.14

lemma *tarski-mult-top-idempotent*:
$x * top = x * top * x * top$
by (*metis sup-commute le-iff-sup mult-assoc star.circ-back-loop-fixpoint star.circ-left-top tarski top-mult-top*)

lemma *tarski-top-omega-below*:
$x * top \leq (x * top)^\omega$
using *omega-induct-mult order.refl mult-assoc tarski-mult-top-idempotent* **by** *auto*

lemma *tarski-top-omega*:
$x * top = (x * top)^\omega$
by (*simp add*: *order.eq-iff mult-top-omega tarski-top-omega-below*)

lemma *tarski-below-top-omega*:
$x \leq (x * top)^\omega$
using *top-right-mult-increasing tarski-top-omega* **by** *auto*

lemma *tarski-mult-omega-omega*:
$(x * y^\omega)^\omega = x * y^\omega$
by (*metis mult-assoc omega-vector tarski-top-omega*)

lemma *tarski-omega-idempotent*:
$x^{\omega\omega} = x^\omega$
by (*metis omega-vector tarski-top-omega*)

lemma *while-denest-2a*:
$w * ((x \star (y * w)) \star z) = w * (((x \star y) * w) \star z)$

proof −
  have $(x^\omega \sqcup x^\star * y * w)^\omega = (x^\star * y * w)^\star * x^\omega * (((x^\star * y * w)^\star * x^\omega)^\omega \sqcup ((x^\star * y * w)^\star * x^\omega)^\star * (x^\star * y * w)^\omega) \sqcup (x^\star * y * w)^\omega$
    by (*metis sup-commute omega-decompose omega-loop-fixpoint*)
  also have ... $\leq (x^\star * y * w)^\star * x^\omega \sqcup (x^\star * y * w)^\omega$
    by (*metis sup-left-isotone mult-assoc mult-right-isotone omega-sub-vector*)
  finally have *1*: $w * (x^\omega \sqcup x^\star * y * w)^\omega \leq (w * x^\star * y)^\star * w * x^\omega \sqcup (w * x^\star * y)^\omega$
    by (*smt sup-commute le-iff-sup mult-assoc mult-left-dist-sup while-def while-slide*)
  have $(x^\omega \sqcup x^\star * y * w)^\star * z = (x^\star * y * w)^\star * x^\omega * ((x^\star * y * w)^\star * x^\omega)^\star * (x^\star * y * w)^\star * z \sqcup (x^\star * y * w)^\star * z$
    by (*smt sup-commute mult-assoc star.circ-sup star.circ-loop-fixpoint*)
  also have ... $\leq (x^\star * y * w)^\star * x^\omega \sqcup (x^\star * y * w)^\star * z$
    by (*smt sup-commute sup-right-isotone mult-assoc mult-right-isotone omega-sub-vector*)
  finally have $w * (x^\omega \sqcup x^\star * y * w)^\star * z \leq (w * x^\star * y)^\star * w * x^\omega \sqcup (w * x^\star * y)^\star * w * z$
    by (*metis mult-assoc mult-left-dist-sup mult-right-isotone star.circ-slide*)
  hence $w * (x^\omega \sqcup x^\star * y * w)^\omega \sqcup w * (x^\omega \sqcup x^\star * y * w)^\star * z \leq (w * x^\star * y)^\star * (w * x^\omega)^\omega \sqcup (w * x^\star * y)^\omega \sqcup (w * x^\star * y)^\star * w * z$
    using *1* **by** (*smt sup-assoc sup-commute le-iff-sup mult-assoc tarski-mult-omega-omega*)
  also have ... $\leq (w * x^\omega \sqcup w * x^\star * y)^\star * (w * x^\omega \sqcup w * x^\star * y)^\omega \sqcup (w * x^\omega \sqcup w * x^\star * y)^\omega \sqcup (w * x^\omega \sqcup w * x^\star * y)^\star * w * z$
    by (*metis sup-mono sup-ge1 sup-ge2 mult-isotone mult-left-isotone omega-isotone star.circ-isotone*)
  also have ... $= (w * x^\omega \sqcup w * x^\star * y)^\omega \sqcup (w * x^\omega \sqcup w * x^\star * y)^\star * w * z$
    by (*simp add*: *star-mult-omega*)
  finally have $w * ((x^\omega \sqcup x^\star * y * w)^\omega \sqcup (x^\omega \sqcup x^\star * y * w)^\star * z) \leq w * ((x^\omega \sqcup x^\star * y) * w)^\omega \sqcup w * ((x^\omega \sqcup x^\star * y) * w)^\star * z$
    by (*smt mult-assoc mult-left-dist-sup omega-slide star.circ-slide*)
  hence *2*: $w * ((x \star (y * w)) \star z) \leq w * (((x \star y) * w) \star z)$
    by (*simp add*: *mult-left-dist-sup while-def mult-assoc*)
  have $w * (((x \star y) * w) \star z) \leq w * ((x \star (y * w)) \star z)$
    by (*simp add*: *mult-right-isotone while-left-isotone while-sub-associative*)
  thus *?thesis*
    using *2 order.antisym* **by** *auto*
qed

lemma *while-denest-3*:
  $(x \star w) \star x^\omega = (x \star w)^\omega$
proof −
  have *1*: $(x \star w) \star x^\omega = (x \star w)^\omega \sqcup (x \star w)^\star * x^{\omega\omega}$
    by (*simp add*: *while-def tarski-omega-idempotent*)
  also have ... $\leq (x \star w)^\omega \sqcup (x \star w)^\star * (x^\omega \sqcup x^\star * w)^\omega$
    using *mult-right-isotone omega-sub-dist semiring.add-left-mono* **by** *blast*
  also have ... $= (x \star w)^\omega$
    by (*simp add*: *star-mult-omega while-def*)

finally show *?thesis*
using *1* **by** (*simp add*: *sup.order-iff*)
qed

lemma *while-denest-4a*:
$(x \star w) \star (x \star (y * z)) = (x \star w) \star ((x \star y) * z)$
proof –
have $(x \star w) \star (x \star (y * z)) = (x \star w)^\omega \sqcup ((x \star w) \star (x^\star * y * z))$
using *while-def while-denest-3 while-left-dist-sup mult-assoc* **by** *auto*
also have ... $\leq (x \star w)^\omega \sqcup ((x \star w) \star ((x \star y) * z))$
using *mult-right-sub-dist-sup-right order.refl semiring.add-mono while-def while-right-isotone* **by** *auto*
finally have *1*: $(x \star w) \star (x \star (y * z)) \leq (x \star w) \star ((x \star y) * z)$
by (*simp add*: *while-def*)
have $(x \star w) \star ((x \star y) * z) \leq (x \star w) \star (x \star (y * z))$
by (*simp add*: *while-right-isotone while-sub-associative*)
thus *?thesis*
using *1 order.antisym* **by** *auto*
qed

Theorem 8.3

subclass *bounded-extended-binary-itering*
apply *unfold-locales*
by (*smt mult-assoc while-denest-2a while-denest-4a while-increasing while-slide*)

Theorem 13.13

lemma *while-mult-top*:
$(x * top) \star z = z \sqcup x * top$
proof –
have *1*: $z \sqcup x * top \leq (x * top) \star z$
by (*metis le-supI sup-ge1 while-def while-increasing tarski-top-omega*)
have $(x * top) \star z = z \sqcup x * top * ((x * top) \star z)$
using *while-left-unfold* **by** *auto*
also have ... $\leq z \sqcup x * top$
using *mult-right-isotone sup-right-isotone top-greatest mult-assoc* **by** *auto*
finally show *?thesis*
using *1 order.antisym* **by** *auto*
qed

lemma *tarski-top-omega-below-2*:
$x * top \leq (x * top) \star bot$
by (*simp add*: *while-mult-top*)

lemma *tarski-top-omega-2*:
$x * top = (x * top) \star bot$
by (*simp add*: *while-mult-top*)

lemma *tarski-below-top-omega-2*:
$x \leq (x * top) \star bot$

using *top-right-mult-increasing tarski-top-omega-2* **by** *auto*

proposition $1 = (x * bot) \star 1$ **nitpick** [*expect=genuine,card=3*] **oops**

end

class *nonstrict-itering-tarski-zero* = *nonstrict-itering-tarski* + *nonstrict-itering-zero*
begin

lemma *while-bot-1*:
$1 = (x * bot) \star 1$
by (*simp add*: *mult-right-zero while-zero*)

Theorem 13 counterexamples

proposition *while-associative*: $(x \star y) * z = x \star (y * z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition $(x \star 1) * y = x \star y$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-one-mult*: $(x \star 1) * x = x \star x$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition $(x \sqcup y) \star z = ((x \star 1) * y) \star ((x \star 1) * z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-mult-top-2*: $(x * top) \star z = z \sqcup x * top * z$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-top-2*: $top \star z = top * z$ **nitpick** [*expect=genuine,card=2*] **oops**

proposition *tarski*: $x = bot \vee top * x * top = top$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-back-loop-is-fixpoint*: *is-fixpoint* $(\lambda x . x * y \sqcup z) (z * (y \star 1))$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition $1 \sqcup x * bot = x \star 1$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $x = x * (x \star 1)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $x * (x \star 1) = x \star 1$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition $x \star 1 = x \star (1 \star 1)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $(x \sqcup y) \star 1 = (x \star (y \star 1)) \star 1$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $z \sqcup y * x = x \Longrightarrow y \star z \leq x$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition $y * x = x ==> y \star x \leq x$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition $z \sqcup x * y = x \Longrightarrow z * (y \star 1) \leq x$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $x * y = x \Longrightarrow x * (y \star 1) \leq x$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition $x * z = z * y \Longrightarrow x \star z \leq z * (y \star 1)$ **nitpick** [*expect=genuine,card=2*] **oops**

proposition *tarski*: $x = bot \vee top * x * top = top$ **nitpick** [*expect=genuine,card=3*] **oops**

proposition *tarski-case*: **assumes** *t1*: $x = bot \longrightarrow P\ x$ **and** *t2*: $top * x * top = top \longrightarrow P\ x$ **shows** $P\ x$ **nitpick** [*expect=genuine,card=3*] **oops**

end

end

10 Tests

theory *Tests*

imports *Subset-Boolean-Algebras.Subset-Boolean-Algebras Base*

begin

context *subset-boolean-algebra-extended*
begin

sublocale *sba-dual*: *subset-boolean-algebra-extended* **where** $uminus = uminus$ **and** $sup = inf$ **and** $minus = \lambda x\ y\ .\ -(-x \sqcap y)$ **and** $inf = sup$ **and** $bot = top$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $less = greater$ **and** $top = bot$
  apply *unfold-locales*
  apply (*simp add*: *inf-associative*)
  apply (*simp add*: *inf-commutative*)
  using *inf-cases-2* **apply** *simp*
  using *inf-closed* **apply** *simp*
  apply *simp*
  apply *simp*
  using *sub-sup-closed sub-sup-demorgan* **apply** *simp*
  apply *simp*
  apply (*simp add*: *inf-commutative less-eq-inf*)
  by (*metis inf-commutative inf-idempotent inf-left-dist-sup sub-less-def sup-absorb sup-right-zero top-double-complement*)

lemma *strict-leq-def*:
  $-x < -y \longleftrightarrow -x \le -y \wedge \neg\ (-y \le -x)$
  by (*simp add*: *sba-dual.sba-dual.sub-less-def sba-dual.sba-dual.sub-less-eq-def*)

lemma *one-def*:
  $top = -bot$
  by *simp*

end

class *tests* = $times + uminus + one + ord + sup + bot +$
  assumes *sub-assoc*: $-x * (-y * -z) = (-x * -y) * -z$
  assumes *sub-comm*: $-x * -y = -y * -x$
  assumes *sub-compl*: $-x = -(--x * -y) * -(--x * --y)$
  assumes *sub-mult-closed*: $-x * -y = --(-x * -y)$

assumes *the-bot-def*: $bot = (THE\ x .\ (\forall\ y .\ x = -y * --y))$
assumes *one-def*: $1 = -\ bot$
assumes *sup-def*: $-x \sqcup -y = -(--x * --y)$
assumes *leq-def*: $-x \leq -y \longleftrightarrow -x * -y = -x$
assumes *strict-leq-def*: $-x < -y \longleftrightarrow -x \leq -y \wedge \neg\ (-y \leq -x)$
begin

sublocale *tests-dual*: *subset-boolean-algebra-extended* **where** $uminus = uminus$ **and** $sup = times$ **and** $minus = \lambda x\ y .\ -(-x * y)$ **and** $inf = sup$ **and** $bot = 1$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $less = greater$ **and** $top = bot$
apply *unfold-locales*
apply (*simp add*: *sub-assoc*)
apply (*simp add*: *sub-comm*)
apply (*simp add*: *sub-compl*)
using *sub-mult-closed* **apply** *simp*
apply (*simp add*: *the-bot-def*)
apply (*simp add*: *one-def the-bot-def*)
apply (*simp add*: *sup-def*)
apply *simp*
apply (*simp add*: *leq-def sub-comm*)
by (*simp add*: *leq-def strict-leq-def sub-comm*)

sublocale *sba*: *subset-boolean-algebra-extended* **where** $uminus = uminus$ **and** $sup = sup$ **and** $minus = \lambda x\ y .\ -(-x \sqcup y)$ **and** $inf = times$ **and** $bot = bot$ **and** $less\text{-}eq = less\text{-}eq$ **and** $less = less$ **and** $top = 1$ **..**

sets and sequences of tests

definition $test\text{-}set :: 'a\ set \Rightarrow bool$
where $test\text{-}set\ A \equiv \forall\, x{\in}A .\ x = --x$

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

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

lemma *sup-left-dist-test-set*:
$test\text{-}set\ A \Longrightarrow test\text{-}set\ \{\ -p \sqcup x \mid x .\ x \in A\ \}$
by (*smt mem-Collect-eq tests-dual.sba-dual.sub-sup-closed test-set-def*)

lemma *sup-right-dist-test-set*:
$test\text{-}set\ A \Longrightarrow test\text{-}set\ \{\ x \sqcup -p \mid x .\ x \in A\ \}$
by (*smt mem-Collect-eq tests-dual.sba-dual.sub-sup-closed test-set-def*)

lemma *test-set-closed*:
$A \subseteq B \Longrightarrow test\text{-}set\ B \Longrightarrow test\text{-}set\ A$
using *test-set-def* **by** *auto*

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

lemma *test-seq-test-set*:
  test-seq $t \Longrightarrow$ *test-set* $\{ t\ n \mid n{::}nat . True \}$
  using *test-seq-def test-set-def* **by** *auto*

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

lemma *nat-test-seq*:
  nat-test $t\ s \Longrightarrow$ *test-seq* t
  by (*simp add*: *nat-test-def test-seq-def*)

primrec *pSum* :: $(nat \Rightarrow 'a) \Rightarrow nat \Rightarrow 'a$
  where *pSum* $f\ 0 = bot$
    | *pSum* $f\ (Suc\ m) = pSum\ f\ m \sqcup f\ m$

lemma *pSum-test*:
  test-seq $t \Longrightarrow pSum\ t\ m = --(pSum\ t\ m)$
  apply (*induct m*)
  apply *simp*
  by (*smt pSum.simps*(*2*) *tests-dual.sba-dual.sub-sup-closed test-seq-def*)

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

lemma *pSum-upper*:
  test-seq $t \Longrightarrow i{<}m \Longrightarrow t\ i \leq pSum\ t\ m$
proof (*induct m*)
  show *test-seq* $t \Longrightarrow i{<}0 \Longrightarrow t\ i \leq pSum\ t\ 0$
    by (*smt less-zeroE*)
next
  fix n
  assume *test-seq* $t \Longrightarrow i{<}n \Longrightarrow t\ i \leq pSum\ t\ n$
  hence *test-seq* $t \Longrightarrow i{<}n \Longrightarrow t\ i \leq pSum\ t\ (Suc\ n)$
    by (*smt* (*z3*) *pSum.simps*(*2*) *pSum-test tests-dual.sba-dual.upper-bound-left*
tests-dual.transitive test-seq-def)
  thus *test-seq* $t \Longrightarrow i{<}Suc\ n \Longrightarrow t\ i \leq pSum\ t\ (Suc\ n)$
    by (*metis less-Suc-eq pSum.simps*(*2*) *pSum-test*
tests-dual.sba-dual.upper-bound-right test-seq-def)
qed

lemma *pSum-below*:
  test-seq $t \Longrightarrow (\forall m{<}k . t\ m * -p \leq -q) \Longrightarrow pSum\ t\ k * -p \leq -q$
  apply (*induct k*)

apply (*simp add*: *tests-dual.top-greatest*)
by (*smt* (*verit*, *ccfv-threshold*) *tests-dual.sup-right-dist-inf pSum.simps*(*2*) *pSum-test test-seq-def sub-mult-closed less-Suc-eq tests-dual.sba-dual.sub-associative tests-dual.sba-dual.sub-less-eq-def*)

lemma *pSum-below-nat*:
nat-test t s $\Longrightarrow$ ($\forall$ *m*<*k* . *t m* $*$ $-p \leq -q$) $\Longrightarrow$ *pSum t k* $*$ $-p \leq -q$
by (*simp add*: *nat-test-seq pSum-below*)

lemma *pSum-below-sum*:
nat-test t s $\Longrightarrow$ *pSum t x* $\leq$ *s*
by (*smt* (*verit*, *ccfv-threshold*) *tests-dual.sup-right-unit nat-test-def one-def pSum-below-nat pSum-test-nat*)

lemma *ascending-chain-sup-left*:
ascending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *ascending-chain* (λn . $-p \sqcup t\ n$) $\wedge$ *test-seq* (λn . $-p \sqcup t\ n$)
by (*smt* (*z3*) *ord.ascending-chain-def tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.sub-sup-right-isotone test-seq-def*)

lemma *ascending-chain-sup-right*:
ascending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *ascending-chain* (λn . $t\ n \sqcup -p$) $\wedge$ *test-seq* (λn . $t\ n \sqcup -p$)
by (*smt ascending-chain-def tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.sub-sup-left-isotone test-seq-def*)

lemma *ascending-chain-mult-left*:
ascending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *ascending-chain* (λn . $-p * t\ n$) $\wedge$ *test-seq* (λn . $-p * t\ n$)
by (*smt* (*z3*) *ascending-chain-def sub-mult-closed tests-dual.sba-dual.reflexive tests-dual.sup-isotone test-seq-def*)

lemma *ascending-chain-mult-right*:
ascending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *ascending-chain* (λn . $t\ n * -p$) $\wedge$ *test-seq* (λn . $t\ n * -p$)
by (*smt* (*z3*) *ascending-chain-def sub-mult-closed tests-dual.sba-dual.reflexive tests-dual.sup-isotone test-seq-def*)

lemma *descending-chain-sup-left*:
descending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *descending-chain* (λn . $-p \sqcup t\ n$) $\wedge$ *test-seq* (λn . $-p \sqcup t\ n$)
by (*smt descending-chain-def tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.sub-sup-right-isotone test-seq-def*)

lemma *descending-chain-sup-right*:
descending-chain t $\Longrightarrow$ *test-seq t* $\Longrightarrow$ *descending-chain* (λn . $t\ n \sqcup -p$) $\wedge$ *test-seq* (λn . $t\ n \sqcup -p$)
by (*smt descending-chain-def tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.sub-sup-left-isotone test-seq-def*)

lemma *descending-chain-mult-left*:
descending-chain $t \Longrightarrow$ *test-seq* $t \Longrightarrow$ *descending-chain* $(\lambda n\ .\ -p * t\ n) \wedge$ *test-seq* $(\lambda n\ .\ -p * t\ n)$
by (*smt* (*z3*) *descending-chain-def sub-mult-closed tests-dual.sba-dual.reflexive tests-dual.sup-isotone test-seq-def*)

lemma *descending-chain-mult-right*:
descending-chain $t \Longrightarrow$ *test-seq* $t \Longrightarrow$ *descending-chain* $(\lambda n\ .\ t\ n * -p) \wedge$ *test-seq* $(\lambda n\ .\ t\ n * -p)$
by (*smt* (*z3*) *descending-chain-def sub-mult-closed tests-dual.sba-dual.reflexive tests-dual.sup-isotone test-seq-def*)

end

end

11 Test Iterings

theory *Test-Iterings*

imports *Stone-Kleene-Relation-Algebras.Iterings Tests*

begin

class *test-itering* = *itering* + *tests* + *while* +
assumes *while-def*: $p \star y = (p * y)^\circ * -p$
begin

lemma *wnf-lemma-5*:
$(-p \sqcup -q) * (-q * x \sqcup --q * y) = -q * x \sqcup --q * -p * y$
by (*smt* (*z3*) *mult-left-dist-sup sup-commute tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.sup-complement-intro tests-dual.sba-dual.sup-idempotent tests-dual.sup-idempotent mult-assoc tests-dual.wnf-lemma-3*)

lemma *test-case-split-left-equal*:
$-z * x = -z * y \Longrightarrow --z * x = --z * y \Longrightarrow x = y$
by (*metis case-split-left-equal tests-dual.inf-complement*)

lemma *preserves-equation*:
$-y * x \leq x * -y \longleftrightarrow -y * x = -y * x * -y$
apply (*rule iffI*)
apply (*simp add*: *test-preserves-equation tests-dual.sub-bot-least*)
by (*simp add*: *test-preserves-equation tests-dual.sub-bot-least*)

Theorem 5

lemma *preserve-test*:
$-y * x \leq x * -y \Longrightarrow -y * x^\circ = -y * x^\circ * -y$
using *circ-simulate preserves-equation* **by** *blast*

Theorem 5

lemma *import-test*:
$-y * x \leq x * -y \Longrightarrow -y * x^{\circ} = -y * (-y * x)^{\circ}$
by (*simp add*: *circ-import tests-dual.sub-bot-least*)

definition *ite* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ (‹- ◁ - ▷ -› [*58*,*58*,*58*] *57*)
where $x \lhd p \rhd y \equiv p * x \sqcup -p * y$

definition *it* :: $'a \Rightarrow 'a \Rightarrow 'a$ (‹- ▷ -› [*58*,*58*] *57*)
where $p \rhd x \equiv p * x \sqcup -p$

definition *assigns* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
where *assigns* $x\ p\ q \equiv x = x * (p * q \sqcup -p * -q)$

definition *preserves* :: $'a \Rightarrow 'a \Rightarrow bool$
where *preserves* $x\ p \equiv p * x \leq x * p \wedge -p * x \leq x * -p$

lemma *ite-neg*:
$x \lhd -p \rhd y = y \lhd --p \rhd x$
by (*simp add*: *ite-def sup-commute*)

lemma *ite-import-true*:
$x \lhd -p \rhd y = -p * x \lhd -p \rhd y$
by (*metis ite-def tests-dual.sup-idempotent mult-assoc*)

lemma *ite-import-false*:
$x \lhd -p \rhd y = x \lhd -p \rhd --p * y$
by (*metis ite-import-true ite-neg*)

lemma *ite-import-true-false*:
$x \lhd -p \rhd y = -p * x \lhd -p \rhd --p * y$
using *ite-import-false ite-import-true* **by** *auto*

lemma *ite-context-true*:
$-p * (x \lhd -p \rhd y) = -p * x$
by (*metis sup-monoid.add-0-left tests-dual.sup-right-zero tests-dual.top-double-complement wnf-lemma-5 sup-bot-right ite-def mult-assoc mult-left-zero*)

lemma *ite-context-false*:
$--p * (x \lhd -p \rhd y) = --p * y$
by (*metis ite-neg ite-context-true*)

lemma *ite-context-import*:
$-p * (x \lhd -q \rhd y) = -p * (x \lhd -p * -q \rhd y)$
by (*smt ite-def mult-assoc tests-dual.sup-complement-intro tests-dual.sub-sup-demorgan tests-dual.sup-idempotent mult-left-dist-sup*)

lemma *ite-conjunction*:
$(x \lhd -q \rhd y) \lhd -p \rhd y = x \lhd -p * -q \rhd y$
by (*smt sup-assoc sup-commute ite-def mult-assoc tests-dual.sub-sup-demorgan mult-left-dist-sup mult-right-dist-sup tests-dual.inf-complement-intro*)

lemma *ite-disjunction*:
$x \lhd -p \rhd (x \lhd -q \rhd y) = x \lhd -p \sqcup -q \rhd y$
by (*smt* (*z3*) *tests-dual.sba-dual.sub-sup-closed sup-assoc ite-def mult-assoc tests-dual.sup-complement-intro tests-dual.sub-sup-demorgan mult-left-dist-sup mult-right-dist-sup tests-dual.inf-demorgan*)

lemma *wnf-lemma-6*:
$(-p \sqcup -q) * (x \lhd --p * -q \rhd y) = (-p \sqcup -q) * (y \lhd -p \rhd x)$
by (*smt* (*z3*) *ite-conjunction ite-context-false ite-context-true semiring.distrib-right tests-dual.sba-dual.inf-cases-2 tests-dual.sba-dual.sub-inf-def tests-dual.sba-dual.sup-complement-intro tests-dual.sub-complement*)

lemma *it-ite*:
$-p \rhd x = x \lhd -p \rhd 1$
by (*simp add*: *it-def ite-def*)

lemma *it-neg*:
$--p \rhd x = 1 \lhd -p \rhd x$
using *it-ite ite-neg* **by** *auto*

lemma *it-import-true*:
$-p \rhd x = -p \rhd -p * x$
using *it-ite ite-import-true* **by** *auto*

lemma *it-context-true*:
$-p * (-p \rhd x) = -p * x$
by (*simp add*: *it-ite ite-context-true*)

lemma *it-context-false*:
$--p * (-p \rhd x) = --p$
using *it-ite ite-context-false* **by** *force*

lemma *while-unfold-it*:
$-p \star x = -p \rhd x * (-p \star x)$
by (*metis circ-loop-fixpoint it-def mult-assoc while-def*)

lemma *while-context-false*:
$--p * (-p \star x) = --p$
by (*metis it-context-false while-unfold-it*)

lemma *while-context-true*:
$-p * (-p \star x) = -p * x * (-p \star x)$
by (*metis it-context-true mult-assoc while-unfold-it*)

lemma *while-zero*:
$bot \star x = 1$
by (*metis circ-zero mult-left-one mult-left-zero one-def while-def*)

lemma *wnf-lemma-7*:
$1 * (bot \star 1) = 1$
by (*simp add*: *while-zero*)

lemma *while-import-condition*:
$-p \star x = -p \star -p * x$
by (*metis mult-assoc tests-dual.sup-idempotent while-def*)

lemma *while-import-condition-2*:
$-p * -q \star x = -p * -q \star -p * x$
by (*metis mult-assoc tests-dual.sup-idempotent sub-comm while-def*)

lemma *wnf-lemma-8*:
$-r * (-p \sqcup --p * -q) \star (x \lhd --p * -q \rhd y) = -r * (-p \sqcup -q) \star (y \lhd -p \rhd x)$
by (*metis mult-assoc while-def wnf-lemma-6 tests-dual.sba-dual.sup-complement-intro*)

Theorem 6 - see Theorem 31 on page 329 of Back and von Wright, Acta Informatica 36:295-334, 1999

lemma *split-merge-loops*:
assumes $--p * y \leq y * --p$
shows $(-p \sqcup -q) \star (x \lhd -p \rhd y) = (-p \star x) * (-q \star y)$
proof −
have $-p \sqcup -q \star (x \lhd -p \rhd y) = (-p * x \sqcup --p * -q * y)^{\circ} * --p * --q$
by (*smt ite-def mult-assoc sup-commute tests-dual.inf-demorgan while-def wnf-lemma-5*)
thus *?thesis*
by (*smt assms circ-sup-1 circ-slide import-test mult-assoc preserves-equation sub-comm while-context-false while-def*)
qed

lemma *assigns-same*:
assigns x $(-p)$ $(-p)$
by (*simp add*: *assigns-def*)

lemma *preserves-equation-test*:
preserves x $(-p) \longleftrightarrow -p * x = -p * x * -p \wedge --p * x = --p * x * --p$
using *preserves-def preserves-equation* **by** *auto*

lemma *preserves-test*:
preserves $(-q)$ $(-p)$
using *tests-dual.sub-commutative preserves-def* **by** *auto*

lemma *preserves-zero*:
  preserves bot $(-p)$
  using *tests-dual.sba-dual.sub-bot-def preserves-test* **by** *blast*

lemma *preserves-one*:
  preserves 1 $(-p)$
  using *preserves-def* **by** *force*

lemma *preserves-sup*:
  preserves x $(-p) \Longrightarrow$ *preserves* y $(-p) \Longrightarrow$ *preserves* $(x \sqcup y)$ $(-p)$
  by (*simp add*: *mult-left-dist-sup mult-right-dist-sup preserves-equation-test*)

lemma *preserves-mult*:
  preserves x $(-p) \Longrightarrow$ *preserves* y $(-p) \Longrightarrow$ *preserves* $(x * y)$ $(-p)$
  by (*smt* (*verit*, *best*) *mult-assoc preserves-equation-test*)

lemma *preserves-ite*:
  preserves x $(-p) \Longrightarrow$ *preserves* y $(-p) \Longrightarrow$ *preserves* $(x \lhd -q \rhd y)$ $(-p)$
  by (*simp add*: *ite-def preserves-mult preserves-sup preserves-test*)

lemma *preserves-it*:
  preserves x $(-p) \Longrightarrow$ *preserves* $(-q \rhd x)$ $(-p)$
  by (*simp add*: *it-ite preserves-ite preserves-one*)

lemma *preserves-circ*:
  preserves x $(-p) \Longrightarrow$ *preserves* (x°) $(-p)$
  by (*meson circ-simulate preserves-def*)

lemma *preserves-while*:
  preserves x $(-p) \Longrightarrow$ *preserves* $(-q \star x)$ $(-p)$
  using *while-def preserves-circ preserves-mult preserves-test* **by** *auto*

lemma *preserves-test-neg*:
  preserves x $(-p) \Longrightarrow$ *preserves* x $(--p)$
  using *preserves-def* **by** *auto*

lemma *preserves-import-circ*:
  preserves x $(-p) \Longrightarrow -p * x^\circ = -p * (-p * x)^\circ$
  using *import-test preserves-def* **by** *blast*

lemma *preserves-simulate*:
  preserves x $(-p) \Longrightarrow -p * x^\circ = -p * x^\circ * -p$
  using *preserve-test preserves-def* **by** *auto*

lemma *preserves-import-ite*:
  assumes *preserves* z $(-p)$
    shows $z * (x \lhd -p \rhd y) = z * x \lhd -p \rhd z * y$
proof −
  have *1*: $-p * z * (x \lhd -p \rhd y) = -p * (z * x \lhd -p \rhd z * y)$

by (*smt assms ite-context-true mult-assoc preserves-equation-test*)
have $--p * z * (x \lhd -p \rhd y) = --p * (z * x \lhd -p \rhd z * y)$
by (*smt* (*z3*) *assms ite-context-false mult-assoc preserves-equation-test*)
thus *?thesis*
using *1* **by** (*metis mult-assoc test-case-split-left-equal*)
qed

lemma *preserves-while-context*:
preserves $x\ (-p) \Longrightarrow -p * (-q \star x) = -p * (-p * -q \star x)$
by (*smt* (*verit*, *del-insts*) *mult-assoc tests-dual.sup-complement-intro tests-dual.sub-sup-demorgan preserves-import-circ preserves-mult preserves-simulate preserves-test while-def*)

lemma *while-ite-context-false*:
assumes *preserves* $y\ (-p)$
shows $--p * (-p \sqcup -q \star (x \lhd -p \rhd y)) = --p * (-q \star y)$
proof –
have $--p * (-p \sqcup -q \star (x \lhd -p \rhd y)) = --p * (--p * -q * y)^\circ * -(-p \sqcup -q)$
by (*smt* (*z3*) *assms import-test mult-assoc preserves-equation preserves-equation-test sub-comm while-def tests-dual.sba-dual.sub-sup-demorgan preserves-test split-merge-loops while-context-false*)
thus *?thesis*
by (*metis* (*no-types*, *lifting*) *assms preserves-def mult.assoc split-merge-loops while-context-false*)
qed

Theorem 7.1

lemma *while-ite-norm*:
assumes *assigns* $z\ (-p)\ (-q)$
and *preserves* $x1\ (-q)$
and *preserves* $x2\ (-q)$
and *preserves* $y1\ (-q)$
and *preserves* $y2\ (-q)$
shows $z * (x1 * (-r1 \star y1) \lhd -p \rhd x2 * (-r2 \star y2)) = z * (x1 \lhd -q \rhd x2) * ((-q * -r1 \sqcup --q * -r2) \star (y1 \lhd -q \rhd y2))$
proof –
have *1*: $-(-q * -r1 \sqcup --q * -r2) = -q * --r1 \sqcup --q * --r2$
by (*smt* (*z3*) *tests-dual.complement-2 tests-dual.sub-sup-closed tests-dual.case-duality tests-dual.sub-sup-demorgan*)
have $-p * -q * x1 * (-q * -r1 * y1 \sqcup --q * -r2 * y2)^\circ * (-q * --r1 \sqcup --q * --r2) = -p * -q * x1 * -q * (-q * (-q * -r1 * y1 \sqcup --q * -r2 * y2))^\circ * (-q * --r1 \sqcup --q * --r2)$
by (*smt* (*verit*, *del-insts*) *assms*(*2*,*4*,*5*) *mult-assoc preserves-sup preserves-equation-test preserves-import-circ preserves-mult preserves-test*)
also have ... $= -p * -q * x1 * -q * (-q * -r1 * y1)^\circ * (-q * --r1 \sqcup --q * --r2)$
using *ite-context-true ite-def mult-assoc* **by** *auto*
finally have *2*: $-p * -q * x1 * (-q * -r1 * y1 \sqcup --q * -r2 * y2)^\circ * (-q *$

$--r1 \sqcup --q * --r2) = -p * -q * x1 * (-r1 * y1)^\circ * --r1$
by (*smt* (*verit*, *del-insts*) *assms ite-context-true ite-def mult-assoc preserves-equation-test preserves-import-circ preserves-mult preserves-simulate preserves-test*)
have $--p * --q * x2 * (-q * -r1 * y1 \sqcup --q * -r2 * y2)^\circ * (-q * --r1 \sqcup --q * --r2) = --p * --q * x2 * --q * (--q * (-q * -r1 * y1 \sqcup --q * -r2 * y2))^\circ * (-q * --r1 \sqcup --q * --r2)$
by (*smt* (*verit*, *del-insts*) *assms mult-assoc preserves-sup preserves-equation-test preserves-import-circ preserves-mult preserves-test preserves-test-neg*)
also have ... $= --p * --q * x2 * --q * (--q * -r2 * y2)^\circ * (-q * --r1 \sqcup --q * --r2)$
using *ite-context-false ite-def mult-assoc* **by** *auto*
finally have $--p * --q * x2 * (-q * -r1 * y1 \sqcup --q * -r2 * y2)^\circ * (-q * --r1 \sqcup --q * --r2) = --p * --q * x2 * (-r2 * y2)^\circ * --r2$
by (*smt* (*verit*, *del-insts*) *assms*(*3*,*5*) *ite-context-false ite-def mult-assoc preserves-equation-test preserves-import-circ preserves-mult preserves-simulate preserves-test preserves-test-neg*)
thus *?thesis*
using *1 2* **by** (*smt* (*z3*) *assms*(*1*) *assigns-def mult-assoc mult-right-dist-sup while-def ite-context-false ite-context-true tests-dual.sub-commutative*)
qed

lemma *while-it-norm*:
assigns $z\ (-p)\ (-q) \Longrightarrow$ *preserves* $x\ (-q) \Longrightarrow$ *preserves* $y\ (-q) \Longrightarrow z * (-p \rhd x * (-r \star y)) = z * (-q \rhd x) * (-q * -r \star y)$
by (*metis sup-bot-right tests-dual.sup-right-zero it-context-true it-ite tests-dual.complement-bot preserves-one while-import-condition-2 while-ite-norm wnf-lemma-7*)

lemma *while-else-norm*:
assigns $z\ (-p)\ (-q) \Longrightarrow$ *preserves* $x\ (-q) \Longrightarrow$ *preserves* $y\ (-q) \Longrightarrow z * (1 \lhd -p \rhd x * (-r \star y)) = z * (1 \lhd -q \rhd x) * (--q * -r \star y)$
by (*metis sup-bot-left tests-dual.sup-right-zero ite-context-false tests-dual.complement-bot preserves-one while-import-condition-2 while-ite-norm wnf-lemma-7*)

lemma *while-while-pre-norm*:
$-p \star x * (-q \star y) = -p \rhd x * (-p \sqcup -q \star (y \lhd -q \rhd x))$
by (*smt sup-commute circ-sup-1 circ-left-unfold circ-slide it-def ite-def mult-assoc mult-left-one mult-right-dist-sup tests-dual.inf-demorgan while-def wnf-lemma-5*)

Theorem 7.2

lemma *while-while-norm*:
assigns $z\ (-p)\ (-r) \Longrightarrow$ *preserves* $x\ (-r) \Longrightarrow$ *preserves* $y\ (-r) \Longrightarrow z * (-p \star x * (-q \star y)) = z * (-r \rhd x) * (-r * (-p \sqcup -q) \star (y \lhd -q \rhd x))$
by (*smt tests-dual.double-negation tests-dual.sub-sup-demorgan tests-dual.inf-demorgan preserves-ite while-it-norm while-while-pre-norm*)

lemma *while-seq-replace*:
assigns $z\ (-p)\ (-q) \Longrightarrow z * (-p \star x * z) * y = z * (-q \star x * z) * y$
by (*smt assigns-def circ-slide mult-assoc tests-dual.wnf-lemma-1 tests-dual.wnf-lemma-2 tests-dual.wnf-lemma-3 tests-dual.wnf-lemma-4 while-def*)

lemma *while-ite-replace*:
assigns $z\ (-p)\ (-q) \Longrightarrow z * (x \lhd -p \rhd y) = z * (x \lhd -q \rhd y)$
by (*smt assigns-def ite-def mult-assoc mult-left-dist-sup sub-comm tests-dual.wnf-lemma-1 tests-dual.wnf-lemma-3*)

lemma *while-post-norm-an*:
assumes *preserves* $y\ (-p)$
shows $(-p \star x) * y = y \lhd --p \rhd (-p \star x * (--p \rhd y))$
proof –
have $-p * (-p * x * (--p * y \sqcup -p))^\circ * --p = -p * x * ((--p * y \sqcup -p) * -p * x)^\circ * (--p * y \sqcup -p) * --p$
by (*metis circ-slide-1 while-def mult-assoc while-context-true*)
also have ... $= -p * x * (--p * y * bot \sqcup -p * x)^\circ * --p * y$
by (*smt assms sup-bot-right mult-assoc tests-dual.sup-complement tests-dual.sup-idempotent mult-left-zero mult-right-dist-sup preserves-equation-test sub-comm*)
finally have $-p * (-p * x * (--p * y \sqcup -p))^\circ * --p = -p * x * (-p * x)^\circ * --p * y$
by (*metis circ-sup-mult-zero sup-commute mult-assoc*)
thus *?thesis*
by (*smt circ-left-unfold tests-dual.double-negation it-def ite-def mult-assoc mult-left-one mult-right-dist-sup while-def*)
qed

lemma *while-post-norm*:
preserves $y\ (-p) \Longrightarrow (-p \star x) * y = -p \star x * (1 \lhd -p \rhd y) \lhd -p \rhd y$
using *it-neg ite-neg while-post-norm-an* **by** *force*

lemma *wnf-lemma-9*:
assumes *assigns* $z\ (-p)\ (-q)$
and *preserves* $x1\ (-q)$
and *preserves* $y1\ (-q)$
and *preserves* $x2\ (-q)$
and *preserves* $y2\ (-q)$
and *preserves* $x2\ (-p)$
and *preserves* $y2\ (-p)$
shows $z * (x1 \lhd -q \rhd x2) * (-q * -p \sqcup -r \star (y1 \lhd -q * -p \rhd y2)) = z * (x1 \lhd -p \rhd x2) * (-p \sqcup -r \star (y1 \lhd -p \rhd y2))$
proof –
have $z * --p * --q * (x1 \lhd -q \rhd x2) * (-q * -p \sqcup -r \star (y1 \lhd -q * -p \rhd y2)) = z * --p * --q * x2 * --q * (--q * (-q * -p \sqcup -r) \star (y1 \lhd -q * -p \rhd y2))$
by (*smt* (*verit, del-insts*) *assms*($3-5$) *tests-dual.double-negation*

ite-context-false mult-assoc tests-dual.sub-sup-demorgan tests-dual.inf-demorgan preserves-equation-test preserves-ite preserves-while-context)
  also have ... $= z * --p * --q * x2 * --q * (--q * -r \star --q * y2)$
    by (*smt sup-bot-left tests-dual.double-negation ite-conjunction ite-context-false mult-assoc tests-dual.sup-complement mult-left-dist-sup mult-left-zero while-import-condition-2*)
  also have ... $= z * --p * --q * x2 * (-r \star y2)$
    by (*metis assms*(*4*,*5*) *mult-assoc preserves-equation-test preserves-test-neg preserves-while-context while-import-condition-2*)
  finally have *1*: $z * --p * --q * (x1 \lhd -q \rhd x2) * (-q * -p \sqcup -r \star (y1 \lhd -q * -p \rhd y2)) = z * --p * --q * (x1 \lhd -q \rhd x2) * (-p \sqcup -r \star (y1 \lhd -p \rhd y2))$
    by (*smt assms*(*6*,*7*) *ite-context-false mult-assoc preserves-equation-test sub-comm while-ite-context-false*)
  have $z * -p * -q * (x1 \lhd -q \rhd x2) * (-q * -p \sqcup -r \star (y1 \lhd -q * -p \rhd y2)) = z * -p * -q * (x1 \lhd -q \rhd x2) * -q * (-q * (-p \sqcup -r) \star -q * (y1 \lhd -p \rhd y2))$
    by (*smt* (*verit*, *del-insts*) *assms*(*2*−*5*) *tests-dual.double-negation ite-context-import mult-assoc tests-dual.sub-sup-demorgan tests-dual.sup-idempotent mult-left-dist-sup tests-dual.inf-demorgan preserves-equation-test preserves-ite preserves-while-context while-import-condition-2*)
  hence $z * -p * -q * (x1 \lhd -q \rhd x2) * (-q * -p \sqcup -r \star (y1 \lhd -q * -p \rhd y2)) = z * -p * -q * (x1 \lhd -q \rhd x2) * (-p \sqcup -r \star (y1 \lhd -p \rhd y2))$
    by (*smt assms*(*2*−*5*) *tests-dual.double-negation mult-assoc tests-dual.sub-sup-demorgan tests-dual.sup-idempotent preserves-equation-test preserves-ite preserves-while-context while-import-condition-2*)
  thus *?thesis*
    using *1* **by** (*smt assms*(*1*) *assigns-def mult-assoc mult-left-dist-sup mult-right-dist-sup while-ite-replace*)
qed

Theorem 7.3

lemma *while-seq-norm*:
  assumes *assigns z1* $(-r1)$ $(-q)$
    and *preserves x2* $(-q)$
    and *preserves y2* $(-q)$
    and *preserves z2* $(-q)$
    and $z1 * z2 = z2 * z1$
    and *assigns z2* $(-q)$ $(-r)$
    and *preserves y1* $(-r)$
    and *preserves z1* $(-r)$
    and *preserves x2* $(-r)$
    and *preserves y2* $(-r)$
  shows $x1 * z1 * z2 * (-r1 \star y1 * z1) * x2 * (-r2 \star y2) = x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd x2) * (-q \sqcup -r2 \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2))$
proof −
  have *1*: *preserves* $(y1 * z1 * (1 \lhd -q \rhd x2))$ $(-r)$

by (*simp add*: *assms*(*7−9*) *ite-def preserves-mult preserves-sup preserves-test*)
hence *2*: *preserves* $(y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)$ $(-r)$
by (*simp add*: *assms*(*10*) *preserves-ite*)
have $x1 * z1 * z2 * (-r1 \star y1 * z1) * x2 * (-r2 \star y2) = x1 * z1 * z2 * (-q \star y1 * z1) * x2 * (-r2 \star y2)$
using *assms*(*1*,*5*) *mult-assoc while-seq-replace* **by** *auto*
also have ... $= x1 * z1 * z2 * (-q \star y1 * z1 * (1 \lhd -q \rhd x2 * (-r2 \star y2)) \lhd -q \rhd x2 * (-r2 \star y2))$
by (*smt assms*(*2*,*3*) *mult-assoc preserves-mult preserves-while while-post-norm*)
also have ... $= x1 * z1 * (z2 * (-q \star y1 * z1 * (1 \lhd -q \rhd x2) * (--q * -r2 \star y2)) \lhd -q \rhd z2 * x2 * (-r2 \star y2))$
by (*smt assms*(*2−4*) *assigns-same mult-assoc preserves-import-ite while-else-norm*)
also have ... $= x1 * z1 * (z2 * (-r \rhd y1 * z1 * (1 \lhd -q \rhd x2)) * (-r * (-q \sqcup -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -q \rhd z2 * x2 * (-r2 \star y2))$
by (*smt assms*(*6−10*) *tests-dual.double-negation tests-dual.sub-sup-demorgan tests-dual.inf-demorgan preserves-ite preserves-mult preserves-one while-while-norm wnf-lemma-8*)
also have ... $= x1 * z1 * z2 * ((-r \rhd y1 * z1 * (1 \lhd -q \rhd x2)) * (-r * (-q \sqcup -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -r \rhd x2 * (-r2 \star y2))$
by (*smt assms*(*4*,*6*) *mult-assoc preserves-import-ite while-ite-replace*)
also have ... $= x1 * z1 * z2 * (-r * (y1 * z1 * (1 \lhd -q \rhd x2)) * (-r * (-q \sqcup -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -r \rhd x2 * (-r2 \star y2))$
by (*smt mult-assoc it-context-true ite-import-true*)
also have ... $= x1 * z1 * z2 * (-r * (y1 * z1 * (1 \lhd -q \rhd x2)) * -r * (-r * (-q \sqcup -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -r \rhd x2 * (-r2 \star y2))$
using *1* **by** (*simp add*: *preserves-equation-test*)
also have ... $= x1 * z1 * z2 * (-r * (y1 * z1 * (1 \lhd -q \rhd x2)) * -r * (-q \sqcup -r2 \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -r \rhd x2 * (-r2 \star y2))$
using *2* **by** (*smt* (*z3*) *tests-dual.sba-dual.sub-sup-closed mult-assoc preserves-while-context*)
also have ... $= x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) * (-q \sqcup -r2 \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2)) \lhd -q \rhd x2 * (-r2 \star y2))$
by (*smt assms*(*6−9*) *tests-dual.double-negation ite-import-true mult-assoc tests-dual.sup-idempotent preserves-equation-test preserves-ite preserves-one while-ite-replace*)
also have ... $= x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -r \rhd x2) * ((-r * (-q \sqcup -r2) \sqcup --r * -r2) \star ((y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2) \lhd -r \rhd y2))$
by (*smt assms*(*6−10*) *tests-dual.double-negation mult-assoc tests-dual.sub-sup-demorgan tests-dual.inf-demorgan preserves-ite preserves-mult preserves-one while-ite-norm*)
also have ... $= x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -r \rhd x2) * ((-r * (-q \sqcup -r2) \sqcup --r * -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -r * -q \rhd y2))$
using *ite-conjunction* **by** *simp*
also have ... $= x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -r \rhd x2) * ((-r * -q \sqcup -r2) \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -r * -q \rhd y2))$
by (*smt* (*z3*) *mult-left-dist-sup sup-assoc tests-dual.sba-dual.sup-cases*

tests-dual.sub-commutative)
  also have ... $= x1 * z1 * z2 * (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd x2) * (-q \sqcup -r2 \star (y1 * z1 * (1 \lhd -q \rhd x2) \lhd -q \rhd y2))$
    using *1* **by** (*metis assms*(*2*,*3*,*6*,*9*,*10*) *mult-assoc wnf-lemma-9*)
  finally show *?thesis*
    .
qed

end

end

12 N-Semirings

theory *N-Semirings*

imports *Test-Iterings Omega-Algebras*

begin

class *n-semiring* = *bounded-idempotent-left-zero-semiring* + *n* + *L* +
  assumes *n-bot* : $n(bot) = bot$
  assumes *n-top* : $n(top) = 1$
  assumes *n-dist-sup* : $n(x \sqcup y) = n(x) \sqcup n(y)$
  assumes *n-export* : $n(n(x) * y) = n(x) * n(y)$
  assumes *n-sub-mult-bot*: $n(x) = n(x * bot) * n(x)$
  assumes *n-L-split* : $x * n(y) * L = x * bot \sqcup n(x * y) * L$
  assumes *n-split* : $x \le x * bot \sqcup n(x * L) * top$
begin

lemma *n-sub-one*:
  $n(x) \le 1$
  by (*metis sup-left-top sup-ge2 n-dist-sup n-top*)

Theorem 15

lemma *n-isotone*:
  $x \le y \Longrightarrow n(x) \le n(y)$
  by (*metis le-iff-sup n-dist-sup*)

lemma *n-mult-idempotent*:
  $n(x) * n(x) = n(x)$
  by (*metis mult-assoc mult-1-right n-export n-sub-mult-bot n-top*)

Theorem 15.3

lemma *n-mult-bot*:
  $n(x) = n(x * bot)$
  by (*metis sup-commute sup-left-top sup-bot-right mult-left-dist-sup mult-1-right n-dist-sup n-sub-mult-bot n-top*)

lemma *n-mult-left-upper-bound*:
$n(x) \leq n(x * y)$
by (*metis mult-right-isotone n-isotone n-mult-bot bot-least*)

lemma *n-mult-right-bot*:
$n(x) * bot = bot$
by (*metis sup-left-top sup-bot-left mult-left-one mult-1-right n-export n-dist-sup n-sub-mult-bot n-top n-bot*)

Theorem 15.9

lemma *n-mult-n*:
$n(x * n(y)) = n(x)$
by (*metis mult-assoc n-mult-right-bot n-mult-bot*)

lemma *n-mult-left-absorb-sup*:
$n(x) * (n(x) \sqcup n(y)) = n(x)$
by (*metis sup-left-top mult-left-dist-sup mult-1-right n-dist-sup n-mult-idempotent n-top*)

lemma *n-mult-right-absorb-sup*:
$(n(x) \sqcup n(y)) * n(y) = n(y)$
by (*metis sup-commute sup-left-top mult-left-one mult-right-dist-sup n-dist-sup n-mult-idempotent n-top*)

lemma *n-sup-left-absorb-mult*:
$n(x) \sqcup n(x) * n(y) = n(x)$
using *mult-left-dist-sup n-mult-idempotent n-mult-left-absorb-sup* **by** *auto*

lemma *n-sup-right-absorb-mult*:
$n(x) * n(y) \sqcup n(y) = n(y)$
using *mult-right-dist-sup n-mult-idempotent n-mult-right-absorb-sup* **by** *auto*

lemma *n-mult-commutative*:
$n(x) * n(y) = n(y) * n(x)$
by (*smt sup-commute mult-left-dist-sup mult-right-dist-sup n-sup-left-absorb-mult n-sup-right-absorb-mult n-export n-mult-idempotent*)

lemma *n-sup-left-dist-mult*:
$n(x) \sqcup n(y) * n(z) = (n(x) \sqcup n(y)) * (n(x) \sqcup n(z))$
by (*metis sup-assoc mult-left-dist-sup mult-right-dist-sup n-sup-right-absorb-mult n-mult-commutative n-mult-left-absorb-sup*)

lemma *n-sup-right-dist-mult*:
$n(x) * n(y) \sqcup n(z) = (n(x) \sqcup n(z)) * (n(y) \sqcup n(z))$
by (*simp add*: *sup-commute n-sup-left-dist-mult*)

lemma *n-order*:
$n(x) \leq n(y) \longleftrightarrow n(x) * n(y) = n(x)$
by (*metis le-iff-sup n-sup-right-absorb-mult n-mult-left-absorb-sup*)

lemma *n-mult-left-lower-bound*:
$n(x) * n(y) \leq n(x)$
by (*simp add*: *sup.orderI n-sup-left-absorb-mult*)

lemma *n-mult-right-lower-bound*:
$n(x) * n(y) \leq n(y)$
by (*simp add*: *le-iff-sup n-sup-right-absorb-mult*)

lemma *n-mult-least-upper-bound*:
$n(x) \leq n(y) \wedge n(x) \leq n(z) \longleftrightarrow n(x) \leq n(y) * n(z)$
by (*metis order.trans mult-left-isotone n-mult-commutative n-mult-right-lower-bound n-order*)

lemma *n-mult-left-divisibility*:
$n(x) \leq n(y) \longleftrightarrow (\exists z . n(x) = n(y) * n(z))$
by (*metis n-mult-commutative n-mult-left-lower-bound n-order*)

lemma *n-mult-right-divisibility*:
$n(x) \leq n(y) \longleftrightarrow (\exists z . n(x) = n(z) * n(y))$
by (*simp add*: *n-mult-commutative n-mult-left-divisibility*)

Theorem 15.1

lemma *n-one*:
$n(1) = bot$
by (*metis mult-left-one n-mult-bot n-bot*)

lemma *n-split-equal*:
$x \sqcup n(x * L) * top = x * bot \sqcup n(x * L) * top$
using *n-split order-trans sup.coboundedl sup-same-context zero-right-mult-decreasing* **by** *blast*

lemma *n-split-top*:
$x * top \leq x * bot \sqcup n(x * L) * top$
by (*metis mult-left-isotone n-split vector-bot-closed vector-mult-closed vector-sup-closed vector-top-closed*)

Theorem 15.2

lemma *n-L*:
$n(L) = 1$
by (*metis sup-bot-left order.antisym mult-left-one n-export n-isotone n-mult-commutative n-split-top n-sub-one n-top*)

Theorem 15.5

lemma *n-split-L*:
$x * L = x * bot \sqcup n(x * L) * L$
by (*metis mult-1-right n-L n-L-split*)

lemma *n-n-L*:

$n(n(x) * L) = n(x)$
by (*simp add*: *n-export n-L*)

lemma *n-L-decreasing*:
$n(x) * L \leq x$
by (*metis mult-left-zero n-L-split order-trans sup.orderI zero-right-mult-decreasing mult-assoc n-mult-bot*)

Theorem 15.10

lemma *n-galois*:
$n(x) \leq n(y) \longleftrightarrow n(x) * L \leq y$
by (*metis order.trans mult-left-isotone n-L-decreasing n-isotone n-n-L*)

Theorem 15.6

lemma *split-L*:
$x * L \leq x * bot \sqcup L$
by (*metis sup-commute sup-left-isotone n-galois n-L n-split-L n-sub-one*)

Theorem 15.7

lemma *L-left-zero*:
$L * x = L$
by (*metis order.antisym mult.left-neutral mult-left-zero zero-right-mult-decreasing n-L n-L-decreasing n-mult-bot mult.assoc*)

Theorem 15.8

lemma *n-mult*:
$n(x * n(y) * L) = n(x * y)$
using *n-L-split n-dist-sup sup.absorb2 n-mult-left-upper-bound n-mult-bot n-n-L*
by *auto*

lemma *n-mult-top*:
$n(x * n(y) * top) = n(x * y)$
by (*metis mult-1-right n-mult n-top*)

Theorem 15.4

lemma *n-top-L*:
$n(x * top) = n(x * L)$
by (*metis mult-1-right n-L n-mult-top*)

lemma *n-top-split*:
$x * n(y) * top \leq x * bot \sqcup n(x * y) * top$
by (*metis mult-assoc n-mult n-mult-right-bot n-split-top*)

lemma *n-mult-right-upper-bound*:
$n(x * y) \leq n(z) \longleftrightarrow n(x) \leq n(z) \land x * n(y) * L \leq x * bot \sqcup n(z) * L$
apply (*rule iffI*)
apply (*metis sup-right-isotone order.eq-iff mult-isotone n-L-split n-mult-left-upper-bound order-trans*)
by (*smt* (*verit, ccfv-threshold*) *n-dist-sup n-export sup.absorb-iff2 n-mult n-mult-commutative n-mult-bot n-n-L*)

lemma *n-preserves-equation*:
  $n(y) * x \leq x * n(y) \longleftrightarrow n(y) * x = n(y) * x * n(y)$
  using *eq-refl test-preserves-equation n-mult-idempotent n-sub-one* **by** *auto*

definition *ni* :: $'a \Rightarrow 'a$
  where $ni\ x = n(x) * L$

lemma *ni-bot*:
  $ni(bot) = bot$
  by (*simp add*: *n-bot ni-def*)

lemma *ni-one*:
  $ni(1) = bot$
  by (*simp add*: *n-one ni-def*)

lemma *ni-L*:
  $ni(L) = L$
  by (*simp add*: *n-L ni-def*)

lemma *ni-top*:
  $ni(top) = L$
  by (*simp add*: *n-top ni-def*)

lemma *ni-dist-sup*:
  $ni(x \sqcup y) = ni(x) \sqcup ni(y)$
  by (*simp add*: *mult-right-dist-sup n-dist-sup ni-def*)

lemma *ni-mult-bot*:
  $ni(x) = ni(x * bot)$
  using *n-mult-bot ni-def* **by** *auto*

lemma *ni-split*:
  $x * ni(y) = x * bot \sqcup ni(x * y)$
  using *n-L-split mult-assoc ni-def* **by** *auto*

lemma *ni-decreasing*:
  $ni(x) \leq x$
  by (*simp add*: *n-L-decreasing ni-def*)

lemma *ni-isotone*:
  $x \leq y \Longrightarrow ni(x) \leq ni(y)$
  using *mult-left-isotone n-isotone ni-def* **by** *auto*

lemma *ni-mult-left-upper-bound*:
  $ni(x) \leq ni(x * y)$
  using *mult-left-isotone n-mult-left-upper-bound ni-def* **by** *force*

lemma *ni-idempotent*:

$ni(ni(x)) = ni(x)$
by ($simp\ add$: n-n-L ni-def)

lemma *ni-below-L*:
$ni(x) \leq L$
using *n-L n-galois n-sub-one ni-def* **by** *auto*

lemma *ni-left-zero*:
$ni(x) * y = ni(x)$
by ($simp\ add$: *L-left-zero mult-assoc ni-def*)

lemma *ni-split-L*:
$x * L = x * bot \sqcup ni(x * L)$
using *n-split-L ni-def* **by** *auto*

lemma *ni-top-L*:
$ni(x * top) = ni(x * L)$
by ($simp\ add$: *n-top-L ni-def*)

lemma *ni-galois*:
$ni(x) \leq ni(y) \longleftrightarrow ni(x) \leq y$
by (*metis n-galois n-n-L ni-def*)

lemma *ni-mult*:
$ni(x * ni(y)) = ni(x * y)$
using *mult-assoc n-mult ni-def* **by** *auto*

lemma *ni-n-order*:
$ni(x) \leq ni(y) \longleftrightarrow n(x) \leq n(y)$
using *n-galois ni-def ni-galois* **by** *auto*

lemma *ni-n-equal*:
$ni(x) = ni(y) \longleftrightarrow n(x) = n(y)$
by (*metis n-n-L ni-def*)

lemma *ni-mult-right-upper-bound*:
$ni(x * y) \leq ni(z) \longleftrightarrow ni(x) \leq ni(z) \wedge x * ni(y) \leq x * bot \sqcup ni(z)$
using *mult-assoc n-mult-right-upper-bound ni-def ni-n-order* **by** *auto*

lemma *n-ni*:
$n(ni(x)) = n(x)$
by ($simp\ add$: *n-n-L ni-def*)

lemma *ni-n*:
$ni(n(x)) = bot$
by (*metis n-mult-right-bot ni-mult-bot ni-bot*)

lemma *ni-n-galois*:
$n(x) \leq n(y) \longleftrightarrow ni(x) \leq y$

by (*simp add*: *n-galois ni-def*)

lemma *n-mult-ni*:
$n(x * ni(y)) = n(x * y)$
using *ni-mult ni-n-equal* **by** *auto*

lemma *ni-mult-n*:
$ni(x * n(y)) = ni(x)$
by (*simp add*: *n-mult-n ni-def*)

lemma *ni-export*:
$ni(n(x) * y) = n(x) * ni(y)$
by (*simp add*: *n-mult-right-bot ni-split*)

lemma *ni-mult-top*:
$ni(x * n(y) * top) = ni(x * y)$
by (*simp add*: *n-mult-top ni-def*)

lemma *ni-n-bot*:
$ni(x) = bot \longleftrightarrow n(x) = bot$
using *n-bot ni-n-equal ni-bot* **by** *force*

lemma *ni-n-L*:
$ni(x) = L \longleftrightarrow n(x) = 1$
using *n-L ni-L ni-n-equal* **by** *force*

proposition *n-bot* : $n(bot) = bot$ **oops**
proposition *n-top* : $n(top) = 1$ **oops**
proposition *n-dist-sup* : $n(x \sqcup y) = n(x) \sqcup n(y)$ **oops**
proposition *n-export* : $n(n(x) * y) = n(x) * n(y)$ **oops**
proposition *n-sub-mult-bot*: $n(x) = n(x * bot) * n(x)$ **oops**
proposition *n-L-split* : $x * n(y) * L = x * bot \sqcup n(x * y) * L$ **oops**
proposition *n-split* : $x \le x * bot \sqcup n(x * L) * top$ **oops**

end

typedef (**overloaded**) *'a nImage* = $\{ x::'a::n\text{-}semiring . (\exists y::'a . x = n(y)) \}$
by *auto*

lemma *simp-nImage*[*simp*]:
$\exists y . Rep\text{-}nImage\ x = n(y)$
using *Rep-nImage* **by** *simp*

setup-lifting *type-definition-nImage*

Theorem 15

instantiation *nImage* :: (*n-semiring*) *bounded-idempotent-semiring*
begin

lift-definition *sup-nImage* :: *'a nImage* ⇒ *'a nImage* ⇒ *'a nImage* **is** *sup*
by (*metis n-dist-sup*)

lift-definition *times-nImage* :: *'a nImage* ⇒ *'a nImage* ⇒ *'a nImage* **is** *times*
by (*metis n-export*)

lift-definition *bot-nImage* :: *'a nImage* **is** *bot*
by (*metis n-bot*)

lift-definition *one-nImage* :: *'a nImage* **is** *1*
using *n-L* **by** *auto*

lift-definition *top-nImage* :: *'a nImage* **is** *1*
using *n-L* **by** *auto*

lift-definition *less-eq-nImage* :: *'a nImage* ⇒ *'a nImage* ⇒ *bool* **is** *less-eq* **.**

lift-definition *less-nImage* :: *'a nImage* ⇒ *'a nImage* ⇒ *bool* **is** *less* **.**

instance
apply *intro-classes*
apply (*simp add*: *less-eq-nImage.rep-eq less-le-not-le less-nImage.rep-eq*)
apply (*simp add*: *less-eq-nImage.rep-eq*)
using *less-eq-nImage.rep-eq* **apply** *force*
apply (*simp add*: *less-eq-nImage.rep-eq Rep-nImage-inject*)
apply (*simp add*: *sup-nImage.rep-eq less-eq-nImage.rep-eq*)
apply (*simp add*: *less-eq-nImage.rep-eq sup-nImage.rep-eq*)
apply (*simp add*: *sup-nImage.rep-eq less-eq-nImage.rep-eq*)
apply (*simp add*: *bot-nImage.rep-eq less-eq-nImage.rep-eq*)
apply (*simp add*: *sup-nImage.rep-eq times-nImage.rep-eq less-eq-nImage.rep-eq mult-left-dist-sup*)
apply (*metis* (*mono-tags*, *lifting*) *sup-nImage.rep-eq times-nImage.rep-eq Rep-nImage-inverse mult-right-dist-sup*)
apply (*smt* (*z3*) *times-nImage.rep-eq Rep-nImage-inverse bot-nImage.rep-eq mult-left-zero*)
using *Rep-nImage-inject one-nImage.rep-eq times-nImage.rep-eq* **apply** *fastforce*
apply (*simp add*: *one-nImage.rep-eq times-nImage.rep-eq less-eq-nImage.rep-eq*)
apply (*smt* (*verit*, *del-insts*) *sup-nImage.rep-eq Rep-nImage Rep-nImage-inject mem-Collect-eq n-sub-one sup.absorb2 top-nImage.rep-eq*)
apply (*simp add*: *less-eq-nImage.rep-eq mult.assoc times-nImage.rep-eq*)
using *Rep-nImage-inject mult.assoc times-nImage.rep-eq* **apply** *fastforce*
using *Rep-nImage-inject one-nImage.rep-eq times-nImage.rep-eq* **apply** *fastforce*
apply (*metis* (*mono-tags*, *lifting*) *sup-nImage.rep-eq times-nImage.rep-eq Rep-nImage-inject mult-left-dist-sup*)
by (*smt* (*z3*) *Rep-nImage-inject bot-nImage.rep-eq n-mult-right-bot simp-nImage times-nImage.rep-eq*)

end

Theorem 15

instantiation *nImage* :: (*n-semiring*) *bounded-distrib-lattice*
begin

lift-definition *inf-nImage* :: $'a$ *nImage* $\Rightarrow$ $'a$ *nImage* $\Rightarrow$ $'a$ *nImage* **is** *times*
  by (*metis n-export*)

instance
  apply *intro-classes*
  apply (*metis* (*mono-tags*) *inf-nImage.rep-eq less-eq-nImage.rep-eq n-mult-left-lower-bound simp-nImage*)
  apply (*metis* (*mono-tags*) *inf-nImage.rep-eq less-eq-nImage.rep-eq n-mult-right-lower-bound simp-nImage*)
  apply (*smt* (*z3*) *inf-nImage-def le-iff-sup less-eq-nImage.rep-eq mult-right-dist-sup n-mult-left-absorb-sup simp-nImage times-nImage.rep-eq times-nImage-def*)
  apply *simp*
  by (*smt* (*z3*) *Rep-nImage-inject inf-nImage.rep-eq n-sup-right-dist-mult simp-nImage sup.commute sup-nImage.rep-eq*)

end

class *n-itering* = *bounded-itering* + *n-semiring*
begin

lemma *mult-L-circ*:
  $(x * L)^\circ = 1 \sqcup x * L$
  by (*metis L-left-zero circ-mult mult-assoc*)

lemma *mult-L-circ-mult*:
  $(x * L)^\circ * y = y \sqcup x * L$
  by (*metis L-left-zero mult-L-circ mult-assoc mult-left-one mult-right-dist-sup*)

lemma *circ-L*:
  $L^\circ = L \sqcup 1$
  by (*metis L-left-zero sup-commute circ-left-unfold*)

lemma *circ-n-L*:
  $x^\circ * n(x) * L = x^\circ * bot$
  by (*metis sup-bot-left circ-left-unfold circ-plus-same mult-left-zero n-L-split n-dist-sup n-mult-bot n-one ni-def ni-split*)

lemma *n-circ-left-unfold*:
  $n(x^\circ) = n(x * x^\circ)$
  by (*metis circ-n-L circ-plus-same n-mult n-mult-bot*)

lemma *ni-circ*:
  $ni(x)^\circ = 1 \sqcup ni(x)$
  by (*simp add*: *mult-L-circ ni-def*)

lemma *circ-ni*:
$x^\circ * ni(x) = x^\circ * bot$
using *circ-n-L ni-def mult-assoc* **by** *auto*

lemma *ni-circ-left-unfold*:
$ni(x^\circ) = ni(x * x^\circ)$
by (*simp add*: *ni-def n-circ-left-unfold*)

lemma *n-circ-import*:
$n(y) * x \leq x * n(y) \Longrightarrow n(y) * x^\circ = n(y) * (n(y) * x)^\circ$
by (*simp add*: *circ-import n-mult-idempotent n-sub-one*)

end

class *n-omega-itering* = *left-omega-conway-semiring* + *n-itering* +
assumes *circ-circ*: $x^{\circ\circ} = L \sqcup x^\star$
begin

lemma *L-below-one-circ*:
$L \leq 1^\circ$
by (*metis sup-left-divisibility circ-circ circ-one*)

lemma *circ-below-L-sup-star*:
$x^\circ \leq L \sqcup x^\star$
by (*metis circ-circ circ-increasing*)

lemma *L-sup-circ-sup-star*:
$L \sqcup x^\circ = L \sqcup x^\star$
by (*metis circ-circ circ-star star-circ*)

lemma *circ-one-L*:
$1^\circ = L \sqcup 1$
using *circ-circ circ-one star-one* **by** *auto*

lemma *one-circ-zero*:
$L = 1^\circ * bot$
by (*metis L-left-zero circ-L circ-ni circ-one-L circ-plus-same ni-L*)

lemma *circ-not-simulate*:
$(\forall x\, y\, z\, .\ x * z \leq z * y \longrightarrow x^\circ * z \leq z * y^\circ) \longrightarrow 1 = bot$
by (*metis L-left-zero circ-one-L order.eq-iff mult-left-one mult-left-zero mult-right-sub-dist-sup-left n-L n-bot bot-least*)

lemma *star-circ-L*:
$x^{\star\circ} = L \sqcup x^\star$
by (*simp add*: *circ-circ star-circ*)

lemma *circ-circ-2*:

$x^{\circ\circ} = L \sqcup x^{\circ}$
by (*simp add*: *L-sup-circ-sup-star circ-circ*)

lemma *circ-sup-6*:
$L \sqcup (x \sqcup y)^{\circ} = (x^{\circ} * y^{\circ})^{\circ}$
by (*metis circ-circ-2 sup-assoc sup-commute circ-sup-1 circ-circ-sup circ-decompose-4*)

lemma *circ-sup-7*:
$(x^{\circ} * y^{\circ})^{\circ} = L \sqcup (x \sqcup y)^{\star}$
using *L-sup-circ-sup-star circ-sup-6* **by** *auto*

end

class *n-omega-algebra-2* = *bounded-left-zero-omega-algebra* + *n-semiring* + *Omega* +
assumes *Omega-def*: $x^{\Omega} = n(x^{\omega}) * L \sqcup x^{\star}$
begin

lemma *mult-L-star*:
$(x * L)^{\star} = 1 \sqcup x * L$
by (*simp add*: *L-left-zero transitive-star mult-assoc*)

lemma *mult-L-omega*:
$(x * L)^{\omega} = x * L$
by (*metis L-left-zero omega-slide*)

lemma *mult-L-sup-star*:
$(x * L \sqcup y)^{\star} = y^{\star} \sqcup y^{\star} * x * L$
by (*metis L-left-zero star.mult-zero-sup-circ-2 sup-commute mult-assoc*)

lemma *mult-L-sup-omega*:
$(x * L \sqcup y)^{\omega} = y^{\omega} \sqcup y^{\star} * x * L$
by (*metis L-left-zero mult-bot-add-omega sup-commute mult-assoc*)

lemma *mult-L-sup-circ*:
$(x * L \sqcup y)^{\Omega} = n(y^{\omega}) * L \sqcup y^{\star} \sqcup y^{\star} * x * L$
by (*smt sup-assoc sup-commute Omega-def le-iff-sup mult-L-sup-omega mult-L-sup-star mult-right-dist-sup n-L-decreasing n-dist-sup*)

lemma *circ-sup-n*:
$(x^{\Omega} * y)^{\Omega} * x^{\Omega} = n((x^{\star} * y)^{\omega}) * L \sqcup ((x^{\star} * y)^{\star} * x^{\star} \sqcup (x^{\star} * y)^{\star} * n(x^{\omega}) * L)$
by (*smt L-left-zero sup-assoc sup-commute Omega-def mult-L-sup-circ mult-assoc mult-left-dist-sup mult-right-dist-sup*)

Theorem 20.6

lemma *n-omega-induct*:
$n(y) \leq n(x * y \sqcup z) \Longrightarrow n(y) \leq n(x^{\omega} \sqcup x^{\star} * z)$
by (*smt sup-commute mult-assoc n-dist-sup n-galois n-mult omega-induct*)

lemma *n-Omega-left-unfold*:
  $1 \sqcup x * x^{\Omega} = x^{\Omega}$
proof −
  have $1 \sqcup x * x^{\Omega} = 1 \sqcup x * n(x^{\omega}) * L \sqcup x * x^{\star}$
    by (*simp add*: *Omega-def semiring.distrib-left sup-assoc mult-assoc*)
  also have ... $= n(x * x^{\omega}) * L \sqcup (1 \sqcup x * x^{\star})$
    by (*metis sup-assoc sup-commute sup-bot-left mult-left-dist-sup n-L-split*)
  also have ... $= n(x^{\omega}) * L \sqcup x^{\star}$
    using *omega-unfold star-left-unfold-equal* **by** *auto*
  also have ... $= x^{\Omega}$
    by (*simp add*: *Omega-def*)
  finally show *?thesis*
    .
qed

lemma *n-Omega-circ-sup*:
  $(x \sqcup y)^{\Omega} = (x^{\Omega} * y)^{\Omega} * x^{\Omega}$
proof −
  have $(x^{\Omega} * y)^{\Omega} * x^{\Omega} = n((x^{\star} * y)^{\omega}) * L \sqcup ((x^{\star} * y)^{\star} * x^{\star} \sqcup (x^{\star} * y)^{\star} * n(x^{\omega}) * L)$
    by (*simp add*: *circ-sup-n*)
  also have ... $= n((x^{\star} * y)^{\omega}) * L \sqcup n((x^{\star} * y)^{\star} * x^{\omega}) * L \sqcup (x^{\star} * y)^{\star} * bot \sqcup (x^{\star} * y)^{\star} * x^{\star}$
    using *n-L-split sup.left-commute sup-commute* **by** *auto*
  also have ... $= n((x^{\star} * y)^{\omega} \sqcup (x^{\star} * y)^{\star} * x^{\omega}) * L \sqcup (x^{\star} * y)^{\star} * x^{\star}$
    by (*smt sup-assoc sup-bot-left mult-left-dist-sup mult-right-dist-sup n-dist-sup*)
  also have ... $= (x \sqcup y)^{\Omega}$
    by (*simp add*: *Omega-def omega-decompose star.circ-sup-9*)
  finally show *?thesis*
    ..
qed

lemma *n-Omega-circ-simulate-right-sup*:
  assumes $z * x \leq y * y^{\Omega} * z \sqcup w$
    shows $z * x^{\Omega} \leq y^{\Omega} * (z \sqcup w * x^{\Omega})$
proof −
  have $z * x \leq y * y^{\Omega} * z \sqcup w$
    by (*simp add*: *assms*)
  also have ... $= y * n(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
    using *L-left-zero Omega-def mult-right-dist-sup semiring.distrib-left mult-assoc* **by** *auto*
  finally have *1*: $z * x \leq n(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
    by (*metis sup-assoc sup-commute sup-bot-left mult-assoc mult-left-dist-sup n-L-split omega-unfold*)
  hence $(n(y^{\omega}) * L \sqcup y^{\star} * z \sqcup y^{\star} * w * n(x^{\omega}) * L \sqcup y^{\star} * w * x^{\star}) * x \leq n(y^{\omega}) * L \sqcup y^{\star} * (n(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w) \sqcup y^{\star} * w * n(x^{\omega}) * L \sqcup y^{\star} * w * x^{\star}$
    by (*smt L-left-zero sup-assoc sup-ge1 sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup mult-right-dist-sup star.circ-back-loop-fixpoint*)

also have ... $= n(y^\omega) * L \sqcup y^\star * n(y^\omega) * L \sqcup y^\star * y * y^\star * z \sqcup y^\star * w \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
using *semiring.distrib-left sup-assoc mult-assoc* **by** *auto*
also have ... $= n(y^\omega) * L \sqcup y^\star * n(y^\omega) * L \sqcup y^\star * y * y^\star * z \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*smt* (*verit*, *ccfv-SIG*) *le-supI1 order.refl semiring.add-mono star.circ-back-loop-prefixpoint sup.bounded-iff sup.coboundedI1 sup.mono sup-left-divisibility sup-right-divisibility sup-same-context*)
also have ... $= n(y^\omega) * L \sqcup y^\star * y * y^\star * z \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*smt sup-assoc sup-commute sup-idem mult-assoc mult-left-dist-sup n-L-split star-mult-omega*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*meson mult-left-isotone order-refl semiring.add-left-mono star.circ-mult-upper-bound star.right-plus-below-circ sup-left-isotone*)
finally have *2*: $z * x^\star \leq n(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*smt le-supI1 le-sup-iff sup-ge1 star.circ-loop-fixpoint star-right-induct*)
have $z * x * x^\omega \leq n(y^\omega) * L \sqcup y * y^\star * z * x^\omega \sqcup w * x^\omega$
using *1* **by** (*smt* (*verit*, *del-insts*) *L-left-zero mult-assoc mult-left-isotone mult-right-dist-sup*)
hence $n(z * x * x^\omega) \leq n(y * y^\star * z * x^\omega \sqcup n(y^\omega) * L \sqcup w * x^\omega)$
by (*simp add*: *n-isotone sup-commute*)
hence $n(z * x^\omega) \leq n(y^\omega \sqcup y^\star * w * x^\omega)$
by (*smt* (*verit*, *del-insts*) *sup-assoc sup-commute left-plus-omega le-iff-sup mult-assoc mult-left-dist-sup n-L-decreasing n-omega-induct omega-unfold star.left-plus-circ star-mult-omega*)
hence $n(z * x^\omega) * L \leq n(y^\omega) * L \sqcup y^\star * w * n(x^\omega) * L$
by (*metis n-dist-sup n-galois n-mult n-n-L*)
hence $z * n(x^\omega) * L \leq z * bot \sqcup n(y^\omega) * L \sqcup y^\star * w * n(x^\omega) * L$
using *n-L-split semiring.add-left-mono sup-assoc* **by** *auto*
also have ... $\leq n(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * n(x^\omega) * L$
by (*smt* (*z3*) *order.trans mult-1-left mult-right-sub-dist-sup-left semiring.add-right-mono star-left-unfold-equal sup-commute zero-right-mult-decreasing*)
finally have $z * n(x^\omega) * L \leq n(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * n(x^\omega) * L \sqcup y^\star * w * x^\star$
using *le-supI1* **by** *blast*
thus *?thesis*
using *2* **by** (*smt L-left-zero Omega-def sup-assoc le-iff-sup mult-assoc mult-left-dist-sup mult-right-dist-sup*)
qed

lemma *n-Omega-circ-simulate-left-sup*:
assumes $x * z \leq z * y^\Omega \sqcup w$
shows $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$
proof –
have $x * (z * n(y^\omega) * L \sqcup z * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star) = x * z * n(y^\omega) * L \sqcup x * z * y^\star \sqcup n(x^\omega) * L \sqcup x * x^\star * w * n(y^\omega) * L \sqcup x * x^\star * w * y^\star$

by (*smt sup-assoc sup-commute mult-assoc mult-left-dist-sup n-L-split omega-unfold*)
also have ... $\leq (z * n(y^\omega) * L \sqcup z * y^\star \sqcup w) * n(y^\omega) * L \sqcup (z * n(y^\omega) * L \sqcup z * y^\star \sqcup w) * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt assms Omega-def sup-assoc sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup mult-right-dist-sup star.circ-loop-fixpoint*)
also have ... $= z * n(y^\omega) * L \sqcup z * y^\star * n(y^\omega) * L \sqcup w * n(y^\omega) * L \sqcup z * y^\star \sqcup w * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt L-left-zero sup-assoc sup-commute sup-idem mult-assoc mult-right-dist-sup star.circ-transitive-equal*)
also have ... $= z * n(y^\omega) * L \sqcup w * n(y^\omega) * L \sqcup z * y^\star \sqcup w * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt sup-assoc sup-commute sup-idem le-iff-sup mult-assoc n-L-split star-mult-omega zero-right-mult-decreasing*)
finally have $x * (z * n(y^\omega) * L \sqcup z * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star) \leq z * n(y^\omega) * L \sqcup z * y^\star \sqcup n(x^\omega) * L \sqcup x^\star * w * n(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt sup-assoc sup-commute sup-idem mult-assoc star.circ-loop-fixpoint*)
thus $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$
by (*smt* (*verit, del-insts*) *L-left-zero Omega-def sup-assoc le-supI1 le-sup-iff sup-ge1 mult-assoc mult-left-dist-sup mult-right-dist-sup star.circ-back-loop-fixpoint star-left-induct*)
qed

end

Theorem 2.6 and Theorem 19

sublocale *n-omega-algebra-2* < *nL-omega*: *itering* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*simp add*: *n-Omega-circ-sup*)
apply (*smt L-left-zero sup-assoc sup-commute sup-bot-left Omega-def mult-assoc mult-left-dist-sup mult-right-dist-sup n-L-split omega-slide star.circ-mult*)
apply (*simp add*: *n-Omega-circ-simulate-right-sup*)
using *n-Omega-circ-simulate-left-sup* **by** *auto*

sublocale *n-omega-algebra-2* < *nL-omega*: *n-omega-itering* **where** *circ* = *Omega*
apply *unfold-locales*
by (*smt Omega-def sup-assoc sup-commute le-iff-sup mult-L-sup-star mult-left-one n-L-split n-top ni-below-L ni-def star-involutive star-mult-omega star-omega-top zero-right-mult-decreasing*)

sublocale *n-omega-algebra-2* < *nL-omega*: *left-zero-kleene-conway-semiring* **where** *circ* = *Omega* ..

sublocale *n-omega-algebra-2* < *nL-star*: *left-omega-conway-semiring* **where** *circ* = *star* ..

context *n-omega-algebra-2*
begin

lemma *circ-sup-8*:
$n((x^\star * y)^\star * x^\omega) * L \leq (x^\star * y)^\Omega * x^\Omega$
by (*metis sup-ge1 nL-omega.circ-sup-4 Omega-def mult-left-isotone n-isotone omega-sum-unfold-3 order-trans*)

lemma *n-split-omega-omega*:
$x^\omega \leq x^\omega * bot \sqcup n(x^\omega) * top$
by (*metis n-split n-top-L omega-vector*)

Theorem 20.1

lemma *n-below-n-star*:
$n(x) \leq n(x^\star)$
by (*simp add*: *n-isotone star.circ-increasing*)

Theorem 20.2

lemma *n-star-below-n-omega*:
$n(x^\star) \leq n(x^\omega)$
by (*metis n-mult-left-upper-bound star-mult-omega*)

lemma *n-below-n-omega*:
$n(x) \leq n(x^\omega)$
using *order.trans n-below-n-star n-star-below-n-omega* **by** *blast*

Theorem 20.4

lemma *star-n-L*:
$x^\star * n(x) * L = x^\star * bot$
by (*metis sup-bot-left mult-left-zero n-L-split n-dist-sup n-mult-bot n-one ni-def ni-split star-left-unfold-equal star-plus*)

lemma *star-L-split*:
assumes $y \leq z$
and $x * z * L \leq x * bot \sqcup z * L$
shows $x^\star * y * L \leq x^\star * bot \sqcup z * L$
proof −
have $x * (x^\star * bot \sqcup z * L) \leq x^\star * bot \sqcup x * z * L$
by (*metis sup-bot-right order.eq-iff mult-assoc mult-left-dist-sup star.circ-loop-fixpoint*)
also have ... $\leq x^\star * bot \sqcup x * bot \sqcup z * L$
using *assms*(*2*) *semiring.add-left-mono sup-assoc* **by** *auto*
also have ... $= x^\star * bot \sqcup z * L$
using *mult-left-isotone star.circ-increasing sup.absorb-iff2 sup-commute* **by** *auto*
finally have $y * L \sqcup x * (x^\star * bot \sqcup z * L) \leq x^\star * bot \sqcup z * L$
by (*metis assms*(*1*) *le-sup-iff sup-ge2 mult-left-isotone order-trans*)
thus *?thesis*
by (*simp add*: *star-left-induct mult-assoc*)
qed

lemma *star-L-split-same*:
$x * y * L \leq x * bot \sqcup y * L \Longrightarrow x^\star * y * L = x^\star * bot \sqcup y * L$
apply (*rule order.antisym*)
apply (*simp add*: *star-L-split*)
by (*metis bot-least le-supI mult-isotone nL-star.star-below-circ star.circ-loop-fixpoint sup.cobounded2 mult-assoc*)

lemma *star-n-L-split-equal*:
$n(x * y) \leq n(y) \Longrightarrow x^\star * n(y) * L = x^\star * bot \sqcup n(y) * L$
by (*simp add*: *n-mult-right-upper-bound star-L-split-same*)

lemma *n-star-mult*:
$n(x * y) \leq n(y) \Longrightarrow n(x^\star * y) = n(x^\star) \sqcup n(y)$
by (*metis n-dist-sup n-mult n-mult-bot n-n-L star-n-L-split-equal*)

Theorem 20.3

lemma *n-omega-mult*:
$n(x^\omega * y) = n(x^\omega)$
by (*simp add*: *n-isotone n-mult-left-upper-bound omega-sub-vector order.eq-iff*)

lemma *n-star-left-unfold*:
$n(x^\star) = n(x * x^\star)$
by (*metis n-mult n-mult-bot star.circ-plus-same star-n-L*)

lemma *ni-star-below-ni-omega*:
$ni(x^\star) \leq ni(x^\omega)$
by (*simp add*: *ni-n-order n-star-below-n-omega*)

lemma *ni-below-ni-omega*:
$ni(x) \leq ni(x^\omega)$
by (*simp add*: *ni-n-order n-below-n-omega*)

lemma *ni-star*:
$ni(x)^\star = 1 \sqcup ni(x)$
by (*simp add*: *mult-L-star ni-def*)

lemma *ni-omega*:
$ni(x)^\omega = ni(x)$
using *mult-L-omega ni-def* **by** *auto*

lemma *ni-omega-induct*:
$ni(y) \leq ni(x * y \sqcup z) \Longrightarrow ni(y) \leq ni(x^\omega \sqcup x^\star * z)$
using *n-omega-induct ni-n-order* **by** *blast*

lemma *star-ni*:
$x^\star * ni(x) = x^\star * bot$
using *ni-def mult-assoc star-n-L* **by** *auto*

lemma *star-ni-split-equal*:

$ni(x * y) \le ni(y) \Longrightarrow x^\star * ni(y) = x^\star * bot \sqcup ni(y)$
using *ni-def ni-mult-right-upper-bound mult-assoc star-L-split-same* **by** *auto*

lemma *ni-star-mult*:
$ni(x * y) \le ni(y) \Longrightarrow ni(x^\star * y) = ni(x^\star) \sqcup ni(y)$
using *mult-right-dist-sup ni-def ni-n-order n-star-mult* **by** *auto*

lemma *ni-omega-mult*:
$ni(x^\omega * y) = ni(x^\omega)$
by (*simp add*: *ni-def n-omega-mult*)

lemma *ni-star-left-unfold*:
$ni(x^\star) = ni(x * x^\star)$
by (*simp add*: *ni-def n-star-left-unfold*)

lemma *n-star-import*:
assumes $n(y) * x \le x * n(y)$
shows $n(y) * x^\star = n(y) * (n(y) * x)^\star$
proof (*rule order.antisym*)
have $n(y) * (n(y) * x)^\star * x \le n(y) * (n(y) * x)^\star$
by (*smt assms mult-assoc mult-right-dist-sup mult-right-sub-dist-sup-left n-mult-idempotent n-preserves-equation star.circ-back-loop-fixpoint*)
thus $n(y) * x^\star \le n(y) * (n(y) * x)^\star$
using *assms eq-refl n-mult-idempotent n-sub-one star.circ-import* **by** *auto*
next
show $n(y) * (n(y) * x)^\star \le n(y) * x^\star$
by (*simp add*: *assms n-mult-idempotent n-sub-one star.circ-import*)
qed

lemma *n-omega-export*:
$n(y) * x \le x * n(y) \Longrightarrow n(y) * x^\omega = (n(y) * x)^\omega$
apply (*rule order.antisym*)
apply (*simp add*: *n-preserves-equation omega-simulation*)
by (*metis mult-right-isotone mult-1-right n-sub-one omega-isotone omega-slide*)

lemma *n-omega-import*:
$n(y) * x \le x * n(y) \Longrightarrow n(y) * x^\omega = n(y) * (n(y) * x)^\omega$
by (*simp add*: *n-mult-idempotent omega-import*)

Theorem 20.5

lemma *star-n-omega-top*:
$x^\star * n(x^\omega) * top = x^\star * bot \sqcup n(x^\omega) * top$
by (*smt* (*verit, del-insts*) *le-supI le-sup-iff sup-right-divisibility order.antisym mult-assoc nL-star.circ-mult-omega nL-star.star-zero-below-circ-mult n-top-split star.circ-loop-fixpoint*)

proposition *n-star-induct-sup*: $n(z \sqcup x * y) \le n(y) \Longrightarrow n(x^\star * z) \le n(y)$ **oops**

end

end

13 Boolean N-Semirings

theory *N-Semirings-Boolean*

imports *N-Semirings*

begin

class *an* =
  fixes *an* :: $'a \Rightarrow 'a$

class *an-semiring* = *bounded-idempotent-left-zero-semiring* + *L* + *n* + *an* + *uminus* +
  assumes *an-complement*: $an(x) \sqcup n(x) = 1$
  assumes *an-dist-sup* : $an(x \sqcup y) = an(x) * an(y)$
  assumes *an-export* : $an(an(x) * y) = n(x) \sqcup an(y)$
  assumes *an-mult-zero* : $an(x) = an(x * bot)$
  assumes *an-L-split* : $x * n(y) * L = x * bot \sqcup n(x * y) * L$
  assumes *an-split* : $an(x * L) * x \leq x * bot$
  assumes *an-uminus* : $-x = an(x * L)$
begin

Theorem 21

lemma *n-an-def*:
  $n(x) = an(an(x) * L)$
  by (*metis an-dist-sup an-export an-split bot-least mult-right-isotone semiring.add-nonneg-eq-0-iff sup.orderE top-greatest vector-bot-closed*)

Theorem 21

lemma *an-complement-bot*:
  $an(x) * n(x) = bot$
  by (*metis an-dist-sup an-split bot-least le-iff-sup mult-left-zero sup-commute n-an-def*)

Theorem 21

lemma *an-n-def*:
  $an(x) = n(an(x) * L)$
  by (*smt* (*verit, ccfv-threshold*) *an-complement-bot an-complement mult.right-neutral mult-left-dist-sup mult-right-dist-sup sup-commute n-an-def*)

lemma *an-case-split-left*:
  $an(z) * x \leq y \wedge n(z) * x \leq y \longleftrightarrow x \leq y$
  by (*metis le-sup-iff an-complement mult-left-one mult-right-dist-sup*)

lemma *an-case-split-right*:
  $x * an(z) \leq y \wedge x * n(z) \leq y \longleftrightarrow x \leq y$

by (*metis le-sup-iff an-complement mult-1-right mult-left-dist-sup*)

lemma *split-sub*:
$x * y \leq z \sqcup x * top$
by (*simp add*: *le-supI2 mult-right-isotone*)

Theorem 21

subclass *n-semiring*
apply *unfold-locales*
apply (*metis an-dist-sup an-split mult-left-zero sup.absorb2 sup-bot-left sup-commute n-an-def*)
apply (*metis sup-left-top an-complement an-dist-sup an-export mult-assoc n-an-def*)
apply (*metis an-dist-sup an-export mult-assoc n-an-def*)
apply (*metis an-dist-sup an-export an-n-def mult-right-dist-sup n-an-def*)
apply (*metis sup-idem an-dist-sup an-mult-zero n-an-def*)
apply (*simp add*: *an-L-split*)
by (*meson an-case-split-left an-split le-supI1 split-sub*)

lemma *n-complement-bot*:
$n(x) * an(x) = bot$
by (*metis an-complement-bot an-n-def n-an-def*)

lemma *an-bot*:
$an(bot) = 1$
by (*metis sup-bot-right an-complement n-bot*)

lemma *an-one*:
$an(1) = 1$
by (*metis sup-bot-right an-complement n-one*)

lemma *an-L*:
$an(L) = bot$
using *an-one n-one n-an-def* **by** *auto*

lemma *an-top*:
$an(top) = bot$
by (*metis mult-left-one n-complement-bot n-top*)

lemma *an-export-n*:
$an(n(x) * y) = an(x) \sqcup an(y)$
by (*metis an-export an-n-def n-an-def*)

lemma *n-export-an*:
$n(an(x) * y) = an(x) * n(y)$
by (*metis an-n-def n-export*)

lemma *n-an-mult-commutative*:
$n(x) * an(y) = an(y) * n(x)$

by (*metis sup-commute an-dist-sup n-an-def*)

lemma *an-mult-commutative*:
$an(x) * an(y) = an(y) * an(x)$
by (*metis sup-commute an-dist-sup*)

lemma *an-mult-idempotent*:
$an(x) * an(x) = an(x)$
by (*metis sup-idem an-dist-sup*)

lemma *an-sub-one*:
$an(x) \leq 1$
using *an-complement sup.cobounded1* **by** *fastforce*

Theorem 21

lemma *an-antitone*:
$x \leq y \Longrightarrow an(y) \leq an(x)$
by (*metis an-n-def an-dist-sup n-order sup.absorb1*)

lemma *an-mult-left-upper-bound*:
$an(x * y) \leq an(x)$
by (*metis an-antitone an-mult-zero mult-right-isotone bot-least*)

lemma *an-mult-right-zero*:
$an(x) * bot = bot$
by (*metis an-n-def n-mult-right-bot*)

lemma *n-mult-an*:
$n(x * an(y)) = n(x)$
by (*metis an-n-def n-mult-n*)

lemma *an-mult-n*:
$an(x * n(y)) = an(x)$
by (*metis an-n-def n-an-def n-mult-n*)

lemma *an-mult-an*:
$an(x * an(y)) = an(x)$
by (*metis an-mult-n an-n-def*)

lemma *an-mult-left-absorb-sup*:
$an(x) * (an(x) \sqcup an(y)) = an(x)$
by (*metis an-n-def n-mult-left-absorb-sup*)

lemma *an-mult-right-absorb-sup*:
$(an(x) \sqcup an(y)) * an(y) = an(y)$
by (*metis an-n-def n-mult-right-absorb-sup*)

lemma *an-sup-left-absorb-mult*:
$an(x) \sqcup an(x) * an(y) = an(x)$

using *an-case-split-right sup-absorb1* **by** *blast*

lemma *an-sup-right-absorb-mult*:
$an(x) * an(y) \sqcup an(y) = an(y)$
using *an-case-split-left sup-absorb2* **by** *blast*

lemma *an-sup-left-dist-mult*:
$an(x) \sqcup an(y) * an(z) = (an(x) \sqcup an(y)) * (an(x) \sqcup an(z))$
by (*metis an-n-def n-sup-left-dist-mult*)

lemma *an-sup-right-dist-mult*:
$an(x) * an(y) \sqcup an(z) = (an(x) \sqcup an(z)) * (an(y) \sqcup an(z))$
by (*simp add*: *an-sup-left-dist-mult sup-commute*)

lemma *an-n-order*:
$an(x) \leq an(y) \longleftrightarrow n(y) \leq n(x)$
by (*smt* (*verit*) *an-n-def an-dist-sup le-iff-sup n-dist-sup n-mult-right-absorb-sup sup.orderE n-an-def*)

lemma *an-order*:
$an(x) \leq an(y) \longleftrightarrow an(x) * an(y) = an(x)$
by (*metis an-n-def n-order*)

lemma *an-mult-left-lower-bound*:
$an(x) * an(y) \leq an(x)$
using *an-case-split-right* **by** *blast*

lemma *an-mult-right-lower-bound*:
$an(x) * an(y) \leq an(y)$
by (*simp add*: *an-sup-right-absorb-mult le-iff-sup*)

lemma *an-n-mult-left-lower-bound*:
$an(x) * n(y) \leq an(x)$
using *an-case-split-right* **by** *blast*

lemma *an-n-mult-right-lower-bound*:
$an(x) * n(y) \leq n(y)$
using *an-case-split-left* **by** *auto*

lemma *n-an-mult-left-lower-bound*:
$n(x) * an(y) \leq n(x)$
using *an-case-split-right* **by** *auto*

lemma *n-an-mult-right-lower-bound*:
$n(x) * an(y) \leq an(y)$
using *an-case-split-left* **by** *blast*

lemma *an-mult-least-upper-bound*:
$an(x) \leq an(y) \land an(x) \leq an(z) \longleftrightarrow an(x) \leq an(y) * an(z)$

by (*metis an-mult-idempotent an-mult-left-lower-bound an-mult-right-lower-bound order.trans mult-isotone*)

lemma *an-mult-left-divisibility*:
$an(x) \leq an(y) \longleftrightarrow (\exists z\ .\ an(x) = an(y) * an(z))$
by (*metis an-mult-commutative an-mult-left-lower-bound an-order*)

lemma *an-mult-right-divisibility*:
$an(x) \leq an(y) \longleftrightarrow (\exists z\ .\ an(x) = an(z) * an(y))$
by (*simp add*: *an-mult-commutative an-mult-left-divisibility*)

lemma *an-split-top*:
$an(x * L) * x * top \leq x * bot$
by (*metis an-split mult-assoc mult-left-isotone mult-left-zero*)

lemma *an-n-L*:
$an(n(x) * L) = an(x)$
using *an-n-def n-an-def* **by** *auto*

lemma *an-galois*:
$an(y) \leq an(x) \longleftrightarrow n(x) * L \leq y$
by (*simp add*: *an-n-order n-galois*)

lemma *an-mult*:
$an(x * n(y) * L) = an(x * y)$
by (*metis an-n-L n-mult*)

lemma *n-mult-top*:
$an(x * n(y) * top) = an(x * y)$
by (*metis an-n-L n-mult-top*)

lemma *an-n-equal*:
$an(x) = an(y) \longleftrightarrow n(x) = n(y)$
by (*metis an-n-L n-an-def*)

lemma *an-top-L*:
$an(x * top) = an(x * L)$
by (*simp add*: *an-n-equal n-top-L*)

lemma *an-case-split-left-equal*:
$an(z) * x = an(z) * y \Longrightarrow n(z) * x = n(z) * y \Longrightarrow x = y$
using *an-complement case-split-left-equal* **by** *blast*

lemma *an-case-split-right-equal*:
$x * an(z) = y * an(z) \Longrightarrow x * n(z) = y * n(z) \Longrightarrow x = y$
using *an-complement case-split-right-equal* **by** *blast*

lemma *an-equal-complement*:
$n(x) \sqcup an(y) = 1 \wedge n(x) * an(y) = bot \longleftrightarrow an(x) = an(y)$

by (*metis sup-commute an-complement an-dist-sup mult-left-one mult-right-dist-sup n-complement-bot*)

lemma *n-equal-complement*:
$n(x) \sqcup an(y) = 1 \wedge n(x) * an(y) = bot \longleftrightarrow n(x) = n(y)$
by (*simp add*: *an-equal-complement an-n-equal*)

lemma *an-shunting*:
$an(z) * x \leq y \longleftrightarrow x \leq y \sqcup n(z) * top$
apply (*rule iffI*)
apply (*meson an-case-split-left le-supI1 split-sub*)
by (*metis sup-bot-right an-case-split-left an-complement-bot mult-assoc mult-left-dist-sup mult-left-zero mult-right-isotone order-refl order-trans*)

lemma *an-shunting-an*:
$an(z) * an(x) \leq an(y) \longleftrightarrow an(x) \leq n(z) \sqcup an(y)$
apply (*rule iffI*)
apply (*smt sup-ge1 sup-ge2 an-case-split-left n-an-mult-left-lower-bound order-trans*)
by (*metis sup-bot-left sup-ge2 an-case-split-left an-complement-bot mult-left-dist-sup mult-right-isotone order-trans*)

lemma *an-L-zero*:
$an(x * L) * x = an(x * L) * x * bot$
by (*metis an-complement-bot n-split-equal sup-monoid.add-0-right vector-bot-closed mult-assoc n-export-an*)

lemma *n-plus-complement-intro-n*:
$n(x) \sqcup an(x) * n(y) = n(x) \sqcup n(y)$
by (*metis sup-commute an-complement an-n-def mult-1-right n-sup-right-dist-mult n-an-mult-commutative*)

lemma *n-plus-complement-intro-an*:
$n(x) \sqcup an(x) * an(y) = n(x) \sqcup an(y)$
by (*metis an-n-def n-plus-complement-intro-n*)

lemma *an-plus-complement-intro-n*:
$an(x) \sqcup n(x) * n(y) = an(x) \sqcup n(y)$
by (*metis an-n-def n-an-def n-plus-complement-intro-n*)

lemma *an-plus-complement-intro-an*:
$an(x) \sqcup n(x) * an(y) = an(x) \sqcup an(y)$
by (*metis an-n-def an-plus-complement-intro-n*)

lemma *n-mult-complement-intro-n*:
$n(x) * (an(x) \sqcup n(y)) = n(x) * n(y)$
by (*simp add*: *mult-left-dist-sup n-complement-bot*)

lemma *n-mult-complement-intro-an*:

$n(x) * (an(x) \sqcup an(y)) = n(x) * an(y)$
by (*simp add*: *semiring.distrib-left n-complement-bot*)

lemma *an-mult-complement-intro-n*:
$an(x) * (n(x) \sqcup n(y)) = an(x) * n(y)$
by (*simp add*: *an-complement-bot mult-left-dist-sup*)

lemma *an-mult-complement-intro-an*:
$an(x) * (n(x) \sqcup an(y)) = an(x) * an(y)$
by (*simp add*: *an-complement-bot semiring.distrib-left*)

lemma *an-preserves-equation*:
$an(y) * x \leq x * an(y) \longleftrightarrow an(y) * x = an(y) * x * an(y)$
by (*metis an-n-def n-preserves-equation*)

lemma *wnf-lemma-1*:
$(n(p * L) * n(q * L) \sqcup an(p * L) * an(r * L)) * n(p * L) = n(p * L) * n(q * L)$
by (*smt sup-commute an-n-def n-sup-left-absorb-mult n-sup-right-dist-mult n-export n-mult-commutative n-mult-complement-intro-n*)

lemma *wnf-lemma-2*:
$(n(p * L) * n(q * L) \sqcup an(r * L) * an(q * L)) * n(q * L) = n(p * L) * n(q * L)$
by (*metis an-mult-commutative n-mult-commutative wnf-lemma-1*)

lemma *wnf-lemma-3*:
$(n(p * L) * n(r * L) \sqcup an(p * L) * an(q * L)) * an(p * L) = an(p * L) * an(q * L)$
by (*metis an-n-def sup-commute wnf-lemma-1 n-an-def*)

lemma *wnf-lemma-4*:
$(n(r * L) * n(q * L) \sqcup an(p * L) * an(q * L)) * an(q * L) = an(p * L) * an(q * L)$
by (*metis an-mult-commutative n-mult-commutative wnf-lemma-3*)

lemma *wnf-lemma-5*:
$n(p \sqcup q) * (n(q) * x \sqcup an(q) * y) = n(q) * x \sqcup an(q) * n(p) * y$
by (*smt sup-bot-right mult-assoc mult-left-dist-sup n-an-mult-commutative n-complement-bot n-dist-sup n-mult-right-absorb-sup*)

definition *ani* :: $'a \Rightarrow 'a$
where *ani* $x \equiv an(x) * L$

lemma *ani-bot*:
$ani(bot) = L$
using *an-bot ani-def* **by** *auto*

lemma *ani-one*:
$ani(1) = L$
using *an-one ani-def* **by** *auto*

lemma *ani-L*:
$ani(L) = bot$
by (*simp add*: *an-L ani-def*)

lemma *ani-top*:
$ani(top) = bot$
by (*simp add*: *an-top ani-def*)

lemma *ani-complement*:
$ani(x) \sqcup ni(x) = L$
by (*metis an-complement ani-def mult-right-dist-sup n-top ni-def ni-top*)

lemma *ani-mult-zero*:
$ani(x) = ani(x * bot)$
using *ani-def an-mult-zero* **by** *auto*

lemma *ani-antitone*:
$y \leq x \Longrightarrow ani(x) \leq ani(y)$
by (*simp add*: *an-antitone ani-def mult-left-isotone*)

lemma *ani-mult-left-upper-bound*:
$ani(x * y) \leq ani(x)$
by (*simp add*: *an-mult-left-upper-bound ani-def mult-left-isotone*)

lemma *ani-involutive*:
$ani(ani(x)) = ni(x)$
by (*simp add*: *ani-def ni-def n-an-def*)

lemma *ani-below-L*:
$ani(x) \leq L$
using *an-case-split-left ani-def* **by** *auto*

lemma *ani-left-zero*:
$ani(x) * y = ani(x)$
by (*simp add*: *ani-def L-left-zero mult-assoc*)

lemma *ani-top-L*:
$ani(x * top) = ani(x * L)$
by (*simp add*: *an-top-L ani-def*)

lemma *ani-ni-order*:
$ani(x) \leq ani(y) \longleftrightarrow ni(y) \leq ni(x)$
by (*metis an-n-L ani-antitone ani-def ani-involutive ni-def*)

lemma *ani-ni-equal*:
$ani(x) = ani(y) \longleftrightarrow ni(x) = ni(y)$
by (*metis ani-ni-order order.antisym order-refl*)

lemma *ni-ani*:
$ni(ani(x)) = ani(x)$
using *an-n-def ani-def ni-def* **by** *auto*

lemma *ani-ni*:
$ani(ni(x)) = ani(x)$
by (*simp add*: *an-n-L ani-def ni-def*)

lemma *ani-mult*:
$ani(x * ni(y)) = ani(x * y)$
using *ani-ni-equal ni-mult* **by** *blast*

lemma *ani-an-order*:
$ani(x) \leq ani(y) \longleftrightarrow an(x) \leq an(y)$
using *an-galois ani-ni-order ni-def ni-galois* **by** *auto*

lemma *ani-an-equal*:
$ani(x) = ani(y) \longleftrightarrow an(x) = an(y)$
by (*metis an-n-def ani-def*)

lemma *n-mult-ani*:
$n(x) * ani(x) = bot$
by (*metis an-L ani-L ani-def mult-assoc n-complement-bot*)

lemma *an-mult-ni*:
$an(x) * ni(x) = bot$
by (*metis an-n-def ani-def n-an-def n-mult-ani ni-def*)

lemma *n-mult-ni*:
$n(x) * ni(x) = ni(x)$
by (*metis n-export n-order ni-def ni-export order-refl*)

lemma *an-mult-ani*:
$an(x) * ani(x) = ani(x)$
by (*metis an-n-def ani-def n-mult-ni ni-def*)

lemma *ani-ni-meet*:
$x \leq ani(y) \Longrightarrow x \leq ni(y) \Longrightarrow x = bot$
by (*metis an-case-split-left an-mult-ni bot-unique mult-right-isotone n-mult-ani*)

lemma *ani-galois*:
$ani(x) \leq y \longleftrightarrow ni(x \sqcup y) = L$
apply (*rule iffI*)
apply (*smt* (*z3*) *an-L an-mult-commutative an-mult-right-zero ani-def an-dist-sup ni-L ni-n-equal sup.absorb1 mult-assoc n-an-def n-complement-bot*)
by (*metis an-L an-galois an-mult-ni an-n-def an-shunting-an ani-def an-dist-sup an-export idempotent-bot-closed n-bot transitive-bot-closed*)

lemma *an-ani*:

$an(ani(x)) = n(x)$
by (*simp add*: *ani-def n-an-def*)

lemma *n-ani*:
$n(ani(x)) = an(x)$
using *an-n-def ani-def* **by** *auto*

lemma *an-ni*:
$an(ni(x)) = an(x)$
by (*simp add*: *an-n-L ni-def*)

lemma *ani-an*:
$ani(an(x)) = L$
by (*metis an-mult-right-zero an-mult-zero an-bot ani-def mult-left-one*)

lemma *ani-n*:
$ani(n(x)) = L$
by (*simp add*: *ani-an n-an-def*)

lemma *ni-an*:
$ni(an(x)) = bot$
using *an-L ani-an ani-def ni-n-bot n-an-def* **by** *force*

lemma *ani-mult-n*:
$ani(x * n(y)) = ani(x)$
by (*simp add*: *an-mult-n ani-def*)

lemma *ani-mult-an*:
$ani(x * an(y)) = ani(x)$
by (*simp add*: *an-mult-an ani-def*)

lemma *ani-export-n*:
$ani(n(x) * y) = ani(x) \sqcup ani(y)$
by (*simp add*: *an-export-n ani-def mult-right-dist-sup*)

lemma *ani-export-an*:
$ani(an(x) * y) = ni(x) \sqcup ani(y)$
by (*simp add*: *ani-def an-export ni-def semiring.distrib-right*)

lemma *ni-export-an*:
$ni(an(x) * y) = an(x) * ni(y)$
by (*simp add*: *an-mult-right-zero ni-split*)

lemma *ani-mult-top*:
$ani(x * n(y) * top) = ani(x * y)$
using *ani-def n-mult-top* **by** *auto*

lemma *ani-an-bot*:
$ani(x) = bot \longleftrightarrow an(x) = bot$

using *an-L ani-L ani-an-equal* **by** *force*

lemma *ani-an-L*:
$ani(x) = L \longleftrightarrow an(x) = 1$
using *an-bot ani-an-equal ani-bot* **by** *force*

Theorem 21

subclass *tests*
apply *unfold-locales*
apply (*simp add*: *mult-assoc*)
apply (*simp add*: *an-mult-commutative an-uminus*)
apply (*smt an-sup-left-dist-mult an-export-n an-n-L an-uminus n-an-def n-complement-bot n-export*)
apply (*metis an-dist-sup an-n-def an-uminus n-an-def*)
using *an-complement-bot an-uminus n-an-def* **apply** *fastforce*
apply (*simp add*: *an-bot an-uminus*)
using *an-export-n an-mult an-uminus n-an-def* **apply** *fastforce*
using *an-order an-uminus* **apply** *force*
by (*simp add*: *less-le-not-le*)

end

class *an-itering* = *n-itering* + *an-semiring* + *while* +
assumes *while-circ-def*: $p \star y = (p * y)^{\circ} * -p$
begin

subclass *test-itering*
apply *unfold-locales*
by (*rule while-circ-def*)

lemma *an-circ-left-unfold*:
$an(x^{\circ}) = an(x * x^{\circ})$
by (*metis an-dist-sup an-one circ-left-unfold mult-left-one*)

lemma *an-circ-x-n-circ*:
$an(x^{\circ}) * x * n(x^{\circ}) \leq x * bot$
by (*metis an-circ-left-unfold an-mult an-split mult-assoc n-mult-right-bot*)

lemma *an-circ-invariant*:
$an(x^{\circ}) * x \leq x * an(x^{\circ})$
proof −
have *1*: $an(x^{\circ}) * x * an(x^{\circ}) \leq x * an(x^{\circ})$
by (*metis an-case-split-left mult-assoc order-refl*)
have $an(x^{\circ}) * x * n(x^{\circ}) \leq x * an(x^{\circ})$
by (*metis an-circ-x-n-circ order-trans mult-right-isotone bot-least*)
thus *?thesis*
using *1 an-case-split-right* **by** *blast*
qed

lemma *ani-circ*:
$ani(x)^\circ = 1 \sqcup ani(x)$
by (*simp add*: *ani-def mult-L-circ*)

lemma *ani-circ-left-unfold*:
$ani(x^\circ) = ani(x * x^\circ)$
by (*simp add*: *an-circ-left-unfold ani-def*)

lemma *an-circ-import*:
$an(y) * x \leq x * an(y) \Longrightarrow an(y) * x^\circ = an(y) * (an(y) * x)^\circ$
by (*metis an-n-def n-circ-import*)

lemma *preserves-L*:
preserves L $(-p)$
using *L-left-zero preserves-equation-test mult-assoc* **by** *force*

end

class *an-omega-algebra* = *n-omega-algebra-2* + *an-semiring* + *while* +
assumes *while-Omega-def*: $p \star y = (p * y)^\Omega * -p$
begin

lemma *an-split-omega-omega*:
$an(x^\omega) * x^\omega \leq x^\omega * bot$
by (*meson an-antitone an-split mult-left-isotone omega-sub-vector order-trans*)

lemma *an-omega-below-an-star*:
$an(x^\omega) \leq an(x^\star)$
by (*simp add*: *an-n-order n-star-below-n-omega*)

lemma *an-omega-below-an*:
$an(x^\omega) \leq an(x)$
by (*simp add*: *an-n-order n-below-n-omega*)

lemma *an-omega-induct*:
$an(x * y \sqcup z) \leq an(y) \Longrightarrow an(x^\omega \sqcup x^\star * z) \leq an(y)$
by (*simp add*: *an-n-order n-omega-induct*)

lemma *an-star-mult*:
$an(y) \leq an(x * y) \Longrightarrow an(x^\star * y) = an(x^\star) * an(y)$
by (*metis an-dist-sup an-n-L an-n-order n-dist-sup n-star-mult*)

lemma *an-omega-mult*:
$an(x^\omega * y) = an(x^\omega)$
by (*simp add*: *an-n-equal n-omega-mult*)

lemma *an-star-left-unfold*:
$an(x^\star) = an(x * x^\star)$
by (*simp add*: *an-n-equal n-star-left-unfold*)

lemma *an-star-x-n-star*:
  $an(x^\star) * x * n(x^\star) \le x * bot$
  by (*metis an-n-L an-split n-mult n-mult-right-bot n-star-left-unfold mult-assoc*)

lemma *an-star-invariant*:
  $an(x^\star) * x \le x * an(x^\star)$
proof −
  have *1*: $an(x^\star) * x * an(x^\star) \le x * an(x^\star)$
    using *an-case-split-left mult-assoc* **by** *auto*
  have $an(x^\star) * x * n(x^\star) \le x * an(x^\star)$
    by (*metis an-star-x-n-star order-trans mult-right-isotone bot-least*)
  thus *?thesis*
    using *1 an-case-split-right* **by** *auto*
qed

lemma *n-an-star-unfold-invariant*:
  $n(an(x^\star) * x^\omega) \le an(x) * n(x * an(x^\star) * x^\omega)$
proof −
  have $n(an(x^\star) * x^\omega) \le an(x)$
    using *an-star-left-unfold an-case-split-right an-mult-left-upper-bound n-export-an* **by** *fastforce*
  thus *?thesis*
    by (*smt an-star-invariant le-iff-sup mult-assoc mult-right-dist-sup n-isotone n-order omega-unfold*)
qed

lemma *ani-omega-below-ani-star*:
  $ani(x^\omega) \le ani(x^\star)$
  by (*simp add*: *an-omega-below-an-star ani-an-order*)

lemma *ani-omega-below-ani*:
  $ani(x^\omega) \le ani(x)$
  by (*simp add*: *an-omega-below-an ani-an-order*)

lemma *ani-star*:
  $ani(x)^\star = 1 \sqcup ani(x)$
  by (*simp add*: *ani-def mult-L-star*)

lemma *ani-omega*:
  $ani(x)^\omega = ani(x) * L$
  by (*simp add*: *L-left-zero ani-def mult-L-omega mult-assoc*)

lemma *ani-omega-induct*:
  $ani(x * y \sqcup z) \le ani(y) \Longrightarrow ani(x^\omega \sqcup x^\star * z) \le ani(y)$
  by (*simp add*: *an-omega-induct ani-an-order*)

lemma *ani-omega-mult*:
  $ani(x^\omega * y) = ani(x^\omega)$

by (*simp add*: *an-omega-mult ani-def*)

lemma *ani-star-left-unfold*:
$ani(x^\star) = ani(x * x^\star)$
by (*simp add*: *an-star-left-unfold ani-def*)

lemma *an-star-import*:
$an(y) * x \leq x * an(y) \Longrightarrow an(y) * x^\star = an(y) * (an(y) * x)^\star$
by (*metis an-n-def n-star-import*)

lemma *an-omega-export*:
$an(y) * x \leq x * an(y) \Longrightarrow an(y) * x^\omega = (an(y) * x)^\omega$
by (*metis an-n-def n-omega-export*)

lemma *an-omega-import*:
$an(y) * x \leq x * an(y) \Longrightarrow an(y) * x^\omega = an(y) * (an(y) * x)^\omega$
by (*simp add*: *an-mult-idempotent omega-import*)

end

Theorem 22

sublocale *an-omega-algebra* < *nL-omega*: *an-itering* **where** *circ* = *Omega*
apply *unfold-locales*
by (*rule while-Omega-def*)

context *an-omega-algebra*
begin

lemma *preserves-star*:
nL-omega.preserves x $(-p) \Longrightarrow$ *nL-omega.preserves* $(x^\star)$ $(-p)$
by (*simp add*: *nL-omega.preserves-def star.circ-simulate*)

end

end

14 Modal N-Semirings

theory *N-Semirings-Modal*

imports *N-Semirings-Boolean*

begin

class *n-diamond-semiring* = *n-semiring* + *diamond* +
assumes *ndiamond-def*: $|x{>}y = n(x * y * L)$
begin

lemma *diamond-x-bot*:

$|x\rangle bot = n(x)$
using *n-mult-bot ndiamond-def mult-assoc* **by** *auto*

lemma *diamond-x-1*:
$|x\rangle 1 = n(x * L)$
by (*simp add*: *ndiamond-def*)

lemma *diamond-x-L*:
$|x\rangle L = n(x * L)$
by (*simp add*: *L-left-zero ndiamond-def mult-assoc*)

lemma *diamond-x-top*:
$|x\rangle top = n(x * L)$
by (*metis mult-assoc n-top-L ndiamond-def top-mult-top*)

lemma *diamond-x-n*:
$|x\rangle n(y) = n(x * y)$
by (*simp add*: *n-mult ndiamond-def*)

lemma *diamond-bot-y*:
$|bot\rangle y = bot$
by (*simp add*: *n-bot ndiamond-def*)

lemma *diamond-1-y*:
$|1\rangle y = n(y * L)$
by (*simp add*: *ndiamond-def*)

lemma *diamond-1-n*:
$|1\rangle n(y) = n(y)$
by (*simp add*: *diamond-1-y n-n-L*)

lemma *diamond-L-y*:
$|L\rangle y = 1$
by (*simp add*: *L-left-zero n-L ndiamond-def*)

lemma *diamond-top-y*:
$|top\rangle y = 1$
by (*metis sup-left-top sup-right-top diamond-L-y mult-right-dist-sup n-dist-sup n-top ndiamond-def*)

lemma *diamond-n-y*:
$|n(x)\rangle y = n(x) * n(y * L)$
by (*simp add*: *n-export ndiamond-def mult-assoc*)

lemma *diamond-n-bot*:
$|n(x)\rangle bot = bot$
by (*simp add*: *n-bot n-mult-right-bot ndiamond-def*)

lemma *diamond-n-1*:

$|n(x)>1 = n(x)$
using *diamond-1-n diamond-1-y diamond-x-1* **by** *auto*

lemma *diamond-n-n*:
$|n(x)>n(y) = n(x) * n(y)$
by (*simp add*: *diamond-x-n n-export*)

lemma *diamond-n-n-same*:
$|n(x)>n(x) = n(x)$
by (*simp add*: *diamond-n-n n-mult-idempotent*)

Theorem 23.1

lemma *diamond-left-dist-sup*:
$|x \sqcup y>z = |x>z \sqcup |y>z$
by (*simp add*: *mult-right-dist-sup n-dist-sup ndiamond-def*)

Theorem 23.2

lemma *diamond-right-dist-sup*:
$|x>(y \sqcup z) = |x>y \sqcup |x>z$
by (*simp add*: *mult-left-dist-sup n-dist-sup ndiamond-def semiring.distrib-right*)

Theorem 23.3

lemma *diamond-associative*:
$|x * y>z = |x>(y * z)$
by (*simp add*: *ndiamond-def mult-assoc*)

Theorem 23.3

lemma *diamond-left-mult*:
$|x * y>z = |x>|y>z$
using *n-mult-ni ndiamond-def ni-def mult-assoc* **by** *auto*

lemma *diamond-right-mult*:
$|x>(y * z) = |x>|y>z$
using *diamond-associative diamond-left-mult* **by** *force*

lemma *diamond-n-export*:
$|n(x) * y>z = n(x) * |y>z$
by (*simp add*: *n-export ndiamond-def mult-assoc*)

lemma *diamond-diamond-export*:
$||x>y>z = |x>y * |z>1$
using *diamond-n-y ndiamond-def* **by** *auto*

lemma *diamond-left-isotone*:
$x \leq y \Longrightarrow |x>z \leq |y>z$
by (*metis diamond-left-dist-sup le-iff-sup*)

lemma *diamond-right-isotone*:
$y \leq z \Longrightarrow |x>y \leq |x>z$

by (*metis diamond-right-dist-sup le-iff-sup*)

lemma *diamond-isotone*:
$w \le y \Longrightarrow x \le z \Longrightarrow |w{>}x \le |y{>}z$
by (*meson diamond-left-isotone diamond-right-isotone order-trans*)

definition *ndiamond-L* :: $'a \Rightarrow 'a \Rightarrow 'a$ (‹‖ - » -› [*50*,*90*] *95*)
where $\|x»y \equiv n(x * y) * L$

lemma *ndiamond-to-L*:
$\|x»y = |x{>}n(y) * L$
by (*simp add*: *diamond-x-n ndiamond-L-def*)

lemma *ndiamond-from-L*:
$|x{>}y = n(\|x»(y * L))$
by (*simp add*: *n-n-L ndiamond-def mult-assoc ndiamond-L-def*)

lemma *diamond-L-ni*:
$\|x»y = ni(x * y)$
by (*simp add*: *ni-def ndiamond-L-def*)

lemma *diamond-L-associative*:
$\|x * y»z = \|x»(y * z)$
by (*simp add*: *diamond-L-ni mult-assoc*)

lemma *diamond-L-left-mult*:
$\|x * y»z = \|x»\|y»z$
using *diamond-L-associative diamond-L-ni ni-mult* **by** *auto*

lemma *diamond-L-right-mult*:
$\|x»(y * z) = \|x»\|y»z$
using *diamond-L-associative diamond-L-left-mult* **by** *auto*

lemma *diamond-L-left-dist-sup*:
$\|x \sqcup y»z = \|x»z \sqcup \|y»z$
by (*simp add*: *diamond-L-ni mult-right-dist-sup ni-dist-sup*)

lemma *diamond-L-x-ni*:
$\|x»ni(y) = ni(x * y)$
using *n-mult-ni ni-def ndiamond-L-def* **by** *auto*

lemma *diamond-L-left-isotone*:
$x \le y \Longrightarrow \|x»z \le \|y»z$
using *mult-left-isotone ni-def ni-isotone ndiamond-L-def* **by** *auto*

lemma *diamond-L-right-isotone*:
$y \le z \Longrightarrow \|x»y \le \|x»z$
using *mult-right-isotone ni-def ni-isotone ndiamond-L-def* **by** *auto*

lemma *diamond-L-isotone*:
  $w \le y \Longrightarrow x \le z \Longrightarrow \| w » x \le \| y » z$
  using *diamond-L-ni mult-isotone ni-isotone* **by** *force*

end

class *n-box-semiring* = *n-diamond-semiring* + *an-semiring* + *box* +
  assumes *nbox-def*: $|x]y = an(x * an(y * L) * L)$
begin

Theorem 23.8

lemma *box-diamond*:
  $|x]y = an(|x>an(y * L) * L)$
  by (*simp add*: *an-n-L nbox-def ndiamond-def*)

Theorem 23.4

lemma *diamond-box*:
  $|x>y = an(|x]an(y * L) * L)$
  using *n-an-def n-mult nbox-def ndiamond-def mult-assoc* **by** *force*

lemma *box-x-bot*:
  $|x]bot = an(x * L)$
  by (*simp add*: *an-bot nbox-def*)

lemma *box-x-1*:
  $|x]1 = an(x)$
  using *an-L an-mult-an nbox-def mult-assoc* **by** *auto*

lemma *box-x-L*:
  $|x]L = an(x)$
  using *box-x-1 L-left-zero nbox-def* **by** *auto*

lemma *box-x-top*:
  $|x]top = an(x)$
  by (*metis box-diamond box-x-1 box-x-bot diamond-top-y*)

lemma *box-x-n*:
  $|x]n(y) = an(x * an(y) * L)$
  by (*simp add*: *an-n-L nbox-def*)

lemma *box-x-an*:
  $|x]an(y) = an(x * y)$
  using *an-mult n-an-def nbox-def* **by** *auto*

lemma *box-bot-y*:
  $|bot]y = 1$
  by (*simp add*: *an-bot nbox-def*)

lemma *box-1-y*:

$|1]y = n(y * L)$
by (*simp add*: *n-an-def nbox-def*)

lemma *box-1-n*:
$|1]n(y) = n(y)$
using *box-1-y diamond-1-n diamond-1-y* **by** *auto*

lemma *box-1-an*:
$|1]an(y) = an(y)$
by (*simp add*: *box-x-an*)

lemma *box-L-y*:
$|L]y = bot$
by (*simp add*: *L-left-zero an-L nbox-def*)

lemma *box-top-y*:
$|top]y = bot$
by (*simp add*: *box-diamond an-L diamond-top-y*)

lemma *box-n-y*:
$|n(x)]y = an(x) \sqcup n(y * L)$
using *an-export-n n-an-def nbox-def mult-assoc* **by** *auto*

lemma *box-an-y*:
$|an(x)]y = n(x) \sqcup n(y * L)$
by (*metis an-n-def box-n-y n-an-def*)

lemma *box-n-bot*:
$|n(x)]bot = an(x)$
by (*simp add*: *box-x-bot an-n-L*)

lemma *box-an-bot*:
$|an(x)]bot = n(x)$
by (*simp add*: *box-x-bot n-an-def*)

lemma *box-n-1*:
$|n(x)]1 = 1$
using *box-x-1 ani-an-L ani-n* **by** *auto*

lemma *box-an-1*:
$|an(x)]1 = 1$
using *box-x-1 ani-an ani-an-L* **by** *fastforce*

lemma *box-n-n*:
$|n(x)]n(y) = an(x) \sqcup n(y)$
using *box-1-n box-1-y box-n-y* **by** *auto*

lemma *box-an-n*:
$|an(x)]n(y) = n(x) \sqcup n(y)$

using *box-x-n an-dist-sup n-an-def n-dist-sup* **by** *auto*

lemma *box-n-an*:
$|n(x)]an(y) = an(x) \sqcup an(y)$
by (*simp add*: *box-x-an an-export-n*)

lemma *box-an-an*:
$|an(x)]an(y) = n(x) \sqcup an(y)$
by (*simp add*: *box-x-an an-export*)

lemma *box-n-n-same*:
$|n(x)]n(x) = 1$
by (*simp add*: *box-n-n an-complement*)

lemma *box-an-an-same*:
$|an(x)]an(x) = 1$
using *box-an-bot an-bot an-complement-bot nbox-def* **by** *auto*

Theorem 23.5

lemma *box-left-dist-sup*:
$|x \sqcup y]z = |x]z * |y]z$
using *an-dist-sup nbox-def semiring.distrib-right* **by** *auto*

lemma *box-right-dist-sup*:
$|x](y \sqcup z) = an(x * an(y * L) * an(z * L) * L)$
by (*simp add*: *an-dist-sup mult-right-dist-sup nbox-def mult-assoc*)

lemma *box-associative*:
$|x * y]z = an(x * y * an(z * L) * L)$
by (*simp add*: *nbox-def*)

Theorem 23.7

lemma *box-left-mult*:
$|x * y]z = |x]|y]z$
using *box-x-an nbox-def mult-assoc* **by** *auto*

lemma *box-right-mult*:
$|x](y * z) = an(x * an(y * z * L) * L)$
by (*simp add*: *nbox-def*)

Theorem 23.6

lemma *box-right-mult-n-n*:
$|x](n(y) * n(z)) = |x]n(y) * |x]n(z)$
by (*smt an-dist-sup an-export-n an-n-L mult-assoc mult-left-dist-sup mult-right-dist-sup nbox-def*)

lemma *box-right-mult-an-n*:
$|x](an(y) * n(z)) = |x]an(y) * |x]n(z)$
by (*metis an-n-def box-right-mult-n-n*)

lemma *box-right-mult-n-an*:
  $|x](n(y) * an(z)) = |x]n(y) * |x]an(z)$
  by (*simp add*: *box-right-mult-an-n box-x-an box-x-n an-mult-commutative n-an-mult-commutative*)

lemma *box-right-mult-an-an*:
  $|x](an(y) * an(z)) = |x]an(y) * |x]an(z)$
  by (*metis an-dist-sup box-x-an mult-left-dist-sup*)

lemma *box-n-export*:
  $|n(x) * y]z = an(x) \sqcup |y]z$
  using *box-left-mult box-n-an nbox-def* **by** *auto*

lemma *box-an-export*:
  $|an(x) * y]z = n(x) \sqcup |y]z$
  using *box-an-an box-left-mult nbox-def* **by** *auto*

lemma *box-left-antitone*:
  $y \le x \Longrightarrow |x]z \le |y]z$
  by (*smt an-mult-commutative an-order box-diamond box-left-dist-sup le-iff-sup*)

lemma *box-right-isotone*:
  $y \le z \Longrightarrow |x]y \le |x]z$
  by (*metis an-antitone mult-left-isotone mult-right-isotone nbox-def*)

lemma *box-antitone-isotone*:
  $y \le w \Longrightarrow x \le z \Longrightarrow |w]x \le |y]z$
  by (*meson box-left-antitone box-right-isotone order.trans*)

definition *nbox-L* :: $'a \Rightarrow 'a \Rightarrow 'a$ (‹‖ - ⟧ -› [*50*,*90*] *95*)
  where $\|x]\!]y \equiv an(x * an(y) * L) * L$

lemma *nbox-to-L*:
  $\|x]\!]y = |x]n(y) * L$
  by (*simp add*: *box-x-n nbox-L-def*)

lemma *nbox-from-L*:
  $|x]y = n(\|x]\!](y * L))$
  using *an-n-def nbox-def nbox-L-def* **by** *auto*

lemma *diamond-x-an*:
  $|x>an(y) = n(x * an(y) * L)$
  by (*simp add*: *ndiamond-def*)

lemma *diamond-1-an*:
  $|1>an(y) = an(y)$
  using *box-1-an box-1-y diamond-1-y* **by** *auto*

lemma *diamond-an-y*:
$|an(x){>}y = an(x) * n(y * L)$
by (*simp add*: *n-export-an ndiamond-def mult-assoc*)

lemma *diamond-an-bot*:
$|an(x){>}bot = bot$
by (*simp add*: *an-mult-right-zero n-bot ndiamond-def*)

lemma *diamond-an-1*:
$|an(x){>}1 = an(x)$
using *an-n-def diamond-x-1* **by** *auto*

lemma *diamond-an-n*:
$|an(x){>}n(y) = an(x) * n(y)$
by (*simp add*: *diamond-x-n n-export-an*)

lemma *diamond-n-an*:
$|n(x){>}an(y) = n(x) * an(y)$
using *an-n-def diamond-n-y* **by** *auto*

lemma *diamond-an-an*:
$|an(x){>}an(y) = an(x) * an(y)$
using *diamond-an-y an-n-def* **by** *auto*

lemma *diamond-an-an-same*:
$|an(x){>}an(x) = an(x)$
by (*simp add*: *diamond-an-an an-mult-idempotent*)

lemma *diamond-an-export*:
$|an(x) * y{>}z = an(x) * |y{>}z$
using *diamond-an-an diamond-box diamond-left-mult* **by** *auto*

lemma *box-ani*:
$|x]y = an(x * ani(y * L))$
by (*simp add*: *ani-def nbox-def mult-assoc*)

lemma *box-x-n-ani*:
$|x]n(y) = an(x * ani(y))$
by (*simp add*: *box-x-n ani-def mult-assoc*)

lemma *box-L-ani*:
$\|x]\!]y = ani(x * ani(y))$
using *box-x-n-ani ani-def nbox-to-L* **by** *auto*

lemma *box-L-left-mult*:
$\|x * y]\!]z = \|x]\!]\|y]\!]z$
using *an-mult n-an-def mult-assoc nbox-L-def* **by** *auto*

lemma *diamond-x-an-ani*:

$|x{>}an(y) = n(x * ani(y))$
by (*simp add*: *ani-def ndiamond-def mult-assoc*)

lemma *box-L-left-antitone*:
$y \le x \Longrightarrow \|x]\!]z \le \|y]\!]z$
by (*simp add*: *box-L-ani ani-antitone mult-left-isotone*)

lemma *box-L-right-isotone*:
$y \le z \Longrightarrow \|x]\!]y \le \|x]\!]z$
using *ani-antitone ani-def mult-right-isotone mult-assoc nbox-L-def* **by** *auto*

lemma *box-L-antitone-isotone*:
$y \le w \Longrightarrow x \le z \Longrightarrow \|w]\!]x \le \|y]\!]z$
using *ani-antitone ani-def mult-isotone mult-assoc nbox-L-def* **by** *force*

end

class *n-box-omega-algebra* = *n-box-semiring* + *an-omega-algebra*
begin

lemma *diamond-omega*:
$|x^\omega{>}y = |x^\omega{>}z$
by (*simp add*: *n-omega-mult ndiamond-def mult-assoc*)

lemma *box-omega*:
$|x^\omega]y = |x^\omega]z$
by (*metis box-diamond diamond-omega*)

lemma *an-box-omega-induct*:
$|x]an(y) * n(z * L) \le an(y) \Longrightarrow |x^\omega \sqcup x^\star]z \le an(y)$
by (*smt an-dist-sup an-omega-induct an-omega-mult box-left-dist-sup box-x-an mult-assoc n-an-def nbox-def*)

lemma *n-box-omega-induct*:
$|x]n(y) * n(z * L) \le n(y) \Longrightarrow |x^\omega \sqcup x^\star]z \le n(y)$
by (*simp add*: *an-box-omega-induct n-an-def*)

lemma *an-box-omega-induct-an*:
$|x]an(y) * an(z) \le an(y) \Longrightarrow |x^\omega \sqcup x^\star]an(z) \le an(y)$
using *an-box-omega-induct an-n-def* **by** *auto*

Theorem 23.13

lemma *n-box-omega-induct-n*:
$|x]n(y) * n(z) \le n(y) \Longrightarrow |x^\omega \sqcup x^\star]n(z) \le n(y)$
using *an-box-omega-induct-an n-an-def* **by** *force*

lemma *n-diamond-omega-induct*:
$n(y) \le |x{>}n(y) \sqcup n(z * L) \Longrightarrow n(y) \le |x^\omega \sqcup x^\star{>}z$
using *diamond-x-n mult-right-dist-sup n-dist-sup n-omega-induct n-omega-mult*

ndiamond-def mult-assoc **by** *force*

lemma *an-diamond-omega-induct*:
$an(y) \le |x{>}an(y) \sqcup n(z * L) \Longrightarrow an(y) \le |x^\omega \sqcup x^\star{>}z$
by (*metis n-diamond-omega-induct an-n-def*)

Theorem 23.9

lemma *n-diamond-omega-induct-n*:
$n(y) \le |x{>}n(y) \sqcup n(z) \Longrightarrow n(y) \le |x^\omega \sqcup x^\star{>}n(z)$
using *box-1-n box-1-y n-diamond-omega-induct* **by** *auto*

lemma *an-diamond-omega-induct-an*:
$an(y) \le |x{>}an(y) \sqcup an(z) \Longrightarrow an(y) \le |x^\omega \sqcup x^\star{>}an(z)$
using *an-diamond-omega-induct an-n-def* **by** *auto*

lemma *box-segerberg-an*:
$|x^\omega \sqcup x^\star]an(y) = an(y) * |x^\omega \sqcup x^\star](n(y) \sqcup |x]an(y))$
proof (*rule order.antisym*)
have $|x^\omega \sqcup x^\star]an(y) \le |x^\omega \sqcup x^\star]|x]an(y)$
by (*smt box-left-dist-sup box-left-mult box-omega sup-right-isotone box-left-antitone mult-right-dist-sup star.right-plus-below-circ*)
hence $|x^\omega \sqcup x^\star]an(y) \le |x^\omega \sqcup x^\star](n(y) \sqcup |x]an(y))$
using *box-right-isotone order-lesseq-imp sup.cobounded2* **by** *blast*
thus $|x^\omega \sqcup x^\star]an(y) \le an(y) * |x^\omega \sqcup x^\star](n(y) \sqcup |x]an(y))$
by (*metis le-sup-iff box-1-an box-left-antitone order-refl star-left-unfold-equal an-mult-least-upper-bound nbox-def*)
next
have $an(y) * |x](n(y) \sqcup |x^\omega \sqcup x^\star]an(y)) * (n(y) \sqcup |x]an(y)) = |x](\ |x^\omega \sqcup x^\star]an(y) * an(y)) * an(y)$
by (*smt sup-bot-left an-export an-mult-commutative box-right-mult-an-an mult-assoc mult-right-dist-sup n-complement-bot nbox-def*)
hence *1*: $an(y) * |x](n(y) \sqcup |x^\omega \sqcup x^\star]an(y)) * (n(y) \sqcup |x]an(y)) \le n(y) \sqcup |x^\omega \sqcup x^\star]an(y)$
by (*smt sup-assoc sup-commute sup-ge2 box-1-an box-left-dist-sup box-left-mult mult-left-dist-sup omega-unfold star-left-unfold-equal star.circ-plus-one*)
have $n(y) * |x](n(y) \sqcup |x^\omega \sqcup x^\star]an(y)) * (n(y) \sqcup |x]an(y)) \le n(y) \sqcup |x^\omega \sqcup x^\star]an(y)$
by (*smt sup-ge1 an-n-def mult-left-isotone n-an-mult-left-lower-bound n-mult-left-absorb-sup nbox-def order-trans*)
thus $an(y) * |x^\omega \sqcup x^\star](n(y) \sqcup |x]an(y)) \le |x^\omega \sqcup x^\star]an(y)$
using *1* **by** (*smt an-case-split-left an-shunting-an mult-assoc n-box-omega-induct-n n-dist-sup nbox-def nbox-from-L*)
qed

Theorem 23.16

lemma *box-segerberg-n*:
$|x^\omega \sqcup x^\star]n(y) = n(y) * |x^\omega \sqcup x^\star](an(y) \sqcup |x]n(y))$
using *box-segerberg-an an-n-def n-an-def* **by** *force*

lemma *diamond-segerberg-an*:
$|x^\omega \sqcup x^\star{>}an(y) = an(y) \sqcup |x^\omega \sqcup x^\star{>}(n(y) * |x{>}an(y))$
by (*smt an-export an-n-L box-diamond box-segerberg-an diamond-box mult-assoc n-an-def*)

Theorem 23.12

lemma *diamond-segerberg-n*:
$|x^\omega \sqcup x^\star{>}n(y) = n(y) \sqcup |x^\omega \sqcup x^\star{>}(an(y) * |x{>}n(y))$
using *diamond-segerberg-an an-n-L n-an-def* **by** *auto*

Theorem 23.11

lemma *diamond-star-unfold-n*:
$|x^\star{>}n(y) = n(y) \sqcup |an(y) * x{>}|x^\star{>}n(y)$
proof −
have $|x^\star{>}n(y) = n(y) \sqcup n(y) * |x * x^\star{>}n(y) \sqcup |an(y) * x * x^\star{>}n(y)$
by (*smt sup-assoc sup-commute sup-bot-right an-complement an-complement-bot diamond-an-n diamond-left-dist-sup diamond-n-export diamond-n-n-same mult-assoc mult-left-one mult-right-dist-sup star-left-unfold-equal*)
thus *?thesis*
by (*metis diamond-left-mult diamond-x-n n-sup-left-absorb-mult*)
qed

lemma *diamond-star-unfold-an*:
$|x^\star{>}an(y) = an(y) \sqcup |n(y) * x{>}|x^\star{>}an(y)$
by (*metis an-n-def diamond-star-unfold-n n-an-def*)

Theorem 23.15

lemma *box-star-unfold-n*:
$|x^\star]n(y) = n(y) * |n(y) * x]|x^\star]n(y)$
by (*smt an-export an-n-L box-diamond diamond-box diamond-star-unfold-an n-an-def n-export*)

lemma *box-star-unfold-an*:
$|x^\star]an(y) = an(y) * |an(y) * x]|x^\star]an(y)$
by (*metis an-n-def box-star-unfold-n*)

Theorem 23.10

lemma *diamond-omega-unfold-n*:
$|x^\omega \sqcup x^\star{>}n(y) = n(y) \sqcup |an(y) * x{>}|x^\omega \sqcup x^\star{>}n(y)$
by (*smt sup-assoc sup-commute diamond-an-export diamond-left-dist-sup diamond-right-dist-sup diamond-star-unfold-n diamond-x-n n-omega-mult n-plus-complement-intro-n omega-unfold*)

lemma *diamond-omega-unfold-an*:
$|x^\omega \sqcup x^\star{>}an(y) = an(y) \sqcup |n(y) * x{>}|x^\omega \sqcup x^\star{>}an(y)$
by (*metis an-n-def diamond-omega-unfold-n n-an-def*)

Theorem 23.14

lemma *box-omega-unfold-n*:
$|x^\omega \sqcup x^\star]n(y) = n(y) * |n(y) * x]|x^\omega \sqcup x^\star]n(y)$
by (*smt an-export an-n-L box-diamond diamond-box diamond-omega-unfold-an n-an-def n-export*)

lemma *box-omega-unfold-an*:
$|x^\omega \sqcup x^\star]an(y) = an(y) * |an(y) * x]|x^\omega \sqcup x^\star]an(y)$
by (*metis an-n-def box-omega-unfold-n*)

lemma *box-cut-iteration-an*:
$|x^\omega \sqcup x^\star]an(y) = |(an(y) * x)^\omega \sqcup (an(y) * x)^\star]an(y)$
apply (*rule order.antisym*)
apply (*meson semiring.add-mono an-case-split-left box-left-antitone omega-isotone order-refl star.circ-isotone*)
by (*smt (z3) an-box-omega-induct-an an-mult-commutative box-omega-unfold-an nbox-def order-refl*)

lemma *box-cut-iteration-n*:
$|x^\omega \sqcup x^\star]n(y) = |(n(y) * x)^\omega \sqcup (n(y) * x)^\star]n(y)$
using *box-cut-iteration-an n-an-def* **by** *auto*

lemma *diamond-cut-iteration-an*:
$|x^\omega \sqcup x^\star>an(y) = |(n(y) * x)^\omega \sqcup (n(y) * x)^\star>an(y)$
using *box-cut-iteration-n diamond-box n-an-def* **by** *auto*

lemma *diamond-cut-iteration-n*:
$|x^\omega \sqcup x^\star>n(y) = |(an(y) * x)^\omega \sqcup (an(y) * x)^\star>n(y)$
using *box-cut-iteration-an an-n-L diamond-box* **by** *auto*

lemma *ni-diamond-omega-induct*:
$ni(y) \le \|x\text{»}ni(y) \sqcup ni(z) \Longrightarrow ni(y) \le \|x^\omega \sqcup x^\star\text{»}z$
by (*metis diamond-L-left-dist-sup diamond-L-x-ni diamond-L-ni ni-dist-sup ni-omega-induct ni-omega-mult*)

lemma *ani-diamond-omega-induct*:
$ani(y) \le \|x\text{»}ani(y) \sqcup ni(z) \Longrightarrow ani(y) \le \|x^\omega \sqcup x^\star\text{»}z$
by (*metis ni-ani ni-diamond-omega-induct*)

lemma *n-diamond-omega-L*:
$|n(x^\omega) * L>y = |x^\omega>y$
using *L-left-zero mult-1-right n-L n-export n-omega-mult ndiamond-def mult-assoc* **by** *auto*

lemma *n-diamond-loop*:
$|x^\Omega>y = |x^\omega \sqcup x^\star>y$
by (*metis Omega-def diamond-left-dist-sup n-diamond-omega-L*)

Theorem 24.1

lemma *cut-iteration-loop*:

$|x^\Omega>n(y) = |(an(y) * x)^\Omega>n(y)$
using *diamond-cut-iteration-n n-diamond-loop* **by** *auto*

lemma *cut-iteration-while-loop*:
$|x^\Omega>n(y) = |(an(y) * x)^\Omega * n(y)>n(y)$
using *cut-iteration-loop diamond-left-mult diamond-n-n-same* **by** *auto*

Theorem 24.1

lemma *cut-iteration-while-loop-2*:
$|x^\Omega>n(y) = |an(y) \star x>n(y)$
by (*metis cut-iteration-while-loop an-uminus n-an-def while-Omega-def*)

lemma *modal-while*:
assumes $-q * -p * L \le x * -p * L \wedge -p \le -q \sqcup -r$
shows $-p \le |n((-q * x)^\omega) * L \sqcup (-q * x)^\star * --q>(-r)$
proof −
have *1*: $--q * -p \le |-q * x>(-p) \sqcup --q * -r$
using *assms mult-right-isotone sup.coboundedI2*
tests-dual.sup-complement-intro **by** *auto*
have $-q * -p = n(-q * -q * -p * L)$
using *an-uminus n-export-an mult-assoc mult-1-right n-L*
tests-dual.sup-idempotent **by** *auto*
also have ... $\le n(-q * x * -p * L)$
by (*metis assms n-isotone mult-right-isotone mult-assoc*)
also have ... $\le |-q * x>(-p) \sqcup --q * -r$
by (*simp add*: *ndiamond-def*)
finally have $-p \le |-q * x>(-p) \sqcup --q * -r$
using *1* **by** (*smt sup-assoc le-iff-sup tests-dual.inf-cases sub-comm*)
thus *?thesis*
by (*smt L-left-zero an-diamond-omega-induct-an an-uminus*
diamond-left-dist-sup mult-assoc n-n-L n-omega-mult ndiamond-def
sub-mult-closed)
qed

lemma *modal-while-loop*:
$-q * -p * L \le x * -p * L \wedge -p \le -q \sqcup -r \Longrightarrow -p \le |(-q * x)^\Omega *$
$--q>(-r)$
by (*metis L-left-zero Omega-def modal-while mult-assoc mult-right-dist-sup*)

Theorem 24.2

lemma *modal-while-loop-2*:
$-q * -p * L \le x * -p * L \wedge -p \le -q \sqcup -r \Longrightarrow -p \le |-q \star x>(-r)$
by (*simp add*: *while-Omega-def modal-while-loop*)

lemma *modal-while-2*:
assumes $-p * L \le x * -p * L$
shows $-p \le |n((-q * x)^\omega) * L \sqcup (-q * x)^\star * --q>(--q)$
proof −
have $-p \le |-q * x>(-p) \sqcup --q$

by (*smt* (*verit*, *del-insts*) *assms an-uminus tests-dual.double-negation n-an-def n-isotone ndiamond-def diamond-an-export sup-assoc sup-commute le-iff-sup tests-dual.inf-complement-intro*)
thus *?thesis*
by (*smt L-left-zero an-diamond-omega-induct-an an-uminus diamond-left-dist-sup mult-assoc tests-dual.sup-idempotent n-n-L n-omega-mult ndiamond-def*)
qed

end

class *n-modal-omega-algebra* = *n-box-omega-algebra* +
assumes *n-star-induct*: $n(x * y) \leq n(y) \longrightarrow n(x^\star * y) \leq n(y)$
begin

lemma *n-star-induct-sup*:
$n(z \sqcup x * y) \leq n(y) \Longrightarrow n(x^\star * z) \leq n(y)$
by (*metis an-dist-sup an-mult-least-upper-bound an-n-order n-mult-right-upper-bound n-star-induct star-L-split*)

lemma *n-star-induct-star*:
$n(x * y) \leq n(y) \Longrightarrow n(x^\star) \leq n(y)$
using *n-star-induct n-star-mult* **by** *auto*

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

lemma *n-star-bot*:
$n(x) = bot \longleftrightarrow n(x^\star) = bot$
by (*metis sup-bot-right le-iff-sup mult-1-right n-one n-star-induct-iff*)

lemma *n-diamond-star-induct*:
$|x{>}n(y) \leq n(y) \Longrightarrow |x^\star{>}n(y) \leq n(y)$
by (*simp add*: *diamond-x-n n-star-induct*)

lemma *n-diamond-star-induct-sup*:
$|x{>}n(y) \sqcup n(z) \leq n(y) \Longrightarrow |x^\star{>}n(z) \leq n(y)$
by (*simp add*: *diamond-x-n n-dist-sup n-star-induct-sup*)

lemma *n-diamond-star-induct-iff*:
$|x{>}n(y) \leq n(y) \longleftrightarrow |x^\star{>}n(y) \leq n(y)$
using *diamond-x-n n-star-induct-iff* **by** *auto*

lemma *an-star-induct*:
$an(y) \leq an(x * y) \Longrightarrow an(y) \leq an(x^\star * y)$
using *an-n-order n-star-induct* **by** *auto*

lemma *an-star-induct-sup*:
  $an(y) \leq an(z \sqcup x * y) \Longrightarrow an(y) \leq an(x^\star * z)$
  using *an-n-order n-star-induct-sup* **by** *auto*

lemma *an-star-induct-star*:
  $an(y) \leq an(x * y) \Longrightarrow an(y) \leq an(x^\star)$
  by (*simp add*: *an-n-order n-star-induct-star*)

lemma *an-star-induct-iff*:
  $an(y) \leq an(x * y) \longleftrightarrow an(y) \leq an(x^\star * y)$
  using *an-n-order n-star-induct-iff* **by** *auto*

lemma *an-star-one*:
  $an(x) = 1 \longleftrightarrow an(x^\star) = 1$
  by (*metis an-n-equal an-bot n-star-bot n-bot*)

lemma *an-box-star-induct*:
  $an(y) \leq |x]an(y) \Longrightarrow an(y) \leq |x^\star]an(y)$
  by (*simp add*: *an-star-induct box-x-an*)

lemma *an-box-star-induct-sup*:
  $an(y) \leq |x]an(y) * an(z) \Longrightarrow an(y) \leq |x^\star]an(z)$
  by (*simp add*: *an-star-induct-sup an-dist-sup an-mult-commutative box-x-an*)

lemma *an-box-star-induct-iff*:
  $an(y) \leq |x]an(y) \longleftrightarrow an(y) \leq |x^\star]an(y)$
  using *an-star-induct-iff box-x-an* **by** *auto*

lemma *box-star-segerberg-an*:
  $|x^\star]an(y) = an(y) * |x^\star](n(y) \sqcup |x]an(y))$
proof (*rule order.antisym*)
  show $|x^\star]an(y) \leq an(y) * |x^\star](n(y) \sqcup |x]an(y))$
    by (*smt* (*verit*) *sup-ge2 box-1-an box-left-dist-sup box-left-mult box-right-isotone mult-right-isotone star.circ-right-unfold*)
next
  have $an(y) * |x^\star](n(y) \sqcup |x]an(y)) \leq an(y) * |x]an(y)$
    by (*metis sup-bot-left an-complement-bot box-an-an box-left-antitone box-x-an mult-left-dist-sup mult-left-one mult-right-isotone star.circ-reflexive*)
  thus $an(y) * |x^\star](n(y) \sqcup |x]an(y)) \leq |x^\star]an(y)$
    by (*smt an-box-star-induct-sup an-case-split-left an-dist-sup an-mult-least-upper-bound box-left-antitone box-left-mult box-right-mult-an-an star.left-plus-below-circ nbox-def*)
qed

lemma *box-star-segerberg-n*:
  $|x^\star]n(y) = n(y) * |x^\star](an(y) \sqcup |x]n(y))$
  using *box-star-segerberg-an an-n-def n-an-def* **by** *auto*

lemma *diamond-segerberg-an*:

$|x^\star{>}an(y) = an(y) \sqcup |x^\star{>}(n(y) * |x{>}an(y))$
by (*smt an-export an-n-L box-diamond box-star-segerberg-an diamond-box mult-assoc n-an-def*)

lemma *diamond-star-segerberg-n*:
$|x^\star{>}n(y) = n(y) \sqcup |x^\star{>}(an(y) * |x{>}n(y))$
using *an-n-def diamond-segerberg-an n-an-def* **by** *auto*

lemma *box-cut-star-iteration-an*:
$|x^\star]an(y) = |(an(y) * x)^\star]an(y)$
by (*smt an-box-star-induct-sup an-mult-commutative an-mult-complement-intro-an order.antisym box-an-export box-star-unfold-an nbox-def order-refl*)

lemma *box-cut-star-iteration-n*:
$|x^\star]n(y) = |(n(y) * x)^\star]n(y)$
using *box-cut-star-iteration-an n-an-def* **by** *auto*

lemma *diamond-cut-star-iteration-an*:
$|x^\star{>}an(y) = |(n(y) * x)^\star{>}an(y)$
using *box-cut-star-iteration-an diamond-box n-an-def* **by** *auto*

lemma *diamond-cut-star-iteration-n*:
$|x^\star{>}n(y) = |(an(y) * x)^\star{>}n(y)$
using *box-cut-star-iteration-an an-n-L diamond-box* **by** *auto*

lemma *ni-star-induct*:
$ni(x * y) \le ni(y) \Longrightarrow ni(x^\star * y) \le ni(y)$
using *n-star-induct ni-n-order* **by** *auto*

lemma *ni-star-induct-sup*:
$ni(z \sqcup x * y) \le ni(y) \Longrightarrow ni(x^\star * z) \le ni(y)$
by (*simp add*: *ni-n-order n-star-induct-sup*)

lemma *ni-star-induct-star*:
$ni(x * y) \le ni(y) \Longrightarrow ni(x^\star) \le ni(y)$
using *ni-n-order n-star-induct-star* **by** *auto*

lemma *ni-star-induct-iff*:
$ni(x * y) \le ni(y) \longleftrightarrow ni(x^\star * y) \le ni(y)$
using *ni-n-order n-star-induct-iff* **by** *auto*

lemma *ni-star-bot*:
$ni(x) = bot \longleftrightarrow ni(x^\star) = bot$
using *ni-n-bot n-star-bot* **by** *auto*

lemma *ni-diamond-star-induct*:
$\|x\text{»}ni(y) \le ni(y) \Longrightarrow \|x^\star\text{»}ni(y) \le ni(y)$
by (*simp add*: *diamond-L-x-ni ni-star-induct*)

lemma *ni-diamond-star-induct-sup*:
$\|x\rangle\!\rangle ni(y) \sqcup ni(z) \leq ni(y) \Longrightarrow \|x^\star\rangle\!\rangle ni(z) \leq ni(y)$
by (*simp add*: *diamond-L-x-ni ni-dist-sup ni-star-induct-sup*)

lemma *ni-diamond-star-induct-iff*:
$\|x\rangle\!\rangle ni(y) \leq ni(y) \longleftrightarrow \|x^\star\rangle\!\rangle ni(y) \leq ni(y)$
using *diamond-L-x-ni ni-star-induct-iff* **by** *auto*

lemma *ani-star-induct*:
$ani(y) \leq ani(x * y) \Longrightarrow ani(y) \leq ani(x^\star * y)$
using *an-star-induct ani-an-order* **by** *blast*

lemma *ani-star-induct-sup*:
$ani(y) \leq ani(z \sqcup x * y) \Longrightarrow ani(y) \leq ani(x^\star * z)$
by (*simp add*: *an-star-induct-sup ani-an-order*)

lemma *ani-star-induct-star*:
$ani(y) \leq ani(x * y) \Longrightarrow ani(y) \leq ani(x^\star)$
using *an-star-induct-star ani-an-order* **by** *auto*

lemma *ani-star-induct-iff*:
$ani(y) \leq ani(x * y) \longleftrightarrow ani(y) \leq ani(x^\star * y)$
using *an-star-induct-iff ani-an-order* **by** *auto*

lemma *ani-star-L*:
$ani(x) = L \longleftrightarrow ani(x^\star) = L$
using *an-star-one ani-an-L* **by** *auto*

lemma *ani-box-star-induct*:
$ani(y) \leq \|x]\!] ani(y) \Longrightarrow ani(y) \leq \|x^\star]\!] ani(y)$
by (*metis an-ani ani-def ani-star-induct-iff n-ani box-L-ani*)

lemma *ani-box-star-induct-iff*:
$ani(y) \leq \|x]\!] ani(y) \longleftrightarrow ani(y) \leq \|x^\star]\!] ani(y)$
using *ani-box-star-induct box-L-left-antitone order-lesseq-imp star.circ-increasing* **by** *blast*

lemma *ani-box-star-induct-sup*:
$ani(y) \leq \|x]\!] ani(y) \Longrightarrow ani(y) \leq ani(z) \Longrightarrow ani(y) \leq \|x^\star]\!] ani(z)$
by (*meson ani-box-star-induct-iff box-L-right-isotone order-trans*)

end

end

15 Approximation

theory *Approximation*

imports *Stone-Kleene-Relation-Algebras.Iterings*

begin

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

class *apx* =
  fixes *apx* :: $'a \Rightarrow 'a \Rightarrow bool$ (**infix** ‹⊑› *50*)

class *apx-order* = *apx* +
  assumes *apx-reflexive*: $x \sqsubseteq x$
  assumes *apx-antisymmetric*: $x \sqsubseteq y \wedge y \sqsubseteq x \longrightarrow x = y$
  assumes *apx-transitive*: $x \sqsubseteq y \wedge y \sqsubseteq z \longrightarrow x \sqsubseteq z$

sublocale *apx-order* < *apx*: *order* **where** *less-eq* = *apx* **and** *less* = $\lambda x\ y\ .\ x \sqsubseteq y \wedge \neg\ y \sqsubseteq x$
  apply *unfold-locales*
  apply *simp*
  apply (*rule apx-reflexive*)
  using *apx-transitive* **apply** *blast*
  by (*simp add*: *apx-antisymmetric*)

context *apx-order*
begin

abbreviation *the-apx-least-fixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a$ (‹κ -› [*201*] *200*) **where** $\kappa\ f \equiv$ *apx.the-least-fixpoint f*
abbreviation *the-apx-least-prefixpoint* :: $('a \Rightarrow 'a) \Rightarrow 'a$ (‹pκ -› [*201*] *200*) **where** $p\kappa\ f \equiv$ *apx.the-least-prefixpoint f*

definition *is-apx-meet* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ **where** *is-apx-meet x y z* $\equiv z \sqsubseteq x \wedge z \sqsubseteq y \wedge (\forall w\ .\ w \sqsubseteq x \wedge w \sqsubseteq y \longrightarrow w \sqsubseteq z)$
definition *has-apx-meet* :: $'a \Rightarrow 'a \Rightarrow bool$ **where** *has-apx-meet x y* $\equiv \exists z$. *is-apx-meet x y z*
definition *the-apx-meet* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** ‹△› *66*) **where** $x \bigtriangleup y \equiv$ *THE z* . *is-apx-meet x y z*

lemma *apx-meet-unique*:
  has-apx-meet x y $\Longrightarrow \exists !z$. *is-apx-meet x y z*
  by (*meson apx-antisymmetric has-apx-meet-def is-apx-meet-def*)

lemma *apx-meet*:
  assumes *has-apx-meet x y*
    shows *is-apx-meet x y* $(x \bigtriangleup y)$
proof –
  have *is-apx-meet x y* (*THE z* . *is-apx-meet x y z*)
    by (*metis apx-meet-unique assms theI*)
  thus *?thesis*

by (*simp add*: *the-apx-meet-def*)
qed

lemma *apx-greatest-lower-bound*:
has-apx-meet $x\ y \Longrightarrow (w \sqsubseteq x \wedge w \sqsubseteq y \longleftrightarrow w \sqsubseteq x \bigtriangleup y)$
by (*meson apx-meet apx-transitive is-apx-meet-def*)

lemma *apx-meet-same*:
is-apx-meet $x\ y\ z \Longrightarrow z = x \bigtriangleup y$
using *apx-meet apx-meet-unique has-apx-meet-def* **by** *blast*

lemma *apx-meet-char*:
is-apx-meet $x\ y\ z \longleftrightarrow$ *has-apx-meet* $x\ y \wedge z = x \bigtriangleup y$
using *apx-meet-same has-apx-meet-def* **by** *auto*

end

class *apx-biorder* = *apx-order* + *order*
begin

lemma *mu-below-kappa*:
has-least-fixpoint $f \Longrightarrow$ *apx.has-least-fixpoint* $f \Longrightarrow \mu\ f \le \kappa\ f$
using *apx.mu-unfold is-least-fixpoint-def least-fixpoint* **by** *auto*

lemma *kappa-below-nu*:
has-greatest-fixpoint $f \Longrightarrow$ *apx.has-least-fixpoint* $f \Longrightarrow \kappa\ f \le \nu\ f$
by (*meson apx.mu-unfold greatest-fixpoint is-greatest-fixpoint-def*)

lemma *kappa-apx-below-mu*:
has-least-fixpoint $f \Longrightarrow$ *apx.has-least-fixpoint* $f \Longrightarrow \kappa\ f \sqsubseteq \mu\ f$
using *apx.is-least-fixpoint-def apx.least-fixpoint mu-unfold* **by** *auto*

lemma *kappa-apx-below-nu*:
has-greatest-fixpoint $f \Longrightarrow$ *apx.has-least-fixpoint* $f \Longrightarrow \kappa\ f \sqsubseteq \nu\ f$
by (*metis apx.is-least-fixpoint-def apx.least-fixpoint nu-unfold*)

end

class *apx-semiring* = *apx-biorder* + *idempotent-left-semiring* + *L* +
assumes *apx-L-least*: $L \sqsubseteq x$
assumes *sup-apx-left-isotone*: $x \sqsubseteq y \longrightarrow x \sqcup z \sqsubseteq y \sqcup z$
assumes *mult-apx-left-isotone*: $x \sqsubseteq y \longrightarrow x * z \sqsubseteq y * z$
assumes *mult-apx-right-isotone*: $x \sqsubseteq y \longrightarrow z * x \sqsubseteq z * y$
begin

lemma *sup-apx-right-isotone*:
$x \sqsubseteq y \Longrightarrow z \sqcup x \sqsubseteq z \sqcup y$
by (*simp add*: *sup-apx-left-isotone sup-commute*)

lemma *sup-apx-isotone*:
$w \sqsubseteq y \Longrightarrow x \sqsubseteq z \Longrightarrow w \sqcup x \sqsubseteq y \sqcup z$
by (*meson apx-transitive sup-apx-left-isotone sup-apx-right-isotone*)

lemma *mult-apx-isotone*:
$w \sqsubseteq y \Longrightarrow x \sqsubseteq z \Longrightarrow w * x \sqsubseteq y * z$
by (*meson apx-transitive mult-apx-left-isotone mult-apx-right-isotone*)

lemma *affine-apx-isotone*:
apx.isotone $(\lambda x\ .\ y * x \sqcup z)$
by (*simp add*: *apx.isotone-def mult-apx-right-isotone sup-apx-left-isotone*)

end

end

16 Strict Recursion

theory *Recursion-Strict*

imports *N-Semirings Approximation*

begin

class *semiring-apx* = *n-semiring* + *apx* +
assumes *apx-def*: $x \sqsubseteq y \longleftrightarrow x \le y \sqcup n(x) * L \wedge y \le x \sqcup n(x) * top$
begin

lemma *apx-n-order-reverse*:
$y \sqsubseteq x \Longrightarrow n(x) \le n(y)$
by (*metis apx-def le-iff-sup n-sup-left-absorb-mult n-dist-sup n-export*)

lemma *apx-n-order*:
$x \sqsubseteq y \Longrightarrow y \sqsubseteq x \Longrightarrow n(x) = n(y)$
by (*simp add*: *apx-n-order-reverse order.antisym*)

lemma *apx-transitive*:
assumes $x \sqsubseteq y$
and $y \sqsubseteq z$
shows $x \sqsubseteq z$
proof −
have $n(y) * L \le n(x) * L$
by (*simp add*: *apx-n-order-reverse assms*(*1*) *mult-left-isotone*)
hence *1*: $x \le z \sqcup n(x) * L$
by (*smt assms sup-assoc sup-right-divisibility apx-def le-iff-sup*)
have $z \le x \sqcup n(x) * top \sqcup n(x \sqcup n(x) * top) * top$
by (*smt* (*verit*) *assms sup-left-isotone order-refl sup-assoc sup-mono apx-def mult-left-isotone n-isotone order-trans*)
also have ... $= x \sqcup n(x) * top$

by (*simp add*: *n-dist-sup n-export n-sup-left-absorb-mult*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

Theorem 16.1

subclass *apx-biorder*
apply *unfold-locales*
apply (*simp add*: *apx-def*)
apply (*smt* (*verit*) *order.antisym le-sup-iff apx-def eq-refl le-iff-sup n-galois apx-n-order*)
using *apx-transitive* **by** *blast*

lemma *sup-apx-left-isotone*:
assumes $x \sqsubseteq y$
shows $x \sqcup z \sqsubseteq y \sqcup z$
proof −
have $x \leq y \sqcup n(x) * L \wedge y \leq x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
hence $z \sqcup x \leq z \sqcup y \sqcup n(z \sqcup x) * L \wedge z \sqcup y \leq z \sqcup x \sqcup n(z \sqcup x) * top$
by (*metis sup-assoc sup-right-isotone mult-right-sub-dist-sup-right n-dist-sup order-trans*)
thus *?thesis*
by (*simp add*: *apx-def sup-commute*)
qed

lemma *mult-apx-left-isotone*:
assumes $x \sqsubseteq y$
shows $x * z \sqsubseteq y * z$
proof −
have $x \leq y \sqcup n(x) * L$
using *assms apx-def* **by** *auto*
hence $x * z \leq y * z \sqcup n(x) * L$
by (*smt* (*verit*, *ccfv-threshold*) *L-left-zero mult-left-isotone semiring.distrib-right mult-assoc*)
hence *1*: $x * z \leq y * z \sqcup n(x * z) * L$
by (*meson mult-left-isotone n-mult-left-upper-bound order-lesseq-imp sup-mono*)
have $y * z \leq x * z \sqcup n(x) * top * z$
by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
hence $y * z \leq x * z \sqcup n(x * z) * top$
using *mult-isotone n-mult-left-upper-bound order.trans sup-right-isotone top-greatest mult-assoc* **by** *presburger*
thus *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

lemma *mult-apx-right-isotone*:
assumes $x \sqsubseteq y$

shows $z * x \sqsubseteq z * y$
proof −
have $x \leq y \sqcup n(x) * L$
using *assms apx-def* **by** *auto*
hence *1*: $z * x \leq z * y \sqcup n(z * x) * L$
by (*smt sup-assoc sup-ge1 sup-bot-right mult-assoc mult-left-dist-sup mult-right-isotone n-L-split*)
have $y \leq x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
hence $z * y \leq z * x \sqcup z * n(x) * top$
by (*smt mult-assoc mult-left-dist-sup mult-right-isotone*)
also have ... $\leq z * x \sqcup n(z * x) * top$
by (*smt* (*verit*) *sup-assoc le-supI le-sup-iff sup-ge1 sup-bot-right mult-left-dist-sup n-L-split n-top-split order-trans*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

Theorem 16.1 and Theorem 16.2

subclass *apx-semiring*
apply *unfold-locales*
apply (*metis sup-right-top sup-ge2 apx-def mult-left-one n-L top-greatest*)
apply (*simp add*: *sup-apx-left-isotone*)
apply (*simp add*: *mult-apx-left-isotone*)
by (*simp add*: *mult-apx-right-isotone*)

Theorem 16.2

lemma *ni-apx-isotone*:
$x \sqsubseteq y \Longrightarrow ni(x) \sqsubseteq ni(y)$
using *apx-n-order-reverse apx-def le-supI1 n-ni ni-def ni-n-order* **by** *force*

Theorem 17

definition *kappa-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-apx-meet* $f \equiv$ *apx.has-least-fixpoint* $f \wedge$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \kappa\ f = \mu\ f \bigtriangleup \nu\ f$

definition *kappa-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-mu-nu* $f \equiv$ *apx.has-least-fixpoint* $f \wedge \kappa\ f = \mu\ f \sqcup n(\nu\ f) * L$

definition *nu-below-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *nu-below-mu-nu* $f \equiv \nu\ f \leq \mu\ f \sqcup n(\nu\ f) * top$

definition *mu-nu-apx-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *mu-nu-apx-nu* $f \equiv \mu\ f \sqcup n(\nu\ f) * L \sqsubseteq \nu\ f$

definition *mu-nu-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *mu-nu-apx-meet* $f \equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \bigtriangleup \nu\ f = \mu\ f \sqcup n(\nu\ f) * L$

definition *apx-meet-below-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *apx-meet-below-nu f* $\equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \bigtriangleup \nu\ f \leq \nu\ f$

lemma *mu-below-l*:
$\mu\ f \leq \mu\ f \sqcup n(\nu\ f) * L$
by *simp*

lemma *l-below-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow \mu\ f \sqcup n(\nu\ f) * L \leq \nu\ f$
by (*simp add*: *mu-below-nu n-L-decreasing*)

lemma *n-l-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow n(\mu\ f \sqcup n(\nu\ f) * L) = n(\nu\ f)$
by (*metis le-iff-sup mu-below-nu n-dist-sup n-n-L*)

lemma *l-apx-mu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow \mu\ f \sqcup n(\nu\ f) * L \sqsubseteq \mu\ f$
by (*simp add*: *apx-def le-supI1 n-l-nu*)

Theorem 17.4 implies Theorem 17.5

lemma *nu-below-mu-nu-mu-nu-apx-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *nu-below-mu-nu f* $\Longrightarrow$ *mu-nu-apx-nu f*
by (*smt* (*z3*) *l-below-nu apx-def le-sup-iff sup.absorb2 sup-commute sup-monoid.add-assoc mu-nu-apx-nu-def n-l-nu nu-below-mu-nu-def*)

Theorem 17.5 implies Theorem 17.6

lemma *mu-nu-apx-nu-mu-nu-apx-meet*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *mu-nu-apx-nu f*
shows *mu-nu-apx-meet f*
proof –
let $?l = \mu\ f \sqcup n(\nu\ f) * L$
have *is-apx-meet* $(\mu\ f)\ (\nu\ f)\ ?l$
apply (*unfold is-apx-meet-def*, *intro conjI*)
apply (*simp add*: *assms*(*1*,*2*) *l-apx-mu*)
using *assms*(*3*) *mu-nu-apx-nu-def* **apply** *blast*
by (*meson assms*(*1*,*2*) *l-below-nu apx-def order-trans sup-ge1 sup-left-isotone*)
thus *?thesis*
by (*simp add*: *apx-meet-char mu-nu-apx-meet-def*)
qed

Theorem 17.6 implies Theorem 17.7

lemma *mu-nu-apx-meet-apx-meet-below-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *mu-nu-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def l-below-nu mu-nu-apx-meet-def* **by** *auto*

Theorem 17.7 implies Theorem 17.4

lemma *apx-meet-below-nu-nu-below-mu-nu*:
assumes *apx-meet-below-nu f*
shows *nu-below-mu-nu f*
proof –
have $\forall m . m \sqsubseteq \mu f \wedge m \sqsubseteq \nu f \wedge m \leq \nu f \longrightarrow \nu f \leq \mu f \sqcup n(m) * top$
by (*smt* (*verit*) *sup-assoc sup-left-isotone sup-right-top apx-def mult-left-dist-sup order-trans*)
thus *?thesis*
by (*smt* (*verit*) *assms sup-right-isotone apx-greatest-lower-bound apx-meet-below-nu-def apx-reflexive mult-left-isotone n-isotone nu-below-mu-nu-def order-trans*)
qed

Theorem 17.1 implies Theorem 17.2

lemma *has-apx-least-fixpoint-kappa-apx-meet*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *apx.has-least-fixpoint f*
shows *kappa-apx-meet f*
proof –
have $\forall w . w \sqsubseteq \mu f \wedge w \sqsubseteq \nu f \longrightarrow w \sqsubseteq \kappa f$
by (*meson assms apx-def order.trans kappa-below-nu mu-below-kappa semiring.add-right-mono*)
hence *is-apx-meet* (μf) (νf) (κf)
by (*simp add*: *assms is-apx-meet-def kappa-apx-below-mu kappa-apx-below-nu*)
thus *?thesis*
by (*simp add*: *assms*(*3*) *kappa-apx-meet-def apx-meet-char*)
qed

Theorem 17.2 implies Theorem 17.7

lemma *kappa-apx-meet-apx-meet-below-nu*:
has-greatest-fixpoint f $\Longrightarrow$ *kappa-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def kappa-apx-meet-def kappa-below-nu* **by** *force*

Theorem 17.7 implies Theorem 17.3

lemma *apx-meet-below-nu-kappa-mu-nu*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *isotone f*
and *apx.isotone f*
and *apx-meet-below-nu f*
shows *kappa-mu-nu f*
proof –
let $?l = \mu f \sqcup n(\nu f) * L$
let $?m = \mu f \bigtriangleup \nu f$
have *1*: $?l \sqsubseteq \nu f$
using *apx-meet-below-nu-nu-below-mu-nu assms*(*1*,*2*,*5*) *mu-nu-apx-nu-def nu-below-mu-nu-mu-nu-apx-nu* **by** *blast*
hence *2*: $?m = ?l$

using *assms*(*1*,*2*) *mu-nu-apx-meet-def mu-nu-apx-nu-def mu-nu-apx-nu-mu-nu-apx-meet* **by** *blast*
have $\mu\ f \le f(?l)$
by (*metis assms*(*1*,*3*) *isotone-def mu-unfold sup-ge1*)
hence *3*: $?l \le f(?l) \sqcup n(?l) * L$
using *assms*(*1*,*2*) *semiring.add-right-mono n-l-nu* **by** *auto*
have $f(?l) \le f(\nu\ f)$
using *assms*(*1*−*3*) *l-below-nu isotone-def* **by** *blast*
also have ... $\le\ ?l \sqcup n(?l) * top$
using *1* **by** (*metis assms*(*2*) *apx-def nu-unfold*)
finally have *4*: $?l \sqsubseteq f(?l)$
using *3 apx-def* **by** *blast*
have *5*: $f(?l) \sqsubseteq \mu\ f$
by (*metis assms*(*1*,*2*,*4*) *apx.isotone-def is-least-fixpoint-def least-fixpoint l-apx-mu*)
have $f(?l) \sqsubseteq \nu\ f$
using *1* **by** (*metis assms*(*2*,*4*) *apx.isotone-def greatest-fixpoint is-greatest-fixpoint-def*)
hence $f(?l) \sqsubseteq\ ?l$
using *2 5 apx-meet-below-nu-def assms*(*5*) *apx-greatest-lower-bound* **by** *fastforce*
hence $f(?l) =\ ?l$
using *4* **by** (*simp add*: *apx.order.antisym*)
thus *?thesis*
using *1* **by** (*smt* (*verit*, *del-insts*) *assms*(*1*,*2*) *sup-left-isotone apx-antisymmetric apx-def apx.least-fixpoint-char greatest-fixpoint apx.is-least-fixpoint-def is-greatest-fixpoint-def is-least-fixpoint-def least-fixpoint n-l-nu order-trans kappa-mu-nu-def*)
qed

Theorem 17.3 implies Theorem 17.1

lemma *kappa-mu-nu-has-apx-least-fixpoint*:
kappa-mu-nu f $\Longrightarrow$ *apx.has-least-fixpoint f*
using *kappa-mu-nu-def* **by** *auto*

Theorem 17.4 implies Theorem 17.3

lemma *nu-below-mu-nu-kappa-mu-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu f* $\Longrightarrow$ *kappa-mu-nu f*
using *apx-meet-below-nu-kappa-mu-nu mu-nu-apx-meet-apx-meet-below-nu mu-nu-apx-nu-mu-nu-apx-meet nu-below-mu-nu-mu-nu-apx-nu* **by** *blast*

Theorem 17.3 implies Theorem 17.4

lemma *kappa-mu-nu-nu-below-mu-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu f* $\Longrightarrow$ *nu-below-mu-nu f*
by (*simp add*: *apx-meet-below-nu-nu-below-mu-nu has-apx-least-fixpoint-kappa-apx-meet kappa-apx-meet-apx-meet-below-nu kappa-mu-nu-def*)

definition *kappa-mu-nu-ni* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *kappa-mu-nu-ni f* $\equiv$ *apx.has-least-fixpoint f* $\wedge$ $\kappa\ f = \mu\ f \sqcup ni(\nu\ f)$

lemma *kappa-mu-nu-ni-kappa-mu-nu*:
  kappa-mu-nu-ni f $\longleftrightarrow$ *kappa-mu-nu f*
  by (*simp add*: *kappa-mu-nu-def kappa-mu-nu-ni-def ni-def*)

lemma *nu-below-mu-nu-kappa-mu-nu-ni*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu f* $\Longrightarrow$ *kappa-mu-nu-ni f*
  by (*simp add*: *kappa-mu-nu-ni-kappa-mu-nu nu-below-mu-nu-kappa-mu-nu*)

lemma *kappa-mu-nu-ni-nu-below-mu-nu*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu-ni f* $\Longrightarrow$ *nu-below-mu-nu f*
  using *kappa-mu-nu-ni-kappa-mu-nu kappa-mu-nu-nu-below-mu-nu* **by** *blast*

end

class *itering-apx* = *n-itering* + *semiring-apx*
begin

Theorem 16.3

lemma *circ-apx-isotone*:
  assumes $x \sqsubseteq y$
    shows $x^\circ \sqsubseteq y^\circ$
proof –
  have *1*: $x \leq y \sqcup n(x) * L \wedge y \leq x \sqcup n(x) * top$
    using *assms apx-def* **by** *auto*
  hence $y^\circ \leq x^\circ \sqcup x^\circ * n(x) * top$
    by (*metis circ-isotone circ-left-top circ-unfold-sum mult-assoc*)
  also have ... $\leq x^\circ \sqcup n(x^\circ * x) * top$
    by (*smt le-sup-iff n-isotone n-top-split order-refl order-trans right-plus-below-circ zero-right-mult-decreasing*)
  also have ... $\leq x^\circ \sqcup n(x^\circ) * top$
    by (*simp add*: *circ-plus-same n-circ-left-unfold*)
  finally have *2*: $y^\circ \leq x^\circ \sqcup n(x^\circ) * top$
    .
  have $x^\circ \leq y^\circ \sqcup y^\circ * n(x) * L$
    using *1* **by** (*metis L-left-zero circ-isotone circ-unfold-sum mult-assoc*)
  also have ... $= y^\circ \sqcup n(y^\circ * x) * L$
    by (*metis sup-assoc sup-bot-right mult-assoc mult-zero-sup-circ-2 n-L-split n-mult-right-bot*)
  also have ... $\leq y^\circ \sqcup n(x^\circ * x) * L \sqcup n(x^\circ) * n(top * x) * L$
    using *2* **by** (*metis sup-assoc sup-right-isotone mult-assoc mult-left-isotone mult-right-dist-sup n-dist-sup n-export n-isotone*)
  finally have $x^\circ \leq y^\circ \sqcup n(x^\circ) * L$
    by (*metis sup-assoc circ-plus-same n-sup-left-absorb-mult n-circ-left-unfold*

n-dist-sup n-export ni-def ni-dist-sup)
  thus *?thesis*
    using *2* **by** (*simp add*: *apx-def*)
qed

end

class *omega-algebra-apx* = *n-omega-algebra-2* + *semiring-apx*

sublocale *omega-algebra-apx* < *star*: *itering-apx* **where** *circ* = *star* **..**

sublocale *omega-algebra-apx* < *nL-omega*: *itering-apx* **where** *circ* = *Omega* **..**

context *omega-algebra-apx*
begin

Theorem 16.4

lemma *omega-apx-isotone*:
  assumes $x \sqsubseteq y$
    shows $x^\omega \sqsubseteq y^\omega$
proof −
  have *1*: $x \leq y \sqcup n(x) * L \wedge y \leq x \sqcup n(x) * top$
    using *assms apx-def* **by** *auto*
  hence $y^\omega \leq x^\star * n(x) * top * (x^\star * n(x) * top)^\omega \sqcup x^\omega \sqcup x^\star * n(x) * top * (x^\star * n(x) * top)^\star * x^\omega$
    by (*smt sup-assoc mult-assoc mult-left-one mult-right-dist-sup omega-decompose omega-isotone omega-unfold star-left-unfold-equal*)
  also have ... $\leq x^\star * n(x) * top \sqcup x^\omega \sqcup x^\star * n(x) * top * (x^\star * n(x) * top)^\star * x^\omega$
    using *mult-top-omega omega-unfold sup-left-isotone* **by** *auto*
  also have ... $= x^\star * n(x) * top \sqcup x^\omega$
    by (*smt* (*z3*) *mult-left-dist-sup sup-assoc sup-commute sup-left-top mult-assoc*)
  also have ... $\leq n(x^\star * x) * top \sqcup x^\star * bot \sqcup x^\omega$
    using *n-top-split semiring.add-left-mono sup-commute* **by** *fastforce*
  also have ... $\leq n(x^\star * x) * top \sqcup x^\omega$
    using *semiring.add-right-mono star-bot-below-omega sup-commute* **by** *fastforce*
  finally have *2*: $y^\omega \leq x^\omega \sqcup n(x^\omega) * top$
    by (*metis sup-commute sup-right-isotone mult-left-isotone n-star-below-n-omega n-star-left-unfold order-trans star.circ-plus-same*)
  have $x^\omega \leq (y \sqcup n(x) * L)^\omega$
    using *1* **by** (*simp add*: *omega-isotone*)
  also have ... $= y^\star * n(x) * L * (y^\star * n(x) * L)^\omega \sqcup y^\omega \sqcup y^\star * n(x) * L * (y^\star * n(x) * L)^\star * y^\omega$
    by (*smt sup-assoc mult-assoc mult-left-one mult-right-dist-sup omega-decompose omega-isotone omega-unfold star-left-unfold-equal*)
  also have ... $= y^\star * n(x) * L \sqcup y^\omega$
    using *L-left-zero sup-assoc sup-monoid.add-commute mult-assoc* **by** *force*
  also have ... $\leq y^\omega \sqcup y^\star * bot \sqcup n(y^\star * x) * L$
    by (*simp add*: *n-L-split sup-assoc sup-commute*)
  also have ... $\leq y^\omega \sqcup n(x^\star * x) * L \sqcup n(x^\star) * n(top * x) * L$

using *1* **by** (*metis sup-right-isotone sup-bot-right apx-def mult-assoc mult-left-dist-sup mult-left-isotone mult-right-dist-sup n-dist-sup n-export n-isotone star.circ-apx-isotone star-mult-omega sup-assoc*)
finally have $x^\omega \le y^\omega \sqcup n(x^\omega) * L$
by (*smt* (*verit*, *best*) *le-supE sup.orderE sup-commute sup-assoc sup-isotone mult-right-dist-sup n-sup-left-absorb-mult n-star-left-unfold ni-def ni-star-below-ni-omega order-refl order-trans star.circ-plus-same*)
thus *?thesis*
using *2* **by** (*simp add*: *apx-def*)
qed

end

class *omega-algebra-apx-extra* = *omega-algebra-apx* +
assumes *n-split-omega*: $x^\omega \le x^\star * bot \sqcup n(x^\omega) * top$
begin

lemma *omega-n-star*:
$x^\omega \sqcup n(x^\star) * top \le x^\star * n(x^\omega) * top$
proof −
have *1*: $n(x^\star) * top \le n(x^\omega) * top$
by (*simp add*: *mult-left-isotone n-star-below-n-omega*)
have ... $\le x^\star * n(x^\omega) * top$
by (*simp add*: *star-n-omega-top*)
thus *?thesis*
using *1* **by** (*metis le-sup-iff n-split-omega order-trans star-n-omega-top*)
qed

lemma *n-omega-zero*:
$n(x^\omega) = bot \longleftrightarrow n(x^\star) = bot \wedge x^\omega \le x^\star * bot$
by (*metis sup-bot-right order.eq-iff mult-left-zero n-mult-bot n-split-omega star-bot-below-omega*)

lemma *n-split-nu-mu*:
$y^\omega \sqcup y^\star * z \le y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * top$
proof −
have $y^\omega \le y^\star * bot \sqcup n(y^\omega \sqcup y^\star * z) * top$
by (*smt sup-ge1 sup-right-isotone mult-left-isotone n-isotone n-split-omega order-trans*)
also have ... $\le y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * top$
using *nL-star.star-zero-below-circ-mult sup-left-isotone* **by** *auto*
finally show *?thesis*
by *simp*
qed

lemma *loop-exists*:
$\nu\ (\lambda x\ .\ y * x \sqcup z) \le \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * top$
by (*metis n-split-nu-mu omega-loop-nu star-loop-mu*)

lemma *loop-apx-least-fixpoint*:
$apx.is\text{-}least\text{-}fixpoint\ (\lambda x\ .\ y * x \sqcup z)\ (\mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * L)$
using *apx.least-fixpoint-char affine-apx-isotone affine-has-greatest-fixpoint affine-has-least-fixpoint affine-isotone kappa-mu-nu-def nu-below-mu-nu-def nu-below-mu-nu-kappa-mu-nu loop-exists* **by** *auto*

lemma *loop-has-apx-least-fixpoint*:
$apx.has\text{-}least\text{-}fixpoint\ (\lambda x\ .\ y * x \sqcup z)$
using *affine-apx-isotone affine-has-greatest-fixpoint affine-has-least-fixpoint affine-isotone kappa-mu-nu-def nu-below-mu-nu-def nu-below-mu-nu-kappa-mu-nu loop-exists* **by** *auto*

lemma *loop-semantics*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * L$
using *apx.least-fixpoint-char loop-apx-least-fixpoint* **by** *auto*

lemma *loop-apx-least-fixpoint-ni*:
$apx.is\text{-}least\text{-}fixpoint\ (\lambda x\ .\ y * x \sqcup z)\ (\mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup ni(\nu\ (\lambda x\ .\ y * x \sqcup z)))$
using *ni-def loop-apx-least-fixpoint* **by** *auto*

lemma *loop-semantics-ni*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup ni(\nu\ (\lambda x\ .\ y * x \sqcup z))$
using *ni-def loop-semantics* **by** *auto*

Theorem 18

lemma *loop-semantics-kappa-mu-nu*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = n(y^\omega) * L \sqcup y^\star * z$
proof –
have $\kappa\ (\lambda x\ .\ y * x \sqcup z) = y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * L$
by (*metis loop-semantics omega-loop-nu star-loop-mu*)
thus *?thesis*
by (*smt sup-assoc sup-commute le-iff-sup mult-right-dist-sup n-L-decreasing n-dist-sup*)
qed

end

class *omega-algebra-apx-extra-2* = *omega-algebra-apx* +
assumes *omega-n-star*: $x^\omega \le x^\star * n(x^\omega) * top$
begin

subclass *omega-algebra-apx-extra*
apply *unfold-locales*
using *omega-n-star star-n-omega-top* **by** *auto*

end

end

17 N-Algebras

theory *N-Algebras*

imports *Stone-Kleene-Relation-Algebras.Iterings Base Lattice-Ordered-Semirings*

begin

class *C-left-n-algebra* = *bounded-idempotent-left-semiring* +
bounded-distrib-lattice + *n* + *L*
begin

abbreviation $C :: 'a \Rightarrow 'a$ **where** $C\ x \equiv n(L) * top \sqcap x$

AACP Theorem 3.38

lemma *C-isotone*:
$x \leq y \longrightarrow C\ x \leq C\ y$
using *inf.sup-right-isotone* **by** *auto*

AACP Theorem 3.40

lemma *C-decreasing*:
$C\ x \leq x$
by *simp*

end

class *left-n-algebra* = *C-left-n-algebra* +
assumes *n-dist-n-add* : $n(x) \sqcup n(y) = n(n(x) * top \sqcup y)$
assumes *n-export* : $n(x) * n(y) = n(n(x) * y)$
assumes *n-left-upper-bound* : $n(x) \leq n(x \sqcup y)$
assumes *n-nL-meet-L-nL0* : $n(L) * x = (x \sqcap L) \sqcup n(L * bot) * x$
assumes *n-n-L-split-n-n-L-L* : $x * n(y) * L = x * bot \sqcup n(x * n(y) * L) * L$
assumes *n-sub-nL* : $n(x) \leq n(L)$
assumes *n-L-decreasing* : $n(x) * L \leq x$
assumes *n-L-T-meet-mult-combined*: $C\ (x * y) * z \leq C\ x * y * C\ z$
assumes *n-n-top-split-n-top* : $x * n(y) * top \leq x * bot \sqcup n(x * y) * top$
assumes *n-top-meet-L-below-L* : $x * top * y \sqcap L \leq x * L * y$
begin

subclass *lattice-ordered-pre-left-semiring* **..**

lemma *n-L-T-meet-mult-below*:
$C\ (x * y) \leq C\ x * y$
proof –
have $C\ (x * y) \leq C\ x * y * C\ 1$
by (*meson order.trans mult-sub-right-one n-L-T-meet-mult-combined*)
also have ... $\leq C\ x * y$
by (*metis mult-1-right mult-left-sub-dist-inf-right*)
finally show *?thesis*
.

qed

AACP Theorem 3.41

lemma *n-L-T-meet-mult-propagate*:
$C\ x * y \leq x * C\ y$
proof −
have $C\ x * y \leq C\ x * 1 * C\ y$
by (*metis mult-1-right mult-assoc n-L-T-meet-mult-combined mult-1-right*)
also have ... $\leq x * C\ y$
by (*simp add*: *mult-right-sub-dist-inf-right*)
finally show *?thesis*
.
qed

AACP Theorem 3.43

lemma *C-n-mult-closed*:
$C\ (n(x) * y) = n(x) * y$
by (*simp add*: *inf.absorb2 mult-isotone n-sub-nL*)

AACP Theorem 3.40

lemma *meet-L-below-C*:
$x \sqcap L \leq C\ x$
by (*simp add*: *le-supI1 n-nL-meet-L-nL0*)

AACP Theorem 3.42

lemma *n-L-T-meet-mult*:
$C\ (x * y) = C\ x * y$
apply (*rule order.antisym*)
apply (*rule n-L-T-meet-mult-below*)
by (*smt (z3) C-n-mult-closed inf.boundedE inf.sup-monoid.add-assoc inf.sup-monoid.add-commute mult-right-sub-dist-inf mult-assoc*)

AACP Theorem 3.42

lemma *C-mult-propagate*:
$C\ x * y = C\ x * C\ y$
by (*smt (z3) C-n-mult-closed order.eq-iff inf.left-commute inf.sup-monoid.add-commute mult-left-sub-dist-inf-right n-L-T-meet-mult-propagate*)

AACP Theorem 3.32

lemma *meet-L-below-n-L*:
$x \sqcap L \leq n(L) * x$
by (*simp add*: *n-nL-meet-L-nL0*)

AACP Theorem 3.27

lemma *n-vector-meet-L*:
$x * top \sqcap L \leq x * L$
by (*metis mult-1-right n-top-meet-L-below-L*)

lemma *n-right-upper-bound*:
$n(x) \leq n(y \sqcup x)$
by (*simp add*: *n-left-upper-bound sup-commute*)

AACP Theorem 3.1

lemma *n-isotone*:
$x \leq y \Longrightarrow n(x) \leq n(y)$
by (*metis le-iff-sup n-left-upper-bound*)

lemma *n-add-left-zero*:
$n(bot) \sqcup n(x) = n(x)$
using *le-iff-sup sup-bot-right sup-right-divisibility n-isotone* **by** *auto*

AACP Theorem 3.13

lemma *n-mult-right-zero-L*:
$n(x) * bot \leq L$
by (*meson bot-least mult-isotone n-L-decreasing n-sub-nL order-trans*)

lemma *n-add-left-top*:
$n(top) \sqcup n(x) = n(top)$
by (*simp add*: *sup-absorb1 n-isotone*)

AACP Theorem 3.18

lemma *n-n-L*:
$n(n(x) * L) = n(x)$
by (*metis order.antisym n-dist-n-add n-export n-sub-nL sup-bot-right sup-commute sup-top-left n-add-left-zero n-right-upper-bound*)

lemma *n-mult-transitive*:
$n(x) * n(x) \leq n(x)$
by (*metis mult-right-isotone n-export n-sub-nL n-n-L*)

lemma *n-mult-left-absorb-add-sub*:
$n(x) * (n(x) \sqcup n(y)) \leq n(x)$
by (*metis mult-right-isotone n-dist-n-add n-export n-sub-nL n-n-L*)

AACP Theorem 3.21

lemma *n-mult-left-lower-bound*:
$n(x) * n(y) \leq n(x)$
by (*metis mult-right-isotone n-export n-sub-nL n-n-L*)

AACP Theorem 3.20

lemma *n-mult-left-zero*:
$n(bot) * n(x) = n(bot)$
by (*metis n-export sup-absorb1 n-add-left-zero n-mult-left-lower-bound*)

lemma *n-mult-right-one*:
$n(x) * n(top) = n(x)$
using *n-dist-n-add n-export sup-commute n-add-left-zero* **by** *fastforce*

lemma *n-L-increasing*:
$n(x) \leq n(n(x) * L)$
by (*simp add*: *n-n-L*)

AACP Theorem 3.2

lemma *n-galois*:
$n(x) \leq n(y) \longleftrightarrow n(x) * L \leq y$
by (*metis mult-left-isotone n-L-decreasing n-L-increasing n-isotone order-trans*)

lemma *n-add-n-top*:
$n(x \sqcup n(x) * top) = n(x)$
by (*metis n-dist-n-add sup.idem sup-commute*)

AACP Theorem 3.6

lemma *n-L-below-nL-top*:
$L \leq n(L) * top$
by (*metis inf-top.left-neutral meet-L-below-n-L*)

AACP Theorem 3.4

lemma *n-less-eq-char-n*:
$x \leq y \longleftrightarrow x \leq y \sqcup L \wedge C\ x \leq y \sqcup n(y) * top$
proof
assume $x \leq y$
thus $x \leq y \sqcup L \wedge C\ x \leq y \sqcup n(y) * top$
by (*simp add*: *inf.coboundedI2 le-supI1*)
next
assume *1*: $x \leq y \sqcup L \wedge C\ x \leq y \sqcup n(y) * top$
hence $x \leq y \sqcup (x \sqcap L)$
using *sup-commute sup-inf-distrib2* **by** *force*
also have ... $\leq y \sqcup C\ x$
using *sup-right-isotone meet-L-below-C* **by** *blast*
also have ... $\leq y \sqcup n(y) * top$
using *1* **by** *simp*
finally have $x \leq y \sqcup (L \sqcap n(y) * top)$
using *1* **by** (*simp add*: *sup-inf-distrib1*)
thus $x \leq y$
by (*metis inf-commute n-L-decreasing order-trans sup-absorb1 n-vector-meet-L*)
qed

AACP Theorem 3.31

lemma *n-L-decreasing-meet-L*:
$n(x) * L \leq x \sqcap L$
using *n-sub-nL n-galois* **by** *auto*

AACP Theorem 3.5

lemma *n-zero-L-zero*:
$n(bot) * L = bot$

by (*simp add*: *le-bot n-L-decreasing*)

lemma *n-L-top-below-L*:
$L * top \leq L$
proof –
have $n(L * bot) * L * top \leq L * bot$
by (*metis dense-top-closed mult-isotone n-L-decreasing zero-vector mult-assoc*)
hence $n(L * bot) * L * top \leq L$
using *order-lesseq-imp zero-right-mult-decreasing* **by** *blast*
hence $n(L) * L * top \leq L$
by (*metis inf.absorb2 n-nL-meet-L-nL0 order.refl sup.absorb-iff1 top-right-mult-increasing mult-assoc*)
thus $L * top \leq L$
by (*metis inf.absorb2 inf.sup-monoid.add-commute n-L-decreasing n-L-below-nL-top n-vector-meet-L*)
qed

AACP Theorem 3.9

lemma *n-L-top-L*:
$L * top = L$
by (*simp add*: *order.antisym top-right-mult-increasing n-L-top-below-L*)

AACP Theorem 3.10

lemma *n-L-below-L*:
$L * x \leq L$
by (*metis mult-right-isotone top.extremum n-L-top-L*)

AACP Theorem 3.7

lemma *n-nL-nT*:
$n(L) = n(top)$
using *order.eq-iff n-sub-nL n-add-left-top* **by** *auto*

AACP Theorem 3.8

lemma *n-L-L*:
$n(L) * L = L$
using *order.antisym meet-L-below-n-L n-L-decreasing-meet-L* **by** *fastforce*

lemma *n-top-L*:
$n(top) * L = L$
using *n-L-L n-nL-nT* **by** *auto*

AACP Theorem 3.23

lemma *n-n-L-split-n-L*:
$x * n(y) * L \leq x * bot \sqcup n(x * y) * L$
by (*metis n-n-L-split-n-n-L-L n-L-decreasing mult-assoc mult-left-isotone mult-right-isotone n-isotone sup-right-isotone*)

AACP Theorem 3.12

lemma *n-L-split-n-L-L*:

$x * L = x * bot \sqcup n(x * L) * L$
apply (*rule order.antisym*)
apply (*metis mult-assoc n-n-L-split-n-L n-L-L*)
by (*simp add*: *mult-right-isotone n-L-decreasing*)

AACP Theorem 3.11

lemma *n-L-split-L*:
$x * L \leq x * bot \sqcup L$
by (*metis n-n-L-split-n-n-L-L n-sub-nL sup-right-isotone mult-assoc n-L-L n-galois*)

AACP Theorem 3.24

lemma *n-split-top*:
$x * n(y) * top \leq x * y \sqcup n(x * y) * top$
proof −
have $x * bot \sqcup n(x * y) * top \leq x * y \sqcup n(x * y) * top$
by (*meson bot-least mult-isotone order.refl sup-left-isotone*)
thus *?thesis*
using *order.trans n-n-top-split-n-top* **by** *blast*
qed

AACP Theorem 3.9

lemma *n-L-L-L*:
$L * L = L$
by (*metis inf.sup-monoid.add-commute inf-absorb1 n-L-below-L n-L-top-L n-vector-meet-L*)

AACP Theorem 3.9

lemma *n-L-top-L-L*:
$L * top * L = L$
by (*simp add*: *n-L-L-L n-L-top-L*)

AACP Theorem 3.19

lemma *n-n-nL*:
$n(x) = n(x) * n(L)$
by (*simp add*: *n-export n-n-L*)

lemma *n-L-mult-idempotent*:
$n(L) * n(L) = n(L)$
using *n-n-nL* **by** *auto*

AACP Theorem 3.22

lemma *n-n-L-n*:
$n(x * n(y) * L) \leq n(x * y)$
by (*simp add*: *mult-right-isotone n-L-decreasing mult-assoc n-isotone*)

AACP Theorem 3.3

lemma *n-less-eq-char*:
$x \leq y \longleftrightarrow x \leq y \sqcup L \wedge x \leq y \sqcup n(y) * top$

by (*meson inf.coboundedI2 le-supI1 n-less-eq-char-n*)

AACP Theorem 3.28

lemma *n-top-meet-L-split-L*:
$x * top * y \sqcap L \leq x * bot \sqcup L * y$
proof –
have $x * top * y \sqcap L \leq x * bot \sqcup n(x * L) * L * y$
by (*smt n-top-meet-L-below-L mult-assoc n-L-L-L n-L-split-n-L-L mult-right-dist-sup mult-left-zero*)
also have ... $\leq x * bot \sqcup x * L * y$
using *mult-left-isotone n-L-decreasing sup-right-isotone* **by** *force*
also have ... $\leq x * bot \sqcup (x * bot \sqcup L) * y$
using *mult-left-isotone sup-right-isotone n-L-split-L* **by** *blast*
also have ... $= x * bot \sqcup x * bot * y \sqcup L * y$
by (*simp add*: *mult-right-dist-sup sup-assoc*)
also have ... $= x * bot \sqcup L * y$
by (*simp add*: *mult-assoc*)
finally show *?thesis*
.
qed

AACP Theorem 3.29

lemma *n-top-meet-L-L-meet-L*:
$x * top * y \sqcap L = x * L * y \sqcap L$
apply (*rule order.antisym*)
apply (*simp add*: *n-top-meet-L-below-L*)
by (*metis inf.sup-monoid.add-commute inf.sup-right-isotone mult-isotone order.refl top.extremum*)

lemma *n-n-top-below-n-L*:
$n(x * top) \leq n(x * L)$
by (*meson order.trans n-L-decreasing-meet-L n-galois n-vector-meet-L*)

AACP Theorem 3.14

lemma *n-n-top-n-L*:
$n(x * top) = n(x * L)$
by (*metis order.antisym mult-right-isotone n-isotone n-n-top-below-n-L top-greatest*)

AACP Theorem 3.30

lemma *n-meet-L-0-below-0-meet-L*:
$(x \sqcap L) * bot \leq x * bot \sqcap L$
by (*meson inf.boundedE inf.boundedI mult-right-sub-dist-inf-left zero-right-mult-decreasing*)

AACP Theorem 3.15

lemma *n-n-L-below-L*:
$n(x) * L \leq x * L$
by (*metis mult-assoc mult-left-isotone n-L-L-L n-L-decreasing*)

lemma *n-n-L-below-n-L-L*:
  $n(x) * L \leq n(x * L) * L$
  by (*simp add*: *mult-left-isotone n-galois n-n-L-below-L*)

AACP Theorem 3.16

lemma *n-below-n-L*:
  $n(x) \leq n(x * L)$
  by (*simp add*: *n-galois n-n-L-below-L*)

AACP Theorem 3.17

lemma *n-below-n-L-mult*:
  $n(x) \leq n(L) * n(x)$
  by (*metis n-export order-trans meet-L-below-n-L n-L-decreasing-meet-L n-isotone n-n-L*)

AACP Theorem 3.33

lemma *n-meet-L-below*:
  $n(x) \sqcap L \leq x$
  by (*meson inf.coboundedI1 inf.coboundedI2 le-supI2 sup.cobounded1 top-right-mult-increasing n-less-eq-char*)

AACP Theorem 3.35

lemma *n-meet-L-top-below-n-L*:
  $(n(x) \sqcap L) * top \leq n(x) * L$
proof −
  have $(n(x) \sqcap L) * top \leq n(x) * top \sqcap L * top$
    by (*meson mult-right-sub-dist-inf*)
  thus *?thesis*
    by (*metis n-L-top-L n-vector-meet-L order-trans*)
qed

AACP Theorem 3.34

lemma *n-meet-L-top-below*:
  $(n(x) \sqcap L) * top \leq x$
  using *order.trans n-L-decreasing n-meet-L-top-below-n-L* **by** *blast*

AACP Theorem 3.36

lemma *n-n-meet-L*:
  $n(x) = n(x \sqcap L)$
  by (*meson order.antisym inf.cobounded1 n-L-decreasing-meet-L n-galois n-isotone*)

lemma *n-T-below-n-meet*:
  $n(x) * top = n(C\ x) * top$
  by (*metis inf.absorb2 inf.sup-monoid.add-assoc meet-L-below-C n-n-meet-L*)

AACP Theorem 3.44

lemma *n-C*:

$n(C\ x) = n(x)$
by (*metis n-T-below-n-meet n-export n-mult-right-one*)

AACP Theorem 3.37

lemma *n-T-meet-L*:
$n(x) * top \sqcap L = n(x) * L$
by (*metis antisym-conv n-L-decreasing-meet-L n-n-L n-n-top-n-L n-vector-meet-L*)

AACP Theorem 3.39

lemma *n-L-top-meet-L*:
$C\ L = L$
by (*simp add*: *n-L-L n-T-meet-L*)

end

class *n-algebra* = *left-n-algebra* + *idempotent-left-zero-semiring*
begin

proposition *n-dist-n-add* : $n(x) \sqcup n(y) = n(n(x) * top \sqcup y)$ **oops**
proposition *n-export* : $n(x) * n(y) = n(n(x) * y)$ **oops**
proposition *n-left-upper-bound* : $n(x) \leq n(x \sqcup y)$ **oops**
proposition *n-nL-meet-L-nL0* : $n(L) * x = (x \sqcap L) \sqcup n(L * bot) * x$ **oops**
proposition *n-n-L-split-n-n-L-L* : $x * n(y) * L = x * bot \sqcup n(x * n(y) * L) * L$ **oops**
proposition *n-sub-nL* : $n(x) \leq n(L)$ **oops**
proposition *n-L-decreasing* : $n(x) * L \leq x$ **oops**
proposition *n-L-T-meet-mult-combined*: $C\ (x * y) * z \leq C\ x * y * C\ z$ **oops**
proposition *n-n-top-split-n-top* : $x * n(y) * top \leq x * bot \sqcup n(x * y) * top$ **oops**
proposition *n-top-meet-L-below-L* : $x * top * y \sqcap L \leq x * L * y$ **oops**

AACP Theorem 3.25

lemma *n-top-split-0*:
$n(x) * top * y \leq x * y \sqcup n(x * bot) * top$
proof –
have *1*: $n(x) * top * y \sqcap L \leq x * y$
using *inf.coboundedI1 mult-left-isotone n-L-decreasing-meet-L n-top-meet-L-L-meet-L* **by** *force*
have $n(x) * top * y = n(x) * n(L) * top * y$
using *n-n-nL* **by** *auto*
also have ... $= n(x) * ((top * y \sqcap L) \sqcup n(L * bot) * top * y)$
by (*metis mult-assoc n-nL-meet-L-nL0*)
also have ... $\leq n(x) * (top * y \sqcap L) \sqcup n(x) * n(L * bot) * top$
by (*metis sup-right-isotone mult-assoc mult-left-dist-sup mult-right-isotone top-greatest*)
also have ... $\leq (n(x) * top * y \sqcap L) \sqcup n(n(x) * L * bot) * top$
by (*smt sup-left-isotone order.trans inf-greatest mult-assoc mult-left-sub-dist-inf-left mult-left-sub-dist-inf-right n-export n-galois n-sub-nL*)

also have ... $\leq x * y \sqcup n(n(x) * L * bot) * top$
using *1 sup-left-isotone* **by** *blast*
also have ... $\leq x * y \sqcup n(x * bot) * top$
using *mult-left-isotone n-galois n-isotone order.refl sup-right-isotone* **by** *auto*
finally show *?thesis*
.
qed

AACP Theorem 3.26

lemma *n-top-split*:
$n(x) * top * y \leq x * y \sqcup n(x * y) * top$
by (*metis order.trans sup-bot-right mult-assoc sup-right-isotone mult-left-isotone mult-left-sub-dist-sup-right n-isotone n-top-split-0*)

proposition *n-zero*: $n(bot) = bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-one*: $n(1) = bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-nL-one*: $n(L) = 1$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-nT-one*: $n(top) = 1$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-n-zero*: $n(x) = n(x * bot)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-dist-add*: $n(x) \sqcup n(y) = n(x \sqcup y)$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *n-L-split*: $x * n(y) * L = x * bot \sqcup n(x * y) * L$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *n-split*: $x \leq x * bot \sqcup n(x * L) * top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-mult-top-1*: $n(x * y) \leq n(x * n(y) * top)$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *l91-1*: $n(L) * x \leq n(x * top) * top$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *meet-domain-top*: $x \sqcap n(y) * top = n(y) * x$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *meet-domain-2*: $x \sqcap n(y) * top \leq n(L) * x$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *n-nL-top-n-top-meet-L-top-2*: $n(L) * x * top \leq n(x * top \sqcap L) * top$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *n-nL-top-n-top-meet-L-top-1*: $n(x * top \sqcap L) * top \leq n(L) * x * top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *l9*: $x * bot \sqcap L \leq n(x * L) * L$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *l18-2*: $n(x * L) * L \leq n(x) * L$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *l51-1*: $n(x) * L \leq (x \sqcap L) * bot$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *l51-2*: $(x \sqcap L) * bot \leq n(x) * L$ **nitpick** [*expect=genuine,card=4*] **oops**

proposition *n-split-equal*: $x \sqcup n(x * L) * top = x * bot \sqcup n(x * L) * top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *n-split-top*: $x * top \leq x * bot \sqcup n(x * L) * top$ **nitpick**

[*expect=genuine*,*card=2*] **oops**
proposition *n-mult*: $n(x * n(y) * L) = n(x * y)$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *n-mult-1*: $n(x * y) \leq n(x * n(y) * L)$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *n-mult-top*: $n(x * n(y) * top) = n(x * y)$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *n-mult-right-upper-bound*: $n(x * y) \leq n(z) \longleftrightarrow n(x) \leq n(z) \wedge x * n(y) * L \leq x * bot \sqcup n(z) * L$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *meet-domain*: $x \sqcap n(y) * z = n(y) * (x \sqcap z)$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *meet-domain-1*: $x \sqcap n(y) * z \leq n(y) * x$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *meet-domain-top-3*: $x \sqcap n(y) * top \leq n(y) * x$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *n-n-top-n-top-split-n-n-top-top*: $n(x) * top \sqcup x * n(y) * top = x * bot \sqcup n(x * n(y) * top) * top$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *n-n-top-n-top-split-n-n-top-top-1*: $x * bot \sqcup n(x * n(y) * top) * top \leq n(x) * top \sqcup x * n(y) * top$ **nitpick** [*expect=genuine*,*card=5*] **oops**
proposition *n-n-top-n-top-split-n-n-top-top-2*: $n(x) * top \sqcup x * n(y) * top \leq x * bot \sqcup n(x * n(y) * top) * top$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *n-nL-top-n-top-meet-L-top*: $n(L) * x * top = n(x * top \sqcap L) * top$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *l18*: $n(x) * L = n(x * L) * L$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *l22*: $x * bot \sqcap L = n(x) * L$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *l22-1*: $x * bot \sqcap L = n(x * L) * L$ **nitpick**
[*expect=genuine*,*card=2*] **oops**
proposition *l22-2*: $x \sqcap L = n(x) * L$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *l22-3*: $x \sqcap L = n(x * L) * L$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *l22-4*: $x \sqcap L \leq n(x) * L$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *l22-5*: $x * bot \sqcap L \leq n(x) * L$ **nitpick** [*expect=genuine*,*card=4*] **oops**
proposition *l23*: $x * top \sqcap L = n(x) * L$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *l51*: $n(x) * L = (x \sqcap L) * bot$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *l91*: $x = x * top \longrightarrow n(L) * x \leq n(x) * top$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *l92*: $x = x * top \longrightarrow n(L) * x \leq n(x \sqcap L) * top$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition $x \sqcap L \leq n(x) * top$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *n-meet-comp*: $n(x) \sqcap n(y) \leq n(x) * n(y)$ **nitpick**
[*expect=genuine*,*card=3*] **oops**

proposition *n-n-meet-L-n-zero*: $n(x) = (n(x) \sqcap L) \sqcup n(x * bot)$ **oops**
proposition *n-below-n-zero*: $n(x) \leq x \sqcup n(x * bot)$ **oops**
proposition *n-n-top-split-n-L-n-zero-top*: $n(x) * top = n(x) * L \sqcup n(x * bot) * top$ **oops**
proposition *n-meet-L-0-0-meet-L*: $(x \sqcap L) * bot = x * bot \sqcap L$ **oops**

end

end

18 Recursion

theory *Recursion*

imports *Approximation N-Algebras*

begin

class *n-algebra-apx* = *n-algebra* + *apx* +
  assumes *apx-def*: $x \sqsubseteq y \longleftrightarrow x \leq y \sqcup L \wedge C\ y \leq x \sqcup n(x) * top$
begin

lemma *apx-transitive-2*:
  assumes $x \sqsubseteq y$
      and $y \sqsubseteq z$
    shows $x \sqsubseteq z$
proof −
  have $C\ z \leq C\ (y \sqcup n(y) * top)$
    using *assms*(*2*) *apx-def le-inf-iff* **by** *blast*
  also have ... $= C\ y \sqcup n(y) * top$
    by (*simp add*: *C-n-mult-closed inf-sup-distrib1*)
  also have ... $\leq x \sqcup n(x) * top \sqcup n(y) * top$
    using *assms*(*1*) *apx-def sup-left-isotone* **by** *blast*
  also have ... $= x \sqcup n(x) * top \sqcup n(C\ y) * top$
    by (*simp add*: *n-C*)
  also have ... $\leq x \sqcup n(x) * top$
    by (*metis assms*(*1*) *sup-assoc sup-idem sup-right-isotone apx-def mult-left-isotone n-add-n-top n-isotone*)
  finally show *?thesis*
    by (*smt assms sup-assoc sup-commute apx-def le-iff-sup*)
qed

lemma *apx-meet-L*:
  assumes $y \sqsubseteq x$
    shows $x \sqcap L \leq y \sqcap L$
proof −
  have $x \sqcap L = C\ x \sqcap L$
    by (*simp add*: *inf.left-commute inf.sup-monoid.add-assoc n-L-top-meet-L*)
  also have ... $\leq (y \sqcup n(y) * top) \sqcap L$
    using *assms apx-def inf.sup-left-isotone* **by** *blast*
  also have ... $= (y \sqcap L) \sqcup (n(y) * top \sqcap L)$
    by (*simp add*: *inf-sup-distrib2*)
  also have ... $\leq (y \sqcap L) \sqcup n(y \sqcap L) * top$
    using *n-n-meet-L sup-right-isotone* **by** *force*

finally show *?thesis*
by (*metis le-iff-sup inf-le2 n-less-eq-char*)
qed

AACP Theorem 4.1

subclass *apx-biorder*
apply *unfold-locales*
apply (*simp add*: *apx-def inf.coboundedI2*)
apply (*metis sup-same-context order.antisym apx-def apx-meet-L relative-equality*)
using *apx-transitive-2* **by** *blast*

lemma *sup-apx-left-isotone-2*:
assumes $x \sqsubseteq y$
shows $x \sqcup z \sqsubseteq y \sqcup z$
proof −
have *1*: $x \sqcup z \leq y \sqcup z \sqcup L$
by (*smt assms sup-assoc sup-commute sup-left-isotone apx-def*)
have $C\ (y \sqcup z) \leq x \sqcup n(x) * top \sqcup C\ z$
using *assms apx-def inf-sup-distrib1 sup-left-isotone* **by** *auto*
also have ... $\leq x \sqcup z \sqcup n(x) * top$
using *inf.coboundedI1 inf.sup-monoid.add-commute sup.cobounded1 sup.cobounded2 sup-assoc sup-least sup-right-isotone* **by** *auto*
also have ... $\leq x \sqcup z \sqcup n(x \sqcup z) * top$
using *mult-isotone n-left-upper-bound semiring.add-left-mono* **by** *force*
finally show *?thesis*
using *1 apx-def* **by** *blast*
qed

lemma *mult-apx-left-isotone-2*:
assumes $x \sqsubseteq y$
shows $x * z \sqsubseteq y * z$
proof −
have $x * z \leq y * z \sqcup L * z$
by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
hence *1*: $x * z \leq y * z \sqcup L$
using *n-L-below-L order-lesseq-imp semiring.add-left-mono* **by** *blast*
have $C\ (y * z) = C\ y * z$
by (*simp add*: *n-L-T-meet-mult*)
also have ... $\leq x * z \sqcup n(x) * top * z$
by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
also have ... $\leq x * z \sqcup n(x * z) * top$
by (*simp add*: *n-top-split*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

lemma *mult-apx-right-isotone-2*:
assumes $x \sqsubseteq y$

shows $z * x \sqsubseteq z * y$
proof –
have $z * x \leq z * y \sqcup z * L$
by (*metis assms apx-def mult-left-dist-sup mult-right-isotone*)
also have ... $\leq z * y \sqcup z * bot \sqcup L$
using *n-L-split-L semiring.add-left-mono sup-assoc* **by** *presburger*
finally have *1*: $z * x \leq z * y \sqcup L$
using *mult-right-isotone sup.absorb-iff1* **by** *auto*
have $C\ (z * y) \leq z * C\ y$
by (*simp add*: *n-L-T-meet-mult n-L-T-meet-mult-propagate*)
also have ... $\leq z * (x \sqcup n(x) * top)$
using *assms apx-def mult-right-isotone* **by** *blast*
also have ... $= z * x \sqcup z * n(x) * top$
by (*simp add*: *mult-left-dist-sup mult-assoc*)
also have ... $\leq z * x \sqcup n(z * x) * top$
by (*simp add*: *n-split-top*)
finally show *?thesis*
using *1 apx-def* **by** *blast*
qed

AACP Theorem 4.1 and Theorem 4.2

subclass *apx-semiring*
apply *unfold-locales*
apply (*simp add*: *apx-def n-L-below-nL-top sup.absorb2*)
using *sup-apx-left-isotone-2* **apply** *blast*
using *mult-apx-left-isotone-2* **apply** *blast*
by (*simp add*: *mult-apx-right-isotone-2*)

AACP Theorem 4.2

lemma *meet-L-apx-isotone*:
$x \sqsubseteq y \Longrightarrow x \sqcap L \sqsubseteq y \sqcap L$
by (*smt* (*verit*) *apx-meet-L apx-def inf.cobounded2 inf.left-commute n-L-top-meet-L n-less-eq-char sup.absorb2*)

AACP Theorem 4.2

lemma *n-L-apx-isotone*:
assumes $x \sqsubseteq y$
shows $n(x) * L \sqsubseteq n(y) * L$
proof –
have $C\ (n(y) * L) \leq n(C\ y) * L$
by (*simp add*: *n-C*)
also have ... $\leq n(x) * L \sqcup n(n(x) * L) * top$
by (*metis assms apx-def n-add-n-top n-galois n-isotone n-n-L*)
finally show *?thesis*
using *apx-def le-inf-iff n-L-decreasing-meet-L sup.absorb2* **by** *auto*
qed

definition *kappa-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-apx-meet* $f \equiv$ *apx.has-least-fixpoint* $f \wedge$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \kappa\ f = \mu\ f \vartriangle \nu\ f$

definition *kappa-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *kappa-mu-nu* $f \equiv apx.has\text{-}least\text{-}fixpoint\ f \wedge \kappa\ f = \mu\ f \sqcup (\nu\ f \sqcap L)$

definition *nu-below-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *nu-below-mu-nu* $f \equiv C\ (\nu\ f) \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\nu\ f) * top$

definition *nu-below-mu-nu-2* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *nu-below-mu-nu-2* $f \equiv C\ (\nu\ f) \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\mu\ f \sqcup (\nu\ f \sqcap L))$
$* top$

definition *mu-nu-apx-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *mu-nu-apx-nu* $f \equiv \mu\ f \sqcup (\nu\ f \sqcap L) \sqsubseteq \nu\ f$

definition *mu-nu-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *mu-nu-apx-meet* $f \equiv has\text{-}apx\text{-}meet\ (\mu\ f)\ (\nu\ f) \wedge \mu\ f \triangle \nu\ f = \mu\ f \sqcup (\nu\ f$
$\sqcap L)$

definition *apx-meet-below-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
 where *apx-meet-below-nu* $f \equiv has\text{-}apx\text{-}meet\ (\mu\ f)\ (\nu\ f) \wedge \mu\ f \triangle \nu\ f \leq \nu\ f$

lemma *mu-below-l*:
 $\mu\ f \leq \mu\ f \sqcup (\nu\ f \sqcap L)$
 by *simp*

lemma *l-below-nu*:
 $has\text{-}least\text{-}fixpoint\ f \Longrightarrow has\text{-}greatest\text{-}fixpoint\ f \Longrightarrow \mu\ f \sqcup (\nu\ f \sqcap L) \leq \nu\ f$
 by (*simp add*: *mu-below-nu*)

lemma *n-l-nu*:
 $has\text{-}least\text{-}fixpoint\ f \Longrightarrow has\text{-}greatest\text{-}fixpoint\ f \Longrightarrow (\mu\ f \sqcup (\nu\ f \sqcap L)) \sqcap L = \nu\ f$
$\sqcap L$
 by (*meson l-below-nu inf.cobounded1 inf.sup-same-context order-trans sup-ge2*)

lemma *l-apx-mu*:
 $\mu\ f \sqcup (\nu\ f \sqcap L) \sqsubseteq \mu\ f$
proof −
 have *1*: $\mu\ f \sqcup (\nu\ f \sqcap L) \leq \mu\ f \sqcup L$
  using *sup-right-isotone* **by** *auto*
 have $C\ (\mu\ f) \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\mu\ f \sqcup (\nu\ f \sqcap L)) * top$
  by (*simp add*: *le-supI1*)
 thus *?thesis*
  using *1 apx-def* **by** *blast*
qed

AACP Theorem 4.8 implies Theorem 4.9

lemma *nu-below-mu-nu-nu-below-mu-nu-2*:
 assumes *nu-below-mu-nu f*
  shows *nu-below-mu-nu-2 f*

proof −
  have $C\ (\nu\ f) = C\ (C\ (\nu\ f))$
    by *auto*
  also have ... $\leq C\ (\mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\nu\ f) * top)$
    using *assms nu-below-mu-nu-def* **by** *auto*
  also have ... $= C\ (\mu\ f \sqcup (\nu\ f \sqcap L)) \sqcup C\ (n(\nu\ f) * top)$
    using *inf-sup-distrib1* **by** *auto*
  also have ... $= C\ (\mu\ f \sqcup (\nu\ f \sqcap L)) \sqcup n(\nu\ f) * top$
    by (*simp add*: *C-n-mult-closed*)
  also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\nu\ f) * top$
    using *inf-le2 sup-left-isotone* **by** *blast*
  also have ... $= \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\nu\ f \sqcap L) * top$
    using *n-n-meet-L* **by** *auto*
  also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\mu\ f \sqcup (\nu\ f \sqcap L)) * top$
    using *mult-isotone n-right-upper-bound semiring.add-left-mono* **by** *auto*
  finally show *?thesis*
    by (*simp add*: *nu-below-mu-nu-2-def*)
qed

AACP Theorem 4.9 implies Theorem 4.8

lemma *nu-below-mu-nu-2-nu-below-mu-nu*:
  assumes *has-least-fixpoint f*
      and *has-greatest-fixpoint f*
      and *nu-below-mu-nu-2 f*
    shows *nu-below-mu-nu f*
proof −
  have $C\ (\nu\ f) \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\mu\ f \sqcup (\nu\ f \sqcap L)) * top$
    using *assms*(*3*) *nu-below-mu-nu-2-def* **by** *blast*
  also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup n(\nu\ f) * top$
    by (*metis assms*(*1*,*2*) *order.eq-iff n-n-meet-L n-l-nu*)
  finally show *?thesis*
    using *nu-below-mu-nu-def* **by** *blast*
qed

lemma *nu-below-mu-nu-equivalent*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ (*nu-below-mu-nu f* $\longleftrightarrow$ *nu-below-mu-nu-2 f*)
  using *nu-below-mu-nu-2-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *blast*

AACP Theorem 4.9 implies Theorem 4.10

lemma *nu-below-mu-nu-2-mu-nu-apx-nu*:
  assumes *has-least-fixpoint f*
      and *has-greatest-fixpoint f*
      and *nu-below-mu-nu-2 f*
    shows *mu-nu-apx-nu f*
proof −
  have $\mu\ f \sqcup (\nu\ f \sqcap L) \leq \nu\ f \sqcup L$
    using *assms*(*1*,*2*) *l-below-nu le-supI1* **by** *blast*

thus *?thesis*
using *assms*(*3*) *apx-def mu-nu-apx-nu-def nu-below-mu-nu-2-def* **by** *blast*
qed

AACP Theorem 4.10 implies Theorem 4.11

lemma *mu-nu-apx-nu-mu-nu-apx-meet*:
assumes *mu-nu-apx-nu f*
shows *mu-nu-apx-meet f*
proof −
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
have *is-apx-meet* $(\mu\ f)\ (\nu\ f)\ ?l$
proof (*unfold is-apx-meet-def*, *intro conjI*)
show $?l \sqsubseteq \mu\ f$
by (*simp add*: *l-apx-mu*)
show $?l \sqsubseteq \nu\ f$
using *assms mu-nu-apx-nu-def* **by** *blast*
show $\forall w.\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow w \sqsubseteq\ ?l$
by (*metis apx-meet-L le-inf-iff sup.absorb1 sup-apx-left-isotone*)
qed
thus *?thesis*
by (*simp add*: *apx-meet-char mu-nu-apx-meet-def*)
qed

AACP Theorem 4.11 implies Theorem 4.12

lemma *mu-nu-apx-meet-apx-meet-below-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *mu-nu-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def l-below-nu mu-nu-apx-meet-def* **by** *auto*

AACP Theorem 4.12 implies Theorem 4.9

lemma *apx-meet-below-nu-nu-below-mu-nu-2*:
assumes *apx-meet-below-nu f*
shows *nu-below-mu-nu-2 f*
proof −
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
have $\forall m\ .\ m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \le \nu\ f \longrightarrow C\ (\nu\ f) \le\ ?l \sqcup n(?l) * top$
proof
fix m
show $m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \le \nu\ f \longrightarrow C\ (\nu\ f) \le\ ?l \sqcup n(?l) * top$
proof
assume *1*: $m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \le \nu\ f$
hence $m \le\ ?l$
by (*smt* (*z3*) *apx-def sup.left-commute sup-inf-distrib1 sup-left-divisibility*)
hence $m \sqcup n(m) * top \le\ ?l \sqcup n(?l) * top$
by (*metis sup-mono mult-left-isotone n-isotone*)
thus $C\ (\nu\ f) \le\ ?l \sqcup n(?l) * top$
using *1 apx-def order.trans* **by** *blast*
qed
qed

thus *?thesis*
by (*smt* (*verit*, *ccfv-threshold*) *assms apx-meet-below-nu-def apx-meet-same apx-meet-unique is-apx-meet-def nu-below-mu-nu-2-def*)
qed

AACP Theorem 4.5 implies Theorem 4.6

lemma *has-apx-least-fixpoint-kappa-apx-meet*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *apx.has-least-fixpoint f*
shows *kappa-apx-meet f*
proof −
have *1*: $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow C\ (\kappa\ f) \leq w \sqcup n(w) * top$
by (*metis assms*(*2*,*3*) *apx-def inf.sup-right-isotone order-trans kappa-below-nu*)
have $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow w \leq \kappa\ f \sqcup L$
by (*metis assms*(*1*,*3*) *sup-left-isotone apx-def mu-below-kappa order-trans*)
hence $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow w \sqsubseteq \kappa\ f$
using *1 apx-def* **by** *blast*
hence *is-apx-meet* $(\mu\ f)\ (\nu\ f)\ (\kappa\ f)$
by (*simp add*: *assms is-apx-meet-def kappa-apx-below-mu kappa-apx-below-nu*)
thus *?thesis*
by (*simp add*: *assms*(*3*) *kappa-apx-meet-def apx-meet-char*)
qed

AACP Theorem 4.6 implies Theorem 4.12

lemma *kappa-apx-meet-apx-meet-below-nu*:
has-greatest-fixpoint f $\Longrightarrow$ *kappa-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def kappa-apx-meet-def kappa-below-nu* **by** *force*

AACP Theorem 4.12 implies Theorem 4.7

lemma *apx-meet-below-nu-kappa-mu-nu*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *isotone f*
and *apx.isotone f*
and *apx-meet-below-nu f*
shows *kappa-mu-nu f*
proof −
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
let $?m = \mu\ f \bigtriangleup \nu\ f$
have *1*: $?m = ?l$
by (*metis assms*(*1*,*2*,*5*) *apx-meet-below-nu-nu-below-mu-nu-2 mu-nu-apx-meet-def mu-nu-apx-nu-mu-nu-apx-meet nu-below-mu-nu-2-mu-nu-apx-nu*)
have *2*: $?l \leq f(?l) \sqcup L$
proof −
have $?l \leq \mu\ f \sqcup L$
using *sup-right-isotone* **by** *auto*
also have ... $= f(\mu\ f) \sqcup L$

by (*simp add*: *assms*(*1*) *mu-unfold*)
also have ... $\leq f(?l) \sqcup L$
using *assms*(*3*) *isotone-def sup-ge1 sup-left-isotone* **by** *blast*
finally show $?l \leq f(?l) \sqcup L$
.
qed
have $C\ (f(?l)) \leq\ ?l \sqcup n(?l) * top$
proof −
have $C\ (f(?l)) \leq C\ (f(\nu\ f))$
using *assms*(*1*−*3*) *l-below-nu inf.sup-right-isotone isotone-def* **by** *blast*
also have ... $= C\ (\nu\ f)$
by (*metis assms*(*2*) *nu-unfold*)
also have ... $\leq\ ?l \sqcup n(?l) * top$
by (*metis assms*(*5*) *apx-meet-below-nu-nu-below-mu-nu-2*
nu-below-mu-nu-2-def)
finally show $C\ (f(?l)) \leq\ ?l \sqcup n(?l) * top$
.
qed
hence *3*: $?l \sqsubseteq f(?l)$
using *2 apx-def* **by** *blast*
have *4*: $f(?l) \sqsubseteq \mu\ f$
proof −
have $?l \sqsubseteq \mu\ f$
by (*simp add*: *l-apx-mu*)
thus $f(?l) \sqsubseteq \mu\ f$
by (*metis assms*(*1*,*4*) *mu-unfold ord.isotone-def*)
qed
have $f(?l) \sqsubseteq \nu\ f$
proof −
have $?l \sqsubseteq \nu\ f$
using *1*
by (*metis apx-meet-below-nu-def assms*(*5*) *apx-meet is-apx-meet-def*)
thus $f(?l) \sqsubseteq \nu\ f$
by (*metis assms*(*2*,*4*) *nu-unfold ord.isotone-def*)
qed
hence $f(?l) \sqsubseteq\ ?l$
using *1 4 apx-meet-below-nu-def assms*(*5*) *apx-meet is-apx-meet-def* **by**
fastforce
hence *5*: $f(?l) =\ ?l$
using *3 apx.order.antisym* **by** *blast*
have $\forall y\ .\ f(y) = y \longrightarrow\ ?l \sqsubseteq y$
proof
fix y
show $f(y) = y \longrightarrow\ ?l \sqsubseteq y$
proof
assume *6*: $f(y) = y$
hence *7*: $?l \leq y \sqcup L$
using *assms*(*1*) *inf.cobounded2 is-least-fixpoint-def least-fixpoint*
semiring.add-mono **by** *blast*

have $y \le \nu f$
using *6 assms(2) greatest-fixpoint is-greatest-fixpoint-def* **by** *auto*
hence $C\ y \le\ ?l \sqcup n(?l) * top$
using *assms(5) apx-meet-below-nu-nu-below-mu-nu-2 inf.sup-right-isotone nu-below-mu-nu-2-def order-trans* **by** *blast*
thus $?l \sqsubseteq y$
using *7 apx-def* **by** *blast*
qed
qed
thus *?thesis*
using *5 apx.least-fixpoint-same apx.has-least-fixpoint-def apx.is-least-fixpoint-def kappa-mu-nu-def* **by** *auto*
qed

AACP Theorem 4.7 implies Theorem 4.5

lemma *kappa-mu-nu-has-apx-least-fixpoint*:
kappa-mu-nu f $\Longrightarrow$ *apx.has-least-fixpoint f*
by (*simp add*: *kappa-mu-nu-def*)

AACP Theorem 4.8 implies Theorem 4.7

lemma *nu-below-mu-nu-kappa-mu-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu f* $\Longrightarrow$ *kappa-mu-nu f*
using *apx-meet-below-nu-kappa-mu-nu mu-nu-apx-meet-apx-meet-below-nu mu-nu-apx-nu-mu-nu-apx-meet nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *blast*

AACP Theorem 4.7 implies Theorem 4.8

lemma *kappa-mu-nu-nu-below-mu-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu f* $\Longrightarrow$ *nu-below-mu-nu f*
by (*simp add*: *apx-meet-below-nu-nu-below-mu-nu-2 has-apx-least-fixpoint-kappa-apx-meet kappa-apx-meet-apx-meet-below-nu kappa-mu-nu-has-apx-least-fixpoint nu-below-mu-nu-2-nu-below-mu-nu*)

definition *kappa-mu-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-mu-nu-L f* $\equiv$ *apx.has-least-fixpoint f* $\wedge\ \kappa\ f = \mu\ f \sqcup n(\nu\ f) * L$

definition *nu-below-mu-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *nu-below-mu-nu-L f* $\equiv C\ (\nu\ f) \le \mu\ f \sqcup n(\nu\ f) * top$

definition *mu-nu-apx-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *mu-nu-apx-nu-L f* $\equiv \mu\ f \sqcup n(\nu\ f) * L \sqsubseteq \nu\ f$

definition *mu-nu-apx-meet-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *mu-nu-apx-meet-L f* $\equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \vartriangle \nu\ f = \mu\ f \sqcup n(\nu\ f) * L$

lemma *n-below-l*:

$x \sqcup n(y) * L \leq x \sqcup (y \sqcap L)$
using *n-L-decreasing-meet-L semiring.add-left-mono* **by** *auto*

lemma *n-equal-l*:
assumes *nu-below-mu-nu-L f*
shows $\mu\ f \sqcup n(\nu\ f) * L = \mu\ f \sqcup (\nu\ f \sqcap L)$
proof −
have $\nu\ f \sqcap L \leq (\mu\ f \sqcup n(\nu\ f) * top) \sqcap L$
by (*meson assms order.trans inf.boundedI inf.cobounded2 meet-L-below-C nu-below-mu-nu-L-def*)
also have ... $\leq \mu\ f \sqcup (n(\nu\ f) * top \sqcap L)$
by (*simp add*: *inf.coboundedI2 inf.sup-monoid.add-commute inf-sup-distrib1*)
also have ... $\leq \mu\ f \sqcup n(\nu\ f) * L$
by (*simp add*: *n-T-meet-L*)
finally have $\mu\ f \sqcup (\nu\ f \sqcap L) \leq \mu\ f \sqcup n(\nu\ f) * L$
by *simp*
thus $\mu\ f \sqcup n(\nu\ f) * L = \mu\ f \sqcup (\nu\ f \sqcap L)$
by (*meson order.antisym n-below-l*)
qed

AACP Theorem 4.14 implies Theorem 4.8

lemma *nu-below-mu-nu-L-nu-below-mu-nu*:
nu-below-mu-nu-L f $\Longrightarrow$ *nu-below-mu-nu f*
by (*metis sup-assoc sup-right-top mult-left-dist-sup n-equal-l nu-below-mu-nu-L-def nu-below-mu-nu-def*)

AACP Theorem 4.14 implies Theorem 4.13

lemma *nu-below-mu-nu-L-kappa-mu-nu-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *kappa-mu-nu-L f*
using *kappa-mu-nu-L-def kappa-mu-nu-def n-equal-l nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-kappa-mu-nu* **by** *force*

AACP Theorem 4.14 implies Theorem 4.15

lemma *nu-below-mu-nu-L-mu-nu-apx-nu-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-nu-L f*
using *mu-nu-apx-nu-L-def mu-nu-apx-nu-def n-equal-l nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *auto*

AACP Theorem 4.14 implies Theorem 4.16

lemma *nu-below-mu-nu-L-mu-nu-apx-meet-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-meet-L f*
using *mu-nu-apx-meet-L-def mu-nu-apx-meet-def mu-nu-apx-nu-mu-nu-apx-meet n-equal-l nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *auto*

AACP Theorem 4.15 implies Theorem 4.14

lemma *mu-nu-apx-nu-L-nu-below-mu-nu-L*:
  assumes *has-least-fixpoint f*
    and *has-greatest-fixpoint f*
    and *mu-nu-apx-nu-L f*
  shows *nu-below-mu-nu-L f*
proof −
  let $?n = \mu\ f \sqcup n(\nu\ f) * L$
  let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
  have $C\ (\nu\ f) \leq\ ?n \sqcup n(?n) * top$
    using *assms*(*3*) *apx-def mu-nu-apx-nu-L-def* **by** *blast*
  also have ... $\leq\ ?n \sqcup n(?l) * top$
    using *mult-left-isotone n-L-decreasing-meet-L n-isotone*
semiring.add-left-mono **by** *auto*
  also have ... $\leq\ ?n \sqcup n(\nu\ f) * top$
    using *assms*(*1*,*2*) *l-below-nu mult-left-isotone n-isotone sup-right-isotone* **by**
auto
  finally show *?thesis*
    by (*metis sup-assoc sup-right-top mult-left-dist-sup nu-below-mu-nu-L-def*)
qed

AACP Theorem 4.13 implies Theorem 4.15

lemma *kappa-mu-nu-L-mu-nu-apx-nu-L*:
  has-greatest-fixpoint f $\Longrightarrow$ *kappa-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-nu-L f*
  using *kappa-mu-nu-L-def kappa-apx-below-nu mu-nu-apx-nu-L-def* **by** *fastforce*

AACP Theorem 4.16 implies Theorem 4.15

lemma *mu-nu-apx-meet-L-mu-nu-apx-nu-L*:
  mu-nu-apx-meet-L f $\Longrightarrow$ *mu-nu-apx-nu-L f*
  using *apx-meet-char is-apx-meet-def mu-nu-apx-meet-L-def mu-nu-apx-nu-L-def*
by *fastforce*

AACP Theorem 4.13 implies Theorem 4.14

lemma *kappa-mu-nu-L-nu-below-mu-nu-L*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu-L f* $\Longrightarrow$
nu-below-mu-nu-L f
  by (*simp add*: *kappa-mu-nu-L-mu-nu-apx-nu-L*
mu-nu-apx-nu-L-nu-below-mu-nu-L)

proposition *nu-below-mu-nu-nu-below-mu-nu-L*: *nu-below-mu-nu f* $\longrightarrow$
nu-below-mu-nu-L f **nitpick** [*expect*=*genuine*,*card*=*3*] **oops**

lemma *unfold-fold-1*:
  isotone f $\Longrightarrow$ *has-least-prefixpoint f* $\Longrightarrow$ *apx.has-least-fixpoint f* $\Longrightarrow f(x) \leq x$
$\Longrightarrow \kappa\ f \leq x \sqcup L$
  by (*metis sup-left-isotone apx-def has-least-fixpoint-def is-least-prefixpoint-def*
least-prefixpoint-char least-prefixpoint-fixpoint order-trans pmu-mu
kappa-apx-below-mu)

lemma *unfold-fold-2*:

assumes *isotone f*
and *apx.isotone f*
and *has-least-prefixpoint f*
and *has-greatest-fixpoint f*
and *apx.has-least-fixpoint f*
and $f(x) \le x$
and $\kappa\ f \sqcap L \le x \sqcap L$
shows $\kappa\ f \le x$
proof –
have $\kappa\ f \sqcap L = \nu\ f \sqcap L$
by (*smt* (*z3*) *apx-meet-L assms*(*4*,*5*) *order.eq-iff inf.cobounded1 kappa-apx-below-nu kappa-below-nu le-inf-iff*)
hence $\kappa\ f = (\kappa\ f \sqcap L) \sqcup \mu\ f$
by (*metis assms*(*1–5*) *apx-meet-below-nu-kappa-mu-nu has-apx-least-fixpoint-kappa-apx-meet sup-commute least-fixpoint-char least-prefixpoint-fixpoint kappa-apx-meet-apx-meet-below-nu kappa-mu-nu-def*)
thus *?thesis*
by (*metis assms*(*1*,*3*,*6*,*7*) *sup-least is-least-prefixpoint-def least-prefixpoint le-inf-iff pmu-mu*)
qed

end

class *n-algebra-apx-2* = *n-algebra* + *apx* +
assumes *apx-def*: $x \sqsubseteq y \longleftrightarrow x \le y \sqcup L \wedge y \le x \sqcup n(x) * top$
begin

lemma *apx-transitive-2*:
assumes $x \sqsubseteq y$
and $y \sqsubseteq z$
shows $x \sqsubseteq z$
proof –
have $z \le y \sqcup n(y) * top$
using *assms*(*2*) *apx-def* **by** *auto*
also have ... $\le x \sqcup n(x) * top \sqcup n(y) * top$
using *assms*(*1*) *apx-def sup-left-isotone* **by** *blast*
also have ... $\le x \sqcup n(x) * top$
by (*metis assms*(*1*) *sup-assoc sup-idem sup-right-isotone apx-def mult-left-isotone n-add-n-top n-isotone*)
finally show *?thesis*
by (*smt assms sup-assoc sup-commute apx-def le-iff-sup*)
qed

lemma *apx-meet-L*:
assumes $y \sqsubseteq x$
shows $x \sqcap L \le y \sqcap L$
proof –
have $x \sqcap L \le (y \sqcap L) \sqcup (n(y) * top \sqcap L)$
by (*metis assms apx-def inf.sup-left-isotone inf-sup-distrib2*)

also have ... $\leq (y \sqcap L) \sqcup n(y \sqcap L) * top$
  using *n-n-meet-L sup-right-isotone* **by** *force*
finally show *?thesis*
  by (*metis le-iff-sup inf-le2 n-less-eq-char*)
qed

AACP Theorem 4.1

subclass *apx-biorder*
  apply *unfold-locales*
  apply (*simp add*: *apx-def*)
  using *apx-def order.eq-iff n-less-eq-char* **apply** *blast*
  using *apx-transitive-2* **by** *blast*

lemma *sup-apx-left-isotone-2*:
  assumes $x \sqsubseteq y$
    shows $x \sqcup z \sqsubseteq y \sqcup z$
proof –
  have *1*: $x \sqcup z \leq y \sqcup z \sqcup L$
    by (*smt assms sup-assoc sup-commute sup-left-isotone apx-def*)
  have $y \sqcup z \leq x \sqcup n(x) * top \sqcup z$
    using *assms apx-def sup-left-isotone* **by** *blast*
  also have ... $\leq x \sqcup z \sqcup n(x \sqcup z) * top$
    by (*metis sup-assoc sup-commute sup-right-isotone mult-left-isotone n-right-upper-bound*)
  finally show *?thesis*
    using *1 apx-def* **by** *auto*
qed

lemma *mult-apx-left-isotone-2*:
  assumes $x \sqsubseteq y$
    shows $x * z \sqsubseteq y * z$
proof –
  have $x * z \leq y * z \sqcup L * z$
    by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
  hence *1*: $x * z \leq y * z \sqcup L$
    using *n-L-below-L order-lesseq-imp semiring.add-left-mono* **by** *blast*
  have $y * z \leq x * z \sqcup n(x) * top * z$
    by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
  also have ... $\leq x * z \sqcup n(x * z) * top$
    by (*simp add*: *n-top-split*)
  finally show *?thesis*
    using *1* **by** (*simp add*: *apx-def*)
qed

lemma *mult-apx-right-isotone-2*:
  assumes $x \sqsubseteq y$
    shows $z * x \sqsubseteq z * y$
proof –
  have $z * x \leq z * y \sqcup z * L$

by (*metis assms apx-def mult-left-dist-sup mult-right-isotone*)
also have ... $\leq z * y \sqcup z * bot \sqcup L$
using *n-L-split-L semiring.add-left-mono sup-assoc* **by** *auto*
finally have *1*: $z * x \leq z * y \sqcup L$
using *mult-right-isotone sup.absorb-iff1* **by** *force*
have $z * y \leq z * (x \sqcup n(x) * top)$
using *assms apx-def mult-right-isotone* **by** *blast*
also have ... $= z * x \sqcup z * n(x) * top$
by (*simp add*: *mult-left-dist-sup mult-assoc*)
also have ... $\leq z * x \sqcup n(z * x) * top$
by (*simp add*: *n-split-top*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

end

end

19 N-Omega-Algebras

theory *N-Omega-Algebras*

imports *Omega-Algebras Recursion*

begin

class *itering-apx* = *bounded-itering* + *n-algebra-apx*
begin

lemma *circ-L*:
$L^\circ = L \sqcup 1$
by (*metis sup-commute mult-top-circ n-L-top-L*)

lemma *C-circ-import*:
$C\ (x^\circ) \leq (C\ x)^\circ$
proof −
have *1*: $C\ x * x^\circ \leq (C\ x)^\circ * C\ x$
using *C-mult-propagate circ-simulate order.eq-iff* **by** *blast*
have $C\ (x^\circ) = C\ (1 \sqcup x * x^\circ)$
by (*simp add*: *circ-left-unfold*)
also have ... $= C\ 1 \sqcup C\ (x * x^\circ)$
by (*simp add*: *inf-sup-distrib1*)
also have ... $\leq 1 \sqcup C\ (x * x^\circ)$
using *sup-left-isotone* **by** *auto*
also have ... $= 1 \sqcup C\ x * x^\circ$
by (*simp add*: *n-L-T-meet-mult*)
also have ... $\leq (C\ x)^\circ$
using *1* **by** (*meson circ-reflexive order.trans le-supI right-plus-below-circ*)

finally show *?thesis*

.

qed

AACP Theorem 4.3 and Theorem 4.4

lemma *circ-apx-isotone*:
assumes $x \sqsubseteq y$
shows $x^\circ \sqsubseteq y^\circ$
proof –
have *1*: $x \leq y \sqcup L \wedge C\ y \leq x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
have $C\ (y^\circ) \leq (C\ y)^\circ$
by (*simp add*: *C-circ-import*)
also have ... $\leq x^\circ \sqcup x^\circ * n(x) * top$
using *1* **by** (*metis circ-isotone circ-left-top circ-unfold-sum mult-assoc*)
also have ... $\leq x^\circ \sqcup (x^\circ * bot \sqcup n(x^\circ * x) * top)$
using *n-n-top-split-n-top sup-right-isotone* **by** *blast*
also have ... $\leq x^\circ \sqcup (x^\circ * bot \sqcup n(x^\circ) * top)$
using *circ-plus-same left-plus-below-circ mult-left-isotone n-isotone sup-right-isotone* **by** *auto*
also have ... $= x^\circ \sqcup n(x^\circ) * top$
by (*meson sup.left-idem sup-relative-same-increasing zero-right-mult-decreasing*)
finally have *2*: $C\ (y^\circ) \leq x^\circ \sqcup n(x^\circ) * top$

.

have $x^\circ \leq y^\circ * L^\circ$
using *1* **by** (*metis circ-sup-1 circ-back-loop-fixpoint circ-isotone n-L-below-L le-iff-sup mult-assoc*)
also have ... $= y^\circ \sqcup y^\circ * L$
using *circ-L mult-left-dist-sup sup-commute* **by** *auto*
also have ... $\leq y^\circ \sqcup y^\circ * bot \sqcup L$
using *n-L-split-L semiring.add-left-mono sup-assoc* **by** *auto*
finally have $x^\circ \leq y^\circ \sqcup L$
using *sup.absorb1 zero-right-mult-decreasing* **by** *force*
thus $x^\circ \sqsubseteq y^\circ$
using *2* **by** (*simp add*: *apx-def*)
qed

end

class *n-omega-algebra-1* = *bounded-left-zero-omega-algebra* + *n-algebra-apx* + *Omega* +
assumes *Omega-def*: $x^\Omega = n(x^\omega) * L \sqcup x^\star$
begin

AACP Theorem 8.13

lemma *C-omega-export*:
$C\ (x^\omega) = (C\ x)^\omega$
proof –

have $C\ (x^\omega) = C\ x * C\ (x^\omega)$
by (*metis C-mult-propagate n-L-T-meet-mult omega-unfold*)
hence *1*: $C\ (x^\omega) \le (C\ x)^\omega$
using *eq-refl omega-induct-mult* **by** *auto*
have $(C\ x)^\omega = C\ (x * (C\ x)^\omega)$
using *n-L-T-meet-mult omega-unfold* **by** *auto*
also have ... $\le C\ (x^\omega)$
by (*metis calculation C-decreasing inf-le1 le-infI omega-induct-mult*)
finally show *?thesis*
using *1 order.antisym* **by** *blast*
qed

AACP Theorem 8.2

lemma *L-mult-star*:
$L * x^\star = L$
by (*metis n-L-top-L star.circ-left-top mult-assoc*)

AACP Theorem 8.3

lemma *mult-L-star*:
$(x * L)^\star = 1 \sqcup x * L$
by (*metis L-mult-star star.circ-mult-1 mult-assoc*)

lemma *mult-L-omega-below*:
$(x * L)^\omega \le x * L$
by (*metis mult-right-isotone n-L-below-L omega-slide*)

AACP Theorem 8.5

lemma *mult-L-sup-star*:
$(x * L \sqcup y)^\star = y^\star \sqcup y^\star * x * L$
using *L-mult-star mult-1-right mult-left-dist-sup star-sup-1 sup-commute mult-L-star mult-assoc* **by** *auto*

lemma *mult-L-sup-omega-below*:
$(x * L \sqcup y)^\omega \le y^\omega \sqcup y^\star * x * L$
proof –
have $(x * L \sqcup y)^\omega \le y^\star * x * L \sqcup (y^\star * x * L)^\star * y^\omega$
by (*metis sup-commute mult-assoc omega-decompose sup-left-isotone mult-L-omega-below*)
also have ... $\le y^\omega \sqcup y^\star * x * L$
by (*smt* (*z3*) *le-iff-sup le-supI mult-left-dist-sup n-L-below-L star-left-induct sup.cobounded2 sup.left-idem sup.orderE sup-assoc sup-commute mult-assoc*)
finally show *?thesis*
.
qed

lemma *n-Omega-isotone*:
$x \le y \Longrightarrow x^\Omega \le y^\Omega$
by (*metis Omega-def sup-mono mult-left-isotone n-isotone omega-isotone star-isotone*)

lemma *n-star-below-Omega*:
$x^\star \leq x^\Omega$
by (*simp add*: *Omega-def*)

lemma *mult-L-star-mult-below*:
$(x * L)^\star * y \leq y \sqcup x * L$
by (*metis sup-right-isotone mult-assoc mult-right-isotone n-L-below-L star-left-induct*)

end

sublocale *n-omega-algebra-1* < *star*: *itering-apx* **where** *circ* = *star* **..**

class *n-omega-algebra* = *n-omega-algebra-1* + *n-algebra-apx* +
assumes *n-split-omega-mult*: $C\ (x^\omega) \leq x^\star * n(x^\omega) * top$
assumes *tarski*: $x * L \leq x * L * x * L$
begin

AACP Theorem 8.4

lemma *mult-L-omega*:
$(x * L)^\omega = x * L$
apply (*rule order.antisym*)
apply (*rule mult-L-omega-below*)
using *omega-induct-mult tarski mult-assoc* **by** *auto*

AACP Theorem 8.6

lemma *mult-L-sup-omega*:
$(x * L \sqcup y)^\omega = y^\omega \sqcup y^\star * x * L$
apply (*rule order.antisym*)
apply (*rule mult-L-sup-omega-below*)
by (*metis le-supI omega-isotone omega-sub-dist-2 sup.cobounded2 sup-commute mult-L-omega mult-assoc*)

AACP Theorem 8.1

lemma *tarski-mult-top-idempotent*:
$x * L = x * L * x * L$
by (*metis omega-unfold mult-L-omega mult-assoc*)

AACP Theorem 8.7

lemma *n-below-n-omega*:
$n(x) \leq n(x^\omega)$
proof −
have $n(x) * L \leq n(x) * L * n(x) * L$
by (*simp add*: *tarski*)
also have ... $\leq x * n(x) * L$
by (*simp add*: *mult-isotone n-L-decreasing*)
finally have $n(x) * L \leq x^\omega$
by (*simp add*: *omega-induct-mult mult-assoc*)

thus *?thesis*
by (*simp add*: *n-galois*)
qed

AACP Theorem 8.14

lemma *n-split-omega-sup-zero*:
$C\ (x^\omega) \leq x^\star * bot \sqcup n(x^\omega) * top$
proof –
have $n(x^\omega) * top \sqcup x * (x^\star * bot \sqcup n(x^\omega) * top) = n(x^\omega) * top \sqcup x * x^\star * bot \sqcup x * n(x^\omega) * top$
by (*simp add*: *mult-left-dist-sup sup-assoc mult-assoc*)
also have ... $\leq n(x^\omega) * top \sqcup x * x^\star * bot \sqcup x * bot \sqcup n(x^\omega) * top$
by (*metis sup-assoc sup-right-isotone n-n-top-split-n-top omega-unfold*)
also have ... $= x * x^\star * bot \sqcup n(x^\omega) * top$
by (*smt sup-assoc sup-commute sup-left-top sup-bot-right mult-assoc mult-left-dist-sup*)
also have ... $\leq x^\star * bot \sqcup n(x^\omega) * top$
by (*metis sup-left-isotone mult-left-isotone star.left-plus-below-circ*)
finally have $x^\star * n(x^\omega) * top \leq x^\star * bot \sqcup n(x^\omega) * top$
using *star-left-induct mult-assoc* **by** *auto*
thus *?thesis*
using *n-split-omega-mult order-trans* **by** *blast*
qed

lemma *n-split-omega-sup*:
$C\ (x^\omega) \leq x^\star \sqcup n(x^\omega) * top$
by (*metis sup-left-isotone n-split-omega-sup-zero order-trans zero-right-mult-decreasing*)

AACP Theorem 8.12

lemma *n-dist-omega-star*:
$n(y^\omega \sqcup y^\star * z) = n(y^\omega) \sqcup n(y^\star * z)$
proof –
have $n(y^\omega \sqcup y^\star * z) = n(C\ (y^\omega) \sqcup C\ (y^\star * z))$
by (*metis inf-sup-distrib1 n-C*)
also have ... $\leq n(C\ (y^\omega) \sqcup y^\star * z)$
using *n-isotone semiring.add-right-mono sup-commute* **by** *auto*
also have ... $\leq n(y^\star * bot \sqcup n(y^\omega) * top \sqcup y^\star * z)$
using *n-isotone semiring.add-right-mono n-split-omega-sup-zero* **by** *blast*
also have ... $= n(y^\omega) \sqcup n(y^\star * z)$
by (*smt sup-assoc sup-commute sup-bot-right mult-left-dist-sup n-dist-n-add*)
finally show *?thesis*
by (*simp add*: *order.antisym n-isotone*)
qed

lemma *mult-L-sup-circ-below*:
$(x * L \sqcup y)^\Omega \leq n(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L$
proof –
have $(x * L \sqcup y)^\Omega \leq n(y^\omega \sqcup y^\star * x * L) * L \sqcup (x * L \sqcup y)^\star$

by (*simp add*: *Omega-def mult-L-sup-omega*)
also have ... $= n(y^\omega) * L \sqcup n(y^\star * x * L) * L \sqcup (x * L \sqcup y)^\star$
by (*simp add*: *semiring.distrib-right mult-assoc n-dist-omega-star*)
also have ... $\le n(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L$
by (*smt* (*z3*) *le-supI sup.cobounded1 sup-assoc sup-commute sup-idem sup-right-isotone mult-L-sup-star n-L-decreasing*)
finally show *?thesis*
.
qed

lemma *n-mult-omega-L-below-zero*:
$n(y * x^\omega) * L \le y * x^\star * bot \sqcup y * n(x^\omega) * L$
proof –
have $n(y * x^\omega) * L \le C\ (y * x^\omega) \sqcap L$
by (*metis n-C n-L-decreasing-meet-L*)
also have ... $\le y * C\ (x^\omega) \sqcap L$
using *inf.sup-left-isotone n-L-T-meet-mult n-L-T-meet-mult-propagate* **by** *auto*
also have ... $\le y * (x^\star * bot \sqcup n(x^\omega) * top) \sqcap L$
using *inf.sup-left-isotone mult-right-isotone n-split-omega-sup-zero* **by** *auto*
also have ... $= (y * x^\star * bot \sqcap L) \sqcup (y * n(x^\omega) * top \sqcap L)$
using *inf-sup-distrib2 mult-left-dist-sup mult-assoc* **by** *auto*
also have ... $\le (y * x^\star * bot \sqcap L) \sqcup y * n(x^\omega) * L$
using *n-vector-meet-L sup-right-isotone* **by** *auto*
also have ... $\le y * x^\star * bot \sqcup y * n(x^\omega) * L$
using *sup-left-isotone* **by** *auto*
finally show *?thesis*
.
qed

AACP Theorem 8.10

lemma *n-mult-omega-L-star-zero*:
$y * x^\star * bot \sqcup n(y * x^\omega) * L = y * x^\star * bot \sqcup y * n(x^\omega) * L$
apply (*rule order.antisym*)
apply (*simp add*: *n-mult-omega-L-below-zero*)
by (*smt sup-assoc sup-commute sup-bot-left sup-right-isotone mult-assoc mult-left-dist-sup n-n-L-split-n-L*)

AACP Theorem 8.11

lemma *n-mult-omega-L-star*:
$y * x^\star \sqcup n(y * x^\omega) * L = y * x^\star \sqcup y * n(x^\omega) * L$
by (*metis zero-right-mult-decreasing n-mult-omega-L-star-zero sup-relative-same-increasing*)

lemma *n-mult-omega-L-below*:
$n(y * x^\omega) * L \le y * x^\star \sqcup y * n(x^\omega) * L$
using *sup-right-divisibility n-mult-omega-L-star* **by** *blast*

lemma *n-omega-L-below-zero*:
$n(x^\omega) * L \le x * x^\star * bot \sqcup x * n(x^\omega) * L$

by (*metis omega-unfold n-mult-omega-L-below-zero*)

lemma *n-omega-L-below*:
$n(x^\omega) * L \le x^\star \sqcup x * n(x^\omega) * L$
by (*metis omega-unfold n-mult-omega-L-below sup-left-isotone star.left-plus-below-circ order-trans*)

lemma *n-omega-L-star-zero*:
$x * x^\star * bot \sqcup n(x^\omega) * L = x * x^\star * bot \sqcup x * n(x^\omega) * L$
by (*metis n-mult-omega-L-star-zero omega-unfold*)

AACP Theorem 8.8

lemma *n-omega-L-star*:
$x^\star \sqcup n(x^\omega) * L = x^\star \sqcup x * n(x^\omega) * L$
by (*metis star.circ-mult-upper-bound star.left-plus-below-circ bot-least n-omega-L-star-zero sup-relative-same-increasing*)

AACP Theorem 8.9

lemma *n-omega-L-star-zero-star*:
$x^\star * bot \sqcup n(x^\omega) * L = x^\star * bot \sqcup x^\star * n(x^\omega) * L$
by (*metis n-mult-omega-L-star-zero star-mult-omega mult-assoc star.circ-transitive-equal*)

AACP Theorem 8.8

lemma *n-omega-L-star-star*:
$x^\star \sqcup n(x^\omega) * L = x^\star \sqcup x^\star * n(x^\omega) * L$
by (*metis zero-right-mult-decreasing n-omega-L-star-zero-star sup-relative-same-increasing*)

lemma *n-Omega-left-unfold*:
$1 \sqcup x * x^\Omega = x^\Omega$
by (*smt Omega-def sup-assoc sup-commute mult-assoc mult-left-dist-sup n-omega-L-star star.circ-left-unfold*)

lemma *n-Omega-left-slide*:
$(x * y)^\Omega * x \le x * (y * x)^\Omega$
proof –
have $(x * y)^\Omega * x \le x * y * n((x * y)^\omega) * L \sqcup (x * y)^\star * x$
by (*smt Omega-def sup-commute sup-left-isotone mult-assoc mult-right-dist-sup mult-right-isotone n-L-below-L n-omega-L-star*)
also have ... $\le x * (y * bot \sqcup n(y * (x * y)^\omega) * L) \sqcup (x * y)^\star * x$
by (*metis mult-right-isotone n-n-L-split-n-L sup-commute sup-right-isotone mult-assoc*)
also have ... $= x * (y * x)^\Omega$
by (*smt (verit, del-insts) le-supI1 star-slide Omega-def sup-assoc sup-commute le-iff-sup mult-assoc mult-isotone mult-left-dist-sup omega-slide star.circ-increasing star.circ-slide bot-least*)
finally show *?thesis*
.

qed

lemma *n-Omega-sup-1*:
$(x \sqcup y)^\Omega = x^\Omega * (y * x^\Omega)^\Omega$
proof −
have *1*: $(x \sqcup y)^\Omega = n((x^\star * y)^\omega) * L \sqcup n((x^\star * y)^\star * x^\omega) * L \sqcup (x^\star * y)^\star * x^\star$
by (*simp add*: *Omega-def omega-decompose semiring.distrib-right star.circ-sup-9 n-dist-omega-star*)
have $n((x^\star * y)^\omega) * L \le (x^\star * y)^\star \sqcup x^\star * (y * n((x^\star * y)^\omega) * L)$
by (*metis n-omega-L-below mult-assoc*)
also have ... $\le (x^\star * y)^\star \sqcup x^\star * y * bot \sqcup x^\star * n((y * x^\star)^\omega) * L$
by (*smt sup-assoc sup-right-isotone mult-assoc mult-left-dist-sup mult-right-isotone n-n-L-split-n-L omega-slide*)
also have ... $= (x^\star * y)^\star \sqcup x^\star * n((y * x^\star)^\omega) * L$
by (*metis sup-commute le-iff-sup star.circ-sub-dist-1 zero-right-mult-decreasing*)
also have ... $\le x^\star * (y * x^\star)^\star \sqcup x^\star * n((y * x^\star)^\omega) * L$
by (*metis star-outer-increasing star-slide star-star-absorb sup-left-isotone*)
also have ... $\le x^\star * (y * x^\Omega)^\Omega$
by (*metis Omega-def sup-commute mult-assoc mult-left-dist-sup mult-right-isotone n-Omega-isotone n-star-below-Omega*)
also have ... $\le x^\Omega * (y * x^\Omega)^\Omega$
by (*simp add*: *mult-left-isotone n-star-below-Omega*)
finally have *2*: $n((x^\star * y)^\omega) * L \le x^\Omega * (y * x^\Omega)^\Omega$
.
have $n((x^\star * y)^\star * x^\omega) * L \le n(x^\omega) * L \sqcup x^\star * (y * x^\star)^\star \sqcup x^\star * (y * x^\star)^\star * y * n(x^\omega) * L$
by (*smt sup-assoc sup-commute mult-left-one mult-right-dist-sup n-mult-omega-L-below star.circ-mult star.circ-slide*)
also have ... $= n(x^\omega) * L * (y * x^\Omega)^\star \sqcup x^\star * (y * x^\Omega)^\star$
by (*smt Omega-def sup-assoc mult-L-sup-star mult-assoc mult-left-dist-sup L-mult-star*)
also have ... $\le x^\Omega * (y * x^\Omega)^\Omega$
by (*simp add*: *Omega-def mult-isotone*)
finally have *3*: $n((x^\star * y)^\star * x^\omega) * L \le x^\Omega * (y * x^\Omega)^\Omega$
.
have $(x^\star * y)^\star * x^\star \le x^\Omega * (y * x^\Omega)^\Omega$
by (*metis star-slide mult-isotone mult-right-isotone n-star-below-Omega order-trans star-isotone*)
hence *4*: $(x \sqcup y)^\Omega \le x^\Omega * (y * x^\Omega)^\Omega$
using *1 2 3* **by** *simp*
have *5*: $x^\Omega * (y * x^\Omega)^\Omega \le n(x^\omega) * L \sqcup x^\star * n((y * x^\Omega)^\omega) * L \sqcup x^\star * (y * x^\Omega)^\star$
by (*smt Omega-def sup-assoc sup-left-isotone mult-assoc mult-left-dist-sup mult-right-dist-sup mult-right-isotone n-L-below-L*)
have $n(x^\omega) * L \le n((x^\star * y)^\star * x^\omega) * L$
by (*metis sup-commute sup-ge1 mult-left-isotone n-isotone star.circ-loop-fixpoint*)
hence *6*: $n(x^\omega) * L \le (x \sqcup y)^\Omega$
using *1 order-lesseq-imp* **by** *fastforce*

have $x^\star * n((y * x^\Omega)^\omega) * L \le x^\star * n((y * x^\star)^\omega \sqcup (y * x^\star)^\star * y * n(x^\omega) * L) * L$
by (*metis Omega-def mult-L-sup-omega-below mult-assoc mult-left-dist-sup mult-left-isotone mult-right-isotone n-isotone*)
also have ... $\le x^\star * bot \sqcup n(x^\star * ((y * x^\star)^\omega \sqcup (y * x^\star)^\star * y * n(x^\omega) * L)) * L$
by (*simp add*: *n-n-L-split-n-L*)
also have ... $\le x^\star \sqcup n((x^\star * y)^\omega \sqcup x^\star * (y * x^\star)^\star * y * n(x^\omega) * L) * L$
using *omega-slide semiring.distrib-left sup-mono zero-right-mult-decreasing mult-assoc* **by** *fastforce*
also have ... $\le x^\star \sqcup n((x^\star * y)^\omega \sqcup (x^\star * y)^\star * n(x^\omega) * L) * L$
by (*smt sup-right-divisibility sup-right-isotone mult-left-isotone n-isotone star.circ-mult*)
also have ... $\le x^\star \sqcup n((x \sqcup y)^\omega) * L$
by (*metis sup-right-isotone mult-assoc mult-left-isotone mult-right-isotone n-L-decreasing n-isotone omega-decompose*)
also have ... $\le (x \sqcup y)^\Omega$
by (*simp add*: *Omega-def le-supI1 star-isotone sup-commute*)
finally have *7*: $x^\star * n((y * x^\Omega)^\omega) * L \le (x \sqcup y)^\Omega$
.
have $x^\star * (y * x^\Omega)^\star \le (x^\star * y)^\star * x^\star \sqcup (x^\star * y)^\star * n(x^\omega) * L$
by (*smt Omega-def sup-right-isotone mult-L-sup-star mult-assoc mult-left-dist-sup mult-left-isotone star.left-plus-below-circ star-slide*)
also have ... $\le (x^\star * y)^\star * x^\star \sqcup n((x^\star * y)^\star * x^\omega) * L$
by (*simp add*: *n-mult-omega-L-star*)
also have ... $\le (x \sqcup y)^\Omega$
by (*smt Omega-def sup-commute sup-right-isotone mult-left-isotone n-right-upper-bound omega-decompose star.circ-sup*)
finally have $n(x^\omega) * L \sqcup x^\star * n((y * x^\Omega)^\omega) * L \sqcup x^\star * (y * x^\Omega)^\star \le (x \sqcup y)^\Omega$
using *6 7* **by** *simp*
hence $x^\Omega * (y * x^\Omega)^\Omega \le (x \sqcup y)^\Omega$
using *5 order.trans* **by** *blast*
thus *?thesis*
using *4 order.antisym* **by** *blast*
qed

end

sublocale *n-omega-algebra* < *nL-omega*: *left-zero-conway-semiring* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*simp add*: *n-Omega-left-unfold*)
apply (*simp add*: *n-Omega-left-slide*)
by (*simp add*: *n-Omega-sup-1*)

context *n-omega-algebra*
begin

AACP Theorem 8.16

lemma *omega-apx-isotone*:
assumes $x \sqsubseteq y$
shows $x^\omega \sqsubseteq y^\omega$
proof −
have *1*: $x \leq y \sqcup L \wedge C\ y \leq x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
have $n(x) * top \sqcup x * (x^\omega \sqcup n(x^\omega) * top) \leq n(x) * top \sqcup x^\omega \sqcup n(x^\omega) * top$
by (*metis le-supI n-split-top sup.coboundedI sup-assoc mult-assoc mult-left-dist-sup sup-right-isotone omega-unfold*)
also have ... $\leq x^\omega \sqcup n(x^\omega) * top$
by (*metis sup-commute sup-right-isotone mult-left-isotone n-below-n-omega sup-assoc sup-idem*)
finally have *2*: $x^\star * n(x) * top \leq x^\omega \sqcup n(x^\omega) * top$
using *star-left-induct mult-assoc* **by** *auto*
have $C\ (y^\omega) = (C\ y)^\omega$
by (*simp add*: *C-omega-export*)
also have ... $\leq (x \sqcup n(x) * top)^\omega$
using *1 omega-isotone* **by** *blast*
also have ... $= (x^\star * n(x) * top)^\omega \sqcup (x^\star * n(x) * top)^\star * x^\omega$
by (*simp add*: *omega-decompose mult-assoc*)
also have ... $\leq x^\star * n(x) * top \sqcup (x^\star * n(x) * top)^\star * x^\omega$
using *mult-top-omega sup-left-isotone* **by** *blast*
also have ... $= x^\star * n(x) * top \sqcup (1 \sqcup x^\star * n(x) * top * (x^\star * n(x) * top)^\star) * x^\omega$
by (*simp add*: *star-left-unfold-equal*)
also have ... $\leq x^\omega \sqcup x^\star * n(x) * top$
by (*smt sup-mono sup-least mult-assoc mult-left-one mult-right-dist-sup mult-right-isotone order-refl top-greatest sup.cobounded2*)
also have ... $\leq x^\omega \sqcup n(x^\omega) * top$
using *2* **by** *simp*
finally have *3*: $C\ (y^\omega) \leq x^\omega \sqcup n(x^\omega) * top$
.
have $x^\omega \leq (y \sqcup L)^\omega$
using *1 omega-isotone* **by** *simp*
also have ... $= (y^\star * L)^\omega \sqcup (y^\star * L)^\star * y^\omega$
by (*simp add*: *omega-decompose*)
also have ... $= y^\star * L * (y^\star * L)^\omega \sqcup (y^\star * L)^\star * y^\omega$
using *omega-unfold* **by** *auto*
also have ... $\leq y^\star * L \sqcup (y^\star * L)^\star * y^\omega$
by (*metis sup-left-isotone n-L-below-L mult-assoc mult-right-isotone*)
also have ... $= y^\star * L \sqcup (1 \sqcup y^\star * L * (y^\star * L)^\star) * y^\omega$
by (*simp add*: *star-left-unfold-equal*)
also have ... $\leq y^\star * L \sqcup y^\omega$
by (*simp add*: *mult-L-star-mult-below star-left-unfold-equal sup-commute*)
also have ... $\leq y^\star * bot \sqcup L \sqcup y^\omega$
using *n-L-split-L sup-left-isotone* **by** *auto*
finally have $x^\omega \leq y^\omega \sqcup L$
by (*simp add*: *star-bot-below-omega sup.absorb1 sup.left-commute sup-commute*)

thus $x^\omega \sqsubseteq y^\omega$
using *3* **by** (*simp add*: *apx-def*)
qed

lemma *combined-apx-left-isotone*:
$x \sqsubseteq y \Longrightarrow n(x^\omega) * L \sqcup x^\star * z \sqsubseteq n(y^\omega) * L \sqcup y^\star * z$
by (*simp add*: *mult-apx-isotone n-L-apx-isotone star.circ-apx-isotone sup-apx-isotone omega-apx-isotone*)

lemma *combined-apx-left-isotone-2*:
$x \sqsubseteq y \Longrightarrow (x^\omega \sqcap L) \sqcup x^\star * z \sqsubseteq (y^\omega \sqcap L) \sqcup y^\star * z$
by (*metis sup-apx-isotone mult-apx-left-isotone omega-apx-isotone star.circ-apx-isotone meet-L-apx-isotone*)

lemma *combined-apx-right-isotone*:
$y \sqsubseteq z \Longrightarrow n(x^\omega) * L \sqcup x^\star * y \sqsubseteq n(x^\omega) * L \sqcup x^\star * z$
by (*simp add*: *mult-apx-isotone sup-apx-left-isotone sup-commute*)

lemma *combined-apx-right-isotone-2*:
$y \sqsubseteq z \Longrightarrow (x^\omega \sqcap L) \sqcup x^\star * y \sqsubseteq (x^\omega \sqcap L) \sqcup x^\star * z$
by (*simp add*: *mult-apx-right-isotone sup-apx-right-isotone*)

lemma *combined-apx-isotone*:
$x \sqsubseteq y \Longrightarrow w \sqsubseteq z \Longrightarrow n(x^\omega) * L \sqcup x^\star * w \sqsubseteq n(y^\omega) * L \sqcup y^\star * z$
by (*simp add*: *mult-apx-isotone n-L-apx-isotone star.circ-apx-isotone sup-apx-isotone omega-apx-isotone*)

lemma *combined-apx-isotone-2*:
$x \sqsubseteq y \Longrightarrow w \sqsubseteq z \Longrightarrow (x^\omega \sqcap L) \sqcup x^\star * w \sqsubseteq (y^\omega \sqcap L) \sqcup y^\star * z$
by (*meson combined-apx-left-isotone-2 combined-apx-right-isotone-2 apx.order.trans*)

lemma *n-split-nu-mu*:
$C\ (y^\omega \sqcup y^\star * z) \le y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * top$
proof –
have $C\ (y^\omega \sqcup y^\star * z) \le C\ (y^\omega) \sqcup y^\star * z$
by (*simp add*: *inf-sup-distrib1 le-supI1 sup-commute*)
also have ... $\le y^\star * bot \sqcup n(y^\omega) * top \sqcup y^\star * z$
using *n-split-omega-sup-zero sup-left-isotone* **by** *blast*
also have ... $\le y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * top$
using *le-supI1 mult-left-isotone mult-right-isotone n-left-upper-bound sup-right-isotone* **by** *force*
finally show *?thesis*
.
qed

lemma *n-split-nu-mu-2*:
$C\ (y^\omega \sqcup y^\star * z) \le y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L) \sqcup n(y^\omega \sqcup y^\star * z) * top$
proof –

have $C\ (y^\omega \sqcup y^\star * z) \leq C\ (y^\omega) \sqcup y^\star * z$
using *inf.sup-left-isotone sup-inf-distrib2* **by** *auto*
also have ... $\leq y^\star * bot \sqcup n(y^\omega) * top \sqcup y^\star * z$
using *n-split-omega-sup-zero sup-left-isotone* **by** *blast*
also have ... $\leq y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * top$
using *le-supI1 mult-left-isotone mult-right-isotone n-left-upper-bound semiring.add-left-mono* **by** *auto*
finally show *?thesis*
using *order-lesseq-imp semiring.add-right-mono sup.cobounded1* **by** *blast*
qed

lemma *loop-exists*:
$C\ (\nu\ (\lambda x\ .\ y * x \sqcup z)) \leq \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * top$
using *omega-loop-nu star-loop-mu n-split-nu-mu* **by** *auto*

lemma *loop-exists-2*:
$C\ (\nu\ (\lambda x\ .\ y * x \sqcup z)) \leq \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * top$
by (*simp add*: *omega-loop-nu star-loop-mu n-split-nu-mu-2*)

lemma *loop-apx-least-fixpoint*:
apx.is-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (\mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * L)$
proof −
have *kappa-mu-nu-L* $(\lambda x\ .\ y * x \sqcup z)$
by (*metis affine-apx-isotone loop-exists affine-has-greatest-fixpoint affine-has-least-fixpoint affine-isotone nu-below-mu-nu-L-def nu-below-mu-nu-L-kappa-mu-nu-L*)
thus *?thesis*
using *apx.least-fixpoint-char kappa-mu-nu-L-def* **by** *force*
qed

lemma *loop-apx-least-fixpoint-2*:
apx.is-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (\mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L))$
proof −
have *kappa-mu-nu* $(\lambda x\ .\ y * x \sqcup z)$
by (*metis affine-apx-isotone affine-has-greatest-fixpoint affine-has-least-fixpoint affine-isotone loop-exists-2 nu-below-mu-nu-def nu-below-mu-nu-kappa-mu-nu*)
thus *?thesis*
using *apx.least-fixpoint-char kappa-mu-nu-def* **by** *force*
qed

lemma *loop-has-apx-least-fixpoint*:
apx.has-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)$
using *apx.least-fixpoint-char loop-apx-least-fixpoint* **by** *blast*

lemma *loop-semantics*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup n(\nu\ (\lambda x\ .\ y * x \sqcup z)) * L$

using *apx.least-fixpoint-char loop-apx-least-fixpoint* **by** *force*

lemma *loop-semantics-2*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L)$
using *apx.least-fixpoint-char loop-apx-least-fixpoint-2* **by** *force*

AACP Theorem 8.15

lemma *loop-semantics-kappa-mu-nu*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = n(y^\omega) * L \sqcup y^\star * z$
proof –
have $\kappa\ (\lambda x\ .\ y * x \sqcup z) = y^\star * z \sqcup n(y^\omega \sqcup y^\star * z) * L$
using *apx.least-fixpoint-char omega-loop-nu star-loop-mu loop-apx-least-fixpoint* **by** *auto*
thus *?thesis*
by (*smt n-dist-omega-star sup-assoc mult-right-dist-sup sup-commute le-iff-sup n-L-decreasing*)
qed

AACP Theorem 8.15

lemma *loop-semantics-kappa-mu-nu-2*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = (y^\omega \sqcap L) \sqcup y^\star * z$
proof –
have $\kappa\ (\lambda x\ .\ y * x \sqcup z) = y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L)$
using *apx.least-fixpoint-char omega-loop-nu star-loop-mu loop-apx-least-fixpoint-2* **by** *auto*
thus *?thesis*
by (*metis sup-absorb2 sup-ge2 sup-inf-distrib1 sup-monoid.add-commute*)
qed

AACP Theorem 8.16

lemma *loop-semantics-apx-left-isotone*:
$w \sqsubseteq y \Longrightarrow \kappa\ (\lambda x\ .\ w * x \sqcup z) \sqsubseteq \kappa\ (\lambda x\ .\ y * x \sqcup z)$
by (*metis loop-semantics-kappa-mu-nu-2 combined-apx-left-isotone-2*)

AACP Theorem 8.16

lemma *loop-semantics-apx-right-isotone*:
$w \sqsubseteq z \Longrightarrow \kappa\ (\lambda x\ .\ y * x \sqcup w) \sqsubseteq \kappa\ (\lambda x\ .\ y * x \sqcup z)$
by (*metis loop-semantics-kappa-mu-nu-2 combined-apx-right-isotone-2*)

lemma *loop-semantics-apx-isotone*:
$v \sqsubseteq y \Longrightarrow w \sqsubseteq z \Longrightarrow \kappa\ (\lambda x\ .\ v * x \sqcup w) \sqsubseteq \kappa\ (\lambda x\ .\ y * x \sqcup z)$
using *apx-transitive-2 loop-semantics-apx-left-isotone loop-semantics-apx-right-isotone* **by** *blast*

end

end

20 N-Omega Binary Iterings

theory *N-Omega-Binary-Iterings*

imports *N-Omega-Algebras Binary-Iterings-Strict*

begin

sublocale *extended-binary-itering* < *left-zero-conway-semiring* **where** *circ* = (λx . $x \star 1$)
  apply *unfold-locales*
  using *while-left-unfold* **apply** *force*
  apply (*metis mult-1-right while-one-mult-below while-slide*)
  by (*simp add*: *while-one-while while-sumstar-2*)

class *binary-itering-apx* = *bounded-binary-itering* + *n-algebra-apx*
begin

lemma *C-while-import*:
  $C\ (x \star z) = C\ (C\ x \star z)$
proof −
  have *1*: $C\ x * (x \star z) \leq C\ x \star (C\ x * z)$
    using *C-mult-propagate while-simulate* **by** *force*
  have $C\ (x \star z) = C\ z \sqcup C\ x * (x \star z)$
    by (*metis inf-sup-distrib1 n-L-T-meet-mult while-left-unfold*)
  also have ... $\leq C\ x \star z$
    using *1* **by** (*metis C-decreasing sup-mono while-right-unfold*)
  finally have $C\ (x \star z) \leq C\ (C\ x \star z)$
    by *simp*
  thus *?thesis*
    by (*metis order.antisym inf.boundedI inf.cobounded1 inf.coboundedI2 inf.sup-monoid.add-commute while-absorb-2 while-increasing*)
qed

lemma *C-while-preserve*:
  $C\ (x \star z) = C\ (x \star C\ z)$
proof −
  have $C\ x * (x \star z) \leq C\ x \star (C\ x * z)$
    using *C-mult-propagate while-simulate* **by** *auto*
  also have ... $\leq x \star (x * C\ z)$
    using *C-decreasing n-L-T-meet-mult-propagate while-isotone* **by** *blast*
  finally have *1*: $C\ x * (x \star z) \leq x \star (x * C\ z)$
    .
  have $C\ (x \star z) = C\ z \sqcup C\ x * (x \star z)$
    by (*metis inf-sup-distrib1 n-L-T-meet-mult while-left-unfold*)
  also have ... $\leq x \star C\ z$
    using *1* **by** (*meson order.trans le-supI while-increasing while-right-plus-below*)
  finally have $C\ (x \star z) \leq C\ (x \star C\ z)$
    by *simp*

thus *?thesis*
by (*meson order.antisym inf.boundedI inf.cobounded1 inf.coboundedI2 inf.eq-refl while-isotone*)
qed

lemma *C-while-import-preserve*:
$C\ (x \star z) = C\ (C\ x \star C\ z)$
using *C-while-import C-while-preserve* **by** *auto*

lemma *while-L-L*:
$L \star L = L$
by (*metis n-L-top-L while-mult-star-exchange while-right-top*)

lemma *while-L-below-sup*:
$L \star x \leq x \sqcup L$
by (*metis while-left-unfold sup-right-isotone n-L-below-L*)

lemma *while-L-split*:
$x \star L \leq (x \star y) \sqcup L$
proof −
have $x \star L \leq (x \star bot) \sqcup L$
by (*metis sup-commute sup-bot-left mult-1-right n-L-split-L while-right-unfold while-simulate-left-plus while-zero*)
thus *?thesis*
by (*metis sup-commute sup-right-isotone order-trans while-right-isotone bot-least*)
qed

lemma *while-n-while-top-split*:
$x \star (n(x \star y) * top) \leq (x \star bot) \sqcup n(x \star y) * top$
proof −
have $x * n(x \star y) * top \leq x * bot \sqcup n(x * (x \star y)) * top$
by (*simp add*: *n-n-top-split-n-top*)
also have ... $\leq n(x \star y) * top \sqcup x * bot$
by (*metis sup-commute sup-right-isotone mult-left-isotone n-isotone while-left-plus-below*)
finally have $x \star (n(x \star y) * top) \leq n(x \star y) * top \sqcup (x \star (x * bot))$
by (*metis mult-assoc mult-1-right while-simulate-left mult-left-zero while-left-top*)
also have ... $\leq (x \star bot) \sqcup n(x \star y) * top$
using *sup-left-isotone while-right-plus-below* **by** *auto*
finally show *?thesis*
.
qed

lemma *circ-apx-right-isotone*:
assumes $x \sqsubseteq y$
shows $z \star x \sqsubseteq z \star y$
proof −

have *1*: $x \le y \sqcup L \wedge C\ y \le x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
hence $z \star x \le (z \star y) \sqcup (z \star L)$
by (*metis while-left-dist-sup while-right-isotone*)
hence *2*: $z \star x \le (z \star y) \sqcup L$
by (*meson le-supI order-lesseq-imp sup.cobounded1 while-L-split*)
have $z \star (n(z \star x) * top) \le (z \star bot) \sqcup n(z \star x) * top$
by (*simp add*: *while-n-while-top-split*)
also have ... $\le (z \star x) \sqcup n(z \star x) * top$
using *sup-left-isotone while-right-isotone* **by** *force*
finally have *3*: $z \star (n(x) * top) \le (z \star x) \sqcup n(z \star x) * top$
by (*metis mult-left-isotone n-isotone order-trans while-increasing while-right-isotone*)
have $C\ (z \star y) \le z \star C\ y$
by (*metis C-while-preserve inf.cobounded2*)
also have ... $\le (z \star x) \sqcup (z \star (n(x) * top))$
using *1* **by** (*metis while-left-dist-sup while-right-isotone*)
also have ... $\le (z \star x) \sqcup n(z \star x) * top$
using *3* **by** *simp*
finally show *?thesis*
using *2 apx-def* **by** *auto*
qed

end

class *extended-binary-itering-apx* = *binary-itering-apx* + *bounded-extended-binary-itering* +
assumes *n-below-while-zero*: $n(x) \le n(x \star bot)$
begin

lemma *circ-apx-right-isotone*:
assumes $x \sqsubseteq y$
shows $x \star z \sqsubseteq y \star z$
proof −
have *1*: $x \le y \sqcup L \wedge C\ y \le x \sqcup n(x) * top$
using *assms apx-def* **by** *auto*
hence $x \star z \le ((y \star 1) * L) \star (y \star z)$
by (*metis while-left-isotone while-sumstar-3*)
also have ... $\le (y \star z) \sqcup (y \star 1) * L$
by (*metis while-productstar sup-right-isotone mult-right-isotone n-L-below-L while-slide*)
also have ... $\le (y \star z) \sqcup L$
by (*meson order.trans le-supI sup.cobounded1 while-L-split while-one-mult-below*)
finally have *2*: $x \star z \le (y \star z) \sqcup L$
.
have $C\ (y \star z) \le C\ y \star z$
by (*metis C-while-import inf.sup-right-divisibility*)
also have ... $\le ((x \star 1) * n(x) * top) \star (x \star z)$

using *1* **by** (*metis while-left-isotone mult-assoc while-sumstar-3*)
also have ... $\leq (x \star z) \sqcup (x \star 1) * n(x) * top$
by (*metis while-productstar sup-left-top sup-right-isotone mult-assoc mult-left-sub-dist-sup-right while-slide*)
also have ... $\leq (x \star z) \sqcup (x \star (n(x) * top))$
using *sup-right-isotone while-one-mult-below mult-assoc* **by** *auto*
also have ... $\leq (x \star z) \sqcup (x \star (n(x \star z) * top))$
by (*metis n-below-while-zero bot-least while-right-isotone n-isotone mult-left-isotone sup-right-isotone order-trans*)
also have ... $\leq (x \star z) \sqcup n(x \star z) * top$
by (*metis sup-assoc sup-right-isotone while-n-while-top-split sup-bot-right while-left-dist-sup*)
finally show *?thesis*
using *2 apx-def* **by** *auto*
qed

proposition *while-top*: $top \star x = L \sqcup top * x$ **oops**
proposition *while-one-top*: $1 \star x = L \sqcup x$ **oops**
proposition *while-unfold-below-1*: $x = y * x \Longrightarrow x \leq y \star 1$ **oops**

proposition *while-square-1*: $x \star 1 = (x * x) \star (x \sqcup 1)$ **oops**
proposition *while-absorb-below-one*: $y * x \leq x \Longrightarrow y \star x \leq 1 \star x$ **oops**
proposition *while-mult-L*: $(x * L) \star z = z \sqcup x * L$ **oops**
proposition *tarski-top-omega-below-2*: $x * L \leq (x * L) \star bot$ **oops**
proposition *tarski-top-omega-2*: $x * L = (x * L) \star bot$ **oops**
proposition *while-separate-right-plus*: $y * x \leq x * (x \star (1 \sqcup y)) \sqcup 1 \Longrightarrow y \star (x \star z) \leq x \star (y \star z)$ **oops**
proposition $y \star (x \star 1) \leq x \star (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**
proposition $y * x \leq (1 \sqcup x) * (y \star 1) \Longrightarrow (x \sqcup y) \star 1 = x \star (y \star 1)$ **oops**

end

class *n-omega-algebra-binary* = *n-omega-algebra* + *while* +
assumes *while-def*: $x \star y = n(x^\omega) * L \sqcup x^\star * y$
begin

lemma *while-omega-inf-L-star*:
$x \star y = (x^\omega \sqcap L) \sqcup x^\star * y$
by (*metis loop-semantics-kappa-mu-nu loop-semantics-kappa-mu-nu-2 while-def*)

lemma *while-one-mult-while-below-1*:
$(y \star 1) * (y \star v) \leq y \star v$
proof −
have $(y \star 1) * (y \star v) \leq y \star (y \star v)$
by (*smt sup-left-isotone mult-assoc mult-right-dist-sup mult-right-isotone n-L-below-L while-def mult-left-one*)
also have ... $= n(y^\omega) * L \sqcup y^\star * n(y^\omega) * L \sqcup y^\star * y^\star * v$
by (*simp add*: *mult-left-dist-sup sup-assoc while-def mult-assoc*)
also have ... $= n(y^\omega) * L \sqcup (y^\star * y^\star * bot \sqcup y^\star * n(y^\omega) * L) \sqcup y^\star * y^\star * v$

by (*metis mult-left-dist-sup star.circ-transitive-equal sup-bot-left mult-assoc*)
also have ... $= n(y^\omega) * L \sqcup (y^\star * y^\star * bot \sqcup n(y^\star * y^\omega) * L) \sqcup y^\star * y^\star * v$
by (*simp add*: *n-mult-omega-L-star-zero*)
also have ... $= n(y^\omega) * L \sqcup n(y^\star * y^\omega) * L \sqcup y^\star * y^\star * v$
by (*smt* (*z3*) *mult-left-dist-sup sup.left-commute sup-bot-left sup-commute*)
finally show *?thesis*
by (*simp add*: *star.circ-transitive-equal star-mult-omega while-def*)
qed

lemma *star-below-while*:
$x^\star * y \leq x \star y$
by (*simp add*: *while-def*)

subclass *bounded-binary-itering*
proof *unfold-locales*
fix $x\ y\ z$
have $z \sqcup x * ((y * x) \star (y * z)) = x * (y * x)^\star * y * z \sqcup x * n((y * x)^\omega) * L \sqcup z$
using *mult-left-dist-sup sup-commute while-def mult-assoc* **by** *auto*
also have ... $= x * (y * x)^\star * y * z \sqcup n(x * (y * x)^\omega) * L \sqcup z$
by (*metis mult-assoc mult-right-isotone bot-least n-mult-omega-L-star-zero sup-relative-same-increasing*)
also have ... $= (x * y)^\star * z \sqcup n(x * (y * x)^\omega) * L$
by (*smt sup-assoc sup-commute mult-assoc star.circ-loop-fixpoint star-slide*)
also have ... $= (x * y) \star z$
by (*simp add*: *omega-slide sup-monoid.add-commute while-def*)
finally show $(x * y) \star z = z \sqcup x * ((y * x) \star (y * z))$
by *simp*
next
fix $x\ y\ z$
have $(x \star y) \star (x \star z) = n((n(x^\omega) * L \sqcup x^\star * y)^\omega) * L \sqcup (n(x^\omega) * L \sqcup x^\star * y)^\star * (x \star z)$
by (*simp add*: *while-def*)
also have ... $= n((x^\star * y)^\omega \sqcup (x^\star * y)^\star * n(x^\omega) * L) * L \sqcup ((x^\star * y)^\star \sqcup (x^\star * y)^\star * n(x^\omega) * L) * (x \star z)$
using *mult-L-sup-omega mult-L-sup-star* **by** *force*
also have ... $= n((x^\star * y)^\omega) * L \sqcup n((x^\star * y)^\star * n(x^\omega) * L) * L \sqcup (x^\star * y)^\star * (x \star z) \sqcup (x^\star * y)^\star * n(x^\omega) * L * (x \star z)$
by (*simp add*: *mult-right-dist-sup n-dist-omega-star sup-assoc mult-assoc*)
also have ... $= n((x^\star * y)^\omega) * L \sqcup n((x^\star * y)^\star * n(x^\omega) * L) * L \sqcup (x^\star * y)^\star * bot \sqcup (x^\star * y)^\star * (x \star z) \sqcup (x^\star * y)^\star * n(x^\omega) * L * (x \star z)$
by (*smt sup-assoc sup-bot-left mult-left-dist-sup*)
also have ... $= n((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * n(x^\omega) * L * (x \star z) \sqcup (x^\star * y)^\star * n(x^\omega) * L \sqcup (x^\star * y)^\star * (x \star z))$
by (*smt n-n-L-split-n-n-L-L sup-commute sup-assoc*)
also have ... $= n((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * n(x^\omega) * L \sqcup (x^\star * y)^\star * (x \star z))$
by (*smt mult-L-omega omega-sub-vector le-iff-sup*)
also have ... $= n((x^\star * y)^\omega) * L \sqcup (x^\star * y)^\star * (x \star z)$
using *mult-left-sub-dist-sup-left sup-absorb2 while-def mult-assoc* **by** *auto*
also have ... $= (x^\star * y)^\star * x^\star * z \sqcup (x^\star * y)^\star * n(x^\omega) * L \sqcup n((x^\star * y)^\omega) * L$

by (*simp add*: *mult-left-dist-sup sup-commute while-def mult-assoc*)
also have ... $= (x^\star * y)^\star * x^\star * z \sqcup n((x^\star * y)^\star * x^\omega) * L \sqcup n((x^\star * y)^\omega) * L$
by (*metis sup-bot-right mult-left-dist-sup sup-assoc n-mult-omega-L-star-zero*)
also have ... $= (x \sqcup y) \star z$
using *n-dist-omega-star omega-decompose semiring.combine-common-factor star.circ-sup-9 sup-commute while-def* **by** *force*
finally show $(x \sqcup y) \star z = (x \star y) \star (x \star z)$
by *simp*
next
fix $x\ y\ z$
show $x \star (y \sqcup z) = (x \star y) \sqcup (x \star z)$
using *semiring.distrib-left sup-assoc sup-commute while-def* **by** *force*
next
fix $x\ y\ z$
show $(x \star y) * z \le x \star (y * z)$
by (*smt sup-left-isotone mult-assoc mult-right-dist-sup mult-right-isotone n-L-below-L while-def*)
next
fix $v\ w\ x\ y\ z$
show $x * z \le z * (y \star 1) \sqcup w \longrightarrow x \star (z * v) \le z * (y \star v) \sqcup (x \star (w * (y \star v)))$
proof
assume *1*: $x * z \le z * (y \star 1) \sqcup w$
have $z * v \sqcup x * (z * (y \star v) \sqcup x^\star * (w * (y \star v))) \le z * v \sqcup x * z * (y \star v) \sqcup x^\star * (w * (y \star v))$
by (*metis sup-assoc sup-right-isotone mult-assoc mult-left-dist-sup mult-left-isotone star.left-plus-below-circ*)
also have ... $\le z * v \sqcup z * (y \star 1) * (y \star v) \sqcup w * (y \star v) \sqcup x^\star * (w * (y \star v))$
using *1* **by** (*metis sup-assoc sup-left-isotone sup-right-isotone mult-left-isotone mult-right-dist-sup*)
also have ... $\le z * v \sqcup z * (y \star v) \sqcup x^\star * (w * (y \star v))$
by (*smt* (*verit*, *ccfv-threshold*) *sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup star.circ-loop-fixpoint while-one-mult-while-below-1 le-supE le-supI*)
also have ... $= z * (y \star v) \sqcup x^\star * (w * (y \star v))$
by (*metis le-iff-sup le-supE mult-right-isotone star.circ-loop-fixpoint star-below-while*)
finally have $x^\star * z * v \le z * (y \star v) \sqcup x^\star * (w * (y \star v))$
using *star-left-induct mult-assoc* **by** *auto*
thus $x \star (z * v) \le z * (y \star v) \sqcup (x \star (w * (y \star v)))$
by (*smt sup-assoc sup-commute sup-right-isotone mult-assoc while-def*)
qed
next
fix $v\ w\ x\ y\ z$
show $z * x \le y * (y \star z) \sqcup w \longrightarrow z * (x \star v) \le y \star (z * v \sqcup w * (x \star v))$
proof
assume $z * x \le y * (y \star z) \sqcup w$
hence *1*: $z * x \le y * y^\star * z \sqcup (y * n(y^\omega) * L \sqcup w)$
by (*simp add*: *mult-left-dist-sup sup.left-commute sup-commute while-def mult-assoc*)

hence $z * x^\star \leq y^\star * (z \sqcup (y * n(y^\omega) * L \sqcup w) * x^\star)$
by (*simp add*: *star-circ-simulate-right-plus*)
also have ... $= y^\star * z \sqcup y^\star * y * n(y^\omega) * L \sqcup y^\star * w * x^\star$
by (*simp add*: *L-mult-star semiring.distrib-left semiring.distrib-right sup-left-commute sup-monoid.add-commute mult-assoc*)
also have ... $= y^\star * z \sqcup n(y^\star * y * y^\omega) * L \sqcup y^\star * w * x^\star$
by (*metis sup-relative-same-increasing mult-isotone n-mult-omega-L-star-zero star.left-plus-below-circ star.right-plus-circ bot-least*)
also have ... $= n(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * x^\star$
using *omega-unfold star-mult-omega sup-commute mult-assoc* **by** *force*
finally have $z * x^\star * v \leq n(y^\omega) * L * v \sqcup y^\star * z * v \sqcup y^\star * w * x^\star * v$
by (*smt le-iff-sup mult-right-dist-sup*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * (z * v \sqcup w * x^\star * v)$
by (*metis n-L-below-L mult-assoc mult-right-isotone sup-left-isotone mult-left-dist-sup sup-assoc*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * (z * v \sqcup w * (x \star v))$
using *mult-right-isotone semiring.add-left-mono mult-assoc star-below-while* **by** *auto*
finally have *2*: $z * x^\star * v \leq y \star (z * v \sqcup w * (x \star v))$
by (*simp add*: *while-def*)
have *3*: $y^\star * y * y^\star * bot \leq y^\star * w * x^\omega$
by (*metis sup-commute sup-bot-left mult-assoc mult-left-sub-dist-sup-left star.circ-loop-fixpoint star.circ-transitive-equal*)
have $z * x^\omega \leq y * y^\star * z * x^\omega \sqcup (y * n(y^\omega) * L \sqcup w) * x^\omega$
using *1* **by** (*metis mult-assoc mult-left-isotone mult-right-dist-sup omega-unfold*)
hence $z * x^\omega \leq y^\omega \sqcup y^\star * y * n(y^\omega) * L * x^\omega \sqcup y^\star * w * x^\omega$
by (*smt sup-assoc sup-commute left-plus-omega mult-assoc mult-left-dist-sup mult-right-dist-sup omega-induct star.left-plus-circ*)
also have ... $\leq y^\omega \sqcup y^\star * y * n(y^\omega) * L \sqcup y^\star * w * x^\omega$
by (*metis sup-left-isotone sup-right-isotone mult-assoc mult-right-isotone n-L-below-L*)
also have ... $= y^\omega \sqcup n(y^\star * y * y^\omega) * L \sqcup y^\star * w * x^\omega$
using *3* **by** (*smt sup-assoc sup-commute sup-relative-same-increasing n-mult-omega-L-star-zero*)
also have ... $= y^\omega \sqcup y^\star * w * x^\omega$
by (*metis mult-assoc omega-unfold star-mult-omega sup-commute le-iff-sup n-L-decreasing*)
finally have $n(z * x^\omega) * L \leq n(y^\omega) * L \sqcup n(y^\star * w * x^\omega) * L$
by (*metis mult-assoc mult-left-isotone mult-right-dist-sup n-dist-omega-star n-isotone*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * (w * (n(x^\omega) * L \sqcup x^\star * bot))$
by (*smt sup-commute sup-right-isotone mult-assoc mult-left-dist-sup n-mult-omega-L-below-zero*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * (w * (n(x^\omega) * L \sqcup x^\star * v))$
by (*metis sup-right-isotone mult-right-isotone bot-least*)
also have ... $\leq n(y^\omega) * L \sqcup y^\star * (z * v \sqcup w * (n(x^\omega) * L \sqcup x^\star * v))$
using *mult-left-sub-dist-sup-right sup-right-isotone* **by** *auto*
finally have *4*: $n(z * x^\omega) * L \leq y \star (z * v \sqcup w * (x \star v))$

using *while-def* **by** *auto*
have $z * (x \star v) = z * n(x^\omega) * L \sqcup z * x^\star * v$
by (*simp add*: *mult-left-dist-sup while-def mult-assoc*)
also have ... $= n(z * x^\omega) * L \sqcup z * x^\star * v$
by (*metis sup-commute sup-relative-same-increasing mult-right-isotone n-mult-omega-L-star-zero bot-least*)
finally show $z * (x \star v) \leq y \star (z * v \sqcup w * (x \star v))$
using *2 4* **by** *simp*
qed
qed

lemma *while-top*:
top $\star x = L \sqcup$ *top* $* x$
by (*metis n-top-L star.circ-top star-omega-top while-def*)

lemma *while-one-top*:
$1 \star x = L \sqcup x$
by (*smt mult-left-one n-top-L omega-one star-one while-def*)

lemma *while-finite-associative*:
$x^\omega = bot \Longrightarrow (x \star y) * z = x \star (y * z)$
by (*metis sup-bot-left mult-assoc n-zero-L-zero while-def*)

lemma *while-while-one*:
$y \star (x \star 1) = n(y^\omega) * L \sqcup y^\star * n(x^\omega) * L \sqcup y^\star * x^\star$
by (*simp add*: *mult-left-dist-sup sup-assoc while-def mult-assoc*)

AACP Theorem 8.17

subclass *bounded-extended-binary-itering*
proof *unfold-locales*
fix *w x y z*
have $w * (x \star y * z) = n(w * n(x^\omega) * L) * L \sqcup w * x^\star * y * z$
by (*smt sup-assoc sup-commute sup-bot-left mult-assoc mult-left-dist-sup n-n-L-split-n-n-L-L while-def*)
also have ... $\leq n((w * n(x^\omega) * L)^\omega) * L \sqcup w * x^\star * y * z$
by (*simp add*: *mult-L-omega*)
also have ... $\leq n((w * (x \star y))^\omega) * L \sqcup w * x^\star * y * z$
by (*smt sup-left-isotone sup-ge1 mult-assoc mult-left-isotone mult-right-isotone n-isotone omega-isotone while-def*)
also have ... $\leq n((w * (x \star y))^\omega) * L \sqcup w * (x \star y) * z$
by (*metis star-below-while mult-assoc mult-left-isotone mult-right-isotone sup-right-isotone*)
also have ... $\leq n((w * (x \star y))^\omega) * L \sqcup (w * (x \star y))^\star * (w * (x \star y) * z)$
using *sup.boundedI sup.cobounded1 while-def while-increasing* **by** *auto*
finally show $w * (x \star y * z) \leq (w * (x \star y)) \star (w * (x \star y) * z)$
using *while-def* **by** *auto*
qed

subclass *extended-binary-itering-apx*

apply *unfold-locales*
by (*metis n-below-n-omega n-left-upper-bound n-n-L order-trans while-def*)

lemma *while-simulate-4-plus*:
assumes $y * x \leq x * (x \star (1 \sqcup y))$
shows $y * x * x^\star \leq x * (x \star (1 \sqcup y))$
proof –
have $x * (x \star (1 \sqcup y)) = x * n(x^\omega) * L \sqcup x * x^\star * (1 \sqcup y)$
by (*simp add*: *mult-left-dist-sup while-def mult-assoc*)
also have ... $= n(x * x^\omega) * L \sqcup x * x^\star * (1 \sqcup y)$
by (*smt n-mult-omega-L-star-zero sup-relative-same-increasing sup-commute sup-bot-right mult-left-sub-dist-sup-right*)
finally have *1*: $x * (x \star (1 \sqcup y)) = n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y$
using *mult-left-dist-sup omega-unfold sup-assoc* **by** *force*
hence $x * x^\star * y * x \leq x * x^\star * n(x^\omega) * L \sqcup x * x^\star * x^\star * x \sqcup x * x^\star * x * x^\star * y$
by (*metis assms mult-assoc mult-right-isotone mult-left-dist-sup star-plus*)
also have ... $= n(x * x^\star * x^\omega) * L \sqcup x * x^\star * x^\star * x \sqcup x * x^\star * x * x^\star * y$
by (*smt* (*z3*) *sup-commute n-mult-omega-L-star omega-unfold semiring.distrib-left star-plus mult-assoc*)
also have ... $= n(x^\omega) * L \sqcup x * x^\star * x \sqcup x * x * x^\star * y$
using *omega-unfold star.circ-plus-same star.circ-transitive-equal star-mult-omega mult-assoc* **by** *auto*
also have ... $\leq n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y$
by (*smt sup-assoc sup-ge2 le-iff-sup mult-assoc mult-right-dist-sup star.circ-increasing star.circ-plus-same star.circ-transitive-equal*)
finally have *2*: $x * x^\star * y * x \leq n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y$
.
have $(n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y) * x \leq n(x^\omega) * L \sqcup x * x^\star * x \sqcup x * x^\star * y * x$
by (*metis mult-right-dist-sup n-L-below-L mult-assoc mult-right-isotone sup-left-isotone*)
also have ... $\leq n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y * x$
by (*smt sup-commute sup-left-isotone mult-assoc mult-right-isotone star.left-plus-below-circ star-plus*)
also have ... $\leq n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y$
using *2* **by** *simp*
finally show *?thesis*
using *1 assms star-right-induct* **by** *force*
qed

lemma *while-simulate-4-omega*:
assumes $y * x \leq x * (x \star (1 \sqcup y))$
shows $y * x^\omega \leq x^\omega$
proof –
have $x * (x \star (1 \sqcup y)) = x * n(x^\omega) * L \sqcup x * x^\star * (1 \sqcup y)$
using *mult-left-dist-sup while-def mult-assoc* **by** *auto*
also have ... $= n(x * x^\omega) * L \sqcup x * x^\star * (1 \sqcup y)$
by (*smt* (*z3*) *mult-1-right mult-left-sub-dist-sup-left n-mult-omega-L-star*

sup-commute sup-relative-same-increasing)
finally have $x * (x \star (1 \sqcup y)) = n(x^\omega) * L \sqcup x * x^\star \sqcup x * x^\star * y$
using *mult-left-dist-sup omega-unfold sup-assoc* **by** *force*
hence $y * x^\omega \leq n(x^\omega) * L * x^\omega \sqcup x * x^\star * x^\omega \sqcup x * x^\star * y * x^\omega$
by (*smt assms le-iff-sup mult-assoc mult-right-dist-sup omega-unfold*)
also have ... $\leq x * x^\star * (y * x^\omega) \sqcup x^\omega$
by (*metis sup-left-isotone mult-L-omega omega-sub-vector mult-assoc omega-unfold star-mult-omega n-L-decreasing le-iff-sup sup-commute*)
finally have $y * x^\omega \leq (x * x^\star)^\omega \sqcup (x * x^\star)^\star * x^\omega$
by (*simp add*: *omega-induct sup-monoid.add-commute*)
thus *?thesis*
by (*metis sup-idem left-plus-omega star-mult-omega*)
qed

lemma *while-square-1*:
$x \star 1 = (x * x) \star (x \sqcup 1)$
by (*metis mult-1-right omega-square star-square-2 while-def*)

lemma *while-absorb-below-one*:
$y * x \leq x \Longrightarrow y \star x \leq 1 \star x$
by (*metis star-left-induct-mult sup-mono n-galois n-sub-nL while-def while-one-top*)

lemma *while-mult-L*:
$(x * L) \star z = z \sqcup x * L$
by (*metis sup-bot-right mult-left-zero while-denest-5 while-one-top while-productstar while-sumstar*)

lemma *tarski-top-omega-below-2*:
$x * L \leq (x * L) \star bot$
by (*simp add*: *while-mult-L*)

lemma *tarski-top-omega-2*:
$x * L = (x * L) \star bot$
by (*simp add*: *while-mult-L*)

proposition *while-sub-mult-one*: $x * (1 \star y) \leq 1 \star x$ **nitpick** [*expect=genuine,card=3*] **oops**
proposition *while-unfold-below*: $x = z \sqcup y * x \longrightarrow x \leq y \star z$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-loop-is-greatest-postfixpoint*: *is-greatest-postfixpoint* $(\lambda x\ .\ y * x \sqcup z)\ (y \star z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-loop-is-greatest-fixpoint*: *is-greatest-fixpoint* $(\lambda x\ .\ y * x \sqcup z)\ (y \star z)$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-denest-3*: $(x \star w) \star x^\omega = (x \star w)^\omega$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *while-mult-top*: $(x * top) \star z = z \sqcup x * top$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *tarski-below-top-omega*: $x \leq (x * L)^\omega$ **nitpick**

[*expect=genuine*,*card=2*] **oops**
proposition *tarski-mult-omega-omega*: $(x * y^\omega)^\omega = x * y^\omega$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition *tarski-below-top-omega-2*: $x \leq (x * L) \star bot$ **nitpick**
[*expect=genuine*,*card=2*] **oops**
proposition $1 = (x * bot) \star 1$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *tarski*: $x = bot \vee top * x * top = top$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition $(x \sqcup y) \star z = ((x \star 1) * y) \star ((x \star 1) * z)$ **nitpick**
[*expect=genuine*,*card=2*] **oops**
proposition *while-top-2*: $top \star z = top * z$ **nitpick** [*expect=genuine*,*card=2*]
oops
proposition *while-mult-top-2*: $(x * top) \star z = z \sqcup x * top * z$ **nitpick**
[*expect=genuine*,*card=2*] **oops**
proposition *while-one-mult*: $(x \star 1) * x = x \star x$ **nitpick**
[*expect=genuine*,*card=4*] **oops**
proposition $(x \star 1) * y = x \star y$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *while-associative*: $(x \star y) * z = x \star (y * z)$ **nitpick**
[*expect=genuine*,*card=2*] **oops**
proposition *while-back-loop-is-fixpoint*: *is-fixpoint* $(\lambda x\ .\ x * y \sqcup z)\ (z * (y \star 1))$
nitpick [*expect=genuine*,*card=4*] **oops**
proposition $1 \sqcup x * bot = x \star 1$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $x = x * (x \star 1)$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $x * (x \star 1) = x \star 1$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition $x \star 1 = x \star (1 \star 1)$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $(x \sqcup y) \star 1 = (x \star (y \star 1)) \star 1$ **nitpick** [*expect=genuine*,*card=3*]
oops
proposition $z \sqcup y * x = x \Longrightarrow y \star z \leq x$ **nitpick** [*expect=genuine*,*card=2*]
oops
proposition $y * x = x \Longrightarrow y \star x \leq x$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition $z \sqcup x * y = x \Longrightarrow z * (y \star 1) \leq x$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition $x * y = x \Longrightarrow x * (y \star 1) \leq x$ **nitpick** [*expect=genuine*,*card=3*]
oops
proposition $x * z = z * y \Longrightarrow x \star z \leq z * (y \star 1)$ **nitpick**
[*expect=genuine*,*card=2*] **oops**

proposition *while-unfold-below-1*: $x = y * x \Longrightarrow x \leq y \star 1$ **nitpick**
[*expect=genuine*,*card=3*] **oops**
proposition $x^\omega \leq x^\omega * x^\omega$ **oops**
proposition *tarski-omega-idempotent*: $x^{\omega\omega} = x^\omega$ **oops**

end

class *n-omega-algebra-binary-strict* = *n-omega-algebra-binary* + *circ* +
  assumes *L-left-zero*: $L * x = L$
  assumes *circ-def*: $x^\circ = n(x^\omega) * L \sqcup x^\star$
begin

subclass *strict-binary-itering*
  apply *unfold-locales*
  apply (*metis while-def mult-assoc L-left-zero mult-right-dist-sup*)
  by (*metis circ-def while-def mult-1-right*)

end

end

21 N-Relation-Algebras

theory *N-Relation-Algebras*

imports *Stone-Relation-Algebras.Relation-Algebras N-Omega-Algebras*

begin

context *bounded-distrib-allegory*
begin

subclass *lattice-ordered-pre-left-semiring* **..**

end

Theorem 37

sublocale *relation-algebra* < *n-algebra* **where** *sup* = *sup* **and** *bot* = *bot* **and** *top* = *top* **and** *inf* = *inf* **and** *n* = *N* **and** *L* = *top*
  apply *unfold-locales*
  using *N-comp-top comp-inf.semiring.distrib-left inf.sup-monoid.add-commute inf-vector-comp* **apply** *simp*
  apply (*metis N-comp compl-sup double-compl mult-assoc mult-right-dist-sup top-mult-top N-comp-N*)
  apply (*metis brouwer.p-antitone inf.sup-monoid.add-commute inf.sup-right-isotone mult-left-isotone p-antitone-sup*)
  apply *simp*
  using *N-vector-top* **apply** *force*
  apply *simp*
  apply (*simp add*: *brouwer.p-antitone-iff top-right-mult-increasing*)
  apply *simp*
  apply (*metis N-comp-top conv-complement-sub double-compl le-supI2 le-iff-sup mult-assoc mult-left-isotone schroeder-3*)
  by *simp*

sublocale *relation-algebra* < *n-algebra-apx* **where** *sup* = *sup* **and** *bot* = *bot* **and** *top* = *top* **and** *inf* = *inf* **and** *n* = *N* **and** *L* = *top* **and** *apx* = *greater-eq*
  apply *unfold-locales*
  using *n-less-eq-char* **by** *force*

no-notation

inverse-divide (**infixl** ‹′/› *70*)

notation
divide (**infixl** ‹′/› *70*)

class *left-residuated-relation-algebra* = *relation-algebra* + *inverse* +
assumes *lres-def*: $x\ /\ y = -(-x * y^T)$
begin

Theorem 32.1

subclass *residuated-pre-left-semiring*
apply *unfold-locales*
by (*metis compl-le-swap1 lres-def schroeder-4*)

end

context *left-residuated-relation-algebra*
begin

Theorem 32.3

lemma *lres-mult-lres-lres*:
$x\ /\ (z * y) = (x\ /\ y)\ /\ z$
by (*metis conv-dist-comp double-compl lres-def mult-assoc*)

Theorem 32.5

lemma *lres-dist-inf*:
$(x \sqcap y)\ /\ z = (x\ /\ z) \sqcap (y\ /\ z)$
by (*metis compl-inf compl-sup lres-def mult-right-dist-sup*)

Theorem 32.6

lemma *lres-add-export-vector*:
assumes *vector x*
shows $(x \sqcup y)\ /\ z = x \sqcup (y\ /\ z)$
proof −
have $(x \sqcup y)\ /\ z = -((-x \sqcap -y) * z^T)$
by (*simp add*: *lres-def*)
also have ... $= -(-x \sqcap (-y * z^T))$
using *assms vector-complement-closed vector-inf-comp* **by** *auto*
also have ... $= x \sqcup (y\ /\ z)$
by (*simp add*: *lres-def*)
finally show *?thesis*
.
qed

Theorem 32.7

lemma *lres-top-vector*:
vector $(x\ /\ top)$
using *equivalence-top-closed lres-def vector-complement-closed*
vector-mult-closed vector-top-closed **by** *auto*

Theorem 32.10

lemma *lres-top-export-inf-mult*:
$((x\ /\ top) \sqcap y) * z = (x\ /\ top) \sqcap (y * z)$
by (*simp add*: *vector-inf-comp lres-top-vector*)

lemma *N-lres*:
$N(x) = x\ /\ top \sqcap 1$
using *lres-def* **by** *auto*

end

class *complete-relation-algebra* = *relation-algebra* + *complete-lattice*
begin

definition *mu* :: $('a \Rightarrow 'a) \Rightarrow 'a$ **where** $mu\ f \equiv Inf\ \{\ y\ .\ f\ y \leq y\ \}$
definition *nu* :: $('a \Rightarrow 'a) \Rightarrow 'a$ **where** $nu\ f \equiv Sup\ \{\ y\ .\ y \leq f\ y\ \}$

lemma *mu-lower-bound*:
$f\ x \leq x \Longrightarrow mu\ f \leq x$
by (*auto simp add*: *mu-def intro*: *Inf-lower*)

lemma *mu-greatest-lower-bound*:
$(\forall y\ .\ f\ y \leq y \longrightarrow x \leq y) \Longrightarrow x \leq mu\ f$
using *lfp-def lfp-greatest mu-def* **by** *auto*

lemma *mu-unfold-1*:
$isotone\ f \Longrightarrow f\ (mu\ f) \leq mu\ f$
by (*metis mu-greatest-lower-bound order-trans mu-lower-bound isotone-def*)

lemma *mu-unfold-2*:
$isotone\ f \Longrightarrow mu\ f \leq f\ (mu\ f)$
by (*simp add*: *mu-lower-bound mu-unfold-1 ord.isotone-def*)

lemma *mu-unfold*:
$isotone\ f \Longrightarrow mu\ f = f\ (mu\ f)$
by (*simp add*: *order.antisym mu-unfold-1 mu-unfold-2*)

lemma *mu-const*:
$mu\ (\lambda x\ .\ y) = y$
by (*simp add*: *isotone-def mu-unfold*)

lemma *mu-lpfp*:
$isotone\ f \Longrightarrow is\text{-}least\text{-}prefixpoint\ f\ (mu\ f)$
by (*simp add*: *is-least-prefixpoint-def mu-lower-bound mu-unfold-1*)

lemma *mu-lfp*:
$isotone\ f \Longrightarrow is\text{-}least\text{-}fixpoint\ f\ (mu\ f)$
by (*metis is-least-fixpoint-def mu-lower-bound mu-unfold order-refl*)

lemma *mu-pmu*:
  isotone f $\Longrightarrow$ $p\mu\ f = mu\ f$
  using *least-prefixpoint-same mu-lpfp* **by** *force*

lemma *mu-mu*:
  isotone f $\Longrightarrow$ $\mu\ f = mu\ f$
  using *least-fixpoint-same mu-lfp* **by** *fastforce*

end

class *omega-relation-algebra* = *relation-algebra* + *star* + *omega* +
  assumes *ra-star-left-unfold* : $1 \sqcup y * y^\star \leq y^\star$
  assumes *ra-star-left-induct* : $z \sqcup y * x \leq x \longrightarrow y^\star * z \leq x$
  assumes *ra-star-right-induct*: $z \sqcup x * y \leq x \longrightarrow z * y^\star \leq x$
  assumes *ra-omega-unfold*: $y^\omega = y * y^\omega$
  assumes *ra-omega-induct*: $x \leq z \sqcup y * x \longrightarrow x \leq y^\omega \sqcup y^\star * z$
begin

subclass *bounded-omega-algebra*
  apply *unfold-locales*
  using *ra-star-left-unfold* **apply** *blast*
  using *ra-star-left-induct* **apply** *blast*
  using *ra-star-right-induct* **apply** *blast*
  using *ra-omega-unfold* **apply** *blast*
  using *ra-omega-induct* **by** *blast*

end

Theorem 38

sublocale *omega-relation-algebra* < *n-omega-algebra* **where** *sup* = *sup* **and** *bot* = *bot* **and** *top* = *top* **and** *inf* = *inf* **and** $n = N$ **and** $L = top$ **and** *apx* = *greater-eq* **and** *Omega* = $\lambda x\ .\ N(x^\omega) * top \sqcup x^\star$
  apply *unfold-locales*
  apply *simp*
  using *n-split-omega-mult omega-vector star-mult-omega* **apply** *force*
  by *simp*

end

22 Domain

theory *Domain*

imports *Stone-Relation-Algebras.Semirings Tests*

begin

context *idempotent-left-semiring*
begin

sublocale *ils*: *il-semiring* **where** *inf* = *times* **and** *sup* = *sup* **and** *bot* = *bot* **and** *less-eq* = *less-eq* **and** *less* = *less* **and** *top* = *1*
  apply *unfold-locales*
  apply (*simp add*: *sup-assoc*)
  apply (*simp add*: *sup-commute*)
  apply *simp*
  apply *simp*
  apply (*simp add*: *mult-assoc*)
  apply (*simp add*: *mult-right-dist-sup*)
  apply *simp*
  apply *simp*
  apply *simp*
  apply (*simp add*: *mult-right-isotone*)
  apply (*simp add*: *le-iff-sup*)
  by (*simp add*: *less-le-not-le*)

end

class *left-zero-domain-semiring* = *idempotent-left-zero-semiring* + *dom* +
  assumes *d-restrict*: $x \sqcup d(x) * x = d(x) * x$
  assumes *d-mult-d* : $d(x * y) = d(x * d(y))$
  assumes *d-plus-one*: $d(x) \sqcup 1 = 1$
  assumes *d-zero* : $d(bot) = bot$
  assumes *d-dist-sup*: $d(x \sqcup y) = d(x) \sqcup d(y)$
begin

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

lemma *d-restrict-equals*:
  $x = d(x) * x$
  by (*metis sup-commute d-plus-one d-restrict mult-left-one mult-right-dist-sup*)

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

lemma *d-fixpoint*:
  $(\exists y . x = d(y)) \longleftrightarrow x = d(x)$
  using *d-idempotent* **by** *auto*

lemma *d-type*:
  $\forall P . (\forall x . x = d(x) \longrightarrow P(x)) \longleftrightarrow (\forall x . P(d(x)))$
  by (*metis d-idempotent*)

lemma *d-mult-sub*:
  $d(x * y) \leq d(x)$
  by (*metis d-dist-sup d-mult-d d-plus-one le-iff-sup mult-left-sub-dist-sup-left mult-1-right*)

lemma *d-sub-one*:
  $x \leq 1 \Longrightarrow x \leq d(x)$
  by (*metis d-restrict-equals mult-right-isotone mult-1-right*)

lemma *d-strict*:
  $d(x) = bot \longleftrightarrow x = bot$
  by (*metis d-restrict-equals d-zero mult-left-zero*)

lemma *d-one*:
  $d(1) = 1$
  by (*metis d-restrict-equals mult-1-right*)

lemma *d-below-one*:
  $d(x) \leq 1$
  by (*simp add*: *d-plus-one le-iff-sup*)

lemma *d-isotone*:
  $x \leq y \Longrightarrow d(x) \leq d(y)$
  by (*metis d-dist-sup le-iff-sup*)

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

lemma *d-export*:
  $d(d(x) * y) = d(x) * d(y)$
  apply (*rule order.antisym*)
  apply (*metis d-below-one d-idempotent d-mult-sub d-restrict-equals d-isotone d-mult-d mult-isotone mult-left-one*)
  by (*metis d-below-one d-sub-one coreflexive-mult-closed d-mult-d*)

lemma *d-mult-idempotent*:
  $d(x) * d(x) = d(x)$
  by (*metis d-export d-restrict-equals*)

lemma *d-commutative*:
  $d(x) * d(y) = d(y) * d(x)$
  by (*metis ils.il-inf-associative order.antisym d-export d-mult-d d-mult-sub d-one d-restrict-equals mult-isotone mult-left-one*)

lemma *d-least-left-preserver*:
  $x \leq d(y) * x \longleftrightarrow d(x) \leq d(y)$
  by (*metis d-below-one d-idempotent d-mult-sub d-restrict-equals order.eq-iff mult-left-isotone mult-left-one*)

lemma *d-weak-locality*:
  $x * y = bot \longleftrightarrow x * d(y) = bot$
  by (*metis d-mult-d d-strict*)

lemma *d-sup-closed*:
  $d(d(x) \sqcup d(y)) = d(x) \sqcup d(y)$
  by (*simp add*: *d-idempotent d-dist-sup*)

lemma *d-mult-closed*:
  $d(d(x) * d(y)) = d(x) * d(y)$
  using *d-export d-mult-d* **by** *auto*

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

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

lemma *d-mult-left-absorb-sup*:
  $d(x) * (d(x) \sqcup d(y)) = d(x)$
  by (*metis sup-commute d-mult-idempotent d-plus-one mult-left-dist-sup mult-1-right*)

lemma *d-sup-left-absorb-mult*:
  $d(x) \sqcup d(x) * d(y) = d(x)$
  using *d-mult-left-lower-bound sup.absorb-iff1* **by** *auto*

lemma *d-sup-left-dist-mult*:
  $d(x) \sqcup d(y) * d(z) = (d(x) \sqcup d(y)) * (d(x) \sqcup d(z))$
  by (*smt sup-assoc d-commutative d-mult-idempotent d-mult-left-absorb-sup mult-left-dist-sup mult-right-dist-sup*)

lemma *d-order*:
  $d(x) \leq d(y) \longleftrightarrow d(x) = d(x) * d(y)$
  by (*metis d-mult-greatest-lower-bound d-mult-left-absorb-sup le-iff-sup order-refl*)

lemma *d-mult-below*:
  $d(x) * y \leq y$
  by (*metis sup-left-divisibility d-plus-one mult-left-one mult-right-dist-sup*)

lemma *d-preserves-equation*:
  $d(y) * x \leq x * d(y) \longleftrightarrow d(y) * x = d(y) * x * d(y)$
  by (*simp add*: *d-below-one d-mult-idempotent test-preserves-equation*)

end

class *left-zero-antidomain-semiring* = *idempotent-left-zero-semiring* + *dom* + *uminus* +
  assumes *a-restrict* : $-x * x = bot$
  assumes *a-plus-mult-d*: $-(x * y) \sqcup -(x * --y) = -(x * --y)$

assumes *a-complement* : $--x \sqcup -x = 1$
assumes *d-def* : $d(x) = --x$
begin

sublocale *aa*: *a-algebra* **where** *minus* $= \lambda x\ y\ .\ -(-x \sqcup y)$ **and** *uminus* $=$ *uminus* **and** *inf* $=$ *times* **and** *sup* $=$ *sup* **and** *bot* $=$ *bot* **and** *less-eq* $=$ *less-eq* **and** *less* $=$ *less* **and** *top* $= 1$
apply *unfold-locales*
apply (*simp add*: *a-restrict*)
using *a-complement sup-commute* **apply** *fastforce*
apply (*simp add*: *a-plus-mult-d le-iff-sup*)
by *simp*

subclass *left-zero-domain-semiring*
apply *unfold-locales*
apply (*simp add*: *d-def aa.double-complement-above*)
apply (*simp add*: *aa.a-d.d3-eq d-def*)
apply (*simp add*: *d-def*)
apply (*simp add*: *d-def*)
by (*simp add*: *d-def aa.l15*)

subclass *tests*
apply *unfold-locales*
apply (*simp add*: *mult-assoc*)
apply (*simp add*: *aa.sba-dual.sub-commutative*)
apply (*simp add*: *aa.sba-dual.sub-complement*)
using *aa.sba-dual.sub-sup-closed* **apply** *simp*
apply *simp*
apply *simp*
apply (*simp add*: *aa.sba-dual.sub-inf-def*)
apply (*simp add*: *aa.less-eq-inf*)
by (*simp add*: *less-le-not-le*)

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

notation
uminus (‹*a*›)

lemma *a-greatest-left-absorber*:
$a(x) * y = bot \longleftrightarrow a(x) \le a(y)$
by (*simp add*: *aa.l10-iff*)

lemma *a-mult-d*:
$a(x * y) = a(x * d(y))$
by (*simp add*: *d-def aa.sba3-complement-inf-double-complement*)

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

lemma *a-plus-left-lower-bound*:
$a(x \sqcup y) \leq a(x)$
by (*simp add*: *aa.l9*)

lemma *a-mult-sup*:
$a(x) * (y \sqcup x) = a(x) * y$
by (*simp add*: *aa.sba3-inf-complement-bot semiring.distrib-left*)

lemma *a-3*:
$a(x) * a(y) * d(x \sqcup y) = bot$
using *d-weak-locality aa.l12 aa.sba3-inf-complement-bot* **by** *force*

lemma *a-export*:
$a(a(x) * y) = d(x) \sqcup a(y)$
using *a-mult-d d-def aa.sba-dual.sub-inf-def* **by** *auto*

lemma *a-fixpoint*:
$\forall x . (a(x) = x \longrightarrow (\forall y . y = bot))$
by (*metis aa.a-d.d-fully-strict aa.sba2-bot-unit aa.sup-idempotent aa.sup-right-zero-var*)

lemma *a-strict*:
$a(x) = 1 \longleftrightarrow x = bot$
using *aa.a-d.d-fully-strict one-def* **by** *fastforce*

lemma *d-complement-zero*:
$d(x) * a(x) = bot$
by (*simp add*: *aa.sba3-inf-complement-bot d-def*)

lemma *a-complement-zero*:
$a(x) * d(x) = bot$
by (*simp add*: *d-def*)

lemma *a-shunting-zero*:
$a(x) * d(y) = bot \longleftrightarrow a(x) \leq a(y)$
by (*simp add*: *aa.less-eq-inf-bot d-def*)

lemma *a-antitone*:
$x \leq y \Longrightarrow a(y) \leq a(x)$
by (*simp add*: *aa.l9*)

lemma *a-mult-deMorgan*:
$a(a(x) * a(y)) = d(x \sqcup y)$
by (*simp add*: *aa.sup-demorgan d-def*)

lemma *a-mult-deMorgan-1*:
$a(a(x) * a(y)) = d(x) \sqcup d(y)$
by (*simp add*: *a-export d-def*)

lemma *a-mult-deMorgan-2*:
$a(d(x) * d(y)) = a(x) \sqcup a(y)$
by (*simp add*: *d-def sup-def*)

lemma *a-plus-deMorgan*:
$a(a(x) \sqcup a(y)) = d(x) * d(y)$
by (*simp add*: *aa.sub-sup-demorgan d-def*)

lemma *a-plus-deMorgan-1*:
$a(d(x) \sqcup d(y)) = a(x) * a(y)$
by (*simp add*: *aa.sup-demorgan d-def*)

lemma *a-mult-left-upper-bound*:
$a(x) \leq a(x * y)$
using *aa.l5 d-def d-mult-sub* **by** *auto*

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

lemma *a-export-d*:
$a(d(x) * y) = a(x) \sqcup a(y)$
using *a-mult-d a-mult-deMorgan-2* **by** *auto*

lemma *a-7*:
$d(x) * a(d(y) \sqcup d(z)) = d(x) * a(y) * a(z)$
by (*simp add*: *a-plus-deMorgan-1 mult-assoc*)

lemma *d-a-shunting*:
$d(x) * a(y) \leq d(z) \longleftrightarrow d(x) \leq d(z) \sqcup d(y)$
using *aa.sba-dual.shunting-right d-def* **by** *auto*

lemma *d-d-shunting*:
$d(x) * d(y) \leq d(z) \longleftrightarrow d(x) \leq d(z) \sqcup a(y)$
using *d-a-shunting d-def* **by** *auto*

lemma *d-cancellation-1*:
$d(x) \leq d(y) \sqcup (d(x) * a(y))$
by (*metis a-d-closed aa.sba2-export aa.sup-demorgan d-def eq-refl le-supE sup-commute*)

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

lemma *a-sup-closed*:
$d(a(x) \sqcup a(y)) = a(x) \sqcup a(y)$
using *aa.sub-sup-closed d-def* **by** *auto*

lemma *a-mult-closed*:
$d(a(x) * a(y)) = a(x) * a(y)$
using *a-d-closed aa.l12* **by** *auto*

lemma *d-a-shunting-zero*:
$d(x) * a(y) = bot \longleftrightarrow d(x) \leq d(y)$
by (*simp add*: *aa.l10-iff d-def*)

lemma *d-d-shunting-zero*:
$d(x) * d(y) = bot \longleftrightarrow d(x) \leq a(y)$
by (*simp add*: *aa.l10-iff d-def*)

lemma *d-compl-intro*:
$d(x) \sqcup d(y) = d(x) \sqcup a(x) * d(y)$
by (*simp add*: *aa.sup-complement-intro d-def*)

lemma *a-compl-intro*:
$a(x) \sqcup a(y) = a(x) \sqcup d(x) * a(y)$
by (*simp add*: *aa.sup-complement-intro d-def*)

lemma *kat-2*:
$y * a(z) \leq a(x) * y \Longrightarrow d(x) * y * a(z) = bot$
by (*smt a-export a-plus-left-lower-bound le-sup-iff d-d-shunting-zero d-export d-strict le-iff-sup mult-assoc*)

lemma *kat-3*:
$d(x) * y * a(z) = bot \Longrightarrow d(x) * y = d(x) * y * d(z)$
by (*metis a-export-d aa.complement-bot d-complement-zero d-def mult-1-right mult-left-dist-sup sup-bot-left*)

lemma *kat-4*:
$d(x) * y = d(x) * y * d(z) \Longrightarrow d(x) * y \leq y * d(z)$
using *d-mult-below mult-assoc* **by** *auto*

lemma *kat-2-equiv*:
$y * a(z) \leq a(x) * y \longleftrightarrow d(x) * y * a(z) = bot$
apply (*rule iffI*)
apply (*simp add*: *kat-2*)
by (*metis aa.top-greatest a-complement sup-bot-left d-def mult-left-one mult-right-dist-sup mult-right-isotone mult-1-right*)

lemma *kat-4-equiv*:
$d(x) * y = d(x) * y * d(z) \longleftrightarrow d(x) * y \leq y * d(z)$
apply (*rule iffI*)
apply (*simp add*: *kat-4*)
apply (*rule order.antisym*)
apply (*metis d-mult-idempotent le-iff-sup mult-assoc mult-left-dist-sup*)
by (*metis d-plus-one le-iff-sup mult-left-dist-sup mult-1-right*)

lemma *kat-3-equiv-opp*:
$a(z) * y * d(x) = bot \longleftrightarrow y * d(x) = d(z) * y * d(x)$
by (*metis a-complement a-restrict sup-bot-left d-a-closed d-def mult-assoc mult-left-one mult-left-zero mult-right-dist-sup*)

lemma *kat-4-equiv-opp*:
$y * d(x) = d(z) * y * d(x) \longleftrightarrow y * d(x) \leq d(z) * y$
using *kat-2-equiv kat-3-equiv-opp d-def* **by** *auto*

lemma *d-restrict-iff*:
$(x \leq y) \longleftrightarrow (x \leq d(x) * y)$
by (*metis d-mult-below d-restrict-equals mult-isotone order-lesseq-imp*)

lemma *d-restrict-iff-1*:
$(d(x) * y \leq z) \longleftrightarrow (d(x) * y \leq d(x) * z)$
by (*metis sup-commute d-export d-mult-left-lower-bound d-plus-one d-restrict-iff mult-left-isotone mult-left-one mult-right-sub-dist-sup-right order-trans*)

end

end

23 Domain Iterings

theory *Domain-Iterings*

imports *Domain Lattice-Ordered-Semirings Omega-Algebras*

begin

class *domain-semiring-lattice* = *left-zero-domain-semiring* + *lattice-ordered-pre-left-semiring*
begin

subclass *bounded-idempotent-left-zero-semiring* **..**

lemma *d-top*:
$d(top) = 1$
by (*metis sup-left-top d-dist-sup d-one d-plus-one*)

lemma *mult-domain-top*:
$x * d(y) * top \leq d(x * y) * top$
by (*smt d-mult-d d-restrict-equals mult-assoc mult-right-isotone top-greatest*)

lemma *domain-meet-domain*:
$d(x \sqcap d(y) * z) \leq d(y)$
by (*metis d-export d-isotone d-mult-greatest-lower-bound inf.cobounded2*)

lemma *meet-domain*:

$x \sqcap d(y) * z = d(y) * (x \sqcap z)$
apply (*rule order.antisym*)
apply (*metis domain-meet-domain d-mult-below d-restrict-equals inf-mono mult-isotone*)
by (*meson d-mult-below le-inf-iff mult-left-sub-dist-inf-right*)

lemma *meet-intro-domain*:
$x \sqcap y = d(y) * x \sqcap y$
by (*metis d-restrict-equals inf-commute meet-domain*)

lemma *meet-domain-top*:
$x \sqcap d(y) * top = d(y) * x$
by (*simp add: meet-domain*)

proposition $d(x) = x * top \sqcap 1$ **nitpick** [*expect=genuine,card=3*] **oops**

lemma *d-galois*:
$d(x) \leq d(y) \longleftrightarrow x \leq d(y) * top$
by (*metis d-export d-isotone d-mult-left-absorb-sup d-plus-one d-restrict-equals d-top mult-isotone top.extremum*)

lemma *vector-meet*:
$x * top \sqcap y \leq d(x) * y$
by (*metis d-galois d-mult-sub inf.sup-monoid.add-commute inf.sup-right-isotone meet-domain-top*)

end

class *domain-semiring-lattice-L* = *domain-semiring-lattice* + *L* +
assumes *l1*: $x * L = x * bot \sqcup d(x) * L$
assumes *l2*: $d(L) * x \leq x * d(L)$
assumes *l3*: $d(L) * top \leq L \sqcup d(L * bot) * top$
assumes *l4*: $L * top \leq L$
assumes *l5*: $x * bot \sqcap L \leq (x \sqcap L) * bot$
begin

lemma *l8*:
$(x \sqcap L) * bot \leq x * bot \sqcap L$
by (*meson inf.boundedE inf.boundedI mult-right-sub-dist-inf-left zero-right-mult-decreasing*)

lemma *l9*:
$x * bot \sqcap L \leq d(x * bot) * L$
by (*metis vector-meet vector-mult-closed zero-vector*)

lemma *l10*:
$L * L = L$
by (*metis d-restrict-equals l1 le-iff-sup zero-right-mult-decreasing*)

lemma *l11*:
$d(x) * L \leq x * L$
by (*metis l1 sup.cobounded2*)

lemma *l12*:
$d(x * bot) * L \leq x * bot$
by (*metis sup-right-divisibility l1 mult-assoc mult-left-zero*)

lemma *l13*:
$d(x * bot) * L \leq x$
using *l12 order-trans zero-right-mult-decreasing* **by** *blast*

lemma *l14*:
$x * L \leq x * bot \sqcup L$
by (*metis d-mult-below l1 sup-right-isotone*)

lemma *l15*:
$x * d(y) * L = x * bot \sqcup d(x * y) * L$
by (*metis d-commutative d-mult-d d-zero l1 mult-assoc mult-left-zero*)

lemma *l16*:
$x * top \sqcap L \leq x * L$
using *inf.order-lesseq-imp l11 vector-meet* **by** *blast*

lemma *l17*:
$d(x) * L \leq d(x * L) * L$
by (*metis d-mult-below l11 le-infE le-infI meet-intro-domain*)

lemma *l18*:
$d(x) * L = d(x * L) * L$
by (*simp add*: *order.antisym d-mult-sub l17 mult-left-isotone*)

lemma *l19*:
$d(x * top * bot) * L \leq d(x * L) * L$
by (*metis d-mult-sub l18 mult-assoc mult-left-isotone*)

lemma *l20*:
$x \leq y \longleftrightarrow x \leq y \sqcup L \wedge x \leq y \sqcup d(y * bot) * top$
apply (*rule iffI*)
apply (*simp add*: *le-supI1*)
by (*smt sup-commute sup-inf-distrib1 l13 le-iff-sup meet-domain-top*)

lemma *l21*:
$d(x * bot) * L \leq x * bot \sqcap L$
by (*simp add*: *d-mult-below l12*)

lemma *l22*:
$x * bot \sqcap L = d(x * bot) * L$
using *l21 order.antisym l9* **by** *auto*

lemma *l23*:
$x * top \sqcap L = d(x) * L$
apply (*rule order.antisym*)
apply (*simp add*: *vector-meet*)
by (*metis d-mult-below inf.le-sup-iff inf-top.left-neutral l1 le-supE mult-left-sub-dist-inf-left*)

lemma *l29*:
$L * d(L) = L$
by (*metis d-preserves-equation d-restrict-equals l2*)

lemma *l30*:
$d(L) * x \leq (x \sqcap L) \sqcup d(L * bot) * x$
by (*metis inf.sup-right-divisibility inf-left-commute inf-sup-distrib1 l3 meet-domain-top*)

lemma *l31*:
$d(L) * x = (x \sqcap L) \sqcup d(L * bot) * x$
by (*smt* (*z3*) *l30 d-dist-sup le-iff-sup meet-intro-domain semiring.combine-common-factor sup-commute sup-inf-absorb zero-right-mult-decreasing*)

lemma *l40*:
$L * x \leq L$
by (*meson bot-least inf.order-trans l4 semiring.mult-left-mono top.extremum*)

lemma *l41*:
$L * top = L$
by (*simp add*: *l40 order.antisym top-right-mult-increasing*)

lemma *l50*:
$x * bot \sqcap L = (x \sqcap L) * bot$
using *order.antisym l5 l8* **by** *force*

lemma *l51*:
$d(x * bot) * L = (x \sqcap L) * bot$
using *l22 l50* **by** *auto*

lemma *l90*:
$L * top * L = L$
by (*simp add*: *l41 l10*)

lemma *l91*:
assumes $x = x * top$
shows $d(L * bot) * x \leq d(x * bot) * top$
proof −
have $d(L * bot) * x \leq d(d(L * bot) * x) * top$
using *d-galois* **by** *blast*

also have ... $= d(d(L * bot) * d(x)) * top$
using *d-mult-d* **by** *auto*
also have ... $= d(d(x) * L * bot) * top$
using *d-commutative d-mult-d ils.il-inf-associative* **by** *auto*
also have ... $\leq d(x * L * bot) * top$
by (*metis d-isotone l11 mult-left-isotone*)
also have ... $\leq d(x * top * bot) * top$
by (*simp add*: *d-isotone mult-left-isotone mult-right-isotone*)
finally show *?thesis*
using *assms* **by** *auto*
qed

lemma *l92*:
assumes $x = x * top$
shows $d(L * bot) * x \leq d((x \sqcap L) * bot) * top$
proof −
have $d(L * bot) * x = d(L) * d(L * bot) * x$
using *d-commutative d-mult-sub d-order* **by** *auto*
also have ... $\leq d(L) * d(x * bot) * top$
by (*metis assms order.eq-iff l91 mult-assoc mult-isotone*)
also have ... $= d(d(x * bot) * L) * top$
by (*simp add*: *d-commutative d-export*)
also have ... $\leq d((x \sqcap L) * bot) * top$
by (*simp add*: *l51*)
finally show *?thesis*
.
qed

end

class *domain-itering-lattice-L* = *bounded-itering* + *domain-semiring-lattice-L*
begin

lemma *mult-L-circ*:
$(x * L)^\circ = 1 \sqcup x * L$
by (*metis circ-back-loop-fixpoint circ-mult l40 le-iff-sup mult-assoc*)

lemma *mult-L-circ-mult-below*:
$(x * L)^\circ * y \leq y \sqcup x * L$
by (*smt sup-right-isotone l40 mult-L-circ mult-assoc mult-left-one*
mult-right-dist-sup mult-right-isotone)

lemma *circ-L*:
$L^\circ = L \sqcup 1$
by (*metis sup-commute l10 mult-L-circ*)

lemma *circ-d0-L*:
$x^\circ * d(x * bot) * L = x^\circ * bot$
by (*metis sup-bot-right circ-loop-fixpoint circ-plus-same d-zero l15 mult-assoc*

mult-left-zero)

lemma *d0-circ-left-unfold*:
  $d(x^\circ * bot) = d(x * x^\circ * bot)$
  by (*metis sup-commute sup-bot-left circ-loop-fixpoint mult-assoc*)

lemma *d-circ-import*:
  $d(y) * x \leq x * d(y) \Longrightarrow d(y) * x^\circ = d(y) * (d(y) * x)^\circ$
  apply (*rule order.antisym*)
  apply (*simp add*: *circ-import d-mult-idempotent d-plus-one le-iff-sup*)
  using *circ-isotone d-mult-below mult-right-isotone* **by** *auto*

end

class *domain-omega-algebra-lattice-L* = *bounded-left-zero-omega-algebra* + *domain-semiring-lattice-L*
begin

lemma *mult-L-star*:
  $(x * L)^\star = 1 \sqcup x * L$
  by (*metis l40 le-iff-sup mult-assoc star.circ-back-loop-fixpoint star.circ-mult*)

lemma *mult-L-omega*:
  $(x * L)^\omega \leq x * L$
  by (*metis l40 mult-right-isotone omega-slide*)

lemma *mult-L-sup-star*:
  $(x * L \sqcup y)^\star = y^\star \sqcup y^\star * x * L$
proof (*rule order.antisym*)
  have $(x * L \sqcup y) * (y^\star \sqcup y^\star * x * L) = x * L * (y^\star \sqcup y^\star * x * L) \sqcup y * (y^\star \sqcup y^\star * x * L)$
    by (*simp add*: *mult-right-dist-sup*)
  also have ... $\leq x * L \sqcup y * (y^\star \sqcup y^\star * x * L)$
    by (*metis sup-left-isotone l40 mult-assoc mult-right-isotone*)
  also have ... $\leq x * L \sqcup y * y^\star \sqcup y^\star * x * L$
    by (*smt sup-assoc sup-commute sup-ge2 mult-assoc mult-left-dist-sup star.circ-loop-fixpoint*)
  also have ... $\leq x * L \sqcup y^\star \sqcup y^\star * x * L$
    by (*meson order-refl star.left-plus-below-circ sup-mono*)
  also have ... $= y^\star \sqcup y^\star * x * L$
    by (*metis sup-assoc sup-commute mult-assoc star.circ-loop-fixpoint star.circ-reflexive star.circ-sup-one-right-unfold star-involutive*)
  finally have $1 \sqcup (x * L \sqcup y) * (y^\star \sqcup y^\star * x * L) \leq y^\star \sqcup y^\star * x * L$
    by (*meson le-supI le-supI1 star.circ-reflexive*)
  thus $(x * L \sqcup y)^\star \leq y^\star \sqcup y^\star * x * L$
    using *star-left-induct* **by** *fastforce*
next
  show $y^\star \sqcup y^\star * x * L \leq (x * L \sqcup y)^\star$
    by (*metis sup-commute le-sup-iff mult-assoc star.circ-increasing*

star.circ-mult-upper-bound star.circ-sub-dist)
qed

lemma *mult-L-sup-omega*:
  $(x * L \sqcup y)^\omega \leq y^\omega \sqcup y^\star * x * L$
proof −
  have *1*: $(y^\star * x * L)^\omega \leq y^\omega \sqcup y^\star * x * L$
    by (*simp add*: *le-supI2 mult-L-omega*)
  have $(y^\star * x * L)^\star * y^\omega \leq y^\omega \sqcup y^\star * x * L$
    by (*metis sup-right-isotone l40 mult-assoc mult-right-isotone star-left-induct*)
  thus *?thesis*
    using *1* **by** (*simp add*: *ils.il-inf-associative omega-decompose*
sup-monoid.add-commute)
qed

end

sublocale *domain-omega-algebra-lattice-L* < *dL-star*: *itering* **where** *circ* = *star*
..

sublocale *domain-omega-algebra-lattice-L* < *dL-star*: *domain-itering-lattice-L*
where *circ* = *star* **..**

context *domain-omega-algebra-lattice-L*
begin

lemma *d0-star-below-d0-omega*:
  $d(x^\star * bot) \leq d(x^\omega * bot)$
  by (*simp add*: *d-isotone star-bot-below-omega-bot*)

lemma *d0-below-d0-omega*:
  $d(x * bot) \leq d(x^\omega * bot)$
  by (*metis d0-star-below-d0-omega d-isotone mult-left-isotone order-trans*
star.circ-increasing)

lemma *star-L-split*:
  assumes $y \leq z$
      and $x * z * L \leq x * bot \sqcup z * L$
    shows $x^\star * y * L \leq x^\star * bot \sqcup z * L$
proof −
  have $x * (x^\star * bot \sqcup z * L) \leq x^\star * bot \sqcup x * z * L$
    by (*metis sup-bot-right order.eq-iff mult-assoc mult-left-dist-sup*
star.circ-loop-fixpoint)
  also have ... $\leq x^\star * bot \sqcup x * bot \sqcup z * L$
    using *assms*(*2*) *semiring.add-left-mono sup-monoid.add-assoc* **by** *auto*
  also have ... $= x^\star * bot \sqcup z * L$
    using *mult-isotone star.circ-increasing sup.absorb-iff1* **by** *force*
  finally have $y * L \sqcup x * (x^\star * bot \sqcup z * L) \leq x^\star * bot \sqcup z * L$
    by (*simp add*: *assms*(*1*) *le-supI1 mult-left-isotone sup-monoid.add-commute*)

thus *?thesis*
by (*simp add*: *star-left-induct mult.assoc*)
qed

lemma *star-L-split-same*:
$x * y * L \le x * bot \sqcup y * L \Longrightarrow x^\star * y * L = x^\star * bot \sqcup y * L$
apply (*rule order.antisym*)
using *star-L-split* **apply** *blast*
by (*metis bot-least ils.il-inf-associative le-supI mult-isotone mult-left-one order-refl star.circ-reflexive*)

lemma *star-d-L-split-equal*:
$d(x * y) \le d(y) \Longrightarrow x^\star * d(y) * L = x^\star * bot \sqcup d(y) * L$
by (*metis sup-right-isotone l15 le-iff-sup mult-right-sub-dist-sup-left star-L-split-same*)

lemma *d0-omega-mult*:
$d(x^\omega * y * bot) = d(x^\omega * bot)$
apply (*rule order.antisym*)
apply (*simp add*: *d-isotone mult-isotone omega-sub-vector*)
by (*metis d-isotone mult-assoc mult-right-isotone bot-least*)

lemma *d-omega-export*:
$d(y) * x \le x * d(y) \Longrightarrow d(y) * x^\omega = (d(y) * x)^\omega$
apply (*rule order.antisym*)
apply (*simp add*: *d-preserves-equation omega-simulation*)
by (*smt le-iff-sup mult-left-dist-sup omega-simulation-2 omega-slide*)

lemma *d-omega-import*:
$d(y) * x \le x * d(y) \Longrightarrow d(y) * x^\omega = d(y) * (d(y) * x)^\omega$
using *d-mult-idempotent omega-import order.refl* **by** *auto*

lemma *star-d-omega-top*:
$x^\star * d(x^\omega) * top = x^\star * bot \sqcup d(x^\omega) * top$
apply (*rule order.antisym*)
apply (*metis le-supI2 mult-domain-top star-mult-omega*)
by (*metis ils.il-inf-associative le-supI mult-left-one mult-left-sub-dist-sup-right mult-right-sub-dist-sup-left star.circ-right-unfold-1 sup-monoid.add-0-right*)

lemma *omega-meet-L*:
$x^\omega \sqcap L = d(x^\omega) * L$
by (*metis l23 omega-vector*)

proposition *d-star-mult*: $d(x * y) \le d(y) \Longrightarrow d(x^\star * y) = d(x^\star * bot) \sqcup d(y)$
oops
proposition *d0-split-omega-omega*: $x^\omega \le x^\omega * bot \sqcup d(x^\omega \sqcap L) * top$ **nitpick** [*expect=genuine*,*card=2*] **oops**

end

end

24 Domain Recursion

theory *Domain-Recursion*

imports *Domain-Iterings Approximation*

begin

class *domain-semiring-lattice-apx* = *domain-semiring-lattice-L* + *apx* +
  assumes *apx-def*: $x \sqsubseteq y \longleftrightarrow x \leq y \sqcup L \wedge d(L) * y \leq x \sqcup d(x * bot) * top$
begin

lemma *apx-transitive*:
  assumes $x \sqsubseteq y$
    and $y \sqsubseteq z$
   shows $x \sqsubseteq z$
proof −
  have *1*: $x \leq z \sqcup L$
   by (*smt assms sup-assoc sup-commute apx-def le-iff-sup*)
  have $d(d(L) * y * bot) * top \leq d((x \sqcup d(x * bot) * top) * bot) * top$
   by (*metis assms*(*1*) *apx-def d-isotone mult-left-isotone*)
  also have ... $\leq d(x * bot) * top$
   by (*metis le-sup-iff d-galois mult-left-isotone mult-right-dist-sup order-refl zero-right-mult-decreasing*)
  finally have *2*: $d(d(L) * y * bot) * top \leq d(x * bot) * top$
   .
  have $d(L) * z = d(L) * (d(L) * z)$
   by (*simp add*: *d-mult-idempotent ils.il-inf-associative*)
  also have ... $\leq d(L) * y \sqcup d(d(L) * y * bot) * top$
   by (*metis assms*(*2*) *apx-def d-export mult-assoc mult-left-dist-sup mult-right-isotone*)
  also have ... $\leq x \sqcup d(x * bot) * top$
   using *2* **by** (*meson assms*(*1*) *apx-def le-supI2 sup-least*)
  finally show *?thesis*
   using *1* **by** (*simp add*: *apx-def*)
qed

lemma *apx-meet-L*:
  assumes $y \sqsubseteq x$
   shows $x \sqcap L \leq y \sqcap L$
proof −
  have $x \sqcap L = d(L) * x \sqcap L$
   using *meet-intro-domain* **by** *auto*
  also have ... $\leq (y \sqcup d(y * bot) * top) \sqcap L$
   using *assms apx-def inf.sup-left-isotone* **by** *blast*
  also have ... $\leq y$

by (*simp add*: *inf.sup-monoid.add-commute inf-sup-distrib1 l13 meet-domain-top*)
finally show *?thesis*
by *simp*
qed

lemma *sup-apx-left-isotone*:
assumes $x \sqsubseteq y$
shows $x \sqcup z \sqsubseteq y \sqcup z$
proof −
have *1*: $x \sqcup z \leq y \sqcup z \sqcup L$
by (*smt assms sup-assoc sup-commute sup-left-isotone apx-def*)
have $d(L) * (y \sqcup z) = d(L) * y \sqcup d(L) * z$
by (*simp add*: *mult-left-dist-sup*)
also have ... $\leq d(L) * y \sqcup z$
by (*simp add*: *d-mult-below le-supI1 sup-commute*)
also have ... $\leq x \sqcup d(x * bot) * top \sqcup z$
using *assms apx-def sup-left-isotone* **by** *blast*
also have ... $\leq x \sqcup z \sqcup d((x \sqcup z) * bot) * top$
by (*simp add*: *d-dist-sup le-iff-sup semiring.distrib-right sup.left-commute sup-monoid.add-assoc*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def*)
qed

subclass *apx-biorder*
apply *unfold-locales*
apply (*metis le-sup-iff sup-ge1 apx-def d-plus-one mult-left-one mult-right-dist-sup*)
apply (*meson apx-meet-L order.antisym apx-def relative-equality sup-same-context*)
using *apx-transitive* **by** *blast*

lemma *mult-apx-left-isotone*:
assumes $x \sqsubseteq y$
shows $x * z \sqsubseteq y * z$
proof −
have $x * z \leq y * z \sqcup L * z$
by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
hence *1*: $x * z \leq y * z \sqcup L$
using *l40 order-lesseq-imp semiring.add-left-mono* **by** *blast*
have $d(L) * y * z \leq x * z \sqcup d(x * bot) * top * z$
by (*metis assms apx-def mult-left-isotone mult-right-dist-sup*)
also have ... $\leq x * z \sqcup d(x * z * bot) * top$
by (*metis sup-right-isotone d-isotone mult-assoc mult-isotone mult-right-isotone top-greatest bot-least*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def mult-assoc*)
qed

lemma *mult-apx-right-isotone*:
assumes $x \sqsubseteq y$
shows $z * x \sqsubseteq z * y$
proof −
have $z * x \leq z * y \sqcup z * L$
by (*metis assms apx-def mult-left-dist-sup mult-right-isotone*)
also have ... $\leq z * y \sqcup z * bot \sqcup L$
using *l14 semiring.add-left-mono sup-monoid.add-assoc* **by** *auto*
finally have *1*: $z * x \leq z * y \sqcup L$
using *mult-right-isotone sup.order-iff* **by** *auto*
have $d(L) * z * y \leq z * d(L) * y$
by (*simp add*: *l2 mult-left-isotone*)
also have ... $\leq z * (x \sqcup d(x * bot) * top)$
by (*metis assms apx-def mult-assoc mult-right-isotone*)
also have ... $= z * x \sqcup z * d(x * bot) * top$
by (*simp add*: *mult-left-dist-sup mult-assoc*)
also have ... $\leq z * x \sqcup d(z * x * bot) * top$
by (*metis sup-right-isotone mult-assoc mult-domain-top*)
finally show *?thesis*
using *1* **by** (*simp add*: *apx-def mult-assoc*)
qed

subclass *apx-semiring*
apply *unfold-locales*
apply (*metis sup-ge2 apx-def l3 mult-right-isotone order-trans top-greatest*)
apply (*simp add*: *sup-apx-left-isotone*)
apply (*simp add*: *mult-apx-left-isotone*)
by (*simp add*: *mult-apx-right-isotone*)

lemma *meet-L-apx-isotone*:
$x \sqsubseteq y \Longrightarrow x \sqcap L \sqsubseteq y \sqcap L$
by (*smt* (*z3*) *inf.cobounded2 sup.coboundedI1 sup-absorb sup-commute apx-def apx-meet-L d-restrict-equals l20 inf-commute meet-domain*)

definition *kappa-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-apx-meet f* $\equiv$ *apx.has-least-fixpoint f* $\wedge$ *has-apx-meet* $(\mu\ f)$ $(\nu\ f)$ $\wedge\ \kappa\ f = \mu\ f \vartriangle \nu\ f$

definition *kappa-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *kappa-mu-nu f* $\equiv$ *apx.has-least-fixpoint f* $\wedge\ \kappa\ f = \mu\ f \sqcup (\nu\ f \sqcap L)$

definition *nu-below-mu-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *nu-below-mu-nu f* $\equiv d(L) * \nu\ f \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d(\nu\ f * bot) * top$

definition *nu-below-mu-nu-2* :: $('a \Rightarrow 'a) \Rightarrow bool$
where *nu-below-mu-nu-2 f* $\equiv d(L) * \nu\ f \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d((\mu\ f \sqcup (\nu\ f \sqcap L)) * bot) * top$

definition *mu-nu-apx-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *mu-nu-apx-nu f* $\equiv \mu\ f \sqcup (\nu\ f \sqcap L) \sqsubseteq \nu\ f$

definition *mu-nu-apx-meet* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *mu-nu-apx-meet f* $\equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \bigtriangleup \nu\ f = \mu\ f \sqcup (\nu\ f \sqcap L)$

definition *apx-meet-below-nu* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *apx-meet-below-nu f* $\equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \bigtriangleup \nu\ f \leq \nu\ f$

lemma *mu-below-l*:
  $\mu\ f \leq \mu\ f \sqcup (\nu\ f \sqcap L)$
  by *simp*

lemma *l-below-nu*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow \mu\ f \sqcup (\nu\ f \sqcap L) \leq \nu\ f$
  by (*simp add*: *mu-below-nu*)

lemma *n-l-nu*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow (\mu\ f \sqcup (\nu\ f \sqcap L)) \sqcap L = \nu\ f \sqcap L$
  by (*meson l-below-nu inf.sup-same-context inf-le1 order-trans sup.cobounded2*)

lemma *l-apx-mu*:
  $\mu\ f \sqcup (\nu\ f \sqcap L) \sqsubseteq \mu\ f$
  by (*simp add*: *apx-def d-mult-below le-supI1 sup-inf-distrib1*)

lemma *nu-below-mu-nu-nu-below-mu-nu-2*:
  assumes *nu-below-mu-nu f*
    shows *nu-below-mu-nu-2 f*
proof −
  have $d(L) * \nu\ f = d(L) * (d(L) * \nu\ f)$
    by (*simp add*: *d-mult-idempotent ils.il-inf-associative*)
  also have ... $\leq d(L) * (\mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d(\nu\ f * bot) * top)$
    using *assms mult-isotone nu-below-mu-nu-def* **by** *blast*
  also have ... $= d(L) * (\mu\ f \sqcup (\nu\ f \sqcap L)) \sqcup d(L) * d(\nu\ f * bot) * top$
    by (*simp add*: *ils.il-inf-associative mult-left-dist-sup*)
  also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d(L) * d(\nu\ f * bot) * top$
    using *d-mult-below sup-left-isotone* **by** *auto*
  also have ... $= \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d(d(\nu\ f * bot) * L) * top$
    by (*simp add*: *d-commutative d-export*)
  also have ... $= \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d((\nu\ f \sqcap L) * bot) * top$
    using *l51* **by** *auto*
  also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d((\mu\ f \sqcup (\nu\ f \sqcap L)) * bot) * top$
    by (*meson d-isotone inf.eq-refl mult-isotone semiring.add-left-mono sup.cobounded2*)
  finally show *?thesis*
    using *nu-below-mu-nu-2-def* **by** *auto*
qed

lemma *nu-below-mu-nu-2-nu-below-mu-nu*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *nu-below-mu-nu-2 f*
shows *nu-below-mu-nu f*
proof −
have $d(L) * \nu\ f \leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d((\mu\ f \sqcup (\nu\ f \sqcap L)) * bot) * top$
using *assms*(*3*) *nu-below-mu-nu-2-def* **by** *blast*
also have ... $\leq \mu\ f \sqcup (\nu\ f \sqcap L) \sqcup d(\nu\ f * bot) * top$
by (*metis assms*(*1*,*2*) *d-isotone inf.sup-monoid.add-commute inf.sup-right-divisibility le-supI le-supI2 mu-below-nu mult-left-isotone sup-left-divisibility*)
finally show *?thesis*
by (*simp add*: *nu-below-mu-nu-def*)
qed

lemma *nu-below-mu-nu-equivalent*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ (*nu-below-mu-nu f* $\longleftrightarrow$ *nu-below-mu-nu-2 f*)
using *nu-below-mu-nu-2-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *blast*

lemma *nu-below-mu-nu-2-mu-nu-apx-nu*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *nu-below-mu-nu-2 f*
shows *mu-nu-apx-nu f*
proof −
have $\mu\ f \sqcup (\nu\ f \sqcap L) \leq \nu\ f \sqcup L$
using *assms*(*1*,*2*) *l-below-nu le-supI1* **by** *blast*
thus *?thesis*
using *assms*(*3*) *apx-def mu-nu-apx-nu-def nu-below-mu-nu-2-def* **by** *blast*
qed

lemma *mu-nu-apx-nu-mu-nu-apx-meet*:
assumes *mu-nu-apx-nu f*
shows *mu-nu-apx-meet f*
proof −
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
have *is-apx-meet* ($\mu\ f$) ($\nu\ f$) *?l*
apply (*unfold is-apx-meet-def*, *intro conjI*)
apply (*simp add*: *l-apx-mu*)
using *assms mu-nu-apx-nu-def* **apply** *blast*
by (*metis apx-meet-L le-supI2 sup.order-iff sup-apx-left-isotone sup-inf-absorb*)
thus *?thesis*
by (*smt apx-meet-char mu-nu-apx-meet-def*)
qed

lemma *mu-nu-apx-meet-apx-meet-below-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *mu-nu-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def l-below-nu mu-nu-apx-meet-def* **by** *auto*

lemma *apx-meet-below-nu-nu-below-mu-nu-2*:
assumes *apx-meet-below-nu f*
shows *nu-below-mu-nu-2 f*
proof −
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
have $\forall m\ .\ m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \leq \nu\ f \longrightarrow d(L) * \nu\ f \leq\ ?l \sqcup d(?l * bot) * top$
proof
fix m
show $m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \leq \nu\ f \longrightarrow d(L) * \nu\ f \leq\ ?l \sqcup d(?l * bot) * top$
proof
assume *1*: $m \sqsubseteq \mu\ f \wedge m \sqsubseteq \nu\ f \wedge m \leq \nu\ f$
hence $m \leq\ ?l$
by (*metis apx-def ils.il-associative sup.orderE sup.orderI sup-inf-distrib1 sup-inf-distrib2*)
hence $m \sqcup d(m * bot) * top \leq\ ?l \sqcup d(?l * bot) * top$
by (*meson d-isotone order.trans le-supI le-supI2 mult-left-isotone sup.cobounded1*)
thus $d(L) * \nu\ f \leq\ ?l \sqcup d(?l * bot) * top$
using *1 apx-def order-lesseq-imp* **by** *blast*
qed
qed
thus *?thesis*
by (*smt* (*verit*) *assms apx-meet-below-nu-def apx-meet-same apx-meet-unique is-apx-meet-def nu-below-mu-nu-2-def*)
qed

lemma *has-apx-least-fixpoint-kappa-apx-meet*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *apx.has-least-fixpoint f*
shows *kappa-apx-meet f*
proof −
have *1*: $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow d(L) * \kappa\ f \leq w \sqcup d(w * bot) * top$
by (*metis assms*(*2*,*3*) *apx-def mult-right-isotone order-trans kappa-below-nu*)
have $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow w \leq \kappa\ f \sqcup L$
by (*metis assms*(*1*,*3*) *sup-left-isotone apx-def mu-below-kappa order-trans*)
hence $\forall w\ .\ w \sqsubseteq \mu\ f \wedge w \sqsubseteq \nu\ f \longrightarrow w \sqsubseteq \kappa\ f$
using *1 apx-def* **by** *blast*
hence *is-apx-meet* $(\mu\ f)\ (\nu\ f)\ (\kappa\ f)$
using *assms apx-meet-char is-apx-meet-def kappa-apx-below-mu kappa-apx-below-nu kappa-apx-meet-def* **by** *presburger*
thus *?thesis*
by (*simp add*: *assms*(*3*) *kappa-apx-meet-def apx-meet-char*)

qed

lemma *kappa-apx-meet-apx-meet-below-nu*:
has-greatest-fixpoint f $\Longrightarrow$ *kappa-apx-meet f* $\Longrightarrow$ *apx-meet-below-nu f*
using *apx-meet-below-nu-def kappa-apx-meet-def kappa-below-nu* **by** *force*

lemma *apx-meet-below-nu-kappa-mu-nu*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *isotone f*
and *apx.isotone f*
and *apx-meet-below-nu f*
shows *kappa-mu-nu f*
proof $-$
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
let $?m = \mu\ f \vartriangle \nu\ f$
have *1*: $?m = ?l$
by (*metis assms*(*1*,*2*,*5*) *apx-meet-below-nu-nu-below-mu-nu-2*
mu-nu-apx-meet-def mu-nu-apx-nu-mu-nu-apx-meet
nu-below-mu-nu-2-mu-nu-apx-nu)
have *2*: $?l \le f(?l) \sqcup L$
proof $-$
have $?l \le \mu\ f \sqcup L$
using *sup-right-isotone* **by** *auto*
also have ... $= f(\mu\ f) \sqcup L$
by (*simp add*: *assms*(*1*) *mu-unfold*)
also have ... $\le f(?l) \sqcup L$
by (*metis assms*(*3*) *sup-left-isotone sup-ge1 isotone-def*)
finally show *?thesis*
.
qed
have $d(L) * f(?l) \le ?l \sqcup d(?l * bot) * top$
proof $-$
have $d(L) * f(?l) \le d(L) * f(\nu\ f)$
by (*metis assms*(*1*$-$*3*) *l-below-nu mult-right-isotone ord.isotone-def*)
also have ... $= d(L) * \nu\ f$
by (*metis assms*(*2*) *nu-unfold*)
also have ... $\le ?l \sqcup d(?l * bot) * top$
using *apx-meet-below-nu-nu-below-mu-nu-2 assms*(*5*) *nu-below-mu-nu-2-def*
by *blast*
finally show *?thesis*
.
qed
hence *3*: $?l \sqsubseteq f(?l)$
using *2* **by** (*simp add*: *apx-def*)
have *4*: $f(?l) \sqsubseteq \mu\ f$
proof $-$
have $?l \sqsubseteq \mu\ f$
by (*simp add*: *l-apx-mu*)

thus *?thesis*
by (*metis assms*(*1*,*4*) *mu-unfold ord.isotone-def*)
qed
have *5*: $f(?l) \sqsubseteq \nu\ f$
proof −
have $?l \sqsubseteq \nu\ f$
by (*meson apx-meet-below-nu-nu-below-mu-nu-2 assms*(*1*,*2*,*5*) *l-below-nu apx-def le-supI1 nu-below-mu-nu-2-def*)
thus *?thesis*
by (*metis assms*(*2*,*4*) *nu-unfold ord.isotone-def*)
qed
hence $f(?l) \sqsubseteq ?l$
using *1 4 apx-meet-below-nu-def assms*(*5*) *apx-greatest-lower-bound* **by** *fastforce*
hence *6*: $f(?l) = ?l$
using *3 apx.order.antisym* **by** *blast*
have $\forall y\ .\ f(y) = y \longrightarrow ?l \sqsubseteq y$
proof
fix *y*
show $f(y) = y \longrightarrow ?l \sqsubseteq y$
proof
assume *7*: $f(y) = y$
hence *8*: $?l \le y \sqcup L$
using *assms*(*1*) *inf.cobounded2 is-least-fixpoint-def least-fixpoint semiring.add-mono* **by** *blast*
have $y \le \nu\ f$
using *7 assms*(*2*) *greatest-fixpoint is-greatest-fixpoint-def* **by** *auto*
hence $d(L) * y \le ?l \sqcup d(?l * bot) * top$
using *3 5* **by** (*smt* (*z3*) *apx.order.trans apx-def semiring.distrib-left sup.absorb-iff2 sup-assoc*)
thus $?l \sqsubseteq y$
using *8* **by** (*simp add*: *apx-def*)
qed
qed
thus *?thesis*
using *1 6* **by** (*smt* (*verit*) *kappa-mu-nu-def apx.is-least-fixpoint-def apx.least-fixpoint-char*)
qed

lemma *kappa-mu-nu-has-apx-least-fixpoint*:
kappa-mu-nu f $\Longrightarrow$ *apx.has-least-fixpoint f*
by (*simp add*: *kappa-mu-nu-def*)

lemma *nu-below-mu-nu-kappa-mu-nu*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu f* $\Longrightarrow$ *kappa-mu-nu f*
using *apx-meet-below-nu-kappa-mu-nu mu-nu-apx-meet-apx-meet-below-nu mu-nu-apx-nu-mu-nu-apx-meet nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *blast*

lemma *kappa-mu-nu-nu-below-mu-nu*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu f* $\Longrightarrow$ *nu-below-mu-nu f*
  by (*simp add*: *apx-meet-below-nu-nu-below-mu-nu-2 has-apx-least-fixpoint-kappa-apx-meet kappa-apx-meet-apx-meet-below-nu kappa-mu-nu-def nu-below-mu-nu-2-nu-below-mu-nu*)

definition *kappa-mu-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *kappa-mu-nu-L f* $\equiv$ *apx.has-least-fixpoint f* $\wedge\ \kappa\ f = \mu\ f \sqcup d(\nu\ f * bot) * L$

definition *nu-below-mu-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *nu-below-mu-nu-L f* $\equiv d(L) * \nu\ f \leq \mu\ f \sqcup d(\nu\ f * bot) * top$

definition *mu-nu-apx-nu-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *mu-nu-apx-nu-L f* $\equiv \mu\ f \sqcup d(\nu\ f * bot) * L \sqsubseteq \nu\ f$

definition *mu-nu-apx-meet-L* :: $('a \Rightarrow 'a) \Rightarrow bool$
  where *mu-nu-apx-meet-L f* $\equiv$ *has-apx-meet* $(\mu\ f)\ (\nu\ f) \wedge \mu\ f \vartriangle \nu\ f = \mu\ f \sqcup d(\nu\ f * bot) * L$

lemma *n-below-l*:
  $x \sqcup d(y * bot) * L \leq x \sqcup (y \sqcap L)$
  using *d-mult-below l13 sup-right-isotone* **by** *auto*

lemma *n-equal-l*:
  assumes *nu-below-mu-nu-L f*
    shows$\mu\ f \sqcup d(\nu\ f * bot) * L = \mu\ f \sqcup (\nu\ f \sqcap L)$
proof −
  have $\nu\ f \sqcap L \leq (\mu\ f \sqcup d(\nu\ f * bot) * top) \sqcap L$
    using *assms l31 nu-below-mu-nu-L-def* **by** *force*
  also have ... $\leq \mu\ f \sqcup d(\nu\ f * bot) * L$
    using *distrib(4) inf.sup-monoid.add-commute meet-domain-top sup-left-isotone* **by** *force*
  finally have $\mu\ f \sqcup (\nu\ f \sqcap L) \leq \mu\ f \sqcup d(\nu\ f * bot) * L$
    by *auto*
  thus *?thesis*
    by (*meson order.antisym n-below-l*)
qed

lemma *nu-below-mu-nu-L-nu-below-mu-nu*:
  nu-below-mu-nu-L f $\Longrightarrow$ *nu-below-mu-nu f*
  using *order-lesseq-imp sup.cobounded1 sup-left-isotone nu-below-mu-nu-L-def nu-below-mu-nu-def* **by** *blast*

lemma *nu-below-mu-nu-L-kappa-mu-nu-L*:
  has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *isotone f* $\Longrightarrow$ *apx.isotone f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *kappa-mu-nu-L f*

using *kappa-mu-nu-L-def kappa-mu-nu-def n-equal-l nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-kappa-mu-nu* **by** *auto*

lemma *nu-below-mu-nu-L-mu-nu-apx-nu-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-nu-L f*
using *mu-nu-apx-nu-L-def mu-nu-apx-nu-def n-equal-l nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *auto*

lemma *nu-below-mu-nu-L-mu-nu-apx-meet-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *nu-below-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-meet-L f*
using *mu-nu-apx-meet-L-def mu-nu-apx-meet-def mu-nu-apx-nu-mu-nu-apx-meet n-equal-l nu-below-mu-nu-2-mu-nu-apx-nu nu-below-mu-nu-L-nu-below-mu-nu nu-below-mu-nu-nu-below-mu-nu-2* **by** *auto*

lemma *mu-nu-apx-nu-L-nu-below-mu-nu-L*:
assumes *has-least-fixpoint f*
and *has-greatest-fixpoint f*
and *mu-nu-apx-nu-L f*
shows *nu-below-mu-nu-L f*
proof −
let $?n = \mu\ f \sqcup d(\nu\ f * bot) * L$
let $?l = \mu\ f \sqcup (\nu\ f \sqcap L)$
have $d(L) * \nu\ f \leq\ ?n \sqcup d(?n * bot) * top$
using *assms*(*3*) *apx-def mu-nu-apx-nu-L-def* **by** *blast*
also have ... $\leq\ ?n \sqcup d(?l * bot) * top$
using *d-isotone mult-left-isotone semiring.add-left-mono n-below-l* **by** *auto*
also have ... $\leq\ ?n \sqcup d(\nu\ f * bot) * top$
by (*meson assms*(*1*,*2*) *l-below-nu d-isotone mult-left-isotone sup-right-isotone*)
finally show *?thesis*
by (*metis sup-assoc sup-right-top mult-left-dist-sup nu-below-mu-nu-L-def*)
qed

lemma *kappa-mu-nu-L-mu-nu-apx-nu-L*:
has-greatest-fixpoint f $\Longrightarrow$ *kappa-mu-nu-L f* $\Longrightarrow$ *mu-nu-apx-nu-L f*
using *kappa-mu-nu-L-def kappa-apx-below-nu mu-nu-apx-nu-L-def* **by** *force*

lemma *mu-nu-apx-meet-L-mu-nu-apx-nu-L*:
mu-nu-apx-meet-L f $\Longrightarrow$ *mu-nu-apx-nu-L f*
using *apx-greatest-lower-bound mu-nu-apx-meet-L-def mu-nu-apx-nu-L-def* **by** *fastforce*

lemma *kappa-mu-nu-L-nu-below-mu-nu-L*:
has-least-fixpoint f $\Longrightarrow$ *has-greatest-fixpoint f* $\Longrightarrow$ *kappa-mu-nu-L f* $\Longrightarrow$ *nu-below-mu-nu-L f*
using *kappa-mu-nu-L-mu-nu-apx-nu-L mu-nu-apx-nu-L-nu-below-mu-nu-L* **by** *auto*

end

class *itering-apx* = *domain-itering-lattice-L* + *domain-semiring-lattice-apx*
begin

lemma *circ-apx-isotone*:
  assumes $x \sqsubseteq y$
    shows $x^\circ \sqsubseteq y^\circ$
proof −
  have *1*: $x \leq y \sqcup L \wedge d(L) * y \leq x \sqcup d(x * bot) * top$
    using *assms apx-def* **by** *auto*
  have $d(L) * y^\circ \leq (d(L) * y)^\circ$
    by (*metis d-circ-import d-mult-below l2*)
  also have ... $\leq x^\circ * (d(x * bot) * top * x^\circ)^\circ$
    using *1* **by** (*metis circ-sup-1 circ-isotone*)
  also have ... $= x^\circ \sqcup x^\circ * d(x * bot) * top$
    by (*metis circ-left-top mult-assoc mult-left-dist-sup mult-1-right mult-top-circ*)
  also have ... $\leq x^\circ \sqcup d(x^\circ * x * bot) * top$
    by (*metis sup-right-isotone mult-assoc mult-domain-top*)
  finally have *2*: $d(L) * y^\circ \leq x^\circ \sqcup d(x^\circ * bot) * top$
    using *circ-plus-same d0-circ-left-unfold* **by** *auto*
  have $x^\circ \leq y^\circ * L^\circ$
    using *1* **by** (*metis circ-sup-1 circ-back-loop-fixpoint circ-isotone l40 le-iff-sup mult-assoc*)
  also have ... $= y^\circ \sqcup y^\circ * L$
    by (*simp add*: *circ-L mult-left-dist-sup sup-commute*)
  also have ... $\leq y^\circ \sqcup y^\circ * bot \sqcup L$
    using *l14 semiring.add-left-mono sup-monoid.add-assoc* **by** *auto*
  finally have $x^\circ \leq y^\circ \sqcup L$
    using *sup.absorb-iff1 zero-right-mult-decreasing* **by** *auto*
  thus *?thesis*
    using *2* **by** (*simp add*: *apx-def*)
qed

end

class *omega-algebra-apx* = *domain-omega-algebra-lattice-L* + *domain-semiring-lattice-apx*

sublocale *omega-algebra-apx* < *star*: *itering-apx* **where** *circ* = *star* **..**

context *omega-algebra-apx*
begin

lemma *omega-apx-isotone*:
  assumes $x \sqsubseteq y$
    shows $x^\omega \sqsubseteq y^\omega$
proof −

have *1*: $x \le y \sqcup L \wedge d(L) * y \le x \sqcup d(x * bot) * top$
using *assms apx-def* **by** *auto*
have $d(L) * y^\omega = (d(L) * y)^\omega$
by (*simp add*: *d-omega-export l2*)
also have ... $\le (x \sqcup d(x * bot) * top)^\omega$
using *1* **by** (*simp add*: *omega-isotone*)
also have ... $= (x^\star * d(x * bot) * top)^\omega \sqcup (x^\star * d(x * bot) * top)^\star * x^\omega$
by (*simp add*: *ils.il-inf-associative omega-decompose*)
also have ... $\le x^\star * d(x * bot) * top \sqcup (x^\star * d(x * bot) * top)^\star * x^\omega$
using *mult-top-omega sup-left-isotone* **by** *blast*
also have ... $= x^\star * d(x * bot) * top \sqcup (1 \sqcup x^\star * d(x * bot) * top * (x^\star * d(x * bot) * top)^\star) * x^\omega$
by (*simp add*: *star-left-unfold-equal*)
also have ... $\le x^\omega \sqcup x^\star * d(x * bot) * top$
by (*smt* (*verit*, *ccfv-threshold*) *sup-mono le-sup-iff mult-assoc mult-left-one mult-right-dist-sup mult-right-isotone order-refl top-greatest*)
also have ... $\le x^\omega \sqcup d(x^\star * x * bot) * top$
by (*metis sup-right-isotone mult-assoc mult-domain-top*)
also have ... $\le x^\omega \sqcup d(x^\star * bot) * top$
by (*simp add*: *dL-star.d0-circ-left-unfold star-plus*)
finally have *2*: $d(L) * y^\omega \le x^\omega \sqcup d(x^\omega * bot) * top$
by (*meson sup-right-isotone d0-star-below-d0-omega mult-left-isotone order-trans*)
have $x^\omega \le (y \sqcup L)^\omega$
using *1* **by** (*simp add*: *omega-isotone*)
also have ... $= (y^\star * L)^\omega \sqcup (y^\star * L)^\star * y^\omega$
by (*simp add*: *omega-decompose*)
also have ... $= y^\star * L * (y^\star * L)^\omega \sqcup (y^\star * L)^\star * y^\omega$
using *omega-unfold* **by** *auto*
also have ... $\le y^\star * L \sqcup (y^\star * L)^\star * y^\omega$
using *mult-L-omega omega-unfold sup-left-isotone* **by** *auto*
also have ... $= y^\star * L \sqcup (1 \sqcup y^\star * L * (y^\star * L)^\star) * y^\omega$
by (*simp add*: *star-left-unfold-equal*)
also have ... $\le y^\star * L \sqcup y^\omega$
by (*simp add*: *dL-star.mult-L-circ-mult-below star-left-unfold-equal sup-commute*)
also have ... $\le y^\star * bot \sqcup L \sqcup y^\omega$
by (*simp add*: *l14 le-supI1*)
finally have $x^\omega \le y^\omega \sqcup L$
using *star-bot-below-omega sup.left-commute sup.order-iff sup-commute* **by** *auto*
thus *?thesis*
using *2* **by** (*simp add*: *apx-def*)
qed

lemma *combined-apx-isotone*:
$x \sqsubseteq y \Longrightarrow (x^\omega \sqcap L) \sqcup x^\star * z \sqsubseteq (y^\omega \sqcap L) \sqcup y^\star * z$
using *meet-L-apx-isotone mult-apx-left-isotone star.circ-apx-isotone sup-apx-isotone omega-apx-isotone* **by** *auto*

lemma *d-split-nu-mu*:
$d(L) * (y^\omega \sqcup y^\star * z) \leq y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L) \sqcup d((y^\omega \sqcup y^\star * z) * bot) * top$
proof −
have $d(L) * y^\omega \leq (y^\omega \sqcap L) \sqcup d(y^\omega * bot) * top$
using *l31 l91 omega-vector sup-right-isotone* **by** *auto*
hence $d(L) * (y^\omega \sqcup y^\star * z) \leq y^\star * z \sqcup (y^\omega \sqcap L) \sqcup d(y^\omega * bot) * top$
by (*smt sup-assoc sup-commute sup-mono d-mult-below mult-left-dist-sup*)
also have ... $\leq y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L) \sqcup d(y^\omega * bot) * top$
by (*simp add*: *le-supI1 le-supI2*)
also have ... $\leq y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L) \sqcup d((y^\omega \sqcup y^\star * z) * bot) * top$
by (*meson d-isotone mult-left-isotone sup.cobounded1 sup-right-isotone*)
finally show *?thesis*
.
qed

lemma *loop-exists*:
$d(L) * \nu\ (\lambda x\ .\ y * x \sqcup z) \leq \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L) \sqcup d(\nu\ (\lambda x\ .\ y * x \sqcup z) * bot) * top$
by (*simp add*: *d-split-nu-mu omega-loop-nu star-loop-mu*)

lemma *loop-apx-least-fixpoint*:
apx.is-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)\ (\mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L))$
using *apx.least-fixpoint-char affine-apx-isotone loop-exists affine-has-greatest-fixpoint affine-has-least-fixpoint affine-isotone nu-below-mu-nu-def nu-below-mu-nu-kappa-mu-nu kappa-mu-nu-def* **by** *auto*

lemma *loop-has-apx-least-fixpoint*:
apx.has-least-fixpoint $(\lambda x\ .\ y * x \sqcup z)$
by (*metis apx.has-least-fixpoint-def loop-apx-least-fixpoint*)

lemma *loop-semantics*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = \mu\ (\lambda x\ .\ y * x \sqcup z) \sqcup (\nu\ (\lambda x\ .\ y * x \sqcup z) \sqcap L)$
using *apx.least-fixpoint-char loop-apx-least-fixpoint* **by** *auto*

lemma *loop-semantics-kappa-mu-nu*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = (y^\omega \sqcap L) \sqcup y^\star * z$
proof −
have $\kappa\ (\lambda x\ .\ y * x \sqcup z) = y^\star * z \sqcup ((y^\omega \sqcup y^\star * z) \sqcap L)$
by (*metis loop-semantics omega-loop-nu star-loop-mu*)
thus *?thesis*
by (*metis sup.absorb2 sup-commute sup-ge2 sup-inf-distrib1*)
qed

lemma *loop-semantics-kappa-mu-nu-domain*:
$\kappa\ (\lambda x\ .\ y * x \sqcup z) = d(y^\omega) * L \sqcup y^\star * z$
by (*simp add*: *omega-meet-L loop-semantics-kappa-mu-nu*)

lemma *loop-semantics-apx-isotone*:
$w \sqsubseteq y \Longrightarrow \kappa\ (\lambda x\ .\ w * x \sqcup z) \sqsubseteq \kappa\ (\lambda x\ .\ y * x \sqcup z)$
by (*metis loop-semantics-kappa-mu-nu combined-apx-isotone*)

end

end

25 Extended Designs

theory *Extended-Designs*

imports *Omega-Algebras Domain*

begin

class *domain-semiring-L-below* = *left-zero-domain-semiring* + *L* +
assumes *L-left-zero-below*: $L * x \leq L$
assumes *mult-L-split*: $x * L = x * bot \sqcup d(x) * L$
begin

lemma *d-zero-mult-L*:
$d(x * bot) * L \leq x$
by (*metis le-sup-iff mult-L-split mult-assoc mult-left-zero zero-right-mult-decreasing*)

lemma *mult-L*:
$x * L \leq x * bot \sqcup L$
by (*metis sup-right-isotone d-mult-below mult-L-split*)

lemma *d-mult-L*:
$d(x) * L \leq x * L$
by (*metis sup-right-divisibility mult-L-split*)

lemma *d-L-split*:
$x * d(y) * L = x * bot \sqcup d(x * y) * L$
by (*metis d-commutative d-mult-d d-zero mult-L-split mult-assoc mult-left-zero*)

lemma *d-mult-mult-L*:
$d(x * y) * L \leq x * d(y) * L$
using *d-L-split* **by** *auto*

lemma *L-L*:
$L * L = L$
by (*metis d-restrict-equals le-iff-sup mult-L-split zero-right-mult-decreasing*)

end

class *antidomain-semiring-L* = *left-zero-antidomain-semiring* + *L* +
  assumes *d-zero-mult-L*: $d(x * bot) * L \leq x$
  assumes *d-L-zero* : $d(L * bot) = 1$
  assumes *mult-L* : $x * L \leq x * bot \sqcup L$
begin

lemma *L-left-zero*:
  $L * x = L$
  by (*metis order.antisym d-L-zero d-zero-mult-L mult-assoc mult-left-one mult-left-zero zero-right-mult-decreasing*)

subclass *domain-semiring-L-below*
  apply *unfold-locales*
  apply (*simp add*: *L-left-zero*)
  apply (*rule order.antisym*)
  apply (*smt d-restrict-equals le-iff-sup mult-L mult-assoc mult-left-dist-sup*)
  by (*metis le-sup-iff d-L-zero d-mult-d d-zero-mult-L mult-assoc mult-right-isotone mult-1-right bot-least*)

end

class *ed-below* = *bounded-left-zero-omega-algebra* + *domain-semiring-L-below* + *Omega* +
  assumes *Omega-def*: $x^\Omega = d(x^\omega) * L \sqcup x^\star$
begin

lemma *Omega-isotone*:
  $x \leq y \Longrightarrow x^\Omega \leq y^\Omega$
  by (*metis Omega-def sup-mono d-isotone mult-left-isotone omega-isotone star.circ-isotone*)

lemma *star-below-Omega*:
  $x^\star \leq x^\Omega$
  using *Omega-def* **by** *auto*

lemma *one-below-Omega*:
  $1 \leq x^\Omega$
  using *order-trans star.circ-reflexive star-below-Omega* **by** *blast*

lemma *L-left-zero-star*:
  $L * x^\star = L$
  by (*meson L-left-zero-below order.antisym star.circ-back-loop-prefixpoint sup.boundedE*)

lemma *L-left-zero-Omega*:
  $L * x^\Omega = L$
  using *L-left-zero-star L-left-zero-below Omega-def mult-left-dist-sup sup.order-iff sup-monoid.add-commute* **by** *auto*

lemma *mult-L-star*:
$(x * L)^\star = 1 \sqcup x * L$
by (*metis L-left-zero-star mult-assoc star.circ-left-unfold*)

lemma *mult-L-omega-below*:
$(x * L)^\omega \leq x * L$
by (*metis L-left-zero-below mult-right-isotone omega-slide*)

lemma *mult-L-sup-star*:
$(x * L \sqcup y)^\star = y^\star \sqcup y^\star * x * L$
by (*metis L-left-zero-star sup-commute mult-assoc star.circ-unfold-sum*)

lemma *mult-L-sup-omega-below*:
$(x * L \sqcup y)^\omega \leq y^\omega \sqcup y^\star * x * L$
proof −
have $(x * L \sqcup y)^\omega = (y^\star * x * L)^\omega \sqcup (y^\star * x * L)^\star * y^\omega$
by (*simp add*: *ils.il-inf-associative omega-decompose sup-commute*)
also have ... $\leq y^\star * x * L \sqcup (y^\star * x * L)^\star * y^\omega$
using *sup-left-isotone mult-L-omega-below* **by** *auto*
also have ... $= y^\star * x * L \sqcup y^\star * x * L * y^\omega \sqcup y^\omega$
by (*smt L-left-zero-star sup-assoc sup-commute mult-assoc star.circ-loop-fixpoint*)
also have ... $\leq y^\omega \sqcup y^\star * x * L$
by (*metis L-left-zero-star sup-commute eq-refl mult-assoc star.circ-back-loop-fixpoint*)
finally show *?thesis*
.
qed

lemma *mult-L-sup-circ*:
$(x * L \sqcup y)^\Omega = d(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L$
proof −
have $(x * L \sqcup y)^\Omega = d((x * L \sqcup y)^\omega) * L \sqcup (x * L \sqcup y)^\star$
by (*simp add*: *Omega-def*)
also have ... $\leq d(y^\omega \sqcup y^\star * x * L) * L \sqcup (x * L \sqcup y)^\star$
by (*metis sup-left-isotone d-isotone mult-L-sup-omega-below mult-left-isotone*)
also have ... $= d(y^\omega) * L \sqcup d(y^\star * x * L) * L \sqcup (x * L \sqcup y)^\star$
by (*simp add*: *d-dist-sup mult-right-dist-sup*)
also have ... $\leq d(y^\omega) * L \sqcup y^\star * x * L * L \sqcup (x * L \sqcup y)^\star$
by (*meson d-mult-L order.refl sup.mono*)
also have ... $= d(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L$
by (*smt L-L sup-assoc sup-commute le-iff-sup mult-L-sup-star mult-assoc order-refl*)
finally have *1*: $(x * L \sqcup y)^\Omega \leq d(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L$
.
have *2*: $d(y^\omega) * L \leq (x * L \sqcup y)^\Omega$
using *Omega-isotone Omega-def* **by** *force*
have $y^\star \sqcup y^\star * x * L \leq (x * L \sqcup y)^\Omega$
by (*metis Omega-def sup-ge2 mult-L-sup-star*)

hence $d(y^\omega) * L \sqcup y^\star \sqcup y^\star * x * L \le (x * L \sqcup y)^\Omega$
using *2* **by** *simp*
thus *?thesis*
using *1* **by** (*simp add*: *order.antisym*)
qed

lemma *circ-sup-d*:
$(x^\Omega * y)^\Omega * x^\Omega = d((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * x^\star \sqcup (x^\star * y)^\star * d(x^\omega) * L)$
proof −
have $(x^\Omega * y)^\Omega * x^\Omega = ((d(x^\omega) * L \sqcup x^\star) * y)^\Omega * x^\Omega$
by (*simp add*: *Omega-def*)
also have ... $= (d(x^\omega) * L * y \sqcup x^\star * y)^\Omega * x^\Omega$
by (*simp add*: *mult-right-dist-sup*)
also have ... $\le (d(x^\omega) * L \sqcup x^\star * y)^\Omega * x^\Omega$
by (*metis L-left-zero-below Omega-isotone sup-left-isotone mult-assoc mult-left-isotone mult-right-isotone*)
also have ... $= (d((x^\star * y)^\omega) * L \sqcup (x^\star * y)^\star \sqcup (x^\star * y)^\star * d(x^\omega) * L) * x^\Omega$
by (*simp add*: *mult-L-sup-circ*)
also have ... $= d((x^\star * y)^\omega) * L * x^\Omega \sqcup (x^\star * y)^\star * x^\Omega \sqcup (x^\star * y)^\star * d(x^\omega) * L * x^\Omega$
using *mult-right-dist-sup* **by** *auto*
also have ... $= d((x^\star * y)^\omega) * L \sqcup (x^\star * y)^\star * x^\Omega \sqcup (x^\star * y)^\star * d(x^\omega) * L$
by (*simp add*: *L-left-zero-Omega mult.assoc*)
also have ... $= d((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * x^\star \sqcup (x^\star * y)^\star * d(x^\omega) * L)$
by (*simp add*: *Omega-def ils.il-inf-associative semiring.distrib-left sup-left-commute sup-monoid.add-commute*)
finally have *1*: $(x^\Omega * y)^\Omega * x^\Omega \le d((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * x^\star \sqcup (x^\star * y)^\star * d(x^\omega) * L)$
.
have $d((x^\star * y)^\omega) * L \le (x^\Omega * y)^\Omega$
using *Omega-isotone Omega-def mult-left-isotone* **by** *auto*
also have ... $\le (x^\Omega * y)^\Omega * x^\Omega$
by (*metis mult-right-isotone mult-1-right one-below-Omega*)
finally have *2*: $d((x^\star * y)^\omega) * L \le (x^\Omega * y)^\Omega * x^\Omega$
.
have *3*: $(x^\star * y)^\star * x^\star \le (x^\Omega * y)^\Omega * x^\Omega$
by (*meson Omega-isotone order.trans mult-left-isotone mult-right-isotone star-below-Omega*)
have $(x^\star * y)^\star * d(x^\omega) * L \le (x^\star * y)^\star * x^\Omega$
by (*metis Omega-def sup-commute mult-assoc mult-left-sub-dist-sup-right*)
also have ... $\le (x^\Omega * y)^\Omega * x^\Omega$
using *Omega-isotone Omega-def mult-left-isotone* **by** *force*
finally have $d((x^\star * y)^\omega) * L \sqcup ((x^\star * y)^\star * x^\star \sqcup (x^\star * y)^\star * d(x^\omega) * L) \le (x^\Omega * y)^\Omega * x^\Omega$
using *2 3* **by** (*simp add*: *sup-assoc*)
thus *?thesis*
using *1* **by** (*simp add*: *order.antisym*)
qed

proposition *mult-L-omega*: $(x * L)^{\omega} = x * L$ **nitpick** [*expect=genuine,card=5*]
oops
proposition *mult-L-sup-omega*: $(x * L \sqcup y)^{\omega} = y^{\omega} \sqcup y^{\star} * x * L$ **nitpick** [*expect=genuine,card=5*] **oops**
proposition *d-Omega-circ-simulate-right-plus*: $z * x \leq y * y^{\Omega} * z \sqcup w \Longrightarrow z * x^{\Omega} \leq y^{\Omega} * (z \sqcup w * x^{\Omega})$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *d-Omega-circ-simulate-left-plus*: $x * z \leq z * y^{\Omega} \sqcup w \Longrightarrow x^{\Omega} * z \leq (z \sqcup x^{\Omega} * w) * y^{\Omega}$ **nitpick** [*expect=genuine,card=3*] **oops**

end

class *ed* = *ed-below* +
 assumes *L-left-zero*: $L * x = L$
begin

lemma *mult-L-omega*:
 $(x * L)^{\omega} = x * L$
 by (*metis L-left-zero omega-slide*)

lemma *mult-L-sup-omega*:
 $(x * L \sqcup y)^{\omega} = y^{\omega} \sqcup y^{\star} * x * L$
 by (*metis L-left-zero ils.il-inf-associative mult-bot-add-omega sup-commute*)

lemma *d-Omega-circ-simulate-right-plus*:
 assumes $z * x \leq y * y^{\Omega} * z \sqcup w$
  shows $z * x^{\Omega} \leq y^{\Omega} * (z \sqcup w * x^{\Omega})$
proof –
 have $z * x \leq y * d(y^{\omega}) * L * z \sqcup y * y^{\star} * z \sqcup w$
  using *assms Omega-def ils.il-inf-associative mult-right-dist-sup semiring.distrib-left* **by** *auto*
 also have ... $\leq y * d(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
  by (*metis L-left-zero-below sup-commute sup-right-isotone mult-assoc mult-right-isotone*)
 also have ... $= y * bot \sqcup d(y * y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
  by (*simp add*: *d-L-split*)
 also have ... $= d(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
  by (*smt sup-assoc sup-commute sup-bot-left mult-assoc mult-left-dist-sup omega-unfold*)
 finally have *1*: $z * x \leq d(y^{\omega}) * L \sqcup y * y^{\star} * z \sqcup w$
  .
 have $(d(y^{\omega}) * L \sqcup y^{\star} * z \sqcup y^{\star} * w * d(x^{\omega}) * L \sqcup y^{\star} * w * x^{\star}) * x = d(y^{\omega}) * L * x \sqcup y^{\star} * z * x \sqcup y^{\star} * w * d(x^{\omega}) * L * x \sqcup y^{\star} * w * x^{\star} * x$
  using *mult-right-dist-sup* **by** *fastforce*
 also have ... $\leq d(y^{\omega}) * L \sqcup y^{\star} * z * x \sqcup y^{\star} * w * d(x^{\omega}) * L * x \sqcup y^{\star} * w * x^{\star} * x$
  by (*metis L-left-zero-below sup-left-isotone mult-assoc mult-right-isotone*)
 also have ... $\leq d(y^{\omega}) * L \sqcup y^{\star} * z * x \sqcup y^{\star} * w * d(x^{\omega}) * L \sqcup y^{\star} * w * x^{\star} * x$
  by (*metis L-left-zero-below sup-commute sup-left-isotone mult-assoc mult-right-isotone*)

also have ... $\leq d(y^\omega) * L \sqcup y^\star * z * x \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*meson star.circ-back-loop-prefixpoint sup.boundedE sup-right-isotone*)
also have ... $\leq d(y^\omega) * L \sqcup y^\star * (d(y^\omega) * L \sqcup y * y^\star * z \sqcup w) \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
using *1* **by** (*smt sup-left-isotone sup-right-isotone le-iff-sup mult-assoc mult-left-dist-sup*)
also have ... $= d(y^\omega) * L \sqcup y^\star * y * y^\star * z \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*smt sup-assoc sup-commute sup-idem mult-assoc mult-left-dist-sup d-L-split star.circ-back-loop-fixpoint star-mult-omega*)
also have ... $\leq d(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
using *mult-isotone order-refl semiring.add-right-mono star.circ-mult-upper-bound star.right-plus-below-circ sup-right-isotone* **by** *auto*
finally have *2*: $z * x^\star \leq d(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*smt le-sup-iff sup-ge1 star.circ-loop-fixpoint star-right-induct*)
have $z * x * x^\omega \leq y * y^\star * z * x^\omega \sqcup d(y^\omega) * L * x^\omega \sqcup w * x^\omega$
using *1* **by** (*metis sup-commute mult-left-isotone mult-right-dist-sup*)
also have ... $\leq y * y^\star * z * x^\omega \sqcup d(y^\omega) * L \sqcup w * x^\omega$
by (*metis L-left-zero eq-refl ils.il-inf-associative*)
finally have $z * x^\omega \leq y^\omega \sqcup y^\star * d(y^\omega) * L \sqcup y^\star * w * x^\omega$
by (*smt sup-assoc sup-commute left-plus-omega mult-assoc mult-left-dist-sup omega-induct omega-unfold star.left-plus-circ*)
hence $z * x^\omega \leq y^\omega \sqcup y^\star * w * x^\omega$
by (*metis sup-commute d-mult-L le-iff-sup mult-assoc mult-right-isotone omega-sub-vector order-trans star-mult-omega*)
hence $d(z * x^\omega) * L \leq d(y^\omega) * L \sqcup y^\star * w * d(x^\omega) * L$
by (*smt sup-assoc sup-commute d-L-split d-dist-sup le-iff-sup mult-right-dist-sup*)
hence $z * d(x^\omega) * L \leq z * bot \sqcup d(y^\omega) * L \sqcup y^\star * w * d(x^\omega) * L$
using *d-L-split sup-assoc sup-right-isotone* **by** *force*
also have ... $\leq y^\star * z \sqcup d(y^\omega) * L \sqcup y^\star * w * d(x^\omega) * L$
by (*smt sup-commute sup-left-isotone sup-ge1 order-trans star.circ-loop-fixpoint zero-right-mult-decreasing*)
finally have $z * d(x^\omega) * L \leq d(y^\omega) * L \sqcup y^\star * z \sqcup y^\star * w * d(x^\omega) * L \sqcup y^\star * w * x^\star$
by (*simp add*: *le-supI2 sup-commute*)
thus *?thesis*
using *2* **by** (*smt L-left-zero Omega-def sup-assoc le-iff-sup mult-assoc mult-left-dist-sup mult-right-dist-sup*)
qed

lemma *d-Omega-circ-simulate-left-plus*:
assumes $x * z \leq z * y^\Omega \sqcup w$
shows $x^\Omega * z \leq (z \sqcup x^\Omega * w) * y^\Omega$
proof –
have $x * (z * d(y^\omega) * L \sqcup z * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star) = x * z * d(y^\omega) * L \sqcup x * z * y^\star \sqcup d(x^\omega) * L \sqcup x * x^\star * w * d(y^\omega) * L \sqcup x * x^\star * w * y^\star$
by (*smt sup-assoc sup-commute mult-assoc mult-left-dist-sup d-L-split*

omega-unfold)
also have ... $\leq (z * d(y^\omega) * L \sqcup z * y^\star \sqcup w) * d(y^\omega) * L \sqcup (z * d(y^\omega) * L \sqcup z * y^\star \sqcup w) * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt assms Omega-def sup-assoc sup-ge2 le-iff-sup mult-assoc mult-left-dist-sup mult-right-dist-sup star.circ-loop-fixpoint*)
also have ... $= z * d(y^\omega) * L \sqcup z * y^\star * d(y^\omega) * L \sqcup w * d(y^\omega) * L \sqcup z * y^\star \sqcup w * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt L-left-zero sup-assoc sup-commute sup-idem mult-assoc mult-right-dist-sup star.circ-transitive-equal*)
also have ... $= z * d(y^\omega) * L \sqcup w * d(y^\omega) * L \sqcup z * y^\star \sqcup w * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt sup-assoc sup-commute sup-idem le-iff-sup mult-assoc d-L-split star-mult-omega zero-right-mult-decreasing*)
finally have $x * (z * d(y^\omega) * L \sqcup z * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star) \leq z * d(y^\omega) * L \sqcup z * y^\star \sqcup d(x^\omega) * L \sqcup x^\star * w * d(y^\omega) * L \sqcup x^\star * w * y^\star$
by (*smt sup-assoc sup-commute sup-idem mult-assoc star.circ-loop-fixpoint*)
thus *?thesis*
by (*smt* (*verit, del-insts*) *L-left-zero Omega-def sup-assoc le-sup-iff sup-ge1 mult-assoc mult-left-dist-sup mult-right-dist-sup star.circ-back-loop-fixpoint star-left-induct*)
qed

end

Theorem 2.5 and Theorem 50.4

sublocale *ed* < *ed-omega*: *itering* **where** *circ* = *Omega*
apply *unfold-locales*
apply (*smt sup-assoc sup-commute sup-bot-left circ-sup-d Omega-def mult-left-dist-sup mult-right-dist-sup d-L-split d-dist-sup omega-decompose star.circ-sup-1 star.circ-slide*)
apply (*smt L-left-zero sup-assoc sup-commute sup-bot-left Omega-def mult-assoc mult-left-dist-sup mult-right-dist-sup d-L-split omega-slide star.circ-mult*)
using *d-Omega-circ-simulate-right-plus* **apply** *blast*
by (*simp add*: *d-Omega-circ-simulate-left-plus*)

sublocale *ed* < *ed-star*: *itering* **where** *circ* = *star* **..**

class *ed-2* = *ed-below* + *antidomain-semiring-L* + *Omega*
begin

subclass *ed*
apply *unfold-locales*
by (*rule L-left-zero*)

end

end

26 Relative Domain

theory *Relative-Domain*

imports *Tests*

begin

class Z =
  fixes $Z :: 'a$

class *relative-domain-semiring* = *idempotent-left-semiring* + *dom* + Z +
  assumes *d-restrict* : $x \le d(x) * x \sqcup Z$
  assumes *d-mult-d* : $d(x * y) = d(x * d(y))$
  assumes *d-below-one*: $d(x) \le 1$
  assumes *d-Z* : $d(Z) = bot$
  assumes *d-dist-sup* : $d(x \sqcup y) = d(x) \sqcup d(y)$
  assumes *d-export* : $d(d(x) * y) = d(x) * d(y)$
begin

lemma *d-plus-one*:
  $d(x) \sqcup 1 = 1$
  by (*simp add*: *d-below-one sup-absorb2*)

Theorem 44.2

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

Theorem 44.3

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

lemma *d-fixpoint*:
  $(\exists y . x = d(y)) \longleftrightarrow x = d(x)$
  using *d-idempotent* **by** *auto*

lemma *d-type*:
  $\forall P . (\forall x . x = d(x) \longrightarrow P(x)) \longleftrightarrow (\forall x . P(d(x)))$
  by (*metis d-idempotent*)

Theorem 44.4

lemma *d-mult-sub*:
  $d(x * y) \le d(x)$
  by (*smt* (*verit*, *ccfv-threshold*) *d-plus-one d-dist-sup d-mult-d le-iff-sup mult.right-neutral mult-left-sub-dist-sup-right sup-commute*)

lemma *d-sub-one*:

$x \leq 1 \Longrightarrow x \leq d(x) \sqcup Z$
by (*metis sup-left-isotone d-restrict mult-right-isotone mult-1-right order-trans*)

lemma *d-one*:
$d(1) \sqcup Z = 1 \sqcup Z$
by (*meson d-sub-one d-below-one order.trans preorder-one-closed sup.cobounded1 sup-same-context*)

Theorem 44.8

lemma *d-strict*:
$d(x) = bot \longleftrightarrow x \leq Z$
by (*metis sup-commute sup-bot-right d-Z d-dist-sup d-restrict le-iff-sup mult-left-zero*)

Theorem 44.1

lemma *d-isotone*:
$x \leq y \Longrightarrow d(x) \leq d(y)$
using *d-dist-sup sup-right-divisibility* **by** *force*

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

lemma *d-mult-idempotent*:
$d(x) * d(x) = d(x)$
by (*smt* (*verit*, *ccfv-threshold*) *d-idempotent d-mult-sub d-Z d-dist-sup d-export d-restrict le-iff-sup sup-bot-left sup-commute*)

Theorem 44.12

lemma *d-least-left-preserver*:
$x \leq d(y) * x \sqcup Z \longleftrightarrow d(x) \leq d(y)$
apply (*rule iffI*)
apply (*smt* (*z3*) *comm-monoid.comm-neutral d-idempotent d-mult-sub d-plus-left-upper-bound d-Z d-dist-sup order-trans sup-absorb2 sup-bot.comm-monoid-axioms*)
by (*smt* (*verit*, *del-insts*) *d-restrict mult-right-dist-sup sup.cobounded1 sup.orderE sup-assoc sup-commute*)

Theorem 44.9

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

lemma *d-sup-closed*:
$d(d(x) \sqcup d(y)) = d(x) \sqcup d(y)$
by (*simp add*: *d-idempotent d-dist-sup*)

lemma *d-mult-closed*:
$d(d(x) * d(y)) = d(x) * d(y)$

using *d-export d-mult-d* **by** *auto*

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

lemma *d-mult-left-absorb-sup*:
$d(x) * (d(x) \sqcup d(y)) = d(x)$
by (*smt d-sup-closed d-export d-mult-idempotent d-idempotent d-mult-sub order.eq-iff mult-left-sub-dist-sup-left*)

lemma *d-sup-left-absorb-mult*:
$d(x) \sqcup d(x) * d(y) = d(x)$
using *d-mult-left-lower-bound sup.absorb-iff1* **by** *auto*

lemma *d-commutative*:
$d(x) * d(y) = d(y) * d(x)$
by (*metis sup-commute order.antisym d-sup-left-absorb-mult d-below-one d-export d-mult-left-absorb-sup mult-assoc mult-left-isotone mult-left-one*)

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

lemma *d-sup-left-dist-mult*:
$d(x) \sqcup d(y) * d(z) = (d(x) \sqcup d(y)) * (d(x) \sqcup d(z))$
by (*metis sup-assoc d-commutative d-dist-sup d-mult-idempotent d-mult-left-absorb-sup mult-right-dist-sup*)

lemma *d-order*:
$d(x) \leq d(y) \longleftrightarrow d(x) = d(x) * d(y)$
by (*metis d-mult-greatest-lower-bound d-mult-left-absorb-sup le-iff-sup order-refl*)

Theorem 44.6

lemma *Z-mult-decreasing*:
$Z * x \leq Z$
by (*metis d-mult-sub bot.extremum d-strict order.eq-iff*)

Theorem 44.5

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

Theorem 44.7

lemma *d-relative-Z*:
$d(x) * x \sqcup Z = x \sqcup Z$
by (*metis sup-ge1 sup-same-context d-below-one d-restrict mult-isotone mult-left-one*)

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

Theorem 44.11

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

lemma *kat-4-equiv*:
$d(x) * y = d(x) * y * d(z) \longleftrightarrow d(x) * y \leq y * d(z)$
apply (*rule iffI*)
apply (*simp add*: *kat-4*)
apply (*rule order.antisym*)
apply (*metis d-mult-idempotent mult-assoc mult-right-isotone*)
by (*metis d-below-one mult-right-isotone mult-1-right*)

lemma *kat-4-equiv-opp*:
$y * d(x) = d(z) * y * d(x) \longleftrightarrow y * d(x) \leq d(z) * y$
apply (*rule iffI*)
using *d-below-one mult-right-isotone* **apply** *fastforce*
apply (*rule order.antisym*)
apply (*metis d-mult-idempotent mult-assoc mult-left-isotone*)
by (*metis d-below-one mult-left-isotone mult-left-one*)

Theorem 44.10

lemma *d-restrict-iff-1*:
$d(x) * y \leq z \longleftrightarrow d(x) * y \leq d(x) * z$
by (*smt* (*verit*, *del-insts*) *d-below-one d-mult-idempotent mult-assoc mult-left-isotone mult-left-one mult-right-isotone order-trans*)

proposition *d-restrict* : $x \leq d(x) * x \sqcup Z$ **oops**
proposition *d-mult-d* : $d(x * y) = d(x * d(y))$ **oops**
proposition *d-below-one*: $d(x) \leq 1$ **oops**
proposition *d-Z* : $d(Z) = bot$ **oops**
proposition *d-dist-sup* : $d(x \sqcup y) = d(x) \sqcup d(y)$ **oops**
proposition *d-export* : $d(d(x) * y) = d(x) * d(y)$ **oops**

end

typedef (**overloaded**) *'a dImage* = { x::*'a*::*relative-domain-semiring* . ($\exists\, y$::*'a* . $x = d(y)$) }
by *auto*

lemma *simp-dImage*[*simp*]:
$\exists\, y$. *Rep-dImage* $x = d(y)$
using *Rep-dImage* **by** *simp*

setup-lifting *type-definition-dImage*

Theorem 44

instantiation *dImage* :: (*relative-domain-semiring*) *bounded-distrib-lattice*
begin

lift-definition *sup-dImage* :: *'a dImage* ⇒ *'a dImage* ⇒ *'a dImage* **is** *sup*
by (*metis d-dist-sup*)

lift-definition *inf-dImage* :: *'a dImage* ⇒ *'a dImage* ⇒ *'a dImage* **is** *times*
by (*metis d-export*)

lift-definition *bot-dImage* :: *'a dImage* **is** *bot*
by (*metis d-zero*)

lift-definition *top-dImage* :: *'a dImage* **is** *d*(*1*)
by *auto*

lift-definition *less-eq-dImage* :: *'a dImage* ⇒ *'a dImage* ⇒ *bool* **is** *less-eq* **.**

lift-definition *less-dImage* :: *'a dImage* ⇒ *'a dImage* ⇒ *bool* **is** *less* **.**

instance
apply *intro-classes*
apply (*simp add*: *less-dImage.rep-eq less-eq-dImage.rep-eq less-le-not-le*)
apply (*simp add*: *less-eq-dImage.rep-eq*)
using *less-eq-dImage.rep-eq* **apply** *simp*
apply (*simp add*: *Rep-dImage-inject less-eq-dImage.rep-eq*)
apply (*metis* (*mono-tags*) *d-idempotent d-mult-sub inf-dImage.rep-eq less-eq-dImage.rep-eq simp-dImage*)
apply (*metis* (*mono-tags*) *d-mult-greatest-lower-bound inf-dImage.rep-eq less-eq-dImage.rep-eq order-refl simp-dImage*)
apply (*metis* (*mono-tags*) *d-mult-greatest-lower-bound inf-dImage.rep-eq less-eq-dImage.rep-eq simp-dImage*)
apply (*simp add*: *less-eq-dImage.rep-eq sup-dImage.rep-eq*)
apply (*simp add*: *less-eq-dImage.rep-eq sup-dImage.rep-eq*)
apply (*simp add*: *less-eq-dImage.rep-eq sup-dImage.rep-eq*)
apply (*simp add*: *bot-dImage.rep-eq less-eq-dImage.rep-eq*)
apply (*smt* (*z3*) *d-below-d-one less-eq-dImage.rep-eq simp-dImage top-dImage.rep-eq*)
by (*smt* (*z3*) *inf-dImage.rep-eq sup-dImage.rep-eq simp-dImage Rep-dImage-inject d-sup-left-dist-mult*)

end

class *bounded-relative-domain-semiring* = *relative-domain-semiring* + *bounded-idempotent-left-semiring*
begin

lemma *Z-top*:
$Z * top = Z$
by (*simp add*: *Z-left-zero-above-one*)

lemma *d-restrict-top*:
$x \le d(x) * top \sqcup Z$
by (*metis sup-left-isotone d-restrict mult-right-isotone order-trans top-greatest*)

proposition *d-one-one*: $d(1) = 1$ **nitpick** [*expect=genuine,card=2*] **oops**

end

class *relative-domain-semiring-split* = *relative-domain-semiring* +
assumes *split-Z*: $x * (y \sqcup Z) \le x * y \sqcup Z$
begin

lemma *d-restrict-iff*:
$(x \le y \sqcup Z) \longleftrightarrow (x \le d(x) * y \sqcup Z)$
proof −
have $x \le y \sqcup Z \longrightarrow x \le d(x) * (y \sqcup Z) \sqcup Z$
by (*smt sup-left-isotone d-restrict le-iff-sup mult-left-sub-dist-sup-left order-trans*)
hence $x \le y \sqcup Z \longrightarrow x \le d(x) * y \sqcup Z$
by (*meson le-supI order-lesseq-imp split-Z sup.cobounded2*)
thus *?thesis*
by (*meson d-restrict-iff-1 le-supI mult-left-sub-dist-sup-left order-lesseq-imp sup.cobounded2*)
qed

end

class *relative-antidomain-semiring* = *idempotent-left-semiring* + *dom* + *Z* + *uminus* +
assumes *a-restrict* : $-x * x \le Z$
assumes *a-mult-d* : $-(x * y) = -(x * --y)$
assumes *a-complement*: $-x * --x = bot$
assumes *a-Z* : $-Z = 1$
assumes *a-export* : $-(--x * y) = -x \sqcup -y$
assumes *a-dist-sup* : $-(x \sqcup y) = -x * -y$
assumes *d-def* : $d(x) = --x$
begin

notation
uminus (‹*a*›)

Theorem 45.7

lemma *a-complement-one*:
$--x \sqcup -x = 1$

by (*metis a-Z a-complement a-export a-mult-d mult-left-one*)

Theorem 45.5 and Theorem 45.6

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

lemma *a-below-one*:
$a(x) \leq 1$
using *a-complement-one sup-right-divisibility* **by** *auto*

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

lemma *a-sup-absorb*:
$(x \sqcup a(y)) * a(a(y)) = x * a(a(y))$
by (*simp add*: *a-complement mult-right-dist-sup*)

Theorem 45.10

lemma *a-greatest-left-absorber*:
$a(x) * y \leq Z \longleftrightarrow a(x) \leq a(y)$
apply (*rule iffI*)
apply (*smt a-Z a-sup-absorb a-dist-sup a-export-a a-mult-d sup-commute d-def le-iff-sup mult-left-one*)
by (*meson a-restrict mult-isotone order.refl order-trans*)

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

Theorem 45.2

subclass *relative-domain-semiring*
apply *unfold-locales*
apply (*smt* (*verit*) *a-Z a-complement-one a-restrict sup-commute sup-ge1 case-split-left d-def order-trans*)
using *a-mult-d d-def* **apply** *force*
apply (*simp add*: *a-below-one d-def*)
apply (*metis a-Z a-complement d-def mult-left-one*)
apply (*simp add*: *a-export-a a-dist-sup d-def*)
using *a-dist-sup a-export d-def* **by** *auto*

Theorem 45.1

subclass *tests*
apply *unfold-locales*
apply (*simp add*: *mult-assoc*)
apply (*metis a-dist-sup sup-commute*)
apply (*smt a-complement a-d-closed a-export-a sup-bot-right d-sup-left-dist-mult*)

apply (*metis a-d-closed a-dist-sup d-def*)
apply (*rule the-equality*[*THEN sym*])
apply (*simp add*: *a-complement*)
apply (*simp add*: *a-complement*)
using *a-d-closed a-Z d-Z d-def* **apply** *force*
using *a-export a-mult-d* **apply** *fastforce*
apply (*metis a-d-closed d-order*)
by (*simp add*: *less-le-not-le*)

lemma *a-plus-mult-d*:
$-(x * y) \sqcup -(x * --y) = -(x * --y)$
using *a-mult-d* **by** *auto*

lemma *a-mult-d-2*:
$a(x * y) = a(x * d(y))$
using *a-mult-d d-def* **by** *auto*

lemma *a-3*:
$a(x) * a(y) * d(x \sqcup y) = bot$
by (*metis a-complement a-dist-sup d-def*)

lemma *a-fixpoint*:
$\forall x . (a(x) = x \longrightarrow (\forall y . y = bot))$
by (*metis a-complement-one mult-1-left mult-left-zero order.refl sup.order-iff tests-dual.one-def*)

Theorem 45.9

lemma *a-strict*:
$a(x) = 1 \longleftrightarrow x \leq Z$
by (*metis a-Z d-def d-strict order.refl tests-dual.sba-dual.double-negation*)

lemma *d-complement-zero*:
$d(x) * a(x) = bot$
by (*simp add*: *d-def tests-dual.sub-commutative*)

lemma *a-complement-zero*:
$a(x) * d(x) = bot$
by (*simp add*: *d-def*)

lemma *a-shunting-zero*:
$a(x) * d(y) = bot \longleftrightarrow a(x) \leq a(y)$
by (*simp add*: *d-def tests-dual.sba-dual.less-eq-inf-bot*)

lemma *a-antitone*:
$x \leq y \Longrightarrow a(y) \leq a(x)$
using *a-plus-left-lower-bound sup-commute sup-right-divisibility* **by** *fastforce*

lemma *a-mult-deMorgan*:
$a(a(x) * a(y)) = d(x \sqcup y)$

by (*simp add*: *a-dist-sup d-def*)

lemma *a-mult-deMorgan-1*:
$a(a(x) * a(y)) = d(x) \sqcup d(y)$
by (*simp add*: *a-mult-deMorgan d-dist-sup*)

lemma *a-mult-deMorgan-2*:
$a(d(x) * d(y)) = a(x) \sqcup a(y)$
using *a-export d-def* **by** *auto*

lemma *a-plus-deMorgan*:
$a(a(x) \sqcup a(y)) = d(x) * d(y)$
by (*simp add*: *a-dist-sup d-def*)

lemma *a-plus-deMorgan-1*:
$a(d(x) \sqcup d(y)) = a(x) * a(y)$
by (*simp add*: *a-dist-sup d-def*)

Theorem 45.8

lemma *a-mult-left-upper-bound*:
$a(x) \leq a(x * y)$
using *a-shunting-zero d-def d-mult-sub tests-dual.less-eq-sup-top* **by** *auto*

Theorem 45.6

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

lemma *a-export-d*:
$a(d(x) * y) = a(x) \sqcup a(y)$
by (*simp add*: *a-export d-def*)

lemma *a-7*:
$d(x) * a(d(y) \sqcup d(z)) = d(x) * a(y) * a(z)$
by (*simp add*: *a-plus-deMorgan-1 mult-assoc*)

lemma *d-a-shunting*:
$d(x) * a(y) \leq d(z) \longleftrightarrow d(x) \leq d(z) \sqcup d(y)$
by (*simp add*: *d-def tests-dual.sba-dual.shunting-right*)

lemma *d-d-shunting*:
$d(x) * d(y) \leq d(z) \longleftrightarrow d(x) \leq d(z) \sqcup a(y)$
by (*simp add*: *d-def tests-dual.sba-dual.shunting-right*)

lemma *d-cancellation-1*:
$d(x) \leq d(y) \sqcup (d(x) * a(y))$
by (*smt* (*z3*) *a-d-closed d-a-shunting d-export eq-refl sup-commute*)

lemma *d-cancellation-2*:

$(d(z) \sqcup d(y)) * a(y) \leq d(z)$
by (*metis d-a-shunting d-dist-sup eq-refl*)

lemma *a-sup-closed*:
$d(a(x) \sqcup a(y)) = a(x) \sqcup a(y)$
using *a-mult-deMorgan tests-dual.sub-inf-def* **by** *auto*

lemma *a-mult-closed*:
$d(a(x) * a(y)) = a(x) * a(y)$
using *d-def tests-dual.sub-sup-closed* **by** *auto*

lemma *d-a-shunting-zero*:
$d(x) * a(y) = bot \longleftrightarrow d(x) \leq d(y)$
using *a-shunting-zero d-def* **by** *force*

lemma *d-d-shunting-zero*:
$d(x) * d(y) = bot \longleftrightarrow d(x) \leq a(y)$
using *d-a-shunting-zero d-def* **by** *auto*

lemma *d-compl-intro*:
$d(x) \sqcup d(y) = d(x) \sqcup a(x) * d(y)$
by (*simp add*: *d-def tests-dual.sba-dual.sup-complement-intro*)

lemma *a-compl-intro*:
$a(x) \sqcup a(y) = a(x) \sqcup d(x) * a(y)$
by (*simp add*: *d-def tests-dual.sba-dual.sup-complement-intro*)

lemma *kat-2*:
$y * a(z) \leq a(x) * y \Longrightarrow d(x) * y * a(z) = bot$
by (*metis d-complement-zero order.eq-iff mult-assoc mult-left-zero mult-right-isotone bot-least*)

Theorem 45.4

lemma *kat-2-equiv*:
$y * a(z) \leq a(x) * y \longleftrightarrow d(x) * y * a(z) = bot$
apply (*rule iffI*)
apply (*simp add*: *kat-2*)
by (*smt* (*verit, best*) *a-Z a-below-one a-complement-one case-split-left d-def mult-assoc mult-right-isotone mult-1-right bot-least*)

lemma *kat-3-equiv-opp*:
$a(z) * y * d(x) = bot \longleftrightarrow y * d(x) = d(z) * y * d(x)$
using *kat-2-equiv d-def kat-4-equiv-opp* **by** *auto*

Theorem 45.4

lemma *kat-3-equiv-opp-2*:
$d(z) * y * a(x) = bot \longleftrightarrow y * a(x) = a(z) * y * a(x)$
by (*metis a-d-closed kat-3-equiv-opp d-def*)

lemma *kat-equiv-6*:
$d(x) * y * a(z) = d(x) * y * bot \longleftrightarrow d(x) * y * a(z) \leq y * bot$
by (*metis d-restrict-iff-1 order.eq-iff mult-left-sub-dist-sup-right tests-dual.sba-dual.sup-right-unit mult-assoc*)

lemma *d-one-one*:
$d(1) = 1$
by (*simp add*: *d-def*)

lemma *case-split-left-sup*:
$-p * x \leq y \wedge --p * x \leq z \Longrightarrow x \leq y \sqcup z$
by (*smt* (*z3*) *a-complement-one case-split-left order-lesseq-imp sup.cobounded2 sup-ge1*)

lemma *test-mult-left-sub-dist-shunt*:
$-p * (--p * x \sqcup Z) \leq Z$
by (*simp add*: *a-greatest-left-absorber a-Z a-dist-sup a-export*)

lemma *test-mult-left-dist-shunt*:
$-p * (--p * x \sqcup Z) = -p * Z$
by (*smt* (*verit*, *ccfv-SIG*) *order.antisym mult-left-sub-dist-sup-right sup.orderE tests-dual.sba-dual.sup-idempotent mult-assoc test-mult-left-sub-dist-shunt tests-dual.sup-absorb*)

proposition *a-restrict* : $-x * x \leq Z$ **oops**
proposition *a-mult-d* : $-(x * y) = -(x * --y)$ **oops**
proposition *a-complement*: $-x * --x = bot$ **oops**
proposition *a-Z* : $-Z = 1$ **oops**
proposition *a-export* : $-(--x * y) = -x \sqcup -y$ **oops**
proposition *a-dist-sup* : $-(x \sqcup y) = -x * -y$ **oops**
proposition *d-def* : $d(x) = --x$ **oops**

end

typedef (**overloaded**) *'a aImage* = { *x*::*'a*::*relative-antidomain-semiring* . ($\exists$ *y*::*'a* . $x = a(y)$) }
by *auto*

lemma *simp-aImage*[*simp*]:
$\exists$ *y* . *Rep-aImage* $x = a(y)$
using *Rep-aImage* **by** *simp*

setup-lifting *type-definition-aImage*

Theorem 45.3

instantiation *aImage* :: (*relative-antidomain-semiring*) *boolean-algebra*
begin

lift-definition *sup-aImage* :: *'a aImage* ⇒ *'a aImage* ⇒ *'a aImage* **is** *sup*
using *tests-dual.sba-dual.sba-dual.inf-closed* **by** *auto*

lift-definition *inf-aImage* :: *'a aImage* ⇒ *'a aImage* ⇒ *'a aImage* **is** *times*
using *tests-dual.sba-dual.inf-closed* **by** *auto*

lift-definition *minus-aImage* :: *'a aImage* ⇒ *'a aImage* ⇒ *'a aImage* **is** $\lambda x\ y\ .\ x * a(y)$
using *tests-dual.sba-dual.inf-closed* **by** *blast*

lift-definition *uminus-aImage* :: *'a aImage* ⇒ *'a aImage* **is** *a*
by *auto*

lift-definition *bot-aImage* :: *'a aImage* **is** *bot*
by (*metis tests-dual.sba-dual.sba-dual.complement-bot*)

lift-definition *top-aImage* :: *'a aImage* **is** *1*
using *a-Z* **by** *auto*

lift-definition *less-eq-aImage* :: *'a aImage* ⇒ *'a aImage* ⇒ *bool* **is** *less-eq* **.**

lift-definition *less-aImage* :: *'a aImage* ⇒ *'a aImage* ⇒ *bool* **is** *less* **.**

instance
apply *intro-classes*
apply (*simp add*: *less-aImage.rep-eq less-eq-aImage.rep-eq less-le-not-le*)
apply (*simp add*: *less-eq-aImage.rep-eq*)
using *less-eq-aImage.rep-eq* **apply** *simp*
apply (*simp add*: *Rep-aImage-inject less-eq-aImage.rep-eq*)
apply (*metis* (*mono-tags*) *a-below-one inf-aImage.rep-eq less-eq-aImage.rep-eq mult.right-neutral mult-right-isotone simp-aImage*)
apply (*metis* (*mono-tags*, *lifting*) *less-eq-aImage.rep-eq a-d-closed a-export bot.extremum-unique inf-aImage.rep-eq kat-equiv-6 mult.assoc mult.left-neutral mult-left-isotone mult-left-zero simp-aImage sup.cobounded1 tests-dual.sba-dual.sba-dual.complement-top*)
apply (*smt* (*z3*) *less-eq-aImage.rep-eq inf-aImage.rep-eq mult-isotone simp-aImage tests-dual.sba-dual.inf-idempotent*)
apply (*simp add*: *less-eq-aImage.rep-eq sup-aImage.rep-eq*)
apply (*simp add*: *less-eq-aImage.rep-eq sup-aImage.rep-eq*)
using *less-eq-aImage.rep-eq sup-aImage.rep-eq* **apply** *force*
apply (*simp add*: *less-eq-aImage.rep-eq bot-aImage.rep-eq*)
apply (*smt* (*z3*) *less-eq-aImage.rep-eq a-below-one simp-aImage top-aImage.rep-eq*)
apply (*metis* (*mono-tags*, *lifting*) *tests-dual.sba-dual.sba-dual.inf-left-dist-sup Rep-aImage-inject inf-aImage.rep-eq sup-aImage.rep-eq simp-aImage*)
apply (*smt* (*z3*) *inf-aImage.rep-eq uminus-aImage.rep-eq Rep-aImage-inject a-complement bot-aImage.rep-eq simp-aImage*)
apply (*smt* (*z3*) *top-aImage.rep-eq Rep-aImage-inject a-complement-one simp-aImage sup-aImage.rep-eq sup-commute uminus-aImage.rep-eq*)

by (*metis* (*mono-tags*) *inf-aImage.rep-eq Rep-aImage-inject minus-aImage.rep-eq uminus-aImage.rep-eq*)

end

class *bounded-relative-antidomain-semiring* = *relative-antidomain-semiring* + *bounded-idempotent-left-semiring*
begin

subclass *bounded-relative-domain-semiring* **..**

lemma *a-top*:
$a(top) = bot$
by (*metis a-plus-left-lower-bound bot-unique sup-right-top tests-dual.sba-dual.complement-top*)

lemma *d-top*:
$d(top) = 1$
using *a-top d-def* **by** *auto*

lemma *shunting-top-1*:
$-p * x \leq y \Longrightarrow x \leq --p * top \sqcup y$
by (*metis sup-commute case-split-left-sup mult-right-isotone top-greatest*)

lemma *shunting-Z*:
$-p * x \leq Z \longleftrightarrow x \leq --p * top \sqcup Z$
apply (*rule iffI*)
apply (*simp add*: *shunting-top-1*)
by (*smt a-top a-Z a-antitone a-dist-sup a-export a-greatest-left-absorber sup-commute sup-bot-right mult-left-one*)

proposition *a-left-dist-sup*: $-p * (y \sqcup z) = -p * y \sqcup -p * z$ **nitpick** [*expect*=*genuine*,*card*=*7*] **oops**
proposition *shunting-top*: $-p * x \leq y \longleftrightarrow x \leq --p * top \sqcup y$ **nitpick** [*expect*=*genuine*,*card*=*7*] **oops**

end

class *relative-left-zero-antidomain-semiring* = *relative-antidomain-semiring* + *idempotent-left-zero-semiring*
begin

lemma *kat-3*:
$d(x) * y * a(z) = bot \Longrightarrow d(x) * y = d(x) * y * d(z)$
by (*metis d-def mult-1-right mult-left-dist-sup sup-monoid.add-0-left tests-dual.inf-complement*)

lemma *a-a-below*:
$a(a(x)) * y \leq y$

using *d-def d-restrict-iff-1* **by** *auto*

lemma *kat-equiv-5*:
$d(x) * y \leq y * d(z) \longleftrightarrow d(x) * y * a(z) = d(x) * y * bot$
proof
assume $d(x) * y \leq y * d(z)$
thus $d(x) * y * a(z) = d(x) * y * bot$
by (*metis d-complement-zero kat-4-equiv mult-assoc*)
next
assume $d(x) * y * a(z) = d(x) * y * bot$
hence $a(a(x)) * y * a(z) \leq y * a(a(z))$
by (*simp add*: *a-a-below d-def mult-isotone*)
thus $d(x) * y \leq y * d(z)$
by (*metis a-a-below a-complement-one case-split-right d-def mult-isotone order-refl*)
qed

lemma *case-split-right-sup*:
$x * -p \leq y \Longrightarrow x * --p \leq z \Longrightarrow x \leq y \sqcup z$
by (*smt* (*verit*, *ccfv-SIG*) *a-complement-one order.trans mult-1-right mult-left-dist-sup sup-commute sup-right-isotone*)

end

class *bounded-relative-left-zero-antidomain-semiring* = *relative-left-zero-antidomain-semiring* + *bounded-idempotent-left-zero-semiring*
begin

lemma *shunting-top*:
$-p * x \leq y \longleftrightarrow x \leq --p * top \sqcup y$
apply (*rule iffI*)
apply (*metis sup-commute case-split-left-sup mult-right-isotone top-greatest*)
by (*metis a-complement sup-bot-left sup-right-divisibility mult-assoc mult-left-dist-sup mult-left-one mult-left-zero mult-right-dist-sup mult-right-isotone order-trans tests-dual.inf-left-unit*)

end

end

27 Relative Modal Operators

theory *Relative-Modal*

imports *Relative-Domain*

begin

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

assumes *diamond-def*: $|x{>}y = d(x * y)$
begin

lemma *diamond-x-1*:
$|x{>}1 = d(x)$
by (*simp add*: *diamond-def*)

lemma *diamond-x-d*:
$|x{>}d(y) = d(x * y)$
using *d-mult-d diamond-def* **by** *auto*

lemma *diamond-x-und*:
$|x{>}d(y) = |x{>}y$
using *diamond-x-d diamond-def* **by** *auto*

lemma *diamond-d-closed*:
$|x{>}y = d(\ |x{>}y)$
by (*simp add*: *d-idempotent diamond-def*)

Theorem 46.11

lemma *diamond-bot-y*:
$|bot{>}y = bot$
by (*simp add*: *d-zero diamond-def*)

lemma *diamond-1-y*:
$|1{>}y = d(y)$
by (*simp add*: *diamond-def*)

Theorem 46.12

lemma *diamond-1-d*:
$|1{>}d(y) = d(y)$
by (*simp add*: *diamond-1-y diamond-x-und*)

Theorem 46.10

lemma *diamond-d-y*:
$|d(x){>}y = d(x) * d(y)$
by (*simp add*: *d-export diamond-def*)

Theorem 46.11

lemma *diamond-d-bot*:
$|d(x){>}bot = bot$
by (*metis diamond-bot-y diamond-d-y d-commutative d-zero*)

Theorem 46.12

lemma *diamond-d-1*:
$|d(x){>}1 = d(x)$
by (*simp add*: *diamond-x-1 d-idempotent*)

lemma *diamond-d-d*:

$|d(x)>d(y) = d(x) * d(y)$
by (*simp add*: *diamond-d-y diamond-x-und*)

Theorem 46.12

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

Theorem 46.2

lemma *diamond-left-dist-sup*:
$|x \sqcup y>z = |x>z \sqcup |y>z$
by (*simp add*: *d-dist-sup diamond-def mult-right-dist-sup*)

Theorem 46.3

lemma *diamond-right-sub-dist-sup*:
$|x>y \sqcup |x>z \leq |x>(y \sqcup z)$
by (*metis d-dist-sup diamond-def le-iff-sup mult-left-sub-dist-sup*)

Theorem 46.4

lemma *diamond-associative*:
$|x * y>z = |x>(y * z)$
by (*simp add*: *diamond-def mult-assoc*)

Theorem 46.4

lemma *diamond-left-mult*:
$|x * y>z = |x>|y>z$
using *diamond-x-und diamond-def mult-assoc* **by** *auto*

lemma *diamond-right-mult*:
$|x>(y * z) = |x>|y>z$
using *diamond-associative diamond-left-mult* **by** *auto*

Theorem 46.6

lemma *diamond-d-export*:
$|d(x) * y>z = d(x) * |y>z$
using *diamond-d-y diamond-def mult-assoc* **by** *auto*

lemma *diamond-diamond-export*:
$||x>y>z = |x>y * |z>1$
using *diamond-d-y diamond-def* **by** *force*

Theorem 46.1

lemma *diamond-left-isotone*:
$x \leq y \Longrightarrow |x>z \leq |y>z$
by (*metis diamond-left-dist-sup le-iff-sup*)

Theorem 46.1

lemma *diamond-right-isotone*:
$y \leq z \Longrightarrow |x>y \leq |x>z$

by (*metis diamond-right-sub-dist-sup le-iff-sup le-sup-iff*)

lemma *diamond-isotone*:
$w \le y \Longrightarrow x \le z \Longrightarrow |w{>}x \le |y{>}z$
by (*meson diamond-left-isotone diamond-right-isotone order-trans*)

lemma *diamond-left-upper-bound*:
$|x{>}y \le |x \sqcup z{>}y$
by (*simp add*: *diamond-left-isotone*)

lemma *diamond-right-upper-bound*:
$|x{>}y \le |x{>}(y \sqcup z)$
by (*simp add*: *diamond-right-isotone*)

lemma *diamond-lower-bound-right*:
$|x{>}(d(y) * d(z)) \le |x{>}d(y)$
by (*simp add*: *diamond-right-isotone d-mult-left-lower-bound*)

lemma *diamond-lower-bound-left*:
$|x{>}(d(y) * d(z)) \le |x{>}d(z)$
using *diamond-lower-bound-right d-commutative* **by** *force*

Theorem 46.5

lemma *diamond-right-sub-dist-mult*:
$|x{>}(d(y) * d(z)) \le |x{>}d(y) * |x{>}d(z)$
using *diamond-lower-bound-left diamond-lower-bound-right d-mult-greatest-lower-bound diamond-def* **by** *force*

Theorem 46.13

lemma *diamond-demodalisation-1*:
$d(x) * |y{>}z \le Z \longleftrightarrow d(x) * y * d(z) \le Z$
by (*metis d-weak-locality diamond-def mult-assoc*)

Theorem 46.14

lemma *diamond-demodalisation-3*:
$|x{>}y \le d(z) \longleftrightarrow x * d(y) \le d(z) * x \sqcup Z$
apply (*rule iffI*)
apply (*smt* (*verit*) *sup-commute sup-right-isotone d-below-one d-restrict diamond-def diamond-x-und mult-left-isotone mult-right-isotone mult-1-right order-trans*)
by (*smt sup-commute sup-bot-left d-Z d-commutative d-dist-sup d-idempotent d-mult-sub d-plus-left-upper-bound diamond-d-y diamond-def diamond-x-und le-iff-sup order-trans*)

Theorem 46.6

lemma *diamond-d-export-2*:
$|d(x) * y{>}z = d(x) * |d(x) * y{>}z$
by (*metis diamond-d-export diamond-left-mult d-mult-idempotent*)

Theorem 46.7

lemma *diamond-d-promote*:
$|x * d(y)>z = |x * d(y)>(d(y) * z)$
by (*metis d-mult-idempotent diamond-def mult-assoc*)

Theorem 46.8

lemma *diamond-d-import-iff*:
$d(x) \le |y>z \longleftrightarrow d(x) \le |d(x) * y>z$
by (*metis diamond-d-export diamond-d-y d-order diamond-def order.eq-iff*)

Theorem 46.9

lemma *diamond-d-import-iff-2*:
$d(x) * d(y) \le |z>w \longleftrightarrow d(x) * d(y) \le |d(y) * z>w$
apply (*rule iffI*)
apply (*metis diamond-associative d-export d-mult-greatest-lower-bound diamond-def order.refl*)
by (*metis diamond-d-y d-mult-greatest-lower-bound diamond-def mult-assoc*)

end

class *relative-box-semiring* = *relative-diamond-semiring* + *relative-antidomain-semiring* + *box* +
assumes *box-def*: $|x]y = a(x * a(y))$
begin

Theorem 47.1

lemma *box-diamond*:
$|x]y = a(|x>a(y))$
by (*simp add*: *box-def d-a-closed diamond-def*)

Theorem 47.2

lemma *diamond-box*:
$|x>y = a(|x]a(y))$
using *box-def d-def d-mult-d diamond-def* **by** *auto*

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

lemma *box-x-1*:
$|x]1 = a(x * bot)$
by (*simp add*: *box-def*)

lemma *box-x-d*:
$|x]d(y) = a(x * a(y))$
by (*simp add*: *box-def d-a-closed*)

lemma *box-x-und*:
$|x]d(y) = |x]y$
by (*simp add*: *box-diamond d-a-closed*)

lemma *box-x-a*:
$|x]a(y) = a(x * y)$
using *a-mult-d box-def* **by** *auto*

Theorem 47.15

lemma *box-bot-y*:
$|bot]y = 1$
using *box-def* **by** *auto*

lemma *box-1-y*:
$|1]y = d(y)$
by (*simp add*: *box-def d-def*)

Theorem 47.16

lemma *box-1-d*:
$|1]d(y) = d(y)$
by (*simp add*: *box-1-y box-x-und*)

lemma *box-1-a*:
$|1]a(y) = a(y)$
by (*simp add*: *box-x-a*)

lemma *box-d-y*:
$|d(x)]y = a(x) \sqcup d(y)$
using *a-export-a box-def d-def* **by** *auto*

lemma *box-a-y*:
$|a(x)]y = d(x) \sqcup d(y)$
by (*simp add*: *a-mult-deMorgan-1 box-def*)

Theorem 47.14

lemma *box-d-bot*:
$|d(x)]bot = a(x)$
by (*simp add*: *box-x-bot d-a-closed*)

lemma *box-a-bot*:
$|a(x)]bot = d(x)$
by (*simp add*: *box-x-bot d-def*)

Theorem 47.15

lemma *box-d-1*:
$|d(x)]1 = 1$
by (*simp add*: *box-d-y d-one-one*)

lemma *box-a-1*:
$|a(x)]1 = 1$
by (*simp add*: *box-x-1*)

Theorem 47.13

lemma *box-d-d*:
$|d(x)]d(y) = a(x) \sqcup d(y)$
by (*simp add*: *box-d-y box-x-und*)

lemma *box-a-d*:
$|a(x)]d(y) = d(x) \sqcup d(y)$
by (*simp add*: *box-a-y box-x-und*)

lemma *box-d-a*:
$|d(x)]a(y) = a(x) \sqcup a(y)$
by (*simp add*: *box-x-a a-export-d*)

lemma *box-a-a*:
$|a(x)]a(y) = d(x) \sqcup a(y)$
by (*simp add*: *box-a-y a-d-closed*)

Theorem 47.15

lemma *box-d-d-same*:
$|d(x)]d(x) = 1$
using *box-x-d d-complement-zero* **by** *auto*

lemma *box-a-a-same*:
$|a(x)]a(x) = 1$
by (*simp add*: *box-def*)

Theorem 47.16

lemma *box-d-below-box*:
$d(x) \leq |d(y)]d(x)$
by (*simp add*: *box-d-d*)

lemma *box-d-closed*:
$|x]y = d(|x]y)$
by (*simp add*: *a-d-closed box-def*)

lemma *box-deMorgan-1*:
$a(|x]y) = |x>a(y)$
by (*simp add*: *diamond-box box-def*)

lemma *box-deMorgan-2*:
$a(|x>y) = |x]a(y)$
using *box-x-a d-a-closed diamond-def* **by** *auto*

Theorem 47.5

lemma *box-left-dist-sup*:
$|x \sqcup y]z = |x]z * |y]z$
by (*simp add*: *a-dist-sup box-def mult-right-dist-sup*)

lemma *box-right-dist-sup*:
$|x](y \sqcup z) = a(x * a(y) * a(z))$

by (*simp add*: *a-dist-sup box-def mult-assoc*)

lemma *box-associative*:
$|x * y]z = a(x * y * a(z))$
by (*simp add*: *box-def*)

Theorem 47.6

lemma *box-left-mult*:
$|x * y]z = |x]|y]z$
using *box-x-a box-def mult-assoc* **by** *force*

lemma *box-right-mult*:
$|x](y * z) = a(x * a(y * z))$
by (*simp add*: *box-def*)

Theorem 47.7

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

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

lemma *box-right-submult-d-a*:
$|x](d(y) * a(z)) \leq |x]d(y) * |x]a(z)$
using *box-right-submult-a-d box-x-a d-def tests-dual.sub-commutative* **by** *auto*

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

Theorem 47.8

lemma *box-d-export*:
$|d(x) * y]z = a(x) \sqcup |y]z$
by (*simp add*: *a-export-d box-def mult-assoc*)

lemma *box-a-export*:
$|a(x) * y]z = d(x) \sqcup |y]z$
using *box-a-y box-d-closed box-left-mult* **by** *auto*

Theorem 47.4

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

Theorem 47.3

lemma *box-right-isotone*:

$y \le z \Longrightarrow |x]y \le |x]z$
by (*metis a-antitone box-def mult-right-isotone*)

lemma *box-antitone-isotone*:
$y \le w \Longrightarrow x \le z \Longrightarrow |w]x \le |y]z$
by (*meson box-left-antitone box-right-isotone order-trans*)

lemma *diamond-1-a*:
$|1>a(y) = a(y)$
by (*simp add*: *d-def diamond-1-y*)

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

lemma *diamond-a-bot*:
$|a(x)>bot = bot$
by (*simp add*: *diamond-a-y d-zero*)

lemma *diamond-a-1*:
$|a(x)>1 = a(x)$
by (*simp add*: *d-def diamond-x-1*)

lemma *diamond-a-d*:
$|a(x)>d(y) = a(x) * d(y)$
by (*simp add*: *diamond-a-y diamond-x-und*)

lemma *diamond-d-a*:
$|d(x)>a(y) = d(x) * a(y)$
by (*simp add*: *a-d-closed diamond-d-y*)

lemma *diamond-a-a*:
$|a(x)>a(y) = a(x) * a(y)$
by (*simp add*: *a-mult-closed diamond-def*)

lemma *diamond-a-a-same*:
$|a(x)>a(x) = a(x)$
by (*simp add*: *diamond-a-a*)

lemma *diamond-a-export*:
$|a(x) * y>z = a(x) * |y>z$
using *diamond-a-y diamond-associative diamond-def* **by** *auto*

lemma *a-box-a-a*:
$a(p) * |a(p)]a(q) = a(p) * a(q)$
using *box-a-a box-a-bot box-x-bot tests-dual.sup-complement-intro* **by** *auto*

lemma *box-left-lower-bound*:
$|x \sqcup y]z \le |x]z$

by (*simp add*: *box-left-antitone*)

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

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

lemma *box-lower-bound-left*:
$|x](d(y) * d(z)) \leq |x]d(z)$
by (*simp add*: *box-right-isotone d-restrict-iff-1*)

Theorem 47.9

lemma *box-d-import*:
$d(x) * |y]z = d(x) * |d(x) * y]z$
using *a-box-a-a box-left-mult box-def d-def* **by** *force*

Theorem 47.10

lemma *box-d-promote*:
$|x * d(y)]z = |x * d(y)](d(y) * z)$
using *a-box-a-a box-x-a box-def d-def mult-assoc* **by** *auto*

Theorem 47.11

lemma *box-d-import-iff*:
$d(x) \leq |y]z \longleftrightarrow d(x) \leq |d(x) * y]z$
using *box-d-export box-def d-def tests-dual.shunting* **by** *auto*

Theorem 47.12

lemma *box-d-import-iff-2*:
$d(x) * d(y) \leq |z]w \longleftrightarrow d(x) * d(y) \leq |d(y) * z]w$
apply (*rule iffI*)
using *box-d-export le-supI2* **apply** *simp*
by (*metis box-d-import d-commutative d-restrict-iff-1*)

Theorem 47.20

lemma *box-demodalisation-2*:
$-p \leq |y](-q) \longleftrightarrow -p * y * --q \leq Z$
by (*simp add*: *a-greatest-left-absorber box-def mult-assoc*)

lemma *box-right-sub-dist-sup*:
$|x]d(y) \sqcup |x]d(z) \leq |x](d(y) \sqcup d(z))$
by (*simp add*: *box-right-isotone*)

lemma *box-diff-var*:
$|x](d(y) \sqcup a(z)) * |x]d(z) \leq |x]d(z)$
by (*simp add*: *box-right-dist-sup box-x-d tests-dual.upper-bound-right*)

Theorem 47.19

lemma *diamond-demodalisation-2*:
$|x{>}y \leq d(z) \longleftrightarrow a(z) * x * d(y) \leq Z$
using *a-antitone a-greatest-left-absorber a-mult-d d-def diamond-def mult-assoc*
by *fastforce*

Theorem 47.17

lemma *box-below-Z*:
$(|x]y) * x * a(y) \leq Z$
by (*simp add*: *a-restrict box-def mult-assoc*)

Theorem 47.18

lemma *box-partial-correctness*:
$|x]1 = 1 \longleftrightarrow x * bot \leq Z$
by (*simp add*: *box-x-1 a-strict*)

lemma *diamond-split*:
$|x{>}y = d(z) * |x{>}y \sqcup a(z) * |x{>}y$
by (*metis d-def diamond-def sup-monoid.add-commute tests-dual.sba-dual.sup-cases tests-dual.sub-commutative*)

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

proposition *box-dist-mult*: $|x](d(y) * d(z)) = |x](d(y)) * |x](d(z))$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *box-demodalisation-3*: $d(x) \leq |y]d(z) \longrightarrow d(x) * y \leq y * d(z) \sqcup Z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *fbox-diff*: $|x](d(y) \sqcup a(z)) \leq |x]y \sqcup a(|x]z)$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *diamond-diff*: $|x{>}y * a(|x{>}z) \leq |x{>}(d(y) * a(z))$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *diamond-diff-var*: $|x{>}d(y) \leq |x{>}(d(y) * a(z)) \sqcup |x{>}d(z)$ **nitpick** [*expect=genuine,card=6*] **oops**

end

class *relative-left-zero-diamond-semiring* = *relative-diamond-semiring* + *relative-domain-semiring* + *idempotent-left-zero-semiring*
begin

lemma *diamond-right-dist-sup*:
$|x{>}(y \sqcup z) = |x{>}y \sqcup |x{>}z$
by (*simp add*: *d-dist-sup diamond-def mult-left-dist-sup*)

end

class *relative-left-zero-box-semiring* = *relative-box-semiring* +
relative-left-zero-antidomain-semiring
begin

subclass *relative-left-zero-diamond-semiring* **..**

lemma *box-right-mult-d-d*:
$|x](d(y) * d(z)) = |x]d(y) * |x]d(z)$
using *a-dist-sup box-d-a box-def d-def mult-left-dist-sup* **by** *auto*

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

lemma *box-right-mult-d-a*:
$|x](d(y) * a(z)) = |x]d(y) * |x]a(z)$
using *box-right-mult-a-d box-def box-x-a d-def* **by** *auto*

lemma *box-right-mult-a-a*:
$|x](a(y) * a(z)) = |x]a(y) * |x]a(z)$
using *a-dist-sup box-def mult-left-dist-sup tests-dual.sub-sup-demorgan* **by** *force*

lemma *box-demodalisation-3*:
assumes $d(x) \le |y]d(z)$
shows $d(x) * y \le y * d(z) \sqcup Z$
proof −
have $d(x) * y * a(z) \le Z$
using *assms a-greatest-left-absorber box-x-d d-def mult-assoc* **by** *auto*
thus *?thesis*
by (*simp add*: *a-a-below case-split-right-sup d-def sup-commute mult-assoc*)
qed

lemma *fbox-diff*:
$|x](d(y) \sqcup a(z)) \le |x]y \sqcup a(\ |x]z)$
by (*smt* (*z3*) *a-compl-intro a-dist-sup a-mult-d a-plus-left-lower-bound sup-commute box-def d-def mult-left-dist-sup tests-dual.sba-dual.shunting*)

lemma *diamond-diff-var*:
$|x{>}d(y) \le |x{>}(d(y) * a(z)) \sqcup |x{>}d(z)$
by (*metis d-cancellation-1 diamond-right-dist-sup diamond-right-isotone sup-commute*)

lemma *diamond-diff*:
$|x{>}y * a(\ |x{>}z) \le |x{>}(d(y) * a(z))$
by (*metis d-a-shunting d-idempotent diamond-def diamond-diff-var diamond-x-und*)

end

end

28 Complete Tests

theory *Complete-Tests*

imports *Tests*

begin

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

lemma *Sup-isotone*:
  test-set $B \Longrightarrow A \subseteq B \Longrightarrow$ *Sup* $A \leq$ *Sup* B
  by (*metis sup-least sup-test sup-upper test-set-closed subset-eq*)

lemma *mult-right-dist-sup*:
  assumes *test-set* A
    shows *Sup* $A * -p =$ *Sup* $\{\ x * -p \mid x\ .\ x \in A\ \}$
proof −
  have *1*: *test-set* $\{\ x * -p \mid x\ .\ x \in A\ \}$
    by (*simp add*: *assms mult-right-dist-test-set*)
  have *2*: *Sup* $\{\ x * -p \mid x\ .\ x \in A\ \} \leq$ *Sup* $A * -p$
    by (*smt* (*verit*, *del-insts*) *assms mem-Collect-eq tests-dual.sub-sup-left-isotone sub-mult-closed sup-test sup-least sup-upper test-set-def*)
  have $\forall x{\in}A\ .\ x \leq --(--$*Sup* $\{\ x * -p \mid x\ .\ x \in A\ \} \sqcup --p)$
  proof
    fix x
    assume *3*: $x \in A$
    hence $x * -p \sqcup --p \leq$ *Sup* $\{\ x * -p \mid x\ .\ x \in A\ \} \sqcup --p$
      using *1* **by** (*smt* (*verit*, *del-insts*) *assms mem-Collect-eq tests-dual.sub-inf-left-isotone sub-mult-closed sup-upper test-set-def sup-test*)
    thus $x \leq --(--$*Sup* $\{\ x * -p \mid x\ .\ x \in A\ \} \sqcup --p)$
      using *1 3* **by** (*smt* (*z3*) *assms tests-dual.inf-closed sub-comm test-set-def sup-test sub-mult-closed tests-dual.sba-dual.shunting-right tests-dual.sba-dual.sub-sup-left-isotone tests-dual.inf-absorb tests-dual.inf-less-eq-cases-3*)
  qed
  hence *Sup* $A \leq --(--$*Sup* $\{\ x * -p \mid x\ .\ x \in A\ \} \sqcup --p)$
    by (*simp add*: *assms sup-least*)
  hence *Sup* $A * -p \leq$ *Sup* $\{\ x * -p \mid x\ .\ x \in A\ \}$
    using *1* **by** (*smt* (*z3*) *assms sup-test tests-dual.sba-dual.shunting tests-dual.sub-commutative tests-dual.sub-sup-closed tests-dual.sub-sup-demorgan*)
  thus *?thesis*
    using *1 2* **by** (*smt* (*z3*) *assms sup-test tests-dual.sba-dual.sub-sup-closed*

tests-dual.antisymmetric tests-dual.inf-demorgan tests-dual.inf-idempotent)
qed

lemma *mult-left-dist-sup*:
  assumes *test-set A*
  shows $-p * Sup\ A = Sup\ \{\ -p * x \mid x\ .\ x \in A\ \}$
proof −
  have *1*: $Sup\ A * -p = Sup\ \{\ x * -p \mid x\ .\ x \in A\ \}$
    by (*simp add*: *assms mult-right-dist-sup*)
  have *2*: $-p * Sup\ A = Sup\ A * -p$
    by (*metis assms sub-comm sup-test*)
  have $\{\ -p * x \mid x\ .\ x \in A\ \} = \{\ x * -p \mid x\ .\ x \in A\ \}$
    by (*metis assms test-set-def tests-dual.sub-commutative*)
  thus *?thesis*
    using *1 2* **by** *simp*
qed

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

lemma *Sum-test*:
  $test\text{-}seq\ t \Longrightarrow Sum\ t = --Sum\ t$
  using *Sum-def sup-test test-seq-test-set* **by** *auto*

lemma *Sum-upper*:
  $test\text{-}seq\ t \Longrightarrow t\ x \leq Sum\ t$
  using *Sum-def sup-upper test-seq-test-set* **by** *auto*

lemma *Sum-least*:
  $test\text{-}seq\ t \Longrightarrow (\forall n\ .\ t\ n \leq -p) \Longrightarrow Sum\ t \leq -p$
  using *Sum-def sup-least test-seq-test-set* **by** *force*

lemma *mult-right-dist-Sum*:
  $test\text{-}seq\ t \Longrightarrow (\forall n\ .\ t\ n * -p \leq -q) \Longrightarrow Sum\ t * -p \leq -q$
  by (*smt* (*verit*, *del-insts*) *CollectD Sum-def sup-least sup-test test-seq-test-set test-set-def tests-dual.sba-dual.shunting-right tests-dual.sba-dual.sub-sup-closed*)

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

lemma *pSum-below-Sum*:
  $test\text{-}seq\ t \Longrightarrow pSum\ t\ m \leq Sum\ t$
  using *Sum-test Sum-upper nat-test-def pSum-below-sum test-seq-def mult-right-dist-Sum* **by** *auto*

lemma *pSum-sup*:
  assumes *test-seq t*

```
    shows pSum t m = Sup { t i | i . i ∈ {..<m} }
proof −
  have 1: test-set { t i | i . i ∈ {..<m} }
    using assms test-seq-test-set test-set-def by auto
  have ∀ y∈{ t i | i . i ∈ {..<m} } . y ≤ −−pSum t m
    using assms pSum-test pSum-upper by force
  hence 2: Sup { t i | i . i ∈ {..<m} } ≤ −−pSum t m
    using 1 by (simp add: sup-least)
  have pSum t m ≤ Sup { t i | i . i ∈ {..<m} }
  proof (induct m)
    case 0
    show ?case
      by (smt (verit, ccfv-SIG) Collect-empty-eq empty-iff lessThan-0
pSum.simps(1) sup-test test-set-def tests-dual.top-greatest)
  next
    case (Suc n)
    have 4: test-set {t i |i. i ∈ {..<n}}
      using assms test-seq-def test-set-def by auto
    have 5: test-set {t i |i. i < Suc n}
      using assms test-seq-def test-set-def by force
    hence 6: Sup {t i |i. i < Suc n} = −−Sup {t i |i. i < Suc n}
      using sup-test by auto
    hence ∀ x∈{t i |i. i ∈ {..<n}} . x ≤ −−Sup {t i |i. i < Suc n}
      using 5 less-Suc-eq sup-upper by fastforce
    hence 7: Sup {t i |i. i ∈ {..<n}} ≤ −−Sup {t i |i. i < Suc n}
      using 4 by (simp add: sup-least)
    have t n ∈ {t i |i. i < Suc n}
      by auto
    hence t n ≤ Sup {t i |i. i < Suc n}
      using 5 by (simp add: sup-upper)
    hence pSum t n ⊔ t n ≤ Sup {t i |i. i <Suc n}
      using Suc 4 6 7 by (smt assms tests-dual.greatest-lower-bound test-seq-def
pSum-test tests-dual.sba-dual.transitive sup-test)
    thus ?case
      by simp
  qed
  thus ?thesis
    using 1 2 by (smt assms tests-dual.antisymmetric sup-test pSum-test)
qed

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

lemma Sum-range:
  Sum f = Sup (range f)
  by (simp add: Sum-def image-def)

lemma Prod-range:
  Prod f = Inf (range f)
```

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

end

end

29 Complete Domain

theory *Complete-Domain*

imports *Relative-Domain Complete-Tests*

begin

class *complete-antidomain-semiring* = *relative-antidomain-semiring* +
complete-tests +
assumes *a-dist-Sum*: *ascending-chain* $f \longrightarrow -(Sum\ f) = Prod\ (\lambda n\ .\ -f\ n)$
assumes *a-dist-Prod*: *descending-chain* $f \longrightarrow -(Prod\ f) = Sum\ (\lambda n\ .\ -f\ n)$
begin

lemma *a-ascending-chain*:
ascending-chain $f \Longrightarrow$ *descending-chain* $(\lambda n\ .\ -f\ n)$
by (*simp add*: *a-antitone ascending-chain-def descending-chain-def*)

lemma *a-descending-chain*:
descending-chain $f \Longrightarrow$ *ascending-chain* $(\lambda n\ .\ -f\ n)$
by (*simp add*: *a-antitone ord.ascending-chain-def ord.descending-chain-def*)

lemma *d-dist-Sum*:
ascending-chain $f \Longrightarrow d(Sum\ f) = Sum\ (\lambda n\ .\ d(f\ n))$
by (*simp add*: *d-def a-ascending-chain a-dist-Prod a-dist-Sum*)

lemma *d-dist-Prod*:
descending-chain $f \Longrightarrow d(Prod\ f) = Prod\ (\lambda n\ .\ d(f\ n))$
by (*simp add*: *d-def a-dist-Sum a-dist-Prod a-descending-chain*)

end

end

30 Preconditions

theory *Preconditions*

imports *Tests*

begin

class *pre* =
  fixes *pre* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** ‹«› *55*)

class *precondition* = *tests* + *pre* +
  assumes *pre-closed*: $x«-q = --(x«-q)$
  assumes *pre-seq*: $x*y«-q = x«y«-q$
  assumes *pre-lower-bound-right*: $x«-p*-q \leq x«-q$
  assumes *pre-one-increasing*: $-q \leq 1«-q$
begin

Theorem 39.2

lemma *pre-sub-distr*:
  $x«-p*-q \leq (x«-p)*(x«-q)$
  by (*smt* (*z3*) *pre-closed pre-lower-bound-right tests-dual.sub-commutative tests-dual.sub-sup-closed tests-dual.least-upper-bound*)

Theorem 39.5

lemma *pre-below-one*:
  $x«-p \leq 1$
  by (*metis pre-closed tests-dual.sub-bot-least*)

lemma *pre-lower-bound-left*:
  $x«-p*-q \leq x«-p$
  using *pre-lower-bound-right tests-dual.sub-commutative* **by** *fastforce*

Theorem 39.1

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

Theorem 39.4 and Theorem 40.9

lemma *pre-below-pre-one*:
  $x«-p \leq x«1$
  using *tests-dual.sba-dual.one-def pre-iso tests-dual.sub-bot-least* **by** *blast*

Theorem 39.3

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

Theorem 39.6

lemma *pre-compose*:
  $-p \leq x«-q \Longrightarrow -q \leq y«-r \Longrightarrow -p \leq x*y«-r$
  by (*metis pre-closed pre-iso tests-dual.transitive pre-seq*)

proposition *pre-test-test*: $-p*(-p«-q) = -p*-q$ **nitpick**
[*expect=genuine,card=2*] **oops**
proposition *pre-test-promote*: $-p«-q = -p«-p*-q$ **nitpick**
[*expect=genuine,card=2*] **oops**

proposition *pre-test*: $-p \ll -q = --p \sqcup -q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-test*: $-p \ll -q = -p * -q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-distr-mult*: $x \ll -p * -q = (x \ll -p) * (x \ll -q)$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *pre-distr-plus*: $x \ll -p \sqcup -q = (x \ll -p) * (x \ll -q)$ **nitpick** [*expect=genuine,card=2*] **oops**

end

class *precondition-test-test* = *precondition* +
  assumes *pre-test-test*: $-p * (-p \ll -q) = -p * -q$
begin

lemma *pre-one*:
  $1 \ll -p = -p$
  by (*metis pre-closed pre-test-test tests-dual.sba-dual.one-def tests-dual.sup-left-unit*)

lemma *pre-import*:
  $-p * (x \ll -q) = -p * (-p * x \ll -q)$
  by (*metis pre-closed pre-seq pre-test-test*)

lemma *pre-import-composition*:
  $-p * (-p * x * y \ll -q) = -p * (x \ll y \ll -q)$
  by (*metis pre-closed pre-seq pre-import*)

lemma *pre-import-equiv*:
  $-p \le x \ll -q \longleftrightarrow -p \le -p * x \ll -q$
  by (*metis leq-def pre-closed pre-import*)

lemma *pre-import-equiv-mult*:
  $-p * -q \le x \ll -s \longleftrightarrow -p * -q \le -q * x \ll -s$
  by (*smt leq-def pre-closed sub-assoc sub-mult-closed pre-import*)

proposition *pre-test-promote*: $-p \ll -q = -p \ll -p * -q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-test*: $-p \ll -q = --p \sqcup -q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-test*: $-p \ll -q = -p * -q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-distr-mult*: $x \ll -p * -q = (x \ll -p) * (x \ll -q)$ **nitpick** [*expect=genuine,card=4*] **oops**
proposition *pre-distr-plus*: $x \ll -p \sqcup -q = (x \ll -p) * (x \ll -q)$ **nitpick** [*expect=genuine,card=2*] **oops**

end

class *precondition-promote* = *precondition* +
  assumes *pre-test-promote*: $-p \ll -q = -p \ll -p * -q$

begin

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

proposition *pre-test-test*: $-p*(-p«-q) = -p*-q$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *pre-test*: $-p«-q = --p⊔-q$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *pre-test*: $-p«-q = -p*-q$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *pre-distr-mult*: $x«-p*-q = (x«-p)*(x«-q)$ **nitpick** [*expect=genuine*,*card=4*] **oops**
proposition *pre-distr-plus*: $x«-p⊔-q = (x«-p)*(x«-q)$ **nitpick** [*expect=genuine*,*card=2*] **oops**

end

class *precondition-test-box* = *precondition* +
assumes *pre-test*: $-p«-q = --p⊔-q$
begin

lemma *pre-test-neg*:
$--p*(-p«-q) = --p$
by (*simp add*: *pre-test*)

lemma *pre-bot*:
$bot«-q = 1$
by (*metis pre-test tests-dual.sba-dual.one-def tests-dual.sba-dual.sup-left-zero tests-dual.top-double-complement*)

lemma *pre-export*:
$-p*x«-q = --p⊔(x«-q)$
by (*metis pre-closed pre-seq pre-test*)

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

lemma *pre-test-test-same*:
$-p«-p = 1$
using *pre-test tests-dual.sba-dual.less-eq-sup-top tests-dual.sba-dual.reflexive* **by** *auto*

lemma *test-below-pre-test-mult*:
$-q \leq -p«-p*-q$
by (*metis pre-test tests-dual.sba-dual.reflexive tests-dual.sba-dual.shunting tests-dual.sub-sup-closed*)

lemma *test-below-pre-test*:
  $-q \leq -p«-q$
  by (*simp add*: *pre-test tests-dual.sba-dual.upper-bound-right*)

lemma *test-below-pre-test-2*:
  $--p \leq -p«-q$
  by (*simp add*: *pre-test tests-dual.sba-dual.upper-bound-left*)

lemma *pre-test-bot*:
  $-p«bot = --p$
  by (*metis pre-test tests-dual.sba-dual.sup-right-unit tests-dual.top-double-complement*)

lemma *pre-test-one*:
  $-p«1 = 1$
  by (*metis pre-seq pre-bot tests-dual.sup-right-zero*)

subclass *precondition-test-test*
  apply *unfold-locales*
  by (*simp add*: *pre-test tests-dual.sup-complement-intro*)

subclass *precondition-promote*
  apply *unfold-locales*
  by (*metis pre-test tests-dual.sba-dual.sub-commutative tests-dual.sub-sup-closed tests-dual.inf-complement-intro*)

proposition *pre-test*: $-p«-q = -p*-q$ **nitpick** [*expect=genuine,card=2*] **oops**
proposition *pre-distr-mult*: $x«-p*-q = (x«-p)*(x«-q)$ **oops**
proposition *pre-distr-plus*: $x«-p\sqcup-q = (x«-p)*(x«-q)$ **nitpick** [*expect=genuine,card=2*] **oops**

end

class *precondition-test-diamond* = *precondition* +
  assumes *pre-test*: $-p«-q = -p*-q$
begin

lemma *pre-test-neg*:
  $--p*(-p«-q) = bot$
  by (*simp add*: *pre-test tests-dual.sub-associative tests-dual.sub-commutative*)

lemma *pre-bot*:
  $bot«-q = bot$
  by (*metis pre-test tests-dual.sup-left-zero tests-dual.top-double-complement*)

lemma *pre-export*:
  $-p*x«-q = -p*(x«-q)$
  by (*metis pre-closed pre-seq pre-test*)

lemma *pre-neg-mult*:
$-p*x«-q \leq -p$
by (*metis pre-closed pre-export tests-dual.upper-bound-left*)

lemma *pre-test-test-same*:
$-p«-p = -p$
by (*simp add*: *pre-test*)

lemma *test-above-pre-test-plus*:
$--p«-p\sqcup-q \leq -q$
using *pre-test tests-dual.sba-dual.inf-complement-intro tests-dual.sub-commutative tests-dual.sub-inf-def tests-dual.upper-bound-left* **by** *auto*

lemma *test-above-pre-test*:
$-p«-q \leq -q$
by (*simp add*: *pre-test tests-dual.upper-bound-right*)

lemma *test-above-pre-test-2*:
$-p«-q \leq -p$
by (*simp add*: *pre-test tests-dual.upper-bound-left*)

lemma *pre-test-bot*:
$-p«bot = bot$
by (*metis pre-test tests-dual.sup-right-zero tests-dual.top-double-complement*)

lemma *pre-test-one*:
$-p«1 = -p$
by (*metis pre-test tests-dual.complement-top tests-dual.sup-right-unit*)

subclass *precondition-test-test*
apply *unfold-locales*
by (*simp add*: *pre-test tests-dual.sub-associative*)

subclass *precondition-promote*
apply *unfold-locales*
by (*metis pre-seq pre-test tests-dual.sup-idempotent*)

proposition *pre-test*: $-p«-q = --p\sqcup-q$ **nitpick** [*expect=genuine*,*card=2*] **oops**
proposition *pre-distr-mult*: $x«-p*-q = (x«-p)*(x«-q)$ **nitpick** [*expect=genuine*,*card=6*] **oops**
proposition *pre-distr-plus*: $x«-p\sqcup-q = (x«-p)*(x«-q)$ **nitpick** [*expect=genuine*,*card=2*] **oops**

end

class *precondition-distr-mult* = *precondition* +
assumes *pre-distr-mult*: $x«-p*-q = (x«-p)*(x«-q)$

begin

proposition *pre-test-test*: $-p*(-p«-q) = -p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test-promote*: $-p«-q = -p«-p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test*: $-p«-q = --p\sqcup-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test*: $-p«-q = -p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-distr-plus*: $x«-p\sqcup-q = (x«-p)*(x«-q)$ **nitpick** $[expect=genuine,card=2]$ **oops**

end

class *precondition-distr-plus* = *precondition* +
  assumes *pre-distr-plus*: $x«-p\sqcup-q = (x«-p)\sqcup(x«-q)$
begin

proposition *pre-test-test*: $-p*(-p«-q) = -p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test-promote*: $-p«-q = -p«-p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test*: $-p«-q = --p\sqcup-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-test*: $-p«-q = -p*-q$ **nitpick** $[expect=genuine,card=2]$ **oops**
proposition *pre-distr-mult*: $x«-p*-q = (x«-p)*(x«-q)$ **nitpick** $[expect=genuine,card=4]$ **oops**

end

end

31 Hoare Calculus

theory *Hoare*

imports *Complete-Tests Preconditions*

begin

class *ite* =
  fixes *ite* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$ (‹- ◁ - ▷ -› [58,58,58] 57)

class *hoare-triple* =
  fixes *hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ (‹- ⦃ - ⦄ -› [54,54,54] 53)

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

Theorem 40.2

lemma *ite-pre-then*:
$-p*(x \lhd -p \rhd y \ll -q) = -p*(x \ll -q)$
proof –
have $-p*(x \lhd -p \rhd y \ll -q) = -p*(x \ll -q) \sqcup bot*(y \ll -q)$
by (*smt* (*z3*) *ite-pre pre-closed tests-dual.sba-dual.sup-right-unit tests-dual.sub-commutative tests-dual.sup-left-zero tests-dual.sup-right-dist-inf tests-dual.top-double-complement tests-dual.wnf-lemma-1*)
thus *?thesis*
by (*metis pre-closed tests-dual.sba-dual.sup-right-unit tests-dual.sub-sup-closed tests-dual.sup-left-zero*)
qed

Theorem 40.3

lemma *ite-pre-else*:
$--p*(x \lhd -p \rhd y \ll -q) = --p*(y \ll -q)$
proof –
have $--p*(x \lhd -p \rhd y \ll -q) = bot*(x \ll -q) \sqcup --p*(y \ll -q)$
by (*smt* (*z3*) *ite-pre pre-closed tests-dual.sub-commutative tests-dual.sub-inf-left-zero tests-dual.sup-left-zero tests-dual.sup-right-dist-inf tests-dual.top-double-complement tests-dual.wnf-lemma-3*)
thus *?thesis*
by (*metis pre-closed tests-dual.sba-dual.sub-sup-demorgan tests-dual.sub-inf-left-zero tests-dual.sup-left-zero*)
qed

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

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

Theorem 40.1

lemma *ite-import-mult*:
$-p*-q \le x \ll -r \Longrightarrow --p*-q \le y \ll -r \Longrightarrow -q \le x \lhd -p \rhd y \ll -r$
by (*smt* (*verit*) *ite-import-mult-else ite-import-mult-then pre-closed tests-dual.sba-dual.inf-less-eq-cases*)

end

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

Theorem 40.4

lemma *while-pre-then*:

$-p{*}(-p{\star}x{«}-q) = -p{*}(x{«}-p{\star}x{«}-q)$
by (*smt pre-closed tests-dual.sub-commutative while-pre tests-dual.wnf-lemma-1*)

Theorem 40.5

lemma *while-pre-else*:
$--p{*}(-p{\star}x{«}-q) = --p{*}-q$
by (*smt pre-closed tests-dual.sub-commutative while-pre tests-dual.wnf-lemma-3*)

Theorem 40.6

lemma *while-pre-sub-1*:
$-p{\star}x{«}-q \le x{*}(-p{\star}x){\lhd}-p{\rhd}1{«}-q$
by (*smt* (*z3*) *ite-import-mult pre-closed pre-one-increasing pre-seq tests-dual.sba-dual.transitive tests-dual.sub-sup-closed tests-dual.upper-bound-right while-pre-else while-pre-then*)

Theorem 40.7

lemma *while-pre-sub-2*:
$-p{\star}x{«}-q \le x{\lhd}-p{\rhd}1{«}-p{\star}x{«}-q$
by (*smt* (*z3*) *ite-import-mult pre-closed pre-one-increasing tests-dual.sba-dual.transitive tests-dual.sub-sup-closed tests-dual.upper-bound-right while-pre-then*)

Theorem 40.8

lemma *while-pre-compl*:
$--p \le -p{\star}x{«}--p$
by (*metis pre-closed tests-dual.sup-idempotent tests-dual.upper-bound-right while-pre-else*)

lemma *while-pre-compl-one*:
$--p \le -p{\star}x{«}1$
by (*metis tests-dual.sba-dual.top-double-complement while-post tests-dual.sup-right-unit while-pre-compl*)

Theorem 40.10

lemma *while-export-equiv*:
$-q \le -p{\star}x{«}1 \longleftrightarrow -p{*}-q \le -p{\star}x{«}1$
by (*smt pre-closed tests-dual.sba-dual.shunting tests-dual.sba-dual.sub-less-eq-def tests-dual.sba-dual.top-double-complement while-pre-compl-one*)

lemma *nat-test-pre*:
assumes *nat-test t s*
and $-q \le s$
and $\forall n .\ t\ n{*}-p{*}-q \le x{«}pSum\ t\ n{*}-q$
shows $-q \le -p{\star}x{«}--p{*}-q$
proof −
have *1*: $-q{*}--p \le -p{\star}x{«}--p{*}-q$
by (*metis pre-closed tests-dual.sub-commutative while-post tests-dual.upper-bound-right while-pre-else*)
have $\forall n .\ t\ n{*}-p{*}-q \le -p{\star}x{«}--p{*}-q$

proof
fix n
show $t\ n{*}{-}p{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
proof (*induct n rule*: *nat-less-induct*)
fix n
have *2*: $t\ n = {-}{-}(t\ n)$
using *assms*(*1*) *nat-test-def* **by** *auto*
assume $\forall\, m{<}n\ .\ t\ m{*}{-}p{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
hence $\forall\, m{<}n\ .\ t\ m{*}{-}p{*}{-}q \sqcup t\ m{*}{-}{-}p{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
using *1* **by** (*smt* (*verit*, *del-insts*) *assms*(*1*) *tests-dual.greatest-lower-bound leq-def nat-test-def pre-closed tests-dual.sub-associative tests-dual.sub-commutative sub-mult-closed*)
hence $\forall\, m{<}n\ .\ t\ m{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
by (*smt* (*verit*, *del-insts*) *assms*(*1*) *tests-dual.sup-right-unit tests-dual.sup-left-dist-inf tests-dual.sup-right-dist-inf nat-test-def tests-dual.inf-complement sub-mult-closed*)
hence $pSum\ t\ n{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
by (*smt assms*(*1*) *pSum-below-nat pre-closed sub-mult-closed*)
hence $t\ n{*}{-}p{*}{-}q{*}({-}p{\star}x{\ll}{-}{-}p{*}{-}q) = t\ n{*}{-}p{*}{-}q$
using *2* **by** (*smt assms*(*1*,*3*) *leq-def pSum-test-nat pre-closed pre-sub-distr sub-assoc sub-comm sub-mult-closed transitive while-pre-then*)
thus $t\ n{*}{-}p{*}{-}q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
using *2* **by** (*smt* (*z3*) *pre-closed tests-dual.sub-sup-closed tests-dual.upper-bound-right*)
qed
qed
hence $-q{*}{-}p \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
by (*smt* (*verit*, *del-insts*) *assms*(*1*,*2*) *leq-def nat-test-def pre-closed tests-dual.sub-associative tests-dual.sub-commutative sub-mult-closed*)
thus *?thesis*
using *1* **by** (*smt* (*z3*) *pre-closed tests-dual.sba-dual.inf-less-eq-cases tests-dual.sub-commutative tests-dual.sub-sup-closed*)
qed

lemma *nat-test-pre-1*:
assumes *nat-test t s*
and $-r \le s$
and $-r \le -q$
and $\forall\, n\ .\ t\ n{*}{-}p{*}{-}q \le x{\ll}pSum\ t\ n{*}{-}q$
shows $-r \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
proof $-$
let $?qs = -q{*}s$
have *1*: $-r \le\ ?qs$
by (*metis assms*(*1*$-$*3*) *nat-test-def tests-dual.least-upper-bound*)
have $\forall\, n\ .\ t\ n{*}{-}p{*}?qs \le x{\ll}pSum\ t\ n{*}?qs$
proof
fix n
have *2*: $pSum\ t\ n \le s$
by (*simp add*: *assms*(*1*) *pSum-below-sum*)

have $t\ n = t\ n * s$
by (*metis assms*(*1*) *nat-test-def tests-dual.sba-dual.less-eq-inf*)
hence $t\ n{*}{-}p{*}?qs = t\ n{*}{-}p{*}{-}q$
by (*smt* (*verit*, *ccfv-threshold*) *assms*(*1*) *nat-test-def tests-dual.sub-sup-closed tests-dual.sub-associative tests-dual.sub-commutative*)
also have $t\ n{*}{-}p{*}{-}q \le x«pSum\ t\ n{*}{-}q$
by (*simp add*: *assms*(*4*))
also have $x«pSum\ t\ n{*}{-}q = x«pSum\ t\ n{*}?qs$
using *2* **by** (*smt* (*verit*, *ccfv-SIG*) *assms*(*1*) *leq-def nat-test-def pSum-test-nat tests-dual.sub-associative tests-dual.sub-commutative*)
finally show $t\ n{*}{-}p{*}?qs \le x«pSum\ t\ n{*}?qs$
.
qed
hence *3*: $?qs \le -p{\star}x«{-}{-}p{*}?qs$
by (*smt* (*verit*, *ccfv-threshold*) *assms*(*1*) *tests-dual.upper-bound-left tests-dual.upper-bound-right nat-test-def nat-test-pre pSum-test-nat pre-closed tests-dual.sub-associative sub-mult-closed transitive*)
have $-p{\star}x«{-}{-}p{*}?qs \le -p{\star}x«{-}{-}p{*}{-}q$
by (*metis assms*(*1*) *nat-test-def pre-lower-bound-left tests-dual.sub-sup-closed while-post*)
thus *?thesis*
using *1 3* **by** (*smt* (*verit*, *del-insts*) *leq-def tests-dual.sub-associative assms*(*1*) *nat-test-def pre-closed sub-mult-closed*)
qed

lemma *nat-test-pre-2*:
assumes *nat-test t s*
and $-r \le s$
and $\forall n\ .\ t\ n{*}{-}p \le x«pSum\ t\ n$
shows $-r \le -p{\star}x«1$
proof −
have *1*: $-r \le -p{\star}x«{-}{-}p{*}s$
by (*smt* (*verit*, *ccfv-threshold*) *assms leq-def nat-test-def nat-test-pre-1 pSum-below-sum pSum-test-nat tests-dual.sub-associative tests-dual.sub-commutative*)
have $-p{\star}x«{-}{-}p{*}s \le -p{\star}x«1$
by (*metis assms*(*1*) *nat-test-def pre-below-pre-one while-post*)
thus *?thesis*
using *1* **by** (*smt* (*verit*) *assms*(*1*) *nat-test-def pre-closed tests-dual.sba-dual.top-double-complement while-post tests-dual.transitive*)
qed

lemma *nat-test-pre-3*:
assumes *nat-test t s*
and $-q \le s$
and $\forall n\ .\ t\ n{*}{-}p{*}{-}q \le x«pSum\ t\ n{*}{-}q$
shows $-q \le -p{\star}x«1$
proof −
have $-p{\star}x«{-}{-}p{*}{-}q \le -p{\star}x«1$

by (*metis pre-below-pre-one sub-mult-closed*)
thus *?thesis*
by (*smt* (*verit*, *ccfv-threshold*) *assms pre-closed tests-dual.sba-dual.top-double-complement tests-dual.sba-dual.transitive tests-dual.sub-sup-closed nat-test-pre*)
qed

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

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

end

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

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

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

lemma *test-expression-test*:
$p \in$ *Test-expression* $\Longrightarrow p = --p$
apply (*induct rule*: *Test-expression.induct*)
apply (*simp add*: *atomic-test-test*)
apply *simp*
by (*metis tests-dual.sub-sup-closed*)

lemma *disj-test*:
$p \in$ *Test-expression* $\Longrightarrow q \in$ *Test-expression* $\Longrightarrow p \sqcup q \in$ *Test-expression*
by (*smt conj-test neg-test tests-dual.sub-inf-def test-expression-test*)

lemma *zero-test-expression*:
bot $\in$ *Test-expression*
by (*simp add*: *Test-expression.atom-test zero-atomic-test*)

lemma *one-test-expression*:
$1 \in$ *Test-expression*

using *Test-expression.simps tests-dual.sba-dual.one-def zero-test-expression* **by** *blast*

lemma *pSum-test-expression*:
$(\forall\, n\ .\ t\ n \in \mathit{Test\text{-}expression}) \Longrightarrow \mathit{pSum}\ t\ m \in \mathit{Test\text{-}expression}$
apply (*induct m*)
apply (*simp add*: *zero-test-expression*)
by (*simp add*: *disj-test*)

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

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

lemma *power-while-program*:
$x \in \mathit{While\text{-}program} \Longrightarrow x\hat{\ }m \in \mathit{While\text{-}program}$
apply (*induct m*)
apply (*simp add*: *one-while-program*)
by (*simp add*: *While-program.seq-prog*)

inductive-set *Pre-expression* :: *'a set*
where *test-pre*: $p \in \mathit{Test\text{-}expression} \Longrightarrow p \in \mathit{Pre\text{-}expression}$
| *neg-pre*: $p \in \mathit{Pre\text{-}expression} \Longrightarrow -p \in \mathit{Pre\text{-}expression}$
| *conj-pre*: $p \in \mathit{Pre\text{-}expression} \Longrightarrow q \in \mathit{Pre\text{-}expression} \Longrightarrow p{*}q \in \mathit{Pre\text{-}expression}$
| *pre-pre*: $p \in \mathit{Pre\text{-}expression} \Longrightarrow x \in \mathit{While\text{-}program} \Longrightarrow x{\ll}p \in \mathit{Pre\text{-}expression}$

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

lemma *disj-pre*:
$p \in \mathit{Pre\text{-}expression} \Longrightarrow q \in \mathit{Pre\text{-}expression} \Longrightarrow p{\sqcup}q \in \mathit{Pre\text{-}expression}$
by (*smt conj-pre neg-pre tests-dual.sub-inf-def pre-expression-test*)

lemma *zero-pre-expression*:

bot ∈ *Pre-expression*
by (*simp add*: *Pre-expression.test-pre zero-test-expression*)

lemma *one-pre-expression*:
1 ∈ *Pre-expression*
by (*simp add*: *Pre-expression.test-pre one-test-expression*)

lemma *pSum-pre-expression*:
$(\forall n\ .\ t\ n \in$ *Pre-expression*$) \Longrightarrow$ *pSum t m* ∈ *Pre-expression*
apply (*induct m*)
apply (*simp add*: *zero-pre-expression*)
by (*simp add*: *disj-pre*)

lemma *aL-pre-expression*:
aL ∈ *Pre-expression*
by (*simp add*: *Pre-expression.pre-pre While-program.while-prog aL-def one-pre-expression one-test-expression one-while-program*)

end

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

definition *tfun* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where *tfun p x q r* ≡ $p \sqcup (x«q*r)$

lemma *tfun-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow$ *tfun p x q r* $= --$*tfun p x q r*
by (*smt tfun-def sub-mult-closed pre-closed tests-dual.inf-closed*)

lemma *tfun-pre-expression*:
x ∈ *While-program* ⟹ *p* ∈ *Pre-expression* ⟹ *q* ∈ *Pre-expression* ⟹ *r* ∈ *Pre-expression* ⟹ *tfun p x q r* ∈ *Pre-expression*
by (*simp add*: *Pre-expression.conj-pre Pre-expression.pre-pre disj-pre tfun-def*)

lemma *tfun-iso*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s = --s \Longrightarrow r \le s \Longrightarrow$ *tfun p x q r* ≤ *tfun p x q s*
by (*smt tfun-def tests-dual.sub-sup-right-isotone pre-iso sub-mult-closed tests-dual.sub-inf-right-isotone pre-closed*)

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

lemma *tseq-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow$ *tseq p x q r m* $= --$*tseq p x q r m*
apply (*induct m*)
apply (*smt tseq-def tfun-test power-zero-id id-def*)
by (*metis tseq-def tfun-test power-succ-unfold-ext*)

lemma *tseq-test-seq*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow$ *test-seq* (*tseq* $p\ x\ q\ r$)
using *test-seq-def tseq-test* **by** *auto*

lemma *tseq-pre-expression*:
$x \in$ *While-program* $\Longrightarrow p \in$ *Pre-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow r \in$ *Pre-expression* $\Longrightarrow$ *tseq* $p\ x\ q\ r\ m \in$ *Pre-expression*
apply (*induct* m)
apply (*smt tseq-def id-def power-zero-id*)
by (*smt tseq-def power-succ-unfold-ext tfun-pre-expression*)

definition *tsum* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where *tsum* $p\ x\ q\ r \equiv$ *Sum* (*tseq* $p\ x\ q\ r$)

lemma *tsum-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow$ *tsum* $p\ x\ q\ r = --$*tsum* $p\ x\ q\ r$
using *Sum-test tseq-test-seq tsum-def* **by** *auto*

lemma *t-fun-test*:
$q = --q \Longrightarrow$ *tfun* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL)) = --$*tfun* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL))$
by (*metis aL-test pre-closed tests-dual.sba-dual.double-negation tfun-def tfun-test*)

lemma *t-fun-pre-expression*:
$x \in$ *While-program* $\Longrightarrow p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow$ *tfun* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL)) \in$ *Pre-expression*
by (*simp add*: *Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre Pre-expression.test-pre While-program.while-prog aL-pre-expression disj-pre tfun-pre-expression*)

lemma *t-seq-test*:
$q = --q \Longrightarrow$ *tseq* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL))\ m = --$*tseq* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL))\ m$
by (*metis aL-test pre-closed tests-dual.sba-dual.double-negation tfun-def tfun-test tseq-test*)

lemma *t-seq-test-seq*:
$q = --q \Longrightarrow$ *test-seq* (*tseq* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL)))$
using *test-seq-def t-seq-test* **by** *auto*

lemma *t-seq-pre-expression*:
$x \in$ *While-program* $\Longrightarrow p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow$ *tseq* $(-p)\ x\ (p\star x« q)\ (-p\sqcup(x«(p\star x« q)*aL))\ m \in$ *Pre-expression*
using *Pre-expression.pre-pre Pre-expression.test-pre Test-expression.neg-test While-program.while-prog aL-pre-expression tfun-def tfun-pre-expression tseq-pre-expression* **by** *auto*

lemma *t-sum-test*:
$q = --q \Longrightarrow tsum\ (-p)\ x\ (p\star x«q)\ (-p\sqcup(x«(p\star x«q)*aL)) = --tsum\ (-p)\ x\ (p\star x«q)\ (-p\sqcup(x«(p\star x«q)*aL))$
using *Sum-test t-seq-test-seq tsum-def* **by** *auto*

definition *tfun2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where $tfun2\ p\ q\ x\ r\ s \equiv p \sqcup q*(x«r*s)$

lemma *tfun2-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s = --s \Longrightarrow tfun2\ p\ q\ x\ r\ s = --tfun2\ p\ q\ x\ r\ s$
by (*smt tfun2-def sub-mult-closed pre-closed tests-dual.inf-closed*)

lemma *tfun2-pre-expression*:
$x \in While\text{-}program \Longrightarrow p \in Pre\text{-}expression \Longrightarrow q \in Pre\text{-}expression \Longrightarrow r \in Pre\text{-}expression \Longrightarrow s \in Pre\text{-}expression \Longrightarrow tfun2\ p\ q\ x\ r\ s \in Pre\text{-}expression$
by (*simp add*: *Pre-expression.conj-pre Pre-expression.pre-pre disj-pre tfun2-def*)

lemma *tfun2-iso*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s1 = --s1 \Longrightarrow s2 = --s2 \Longrightarrow s1 \le s2 \Longrightarrow tfun2\ p\ q\ x\ r\ s1 \le tfun2\ p\ q\ x\ r\ s2$
by (*smt tfun2-def tests-dual.sub-inf-right-isotone pre-iso sub-mult-closed tests-dual.sub-sup-right-isotone pre-closed*)

definition *tseq2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow nat \Rightarrow 'a$
where $tseq2\ p\ q\ x\ r\ s\ m \equiv (tfun2\ p\ q\ x\ r\ \widehat{}\ m)\ s$

lemma *tseq2-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s = --s \Longrightarrow tseq2\ p\ q\ x\ r\ s\ m = --tseq2\ p\ q\ x\ r\ s\ m$
apply (*induct m*)
apply (*smt tseq2-def power-zero-id id-def*)
by (*smt tseq2-def tfun2-test power-succ-unfold-ext*)

lemma *tseq2-test-seq*:
$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s = --s \Longrightarrow test\text{-}seq\ (tseq2\ p\ q\ x\ r\ s)$
using *test-seq-def tseq2-test* **by** *force*

lemma *tseq2-pre-expression*:
$x \in While\text{-}program \Longrightarrow p \in Pre\text{-}expression \Longrightarrow q \in Pre\text{-}expression \Longrightarrow r \in Pre\text{-}expression \Longrightarrow s \in Pre\text{-}expression \Longrightarrow tseq2\ p\ q\ x\ r\ s\ m \in Pre\text{-}expression$
apply (*induct m*)
apply (*smt tseq2-def id-def power-zero-id*)
by (*smt tseq2-def power-succ-unfold-ext tfun2-pre-expression*)

definition *tsum2* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
where $tsum2\ p\ q\ x\ r\ s \equiv Sum\ (tseq2\ p\ q\ x\ r\ s)$

lemma *tsum2-test*:

$p = --p \Longrightarrow q = --q \Longrightarrow r = --r \Longrightarrow s = --s \Longrightarrow tsum2\ p\ q\ x\ r\ s = --tsum2\ p\ q\ x\ r\ s$
using *Sum-test tseq2-test-seq tsum2-def* **by** *force*

lemma *t-fun2-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow tfun2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL)) = --tfun2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))$
by (*smt* (*z3*) *aL-test pre-closed tests-dual.sub-sup-closed tfun2-def tfun2-test*)

lemma *t-fun2-pre-expression*:
$x \in While\text{-}program \Longrightarrow p \in Test\text{-}expression \Longrightarrow q \in Pre\text{-}expression \Longrightarrow tfun2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL)) \in Pre\text{-}expression$
by (*simp add*: *Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre Pre-expression.test-pre While-program.while-prog aL-pre-expression disj-pre tfun2-pre-expression*)

lemma *t-seq2-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))\ m = --tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))\ m$
by (*smt* (*z3*) *aL-test pre-closed tests-dual.sub-sup-closed tfun2-def tfun2-test tseq2-test*)

lemma *t-seq2-test-seq*:
$p = --p \Longrightarrow q = --q \Longrightarrow test\text{-}seq\ (tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL)))$
using *test-seq-def t-seq2-test* **by** *auto*

lemma *t-seq2-pre-expression*:
$x \in While\text{-}program \Longrightarrow p \in Test\text{-}expression \Longrightarrow q \in Pre\text{-}expression \Longrightarrow tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))\ m \in Pre\text{-}expression$
by (*simp add*: *Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre Pre-expression.test-pre While-program.while-prog aL-pre-expression disj-pre tseq2-pre-expression*)

lemma *t-sum2-test*:
$p = --p \Longrightarrow q = --q \Longrightarrow tsum2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL)) = --tsum2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))$
using *Sum-test t-seq2-test-seq tsum2-def* **by** *auto*

lemma *t-seq2-below-t-seq*:
assumes $p \in Test\text{-}expression$
and $q \in Pre\text{-}expression$
shows $tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))\ m \le tseq\ (-p)\ x\ (p\star x«q)\ (-p\sqcup(x«(p\star x«q)*aL))\ m$
proof −
let $?t2 = tseq2\ (-p*q)\ p\ x\ (p\star x«q)\ (-p*q\sqcup p*(x«(p\star x«q)*aL))$

let *?t* = *tseq* (−*p*) *x* (*p*⋆*x*«*q*) (−*p*⊔(*x*«(*p*⋆*x*«*q*)∗*aL*))
show *?thesis*
proof (*induct m*)
case *0*
show *?t2 0* ≤ *?t 0*
by (*smt assms aL-test id-def tests-dual.upper-bound-left tests-dual.upper-bound-right tests-dual.inf-isotone power-zero-id pre-closed pre-expression-test sub-mult-closed test-pre tseq2-def tseq-def*)
next
fix *m*
assume *?t2 m* ≤ *?t m*
hence *1*: *?t2* (*Suc m*) ≤ *tfun2* (− *p* ∗ *q*) *p x* (*p* ⋆ *x* « *q*) (*?t m*)
by (*smt assms power-succ-unfold-ext pre-closed pre-expression-test sub-mult-closed t-seq2-test t-seq-test test-pre tfun2-iso tseq2-def*)
have ... ≤ *?t* (*Suc m*)
by (*smt assms tests-dual.upper-bound-left tests-dual.upper-bound-right tests-dual.inf-isotone power-succ-unfold-ext pre-closed pre-expression-test sub-mult-closed t-seq-test test-pre tfun2-def tfun-def tseq-def*)
thus *?t2* (*Suc m*) ≤ *?t* (*Suc m*)
using *1* **by** (*smt* (*verit*, *del-insts*) *assms pre-closed pre-expression-test test-expression-test tests-dual.sba-dual.transitive tests-dual.sub-sup-closed t-seq2-test t-seq-test tfun2-test*)
qed
qed

lemma *t-seq2-below-t-sum*:
p ∈ *Test-expression* ⟹ *q* ∈ *Pre-expression* ⟹ *x* ∈ *While-program* ⟹ *tseq2* (−*p*∗*q*) *p x* (*p*⋆*x*«*q*) (−*p*∗*q*⊔*p*∗(*x*«(*p*⋆*x*«*q*)∗*aL*)) *m* ≤ *tsum* (−*p*) *x* (*p*⋆*x*«*q*) (−*p*⊔(*x*«(*p*⋆*x*«*q*)∗*aL*))
by (*smt* (*verit*, *del-insts*) *Sum-upper pre-expression-test t-seq2-below-t-seq t-seq2-test t-seq-test t-sum-test test-pre test-seq-def tsum-def leq-def tests-dual.sub-associative*)

lemma *t-sum2-below-t-sum*:
p ∈ *Test-expression* ⟹ *q* ∈ *Pre-expression* ⟹ *x* ∈ *While-program* ⟹ *tsum2* (−*p*∗*q*) *p x* (*p*⋆*x*«*q*) (−*p*∗*q*⊔*p*∗(*x*«(*p*⋆*x*«*q*)∗*aL*)) ≤ *tsum* (−*p*) *x* (*p*⋆*x*«*q*) (−*p*⊔(*x*«(*p*⋆*x*«*q*)∗*aL*))
by (*smt Sum-least pre-expression-test t-seq2-below-t-sum t-seq2-test t-sum-test test-pre test-seq-def tsum2-def*)

lemma *t-seq2-below-w*:
p ∈ *Test-expression* ⟹ *q* ∈ *Pre-expression* ⟹ *x* ∈ *While-program* ⟹ *tseq2* (−*p*∗*q*) *p x* (*p*⋆*x*«*q*) (−*p*∗*q*⊔*p*∗(*x*«(*p*⋆*x*«*q*)∗*aL*)) *m* ≤ *p*⋆*x*«*q*
apply (*cases m*)
apply (*smt aL-test id-def tests-dual.upper-bound-left tests-dual.sub-sup-right-isotone tests-dual.inf-commutative tests-dual.sub-inf-right-isotone power-zero-id pre-closed pre-expression-test pre-iso sub-mult-closed test-pre tseq2-def while-pre*)
by (*smt tseq2-def power-succ-unfold-ext tests-dual.upper-bound-left*

tests-dual.sub-sup-right-isotone tests-dual.inf-commutative tests-dual.sub-inf-right-isotone pre-closed pre-expression-test pre-iso sub-mult-closed t-seq2-test test-pre tseq2-def while-pre tfun2-def)

lemma *t-sum2-below-w*:
$p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow x \in$ *While-program* $\Longrightarrow$ *tsum2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL)) \leq p\star x«q$
by (*smt Sum-least pre-closed pre-expression-test t-seq2-below-w t-seq2-test-seq test-pre tsum2-def*)

lemma *t-sum2-w*:
assumes $aL = 1$
and $p \in$ *Test-expression*
and $q \in$ *Pre-expression*
and $x \in$ *While-program*
shows *tsum2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL)) = p\star x«q$
proof $-$
let $?w = p\star x«q$
let $?s = -p*q\sqcup p*(x«?w*aL)$
have $?w =$ *tseq2* $(-p*q)$ p x $?w$ $?s$ 0
by (*smt assms*$(1-3)$ *tests-dual.sup-right-unit id-def tests-dual.inf-commutative power-zero-id pre-closed pre-expression-test sub-mult-closed test-expression-test tseq2-def while-pre*)
hence $?w \leq$ *tsum2* $(-p*q)$ p x $?w$ $?s$
by (*smt assms*$(2,3)$ *Sum-upper pre-expression-test t-seq2-test-seq test-pre tsum2-def*)
thus *?thesis*
by (*smt assms*$(2-4)$ *tests-dual.antisymmetric pre-closed pre-expression-test t-sum2-test t-sum2-below-w test-pre*)
qed

inductive *derived-hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow$ *bool* (‹- ⦇ - ⦈ -› [$54,54,54$] 53)
where *atom-trip*: $p \in$ *Pre-expression* $\Longrightarrow x \in$ *Atomic-program* $\Longrightarrow x«p⦇x⦈p$
| *seq-trip*: $p⦇x⦈q \wedge q⦇y⦈r \Longrightarrow p⦇x*y⦈r$
| *cond-trip*: $p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow p*q⦇x⦈r \wedge -p*q⦇y⦈r \Longrightarrow q⦇x\lhd p\rhd y⦈r$
| *while-trip*: $p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow$ *test-seq* $t \wedge q \leq$ *Sum* $t \Longrightarrow t\ 0*p*q⦇x⦈aL*q \Longrightarrow (\forall\, n>0\ .\ t\ n*p*q⦇x⦈pSum\ t\ n*q) \Longrightarrow q⦇p\star x⦈-p*q$
| *cons-trip*: $p \in$ *Pre-expression* $\Longrightarrow s \in$ *Pre-expression* $\Longrightarrow p \leq q \wedge q⦇x⦈r \Longrightarrow r \leq s \Longrightarrow p⦇x⦈s$

lemma *derived-type*:
$p⦇x⦈q \Longrightarrow p \in$ *Pre-expression* $\wedge\ q \in$ *Pre-expression* $\wedge\ x \in$ *While-program*
apply (*induct rule*: *derived-hoare-triple.induct*)
apply (*simp add*: *Pre-expression.pre-pre While-program.atom-prog*)
using *While-program.seq-prog* **apply** *blast*
using *While-program.cond-prog* **apply** *blast*
using *Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.test-pre*

While-program.while-prog **apply** *simp*
  by *blast*

lemma *cons-pre-trip*:
  $p \in Pre\text{-}expression \Longrightarrow q(\!|y|\!)r \Longrightarrow p{*}q(\!|y|\!)r$
  by (*metis cons-trip derived-type Pre-expression.conj-pre pre-expression-test tests-dual.sba-dual.reflexive tests-dual.upper-bound-right*)

lemma *cons-post-trip*:
  $q \in Pre\text{-}expression \Longrightarrow r \in Pre\text{-}expression \Longrightarrow p(\!|y|\!)q{*}r \longrightarrow p(\!|y|\!)r$
  by (*metis cons-trip derived-type pre-expression-test tests-dual.sba-dual.reflexive tests-dual.upper-bound-right*)

definition *valid-hoare-triple* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$ (‹- ⟨ - ⟩ -› [54,54,54] 53)
  where $p\langle x\rangle q \equiv (p \in Pre\text{-}expression \wedge q \in Pre\text{-}expression \wedge x \in While\text{-}program \wedge p \leq x«q)$

end

class *hoare-calculus-sound* = *hoare-calculus* +
  assumes *while-soundness*: $-p{*}-q \leq x«-q \longrightarrow aL{*}-q \leq -p{\star}x«-q$
begin

lemma *while-soundness-0*:
  $-p{*}-q \leq x«-q \Longrightarrow -q{*}aL \leq -p{\star}x«--p{*}-q$
  by (*smt while-soundness aL-test sub-comm while-post*)

lemma *while-soundness-1*:
  assumes *test-seq t*
    and $-q \leq Sum\ t$
    and $t\ 0{*}-p{*}-q \leq x«aL{*}-q$
    and $\forall n{>}0\ .\ t\ n{*}-p{*}-q \leq x«pSum\ t\ n{*}-q$
   shows $-q \leq -p{\star}x«--p{*}-q$
proof −
  have $\forall n\ .\ t\ n{*}-p{*}-q \leq x«-q$
  proof
   fix *n*
   show $t\ n{*}-p{*}-q \leq x«-q$
   proof (*cases n*)
    case *0*
    thus *?thesis*
     by (*smt* (*z3*) *assms*(*1*) *assms*(*3*) *aL-test leq-def pre-closed pre-lower-bound-right test-seq-def tests-dual.sub-associative tests-dual.sub-sup-closed*)
   next
    case (*Suc m*)
    hence *1*: $t\ n{*}-p{*}-q \leq x«pSum\ t\ n{*}-q$
     using *assms*(*4*) **by** *blast*
    have $x«pSum\ t\ n{*}-q \leq x«-q$

by (*metis assms*(*1*) *pSum-test pre-lower-bound-right*)
thus *?thesis*
using *1* **by** (*smt* (*verit*, *del-insts*) *assms*(*1*) *pSum-test pre-closed sub-mult-closed test-seq-def leq-def tests-dual.sub-associative*)
qed
qed
hence *2*: $-p*-q \leq x«-q$
by (*smt assms*(*1*,*2*) *Sum-test leq-def mult-right-dist-Sum pre-closed sub-assoc sub-comm sub-mult-closed test-seq-def*)
have $\forall n . t\ n*-q \leq -p\star x«--p*-q \wedge pSum\ t\ n*-q \leq -p\star x«--p*-q$
proof
fix *n*
show $t\ n*-q \leq -p\star x«--p*-q \wedge pSum\ t\ n*-q \leq -p\star x«--p*-q$
proof (*induct n rule*: *nat-less-induct*)
fix *n*
assume *3*: $\forall m<n . t\ m*-q \leq -p\star x«--p*-q \wedge pSum\ t\ m*-q \leq -p\star x«--p*-q$
have *4*: $pSum\ t\ n*-q \leq -p\star x«--p*-q$
proof (*cases n*)
case *0*
thus *?thesis*
by (*metis pSum.simps*(*1*) *pre-closed sub-mult-closed tests-dual.top-greatest tests-dual.sba-dual.less-eq-inf tests-dual.top-double-complement*)
next
case (*Suc m*)
hence $pSum\ t\ n*-q = (pSum\ t\ m \sqcup t\ m)*-q$
by *simp*
also have ... $= pSum\ t\ m*-q \sqcup t\ m*-q$
by (*metis* (*full-types*) *assms*(*1*) *pSum-test test-seq-def tests-dual.sup-right-dist-inf*)
also have ... $\leq -p\star x«--p*-q$
proof −
have $pSum\ t\ m*-q = --(pSum\ t\ m*-q) \wedge t\ m*-q = --(t\ m*-q) \wedge -p\star x«--p*-q = --(-p\star x«--p*-q)$
apply (*intro conjI*)
apply (*metis assms*(*1*) *pSum-test tests-dual.sub-sup-closed*)
apply (*metis assms*(*1*) *test-seq-def tests-dual.sub-sup-closed*)
by (*metis pre-closed tests-dual.sub-sup-closed*)
thus *?thesis*
using *3* **by** (*smt* (*z3*) *lessI Suc tests-dual.greatest-lower-bound sub-mult-closed*)
qed
finally show *?thesis*
.
qed
hence *5*: $x«pSum\ t\ n*-q \leq x«-p\star x«--p*-q$
by (*smt assms pSum-test pre-closed pre-iso sub-mult-closed*)
have *6*: $-p*(t\ n*-q) \leq -p*(-p\star x«--p*-q)$
proof (*cases n*)

```
        case 0
        thus ?thesis
          using 2 by (smt assms(1,3) aL-test leq-def tests-dual.sup-idempotent
tests-dual.sub-sup-right-isotone pre-closed pre-lower-bound-left sub-assoc
sub-comm sub-mult-closed test-seq-def transitive while-pre-then while-soundness-0)
      next
        case (Suc m)
        hence −p∗(t n∗−q) ≤ x«pSum t n∗−q
          by (smt assms(1,4) test-seq-def tests-dual.sub-associative
tests-dual.sub-commutative zero-less-Suc)
        hence −p∗(t n∗−q) ≤ x«−p⋆x«−−p∗−q
          using 5 by (smt assms(1) tests-dual.least-upper-bound pSum-test
pre-closed sub-mult-closed test-seq-def leq-def)
        hence −p∗(t n∗−q) ≤ −p∗(x«−p⋆x«−−p∗−q)
          by (smt assms(1) tests-dual.upper-bound-left pre-closed sub-mult-closed
test-seq-def leq-def tests-dual.sub-associative)
        thus ?thesis
          using while-post while-pre-then by auto
      qed
      have −−p∗(t n∗−q) ≤ −−p∗(−p⋆x«−−p∗−q)
        by (smt assms(1) leq-def tests-dual.upper-bound-right sub-assoc sub-comm
sub-mult-closed test-seq-def while-pre-else)
      thus t n∗−q ≤ −p⋆x«−−p∗−q ∧ pSum t n∗−q ≤ −p⋆x«−−p∗−q
        using 4 6 by (smt assms(1) tests-dual.sup-less-eq-cases-2 pre-closed
sub-mult-closed test-seq-def)
    qed
  qed
  thus ?thesis
    by (smt assms(1,2) Sum-test leq-def mult-right-dist-Sum pre-closed sub-comm
sub-mult-closed)
qed

lemma while-soundness-2:
  assumes test-seq t
      and −r ≤ Sum t
      and ∀n . t n∗−p ≤ x«pSum t n
    shows −r ≤ −p⋆x«1
proof −
  have 1: ∀n>0 . t n∗−p∗Sum t ≤ x«pSum t n∗Sum t
    by (smt (z3) assms(1,3) Sum-test Sum-upper leq-def pSum-below-Sum
pSum-test test-seq-def tests-dual.sub-associative tests-dual.sub-commutative)
  have 2: t 0∗−p∗Sum t ≤ x«bot
    by (smt assms(1,3) Sum-test Sum-upper leq-def sub-assoc sub-comm
test-seq-def pSum.simps(1))
  have x«bot ≤ x«aL∗Sum t
    by (smt assms(1) Sum-test aL-test pre-iso sub-mult-closed
tests-dual.top-double-complement tests-dual.top-greatest)
  hence t 0∗−p∗Sum t ≤ x«aL∗Sum t
    using 2 by (smt (z3) assms(1) Sum-test aL-test leq-def pSum.simps(1)
```

pSum-test pre-closed test-seq-def tests-dual.sub-associative tests-dual.sub-sup-closed)
hence *3*: $Sum\ t \leq -p\star x«--p*Sum\ t$
using *1* **by** (*smt* (*verit*, *del-insts*) *assms*(*1*) *Sum-test tests-dual.sba-dual.one-def tests-dual.sup-right-unit tests-dual.upper-bound-left while-soundness-1*)
have $-p\star x«--p*Sum\ t \leq -p\star x«1$
by (*metis assms*(*1*) *Sum-test pre-below-pre-one tests-dual.sub-sup-closed*)
hence $Sum\ t \leq -p\star x«1$
using *3* **by** (*smt* (*z3*) *assms*(*1*) *Sum-test pre-closed tests-dual.sba-dual.one-def while-post tests-dual.transitive*)
thus *?thesis*
by (*smt* (*z3*) *assms*(*1*,*2*) *Sum-test pre-closed tests-dual.sba-dual.one-def tests-dual.transitive*)
qed

theorem *soundness*:
$p(\!|x|\!)q \Longrightarrow p\langle x\rangle q$
apply (*induct rule*: *derived-hoare-triple.induct*)
apply (*metis Pre-expression.pre-pre While-program.atom-prog pre-expression-test tests-dual.sba-dual.reflexive valid-hoare-triple-def*)
apply (*metis valid-hoare-triple-def pre-expression-test pre-compose While-program.seq-prog*)
apply (*metis valid-hoare-triple-def ite-import-mult pre-expression-test cond-prog test-pre*)
apply (*smt* (*verit*, *del-insts*) *Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.test-pre While-program.while-prog pre-expression-test valid-hoare-triple-def while-soundness-1*)
by (*metis pre-expression-test pre-iso pre-pre tests-dual.sba-dual.transitive valid-hoare-triple-def*)

end

class *hoare-calculus-pre-complete* = *hoare-calculus* +
assumes *aL-pre-import*: $(x«-q)*aL \leq x«-q*aL$
assumes *pre-right-dist-Sum*: $x \in While\text{-}program \wedge ascending\text{-}chain\ t \wedge test\text{-}seq\ t \longrightarrow x«Sum\ t = Sum\ (\lambda n\ .\ x«t\ n)$
begin

lemma *aL-pre-import-equal*:
$(x«-q)*aL = (x«-q*aL)*aL$
proof −
have *1*: $(x«-q)*aL \leq (x«-q*aL)*aL$
by (*smt* (*z3*) *aL-pre-import aL-test pre-closed tests-dual.sub-sup-closed tests-dual.least-upper-bound tests-dual.upper-bound-right*)
have $(x«-q*aL)*aL \leq (x«-q)*aL$
by (*smt* (*verit*, *ccfv-threshold*) *aL-test pre-closed pre-lower-bound-left tests-dual.sba-dual.inf-isotone tests-dual.sba-dual.reflexive tests-dual.sub-sup-closed*)

thus *?thesis*
using *1* **by** (*smt* (*z3*) *tests-dual.antisymmetric aL-test pre-closed tests-dual.sub-sup-closed*)
qed

lemma *aL-pre-below-t-seq2*:
assumes $p \in$ *Test-expression*
and $q \in$ *Pre-expression*
shows $(p\star x«q)*aL \leq$ *tseq2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL))$ *0*
proof (*unfold tseq2-def power-zero-id id-def while-pre*)
have $(p\star x«q)*aL = (p*(x«p\star x«q) \sqcup -p*q)*aL$
by (*metis assms while-pre test-pre pre-expression-test*)
also have ... $= p*(x«p\star x«q)*aL \sqcup -p*q*aL$
by (*smt* (*z3*) *assms aL-test tests-dual.sup-right-dist-inf pre-closed pre-expression-test sub-mult-closed test-pre*)
also have ... $= p*((x«p\star x«q)*aL) \sqcup -p*q*aL$
by (*smt assms aL-test pre-closed pre-expression-test test-pre sub-assoc*)
also have ... $\leq p*(x«(p\star x«q)*aL) \sqcup -p*q$
proof −
have *1*: $(x«p\star x«q)*aL \leq x«(p\star x«q)*aL$
by (*metis assms*(*2*) *pre-closed pre-expression-test aL-pre-import*)
have $-p*q*aL \leq -p*q$
by (*metis assms*(*2*) *aL-test pre-expression-test tests-dual.sub-sup-closed tests-dual.upper-bound-left*)
thus *?thesis*
using *1* **by** (*smt assms aL-test pre-closed pre-expression-test test-pre tests-dual.sub-sup-closed tests-dual.sub-sup-right-isotone tests-dual.inf-isotone*)
qed
also have ... $= -p*q \sqcup p*(x«(p\star x«q)*aL)$
by (*smt assms aL-test tests-dual.inf-commutative pre-closed pre-expression-test test-pre tests-dual.sub-sup-closed*)
finally show $(p\star x«q)*aL \leq -p*q \sqcup p*(x«(p\star x«q)*aL)$
.
qed

lemma *t-seq2-ascending*:
assumes $p \in$ *Test-expression*
and $q \in$ *Pre-expression*
and $x \in$ *While-program*
shows *tseq2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL))$ $m \leq$ *tseq2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL))$ (*Suc* m)
proof (*induct* m)
let $?w = p\star x«q$
let $?r = -p*q\sqcup p*(x«?w*aL)$
case *0*
have *1*: $?w*aL = --(?w*aL)$
by (*simp add*: *assms Pre-expression.conj-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression pre-expression-test*)
have *2*: $?r = --?r$

by (*simp add*: *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre Pre-expression.test-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test*)
have $?w{*}aL \leq ?r$
by (*metis aL-pre-below-t-seq2 assms*(*1*,*2*) *id-def tseq2-def power-zero-id*)
hence $?w{*}aL \leq ?w{*}?r$
using *1 2* **by** (*smt* (*verit*, *ccfv-threshold*) *assms Pre-expression.pre-pre While-program.while-prog aL-test pre-expression-test tests-dual.sub-associative tests-dual.sub-sup-right-isotone tests-dual.sba-dual.less-eq-inf tests-dual.sba-dual.reflexive*)
hence $x«?w{*}aL \leq x«(?w{*}?r)$
by (*smt* (*verit*, *ccfv-threshold*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test pre-iso test-pre*)
hence $p{*}(x«?w{*}aL) \leq p{*}(x«(?w{*}?r))$
by (*smt* (*z3*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test test-pre tests-dual.sub-sup-right-isotone*)
hence $?r \leq -p{*}q \sqcup p{*}(x«(?w{*}?r))$
by (*smt* (*verit*, *del-insts*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test test-pre tests-dual.sba-dual.reflexive tests-dual.inf-isotone*)
thus *?case*
by (*unfold tseq2-def power-zero-id power-succ-unfold-ext id-def tfun2-def*)
next
let $?w = p{\star}x«q$
let $?r = -p{*}q \sqcup p{*}(x«?w{*}aL)$
let $?t = tseq2\ (-p{*}q)\ p\ x\ ?w\ ?r$
case (*Suc m*)
hence $?w{*}?t\ m \leq ?w{*}?t\ (Suc\ m)$
by (*smt* (*z3*) *assms*(*1*,*2*) *pre-closed pre-expression-test t-seq2-test test-expression-test tests-dual.sub-sup-right-isotone*)
hence $x«?w{*}?t\ m \leq x«?w{*}?t\ (Suc\ m)$
by (*smt* (*z3*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test pre-iso test-pre tseq2-pre-expression*)
hence $p{*}(x«?w{*}?t\ m) \leq p{*}(x«?w{*}?t\ (Suc\ m))$
by (*smt* (*z3*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test test-pre tests-dual.sub-sup-right-isotone tseq2-pre-expression*)
hence $-p{*}q \sqcup p{*}(x«?w{*}?t\ m) \leq -p{*}q \sqcup p{*}(x«?w{*}?t\ (Suc\ m))$
by (*smt* (*z3*) *assms Pre-expression.conj-pre Pre-expression.neg-pre Pre-expression.pre-pre While-program.while-prog aL-pre-expression disj-pre pre-expression-test test-pre tests-dual.sba-dual.reflexive tests-dual.inf-isotone tseq2-pre-expression*)
thus *?case*
by (*smt tseq2-def power-succ-unfold-ext tfun2-def*)
qed

lemma *t-seq2-ascending-chain*:
$p \in$ *Test-expression* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow x \in$ *While-program* $\Longrightarrow$
ascending-chain (*tseq2* $(-p*q)$ p x $(p\star x«q)$ $(-p*q\sqcup p*(x«(p\star x«q)*aL)))$
by (*simp add*: *ord.ascending-chain-def t-seq2-ascending*)

end

class *hoare-calculus-complete* = *hoare-calculus-pre-complete* +
assumes *while-completeness*: $-p*(x«-q) \leq -q \longrightarrow -p\star x«-q \leq -q\sqcup aL$
begin

lemma *while-completeness-var*:
assumes $-p*(x«-q)\sqcup-r \leq -q$
shows $-p\star x«-r \leq -q\sqcup aL$
proof –
have *1*: $-p\star x«-q \leq -q\sqcup aL$
by (*smt assms pre-closed tests-dual.sub-sup-closed*
tests-dual.greatest-lower-bound while-completeness)
have $-p\star x«-r \leq -p\star x«-q$
by (*smt assms pre-closed tests-dual.sub-sup-closed*
tests-dual.greatest-lower-bound pre-iso)
thus *?thesis*
using *1* **by** (*smt* (*z3*) *aL-test pre-closed tests-dual.sba-dual.sub-sup-closed*
tests-dual.sba-dual.transitive)
qed

lemma *while-completeness-sum*:
assumes $p \in$ *Test-expression*
and $q \in$ *Pre-expression*
and $x \in$ *While-program*
shows $p\star x«q \leq$ *tsum* $(-p)$ x $(p\star x«q)$ $(-p\sqcup(x«(p\star x«q)*aL))$
proof –
let $?w = p\star x«q$
let $?r = -p*q\sqcup p*(x«?w*aL)$
let $?t =$ *tseq2* $(-p*q)$ p x $?w$ $?r$
let $?ts =$ *tsum2* $(-p*q)$ p x $?w$ $?r$
have *1*: $?w = --?w$
by (*metis assms*(*2*) *pre-expression-test pre-closed*)
have *2*: $?r = --?r$
by (*simp add*: *assms Pre-expression.conj-pre Pre-expression.neg-pre*
Pre-expression.pre-pre Pre-expression.test-pre While-program.while-prog
aL-pre-expression disj-pre pre-expression-test)
have *3*: $?ts = --?ts$
by (*meson assms*(*1*) *assms*(*2*) *pre-expression-test t-sum2-test*
test-expression-test)
have *4*: *test-seq* $?t$
by (*simp add*: *assms*(*1*) *assms*(*2*) *pre-expression-test t-seq2-test-seq*
test-expression-test)
have $-p*q \leq ?r$

by (smt (z3) assms(1,2) aL-test pre-closed pre-expression-test sub-mult-closed test-pre tests-dual.lower-bound-left)
hence 5: −p∗q ≤ ?ts
using 1 2 3 **by** (smt assms Sum-upper id-def tests-dual.sba-dual.transitive power-zero-id pre-expression-test sub-mult-closed test-pre tseq2-def tseq2-test-seq tsum2-def)
have ∀ n . p∗(x« ?t n) ≤ ?ts
proof (rule allI, unfold tsum2-def)
fix n
have 6: p∗(x« ?t n) ≤ ?t (Suc n)
using 4 **by** (smt assms leq-def power-succ-unfold-ext pre-closed pre-expression-test tests-dual.sub-commutative sub-mult-closed t-seq2-below-w test-pre test-seq-def tfun2-def tseq2-def tests-dual.lower-bound-right)
have ?t (Suc n) ≤ Sum ?t
using 4 Sum-upper **by** auto
thus p∗(x« ?t n) ≤ Sum ?t
using 3 4 6 **by** (smt assms(1) pre-closed pre-expression-test sub-mult-closed test-pre test-seq-def tests-dual.transitive tsum2-def)
qed
hence p∗(x« ?ts) ≤ ?ts
using 3 4 **by** (smt assms mult-left-dist-Sum pre-closed pre-right-dist-Sum t-seq2-ascending-chain test-expression-test test-seq-def tsum2-def)
hence p∗(x« ?ts)⊔−p∗q ≤ ?ts
using 3 5 **by** (smt assms(1,2) tests-dual.greatest-lower-bound pre-closed pre-expression-test sub-mult-closed test-pre)
hence ?w ≤ ?ts⊔aL
using 1 3 **by** (smt assms(1,2) pre-expression-test while-post sub-mult-closed t-sum2-below-t-sum t-sum-test test-pre transitive while-completeness-var)
hence ?w = ?w∗(?ts⊔aL)
using 1 3 **by** (smt aL-test tests-dual.sba-dual.less-eq-inf tests-dual.sba-dual.sub-sup-closed)
also have ... = ?w∗?ts⊔?w∗aL
using 1 3 **by** (smt aL-test tests-dual.sup-left-dist-inf)
also have ... ≤ ?ts⊔?t 0
using 1 3 4 **by** (smt (z3) assms(1,2) aL-pre-below-t-seq2 tests-dual.upper-bound-right aL-test test-seq-def tests-dual.sub-sup-closed tests-dual.inf-isotone)
also have ... = ?ts
using 3 4 **by** (smt Sum-upper tsum2-def test-seq-def tests-dual.less-eq-inf)
finally have ?w ≤ ?ts
.
thus ?thesis
using 1 3 **by** (metis assms t-sum2-below-t-sum t-sum2-below-w tests-dual.antisymmetric)
qed

lemma while-complete:
assumes p ∈ Test-expression
and q ∈ Pre-expression

and $x \in$ *While-program*
and $\forall\, r{\in}$*Pre-expression* . $x«r(\!|x|\!)r$
shows $p{\star}x«q(\!|p{\star}x|\!)q$
proof $-$
let $?w = p{\star}x«q$
let $?t = tseq\ (-p)\ x\ ?w\ (-p\sqcup(x«?w{*}aL))$
have *1*: $?w \in$ *Pre-expression*
by (*simp add*: *assms*(*1*$-$*3*) *Pre-expression.pre-pre While-program.while-prog*)
have *2*: *test-seq ?t*
by (*simp add*: *assms*(*2*) *pre-expression-test t-seq-test-seq*)
hence *3*: $?w \leq Sum\ ?t$
using *assms*(*1*$-$*3*) *tsum-def while-completeness-sum* **by** *auto*
have *4*: $p = --p$
by (*simp add*: *assms*(*1*) *test-expression-test*)
have $x«?w{*}aL = --(x«?w{*}aL)$
using *1* **by** (*simp add*: *assms*(*3*) *Pre-expression.conj-pre*
Pre-expression.pre-pre aL-pre-expression pre-expression-test)
hence *5*: $(-p\sqcup(x«?w{*}aL)){*}p = (x«?w{*}aL){*}p$
using *4* **by** (*metis tests-dual.sba-dual.inf-complement-intro*)
have $x«aL{*}?w(\!|x|\!)aL{*}?w$
using *1* **by** (*simp add*: *assms*(*4*) *Pre-expression.conj-pre aL-pre-expression*)
hence $x«?w{*}aL(\!|x|\!)aL{*}?w$
using *1* **by** (*metis aL-test pre-expression-test sub-comm*)
hence $(x«?w{*}aL){*}p{*}?w(\!|x|\!)aL{*}?w$
using *1* **by** (*smt* (*z3*) *assms*(*1*) *Pre-expression.conj-pre Pre-expression.test-pre*
derived-hoare-triple.cons-trip derived-type pre-expression-test sub-assoc
tests-dual.sba-dual.reflexive tests-dual.upper-bound-left)
hence $(-p\sqcup(x«?w{*}aL)){*}p{*}?w(\!|x|\!)aL{*}?w$
using *5* **by** *simp*
hence *6*: $?t\ 0{*}p{*}?w(\!|x|\!)aL{*}?w$
by (*unfold tseq-def power-zero-id id-def*)
have $\forall\, n{>}0\ .\ ?t\ n{*}p{*}?w(\!|x|\!)pSum\ ?t\ n{*}?w$
proof (*rule allI*, *rule impI*)
fix n
assume $0{<}(n{::}nat)$
from *this* **obtain** m **where** *7*: $n = Suc\ m$
by (*auto dest*: *less-imp-Suc-add*)
hence $?t\ m{*}?w \leq pSum\ ?t\ n{*}?w$
using *1 2* **by** (*smt pSum.simps*(*2*) *pSum-test pre-expression-test test-seq-def*
tests-dual.lower-bound-right tests-dual.sba-dual.inf-isotone
tests-dual.sba-dual.reflexive)
thus $?t\ n{*}p{*}?w(\!|x|\!)pSum\ ?t\ n{*}?w$
using *1 7* **by** (*smt assms conj-pre cons-trip tests-dual.upper-bound-left*
tests-dual.sba-dual.inf-complement-intro pSum-pre-expression
power-succ-unfold-ext pre-closed pre-expression-test sub-assoc sub-comm
t-seq-pre-expression test-pre tfun-def tseq-def)
qed
hence $?w(\!|p{\star}x|\!){-}p{*}?w$
using *1 2 3 6 assms while-trip* **by** *auto*

hence *?w*⦇$p{\star}x$⦈$-p{*}q$
using *4* **by** (*metis assms*(*2*) *while-pre-else pre-expression-test while-pre-else*)
thus *?thesis*
using *assms*(*1*,*2*) *Pre-expression.neg-pre Pre-expression.test-pre cons-post-trip*
by *blast*
qed

lemma *pre-completeness*:
$x \in$ *While-program* $\Longrightarrow q \in$ *Pre-expression* $\Longrightarrow x$«q⦇x⦈q
apply (*induct arbitrary*: *q rule*: *While-program.induct*)
apply (*simp add*: *derived-hoare-triple.atom-trip*)
apply (*metis pre-pre pre-seq seq-trip pre-expression-test*)
apply (*smt cond-prog cond-trip cons-pre-trip ite-pre-else ite-pre-then neg-pre pre-pre pre-expression-test test-pre*)
by (*simp add*: *while-complete*)

theorem *completeness*:
$p\langle x\rangle q \Longrightarrow p$⦇$x$⦈$q$
by (*metis valid-hoare-triple-def pre-completeness tests-dual.reflexive pre-expression-test cons-trip*)

end

class *hoare-calculus-sound-complete* = *hoare-calculus-sound* + *hoare-calculus-complete*
begin

Theorem 41

theorem *soundness-completeness*:
p⦇x⦈$q \longleftrightarrow p\langle x\rangle q$
using *completeness soundness* **by** *blast*

end

class *hoare-rules* = *whiledo* + *complete-tests* + *hoare-triple* +
assumes *rule-pre*: x«$-q${|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|}$-r \longrightarrow -q${|$x \lhd -p \rhd y$|}$-r$
assumes *rule-while*: *test-seq t* $\wedge -q \le$ *Sum t* $\wedge$ *t 0*$*-p*-q${|x|}*aL*$*-q \wedge (\forall n{>}0$. *t n*$*-p*-q${|x|}*pSum t n*$*-q) \longrightarrow -q${|$-p{\star}x$|}$--p{*}-q$
assumes *rule-cons*: $-p \le -q \wedge -q${|x|}$-r \wedge -r \le -s \longrightarrow -p${|$x$|}$-s$
assumes *rule-disj*: $-p${|x|}$-r \wedge -q${|x|}$-s \longrightarrow -p\sqcup-q${|$x$|}$-r\sqcup-s$
begin

lemma *rule-cons-pre*:
$-p \le -q \Longrightarrow -q${|$x$|}$-r \Longrightarrow -p${|$x$|}$-r$
using *rule-cons tests-dual.sba-dual.reflexive* **by** *blast*

lemma *rule-cons-pre-mult*:

$-q\{\!|x|\!\}-r \Longrightarrow -p*-q\{\!|x|\!\}-r$
by (*metis tests-dual.sub-sup-closed rule-cons-pre tests-dual.upper-bound-right*)

lemma *rule-cons-pre-plus*:
$-p\sqcup-q\{\!|x|\!\}-r \Longrightarrow -p\{\!|x|\!\}-r$
by (*metis tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.upper-bound-left rule-cons-pre*)

lemma *rule-cons-post*:
$-q\{\!|x|\!\}-r \Longrightarrow -r \leq -s \Longrightarrow -q\{\!|x|\!\}-s$
using *rule-cons tests-dual.sba-dual.reflexive* **by** *blast*

lemma *rule-cons-post-mult*:
$-q\{\!|x|\!\}-r*-s \Longrightarrow -q\{\!|x|\!\}-s$
by (*metis rule-cons-post tests-dual.upper-bound-left sub-comm sub-mult-closed*)

lemma *rule-cons-post-plus*:
$-q\{\!|x|\!\}-r \Longrightarrow -q\{\!|x|\!\}-r\sqcup-s$
by (*metis tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.upper-bound-left rule-cons-post*)

lemma *rule-disj-pre*:
$-p\{\!|x|\!\}-r \Longrightarrow -q\{\!|x|\!\}-r \Longrightarrow -p\sqcup-q\{\!|x|\!\}-r$
by (*metis rule-disj tests-dual.sba-dual.sup-idempotent*)

end

class *hoare-calculus-valid* = *hoare-calculus-sound-complete* + *hoare-triple* +
assumes *hoare-triple-valid*: $-p\{\!|x|\!\}-q \longleftrightarrow -p \leq x«-q$
begin

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

lemma *valid-rule-disj*:
assumes $-p\{\!|x|\!\}-r$
and $-q\{\!|x|\!\}-s$
shows $-p\sqcup-q\{\!|x|\!\}-r\sqcup-s$
proof –
have $x«-r \leq x«-r\sqcup-s \land x«-s \leq x«-r\sqcup-s$
by (*metis pre-iso tests-dual.sba-dual.sub-sup-closed tests-dual.sba-dual.upper-bound-left tests-dual.sba-dual.upper-bound-right*)

thus *?thesis*
by (*smt assms hoare-triple-valid tests-dual.greatest-lower-bound tests-dual.sba-dual.sub-sup-closed pre-closed tests-dual.transitive*)
qed

subclass *hoare-rules*
apply *unfold-locales*
apply (*metis hoare-triple-valid pre-closed tests-dual.sba-dual.reflexive*)
apply (*meson hoare-triple-valid pre-compose*)
apply (*smt hoare-triple-valid ite-import-mult sub-mult-closed*)
apply (*smt* (*verit*, *del-insts*) *hoare-triple-valid aL-test pSum-test sba-dual.sub-sup-closed sub-mult-closed test-seq-def while-soundness-1*)
apply (*smt hoare-triple-valid pre-iso tests-dual.transitive pre-closed*)
by (*simp add*: *valid-rule-disj*)

lemma *nat-test-rule-while*:
$nat\text{-}test\ t\ s \Longrightarrow -q \le s \Longrightarrow (\forall n\ .\ t\ n{*}{-}p{*}{-}q\{\!|x|\!\}pSum\ t\ n{*}{-}q) \Longrightarrow -q\{\!|{-}p{\star}x|\!\}{-}{-}p{*}{-}q$
by (*smt* (*verit*, *ccfv-threshold*) *hoare-triple-valid nat-test-def nat-test-pre pSum-test-nat sub-mult-closed*)

lemma *test-seq-rule-while*:
$test\text{-}seq\ t \Longrightarrow -q \le Sum\ t \Longrightarrow t\ 0{*}{-}p{*}{-}q\{\!|x|\!\}aL{*}{-}q \Longrightarrow (\forall n{>}0\ .\ t\ n{*}{-}p{*}{-}q\{\!|x|\!\}pSum\ t\ n{*}{-}q) \Longrightarrow -q\{\!|{-}p{\star}x|\!\}{-}{-}p{*}{-}q$
by (*smt* (*verit*, *del-insts*) *hoare-triple-valid aL-test pSum-test sub-mult-closed test-seq-def while-soundness-1*)

lemma *rule-bot*:
$bot\{\!|x|\!\}{-}p$
by (*metis hoare-triple-valid pre-closed tests-dual.top-double-complement tests-dual.top-greatest*)

lemma *rule-skip*:
$-p\{\!|1|\!\}{-}p$
by (*simp add*: *hoare-triple-valid pre-one-increasing*)

lemma *rule-example-4*:
assumes *test-seq t*
and $Sum\ t = 1$
and $t\ 0{*}{-}p1{*}{-}p3 = bot$
and $-p1\{\!|z1|\!\}{-}p1{*}{-}p2$
and $\forall n{>}0\ .\ t\ n{*}{-}p1{*}{-}p2{*}{-}p3\{\!|z2|\!\}pSum\ t\ n{*}{-}p1{*}{-}p2$
shows $-p1\{\!|z1{*}({-}p3{\star}z2)|\!\}{-}p2{*}{-}{-}p3$
proof −
have $t\ 0{*}{-}p3{*}({-}p1{*}{-}p2) = bot$
by (*smt* (*verit*, *ccfv-threshold*) *assms*(*1*,*3*) *sub-assoc sub-comm sub-mult-closed test-seq-def tests-dual.sup-right-zero*)
hence *1*: $t\ 0{*}{-}p3{*}({-}p1{*}{-}p2)\{\!|z2|\!\}aL{*}({-}p1{*}{-}p2)$
by (*metis aL-test sub-mult-closed rule-bot*)

have $\forall\, n{>}0$. $t\ n{*}{-}p3{*}({-}p1{*}{-}p2)\{\!|z2|\!\}pSum\ t\ n{*}({-}p1{*}{-}p2)$
by (*smt assms*(*1*,*5*) *lower-bound-left pSum-test rule-cons-pre sub-assoc sub-comm sub-mult-closed test-seq-def*)
hence $-p1{*}{-}p2\{\!|{-}p3{\star}z2|\!\}{-}{-}p3{*}({-}p1{*}{-}p2)$
using *1* **by** (*smt* (*verit*, *del-insts*) *assms*(*1*,*2*) *tests-dual.sub-bot-least rule-while sub-mult-closed*)
thus *?thesis*
by (*smt assms*(*4*) *tests-dual.upper-bound-left rule-cons-post rule-seq sub-assoc sub-comm sub-mult-closed*)
qed

end

class *hoare-calculus-pc-2* = *hoare-calculus-sound* + *hoare-calculus-pre-complete* +
assumes *aL-one*: $aL = 1$
begin

subclass *hoare-calculus-sound-complete*
apply *unfold-locales*
by (*simp add*: *aL-one pre-below-one*)

lemma *while-soundness-pc*:
assumes $-p{*}{-}q \le x{\ll}{-}q$
shows $-q \le -p{\star}x{\ll}{-}{-}p{*}{-}q$
proof −
let $?t = \lambda x\ .\ 1$
have *1*: *test-seq ?t*
by (*simp add*: *test-seq-def*)
hence *2*: $-q \le Sum\ ?t$
by (*metis Sum-test Sum-upper tests-dual.sba-dual.one-def tests-dual.antisymmetric tests-dual.sub-bot-least*)
have *3*: $?t\ 0{*}{-}p{*}{-}q \le x{\ll}aL{*}{-}q$
using *1* **by** (*simp add*: *assms aL-one*)
have $\forall\, n{>}0$. $?t\ n{*}{-}p{*}{-}q \le x{\ll}pSum\ ?t\ n{*}{-}q$
using *1* **by** (*metis assms pSum-test pSum-upper tests-dual.sba-dual.one-def tests-dual.antisymmetric tests-dual.sub-bot-least tests-dual.sup-left-unit*)
thus *?thesis*
using *1 2 3 aL-one while-soundness-0* **by** *auto*
qed

end

class *hoare-calculus-pc* = *hoare-calculus-sound* + *hoare-calculus-pre-complete* +
assumes *pre-one-one*: $x{\ll}1 = 1$
begin

subclass *hoare-calculus-pc-2*
apply *unfold-locales*
by (*simp add*: *aL-def pre-one-one*)

end

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

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

lemma *rule-alternation*:
  $-p\{\!|x|\!\}-q \Longrightarrow -q\{\!|y|\!\}-p \Longrightarrow -p\{\!|-r{\star}x{*}y|\!\}--r{*}-p$
  by (*meson rule-cons-pre-mult rule-seq rule-while-pc*)

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

lemma *rule-example-3*:
  assumes $-p{*}-q\{\!|x|\!\}--p{*}-q$
    and $--p{*}-r\{\!|x|\!\}-p{*}-r$
    and $-p{*}-r\{\!|y|\!\}-p{*}-q$
    and $--p{*}-q\{\!|z|\!\}--p{*}-r$
   shows $-p{*}-q\sqcup--p{*}-r\{\!|-s{\star}x{*}(y\lhd-p\rhd z)|\!\}--s{*}(-p{*}-q\sqcup--p{*}-r)$
proof −
  have *t1*: $-p{*}-q\sqcup--p{*}-r\{\!|x|\!\}--p{*}-q\sqcup-p{*}-r$
    by (*smt assms*(*1*,*2*) *rule-disj sub-mult-closed*)
  have $-p{*}-r\{\!|y|\!\}-p{*}-q\sqcup--p{*}-r$
    by (*smt assms*(*3*) *rule-cons-post-plus sub-mult-closed*)
  hence *t2*: $-p{*}(--p{*}-q\sqcup-p{*}-r)\{\!|y|\!\}-p{*}-q\sqcup--p{*}-r$
    by (*smt* (*z3*) *tests-dual.sba-dual.less-eq-inf tests-dual.sba-dual.reflexive tests-dual.sba-dual.sub-sup-closed tests-dual.sub-associative tests-dual.sub-sup-closed tests-dual.upper-bound-left tests-dual.wnf-lemma-3*)
  have $--p{*}-q\{\!|z|\!\}-p{*}-q\sqcup--p{*}-r$
    by (*smt assms*(*4*) *tests-dual.inf-commutative rule-cons-post-plus sub-mult-closed*)
  hence $--p{*}(--p{*}-q\sqcup-p{*}-r)\{\!|z|\!\}-p{*}-q\sqcup--p{*}-r$
    by (*smt* (*z3*) *tests-dual.sba-dual.one-def tests-dual.sba-dual.sup-absorb tests-dual.sba-dual.sup-complement-intro tests-dual.sba-dual.sup-right-unit tests-dual.sub-sup-closed tests-dual.sup-complement-intro tests-dual.sup-left-dist-inf tests-dual.sup-right-unit tests-dual.top-double-complement*)
  hence $--p{*}-q\sqcup-p{*}-r\{\!|y\lhd-p\rhd z|\!\}-p{*}-q\sqcup--p{*}-r$
    using *t2* **by** (*smt tests-dual.inf-closed rule-cond sub-mult-closed*)
  hence $-s{*}(-p{*}-q\sqcup--p{*}-r)\{\!|x{*}(y\lhd-p\rhd z)|\!\}-p{*}-q\sqcup--p{*}-r$
    using *t1* **by** (*smt tests-dual.inf-closed rule-cons-pre-mult rule-seq sub-mult-closed*)
  thus *?thesis*

by (*smt tests-dual.inf-closed rule-while-pc sub-mult-closed*)
qed

end

class *hoare-calculus-tc* = *hoare-calculus* + *precondition-test-test* + *precondition-distr-mult* +
assumes *while-bnd*: $p \in$ *Test-expression* $\wedge$ $q \in$ *Pre-expression* $\wedge$ $x \in$ *While-program* $\longrightarrow$ $p{\star}x{«}q \leq$ *Sum* $(\lambda n\ .\ (p{*}x)\widehat{\ }n{«}bot)$
begin

lemma
assumes $p \in$ *Test-expression*
and $q \in$ *Pre-expression*
and $x \in$ *While-program*
shows $p{\star}x{«}q \leq$ *tsum* $(-p)$ x $(p{\star}x{«}q)$ $(-p{\sqcup}(x{«}(p{\star}x{«}q){*}aL))$
proof −
let *?w* = $p{\star}x{«}q$
let *?s* = $-p{\sqcup}(x{«}$ *?w*$*aL)$
let *?t* = *tseq* $(-p)$ x *?w* *?s*
let *?b* = $\lambda n\ .\ (p{*}x)\widehat{\ }n{«}bot$
have *2*: *test-seq ?t*
by (*simp add*: *assms*(*2*) *pre-expression-test t-seq-test-seq*)
have *3*: *test-seq ?b*
using *pre-closed test-seq-def tests-dual.sba-dual.complement-top* **by** *blast*
have *4*: *?w* = −− *?w*
by (*metis assms*(*2*) *pre-expression-test pre-closed*)
have *?w* ≤ *Sum ?b*
using *assms while-bnd* **by** *blast*
hence *5*: *?w* = *Sum ?b*∗*?w*
using *3 4* **by** (*smt Sum-test leq-def sub-comm*)
have ∀ *n* . *?b n*∗*?w* ≤ *?t n*
proof
fix *n*
show *?b n*∗*?w* ≤ *?t n*
proof (*induct n*)
show *?b 0*∗*?w* ≤ *?t 0*
using *2 4* **by** (*metis power.power-0 pre-one test-seq-def tests-dual.sup-left-zero tests-dual.top-double-complement tests-dual.top-greatest*)
next
fix *n*
assume *6*: *?b n*∗*?w* ≤ *?t n*
have −*p* ≤ *?t* (*Suc n*)
apply (*unfold tseq-def power-succ-unfold-ext*)
by (*smt assms*(*2*) *pre-expression-test t-seq-test pre-closed sub-mult-closed tfun-def tseq-def tests-dual.lower-bound-left*)
hence *7*: −*p*∗*?b* (*Suc n*)∗*?w* ≤ *?t* (*Suc n*)
using *2 3 4* **by** (*smt tests-dual.upper-bound-left sub-mult-closed test-seq-def tests-dual.transitive*)

have *8*: $p*?b\ (Suc\ n)*?w \le x«?w*(?b\ n*?w)$
by (*smt assms*(*1*,*2*) *tests-dual.upper-bound-right tests-dual.sup-idempotent power-Suc pre-closed pre-distr-mult pre-expression-test pre-import-composition sub-assoc sub-comm sub-mult-closed test-expression-test while-pre-then tests-dual.top-double-complement*)
have *9*: $... \le x«?w*?t\ n$
using *2 3 4 6* **by** (*smt tests-dual.sub-sup-right-isotone pre-iso sub-mult-closed test-seq-def*)
have $... \le ?t\ (Suc\ n)$
using *2 4* **by** (*smt power-succ-unfold-ext pre-closed sub-mult-closed test-seq-def tfun-def tseq-def tests-dual.lower-bound-right*)
hence $p*?b\ (Suc\ n)*?w \le ?t\ (Suc\ n)$
using *2 3 4 8 9* **by** (*smt assms*(*1*) *pre-closed sub-mult-closed test-expression-test test-seq-def tests-dual.transitive*)
thus $?b\ (Suc\ n)*?w \le ?t\ (Suc\ n)$
using *2 3 4 7* **by** (*smt assms*(*1*) *tests-dual.sup-less-eq-cases sub-assoc sub-mult-closed test-expression-test test-seq-def*)
qed
qed
hence $Sum\ ?b*?w \le tsum\ (-p)\ x\ ?w\ ?s$
using *3 4* **by** (*smt assms*(*2*) *Sum-upper mult-right-dist-Sum pre-expression-test sub-mult-closed t-seq-test t-sum-test test-seq-def tests-dual.transitive tsum-def*)
thus *?thesis*
using *5* **by** *auto*
qed

end

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

definition $bnd :: 'a \Rightarrow 'a$
where $bnd\ x \equiv Sup\ \{\ x\hat{\ }n«bot \mid n::nat\ .\ True\ \}$

lemma *bnd-test-set*:
test-set $\{\ x\hat{\ }n«bot \mid n::nat\ .\ True\ \}$
by (*smt* (*verit*, *del-insts*) *CollectD pre-closed test-set-def tests-dual.top-double-complement*)

lemma *bnd-test*:
$bnd\ x = --bnd\ x$
using *bnd-def bnd-test-set sup-test* **by** *auto*

lemma *bnd-upper*:
$x\hat{\ }m«bot \le bnd\ x$
proof −
have $x\hat{\ }m«bot \in \{\ x\hat{\ }m«bot \mid m::nat\ .\ True\ \}$
by *auto*

thus *?thesis*
using *bnd-def bnd-test-set sup-upper* **by** *auto*
qed

lemma *bnd-least*:
assumes $\forall n . x\hat{\ }n«bot \leq -p$
shows *bnd* $x \leq -p$
proof –
have $\forall y \in \{ x\hat{\ }n«bot \mid n::nat . True \} . y \leq -p$
using *assms* **by** *blast*
thus *?thesis*
using *bnd-def bnd-test-set sup-least* **by** *auto*
qed

lemma *mult-right-dist-bnd*:
assumes $\forall n . (x\hat{\ }n«bot)*-p \leq -q$
shows *bnd* $x*-p \leq -q$
proof –
have *Sup* $\{ y*-p \mid y . y \in \{ x\hat{\ }n«bot \mid n::nat . True \} \} \leq -q$
by (*smt assms mem-Collect-eq tests-dual.complement-bot pre-closed sub-mult-closed sup-least test-set-def*)
thus *?thesis*
using *bnd-test-set bnd-def mult-right-dist-sup* **by** *simp*
qed

lemma *tests-complete*:
nat-test $(\lambda n . (-p*x)\hat{\ }n«bot)$ $(bnd(-p*x))$
using *bnd-test bnd-upper mult-right-dist-bnd nat-test-def tests-dual.complement-bot pre-closed* **by** *blast*

end

end

32 Hoare Calculus and Modal Operators

theory *Hoare-Modal*

imports *Stone-Kleene-Relation-Algebras.Kleene-Algebras Complete-Domain Hoare Relative-Modal*

begin

class *box-precondition* = *relative-box-semiring* + *pre* +
assumes *pre-def*: $x«p = |x]p$
begin

Theorem 47

subclass *precondition*

apply *unfold-locales*
apply (*simp add*: *box-x-a pre-def*)
apply (*simp add*: *box-left-mult pre-def*)
using *box-def box-right-submult-a-a pre-def tests-dual.sba-dual.greatest-lower-bound* **apply** *fastforce*
by (*simp add*: *box-1-a pre-def*)

subclass *precondition-test-test*
apply *unfold-locales*
by (*simp add*: *a-box-a-a pre-def*)

subclass *precondition-promote*
apply *unfold-locales*
using *a-mult-d box-def pre-def pre-test-test* **by** *auto*

subclass *precondition-test-box*
apply *unfold-locales*
by (*simp add*: *box-a-a d-def pre-def*)

lemma *pre-Z*:
$-p \leq x«-q \longleftrightarrow -p * x * --q \leq Z$
by (*simp add*: *box-demodalisation-2 pre-def*)

lemma *pre-left-dist-add*:
$x \sqcup y«-q = (x«-q) * (y«-q)$
by (*simp add*: *box-left-dist-sup pre-def*)

lemma *pre-left-antitone*:
$x \leq y \Longrightarrow y«-q \leq x«-q$
by (*simp add*: *box-antitone-isotone pre-def*)

lemma *pre-promote-neg*:
$(x«-q) * x * --q \leq Z$
by (*simp add*: *box-below-Z pre-def*)

lemma *pre-pc-Z*:
$x«1 = 1 \longleftrightarrow x * bot \leq Z$
by (*simp add*: *a-strict box-x-1 pre-def*)

proposition *pre-sub-promote*: $(x«-q) * x \leq (x«-q) * x * -q \sqcup Z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *pre-promote*: $(x«-q) * x \sqcup Z = (x«-q) * x * -q \sqcup Z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *pre-mult-sub-promote*: $(x*y«-q) * x \leq (x*y«-q) * x * (y«-q) \sqcup Z$ **nitpick** [*expect=genuine,card=6*] **oops**
proposition *pre-mult-promote*: $(x*y«-q) * x * (y«-q) \sqcup Z = (x*y«-q) * x \sqcup Z$ **nitpick** [*expect=genuine,card=6*] **oops**

end

class *left-zero-box-precondition* = *box-precondition* +
relative-left-zero-antidomain-semiring
begin

lemma *pre-sub-promote*:
  $(x«-q) * x \leq (x«-q) * x * -q \sqcup Z$
  using *case-split-right-sup pre-promote-neg* **by** *blast*

lemma *pre-promote*:
  $(x«-q) * x \sqcup Z = (x«-q) * x * -q \sqcup Z$
  apply (*rule sup-same-context*)
  apply (*simp add*: *pre-sub-promote*)
  by (*metis a-below-one le-supI1 mult-1-right mult-right-isotone*)

lemma *pre-mult-sub-promote*:
  $(x*y«-q) * x \leq (x*y«-q) * x * (y«-q) \sqcup Z$
  by (*metis pre-closed pre-seq pre-sub-promote*)

lemma *pre-mult-promote-sub*:
  $(x*y«-q) * x * (y«-q) \leq (x*y«-q) * x$
  by (*metis mult-right-isotone mult-1-right pre-below-one*)

lemma *pre-mult-promote*:
  $(x*y«-q) * x * (y«-q) \sqcup Z = (x*y«-q) * x \sqcup Z$
  by (*metis sup-ge1 sup-same-context order-trans pre-mult-sub-promote*
pre-mult-promote-sub)

end

class *diamond-precondition* = *relative-box-semiring* + *pre* +
  assumes *pre-def*: $x«p = |x>p$
begin

Theorem 47

subclass *precondition*
  apply *unfold-locales*
  apply (*simp add*: *d-def diamond-def pre-def*)
  apply (*simp add*: *diamond-left-mult pre-def*)
  apply (*metis a-antitone a-dist-sup box-antitone-isotone box-deMorgan-1*
order.refl pre-def sup-right-divisibility)
  by (*simp add*: *diamond-1-a pre-def*)

subclass *precondition-test-test*
  apply *unfold-locales*
  by (*metis diamond-a-a-same diamond-a-export diamond-associative*
diamond-right-mult pre-def)

subclass *precondition-promote*

apply *unfold-locales*
using *d-def diamond-def pre-def pre-test-test tests-dual.sub-sup-closed* **by** *force*

subclass *precondition-test-diamond*
apply *unfold-locales*
by (*simp add*: *diamond-a-a pre-def*)

lemma *pre-left-dist-add*:
$x \sqcup y \ll -q = (x \ll -q) \sqcup (y \ll -q)$
by (*simp add*: *diamond-left-dist-sup pre-def*)

lemma *pre-left-isotone*:
$x \leq y \Longrightarrow x \ll -q \leq y \ll -q$
by (*metis diamond-left-isotone pre-def*)

end

class *box-while* = *box-precondition* + *bounded-left-conway-semiring* + *ite* + *while* +
assumes *ite-def*: $x \lhd p \rhd y = p * x \sqcup -p * y$
assumes *while-def*: $p \star x = (p * x)^\circ * -p$
begin

subclass *bounded-relative-antidomain-semiring* **..**

lemma *Z-circ-left-zero*:
$Z * x^\circ = Z$
using *Z-left-zero-above-one circ-plus-one sup.absorb-iff2* **by** *auto*

subclass *ifthenelse*
apply *unfold-locales*
by (*smt a-d-closed box-a-export box-left-dist-sup box-x-a tests-dual.case-duality d-def ite-def pre-def*)

Theorem 48.1

subclass *whiledo*
apply *unfold-locales*
apply (*smt circ-loop-fixpoint ite-def ite-pre mult-assoc mult-1-right pre-one pre-seq while-def*)
using *pre-mult-test-promote while-def* **by** *auto*

lemma *pre-while-1*:
$-p*(-p \star x) \ll 1 = -p \star x \ll 1$
proof –
have $--p*(-p*(-p \star x) \ll 1) = --p*(-p \star x \ll 1)$
by (*metis mult-1-right pre-closed pre-seq pre-test-neg tests-dual.sba-dual.top-double-complement while-pre-else*)
thus *?thesis*
by (*smt* (*z3*) *pre-closed pre-import tests-dual.sba-dual.top-double-complement*

tests-dual.sup-eq-cases)
qed

lemma *aL-one-circ*:
  $aL = a(1^\circ * bot)$
  by (*metis aL-def box-left-mult box-x-a idempotent-bot-closed idempotent-one-closed pre-def tests-dual.sba-dual.one-def while-def tests-dual.one-def*)

end

class *diamond-while* = *diamond-precondition* + *bounded-left-conway-semiring* + *ite* + *while* +
  assumes *ite-def*: $x \lhd p \rhd y = p * x \sqcup -p * y$
  assumes *while-def*: $p \star x = (p * x)^\circ * -p$
begin

subclass *bounded-relative-antidomain-semiring* **..**

lemma *Z-circ-left-zero*:
  $Z * x^\circ = Z$
  by (*simp add*: *Z-left-zero-above-one circ-reflexive*)

subclass *ifthenelse*
  apply *unfold-locales*
  by (*simp add*: *ite-def pre-export pre-left-dist-add*)

Theorem 48.2

subclass *whiledo*
  apply *unfold-locales*
  apply (*smt circ-loop-fixpoint ite-def ite-pre mult-assoc mult-1-right pre-one pre-seq while-def*)
  by (*simp add*: *pre-mult-test-promote while-def*)

lemma *aL-one-circ*:
  $aL = d(1^\circ * bot)$
  by (*metis aL-def tests-dual.complement-bot diamond-x-1 mult-left-one pre-def while-def*)

end

class *box-while-program* = *box-while* + *atoms*
begin

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

end

class *diamond-while-program* = *diamond-while* + *atoms*

begin

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

end

class *box-hoare-calculus* = *box-while-program* + *complete-antidomain-semiring*
begin

subclass *hoare-calculus* **..**

end

class *diamond-hoare-calculus* = *diamond-while-program* +
complete-antidomain-semiring
begin

subclass *hoare-calculus* **..**

end

class *box-hoare-sound* = *box-hoare-calculus* + *relative-domain-semiring-split* +
left-kleene-conway-semiring +
  assumes *aL-circ*: $aL * x^\circ \le x^\star$
begin

lemma *aL-circ-ext*:
  $|x^\star]y \le |aL * x^\circ]y$
  by (*simp add*: *aL-circ box-left-antitone*)

lemma *box-star-induct*:
  assumes $-p \le |x](-p)$
    shows $-p \le |x^\star](-p)$
proof $-$
  have *1*: $x*--p*top \le Z \sqcup --p*top$
    by (*metis assms Z-top sup-commute box-demodalisation-2 mult-assoc
mult-left-isotone shunting-Z*)
  have $x*(Z \sqcup --p*top) \le x*--p*top \sqcup Z$
    using *split-Z sup-monoid.add-commute mult-assoc* **by** *force*
  also have ... $\le Z \sqcup --p*top$
    using *1* **by** *simp*
  finally have $x*(Z \sqcup --p*top) \sqcup --p \le Z \sqcup --p*top$
    using *le-supI2 sup.bounded-iff top-right-mult-increasing* **by** *auto*
  thus *?thesis*
    by (*metis sup-commute box-demodalisation-2 mult-assoc shunting-Z
star-left-induct*)
qed

lemma *box-circ-induct*:

$-p \leq |x](-p) \Longrightarrow -p{*}aL \leq |x^\circ](-p)$
by (*smt aL-circ-ext aL-test box-left-mult box-star-induct order-trans tests-dual.inf-commutative pre-closed pre-def pre-test tests-dual.shunting-right*)

lemma *a-while-soundness*:
assumes $-p{*}-q \leq |x](-q)$
shows $aL{*}-q \leq |(-p{*}x)^\circ{*}--p](-q)$
proof −
have $|(-p{*}x)^\circ](-q) \leq |(-p{*}x)^\circ{*}--p](-q)$
by (*meson box-left-antitone circ-mult-upper-bound circ-reflexive order.refl order.trans tests-dual.sub-bot-least*)
thus *?thesis*
by (*smt assms box-import-shunting box-circ-induct order-trans sub-comm aL-test*)
qed

subclass *hoare-calculus-sound*
apply *unfold-locales*
by (*simp add*: *a-while-soundness pre-def while-def*)

end

class *diamond-hoare-sound* = *diamond-hoare-calculus* + *left-kleene-conway-semiring* +
assumes *aL-circ*: $aL * x^\circ \leq x^\star$
begin

lemma *aL-circ-equal*:
$aL * x^\circ = aL * x^\star$
apply (*rule order.antisym*)
using *aL-circ aL-one-circ d-restrict-iff-1* **apply** *force*
by (*simp add*: *mult-right-isotone star-below-circ*)

lemma *aL-zero*:
aL = *bot*
by (*smt aL-circ-equal aL-one-circ d-export d-mult-idempotent diamond-d-bot diamond-def mult-assoc mult-1-right star-one*)

subclass *hoare-calculus-sound*
apply *unfold-locales*
using *aL-zero* **by** *auto*

end

class *box-hoare-complete* = *box-hoare-calculus* + *left-kleene-conway-semiring* +
assumes *box-circ-induct-2*: $-p{*}|x](-q) \leq -q \longrightarrow |x^\circ](-p) \leq -q \sqcup aL$
assumes *aL-zero-or-one*: $aL = bot \vee aL = 1$
assumes *while-mult-left-dist-Prod*: $x \in$ *While-program* $\wedge$ *descending-chain t* $\wedge$ *test-seq t* $\longrightarrow x{*}Prod\ t = Prod\ (\lambda n\ .\ x{*}t\ n)$

begin

subclass *hoare-calculus-complete*
  apply *unfold-locales*
  apply (*metis aL-zero-or-one bot-least order.eq-iff mult-1-right pre-closed tests-dual.sup-right-zero*)
  subgoal
    apply (*unfold pre-def box-def*)
    by (*metis a-ascending-chain a-dist-Prod a-dist-Sum descending-chain-left-mult while-mult-left-dist-Prod test-seq-def*)
  by (*smt box-circ-induct-2 tests-dual.double-negation tests-dual.greatest-lower-bound tests-dual.upper-bound-left mult-right-dist-sup pre-closed pre-def pre-import pre-seq pre-test sub-mult-closed while-def*)

end

class *diamond-hoare-complete* = *diamond-hoare-calculus* + *relative-domain-semiring-split* + *left-kleene-conway-semiring* +
  assumes *dL-circ*: $-aL*x^\circ \leq x^\star$
  assumes *aL-zero-or-one*: $aL = bot \vee aL = 1$
  assumes *while-mult-left-dist-Sum*: $x \in$ *While-program* $\wedge$ *ascending-chain* $t \wedge$ *test-seq* $t \longrightarrow x*Sum\ t = Sum\ (\lambda n\ .\ x*t\ n)$
begin

lemma *diamond-star-induct-var*:
  assumes $|x>(d\ p) \leq d\ p$
    shows $|x^\star>(d\ p) \leq d\ p$
proof −
  have $x * (d\ p * x^\star \sqcup Z) \leq d\ p * x * x^\star \sqcup Z * x^\star \sqcup Z$
    by (*metis assms sup-left-isotone d-mult-d diamond-def diamond-demodalisation-3 mult-assoc mult-left-isotone mult-right-dist-sup order-trans split-Z*)
  also have ... $\leq d\ p * x^\star \sqcup Z$
    by (*metis Z-mult-decreasing mult-right-isotone star.left-plus-below-circ sup.bounded-iff sup-ge1 sup-mono sup-monoid.add-commute mult-assoc*)
  finally show *?thesis*
    by (*smt sup-commute le-sup-iff sup-ge2 d-mult-d diamond-def diamond-demodalisation-3 order-trans star.circ-back-loop-prefixpoint star-left-induct*)
qed

lemma *diamond-star-induct*:
  $d\ q \sqcup |x>(d\ p) \leq d\ p \Longrightarrow |x^\star>(d\ q) \leq d\ p$
  by (*metis le-sup-iff diamond-star-induct-var diamond-right-isotone order-trans*)

lemma *while-completeness-1*:
  assumes $-p*(x«-q) \leq -q$
    shows $-p\star x«-q \leq -q \sqcup aL$
proof −

have $--p*-q \sqcup |-p*x>(-q) \leq -q$
using *assms pre-def pre-export tests-dual.upper-bound-right* **by** *auto*
hence $|(-p*x)^\star>(--p*-q) \leq -q$
by (*smt diamond-star-induct d-def sub-mult-closed tests-dual.double-negation*)
hence $|-aL*(-p*x)^\circ>(--p*-q) \leq -q$
by (*meson dL-circ diamond-isotone order.eq-iff order.trans*)
thus *?thesis*
by (*smt aL-test diamond-a-export diamond-def mult-assoc tests-dual.inf-commutative pre-closed pre-def tests-dual.shunting while-def*)
qed

subclass *hoare-calculus-complete*
apply *unfold-locales*
apply (*metis aL-test aL-zero-or-one bot-least order.eq-iff pre-closed pre-test pre-test-one tests-dual.sup-right-zero*)
subgoal
apply (*unfold pre-def diamond-def*)
by (*simp add: ascending-chain-left-mult d-dist-Sum while-mult-left-dist-Sum*)
by (*simp add: while-completeness-1*)

end

class *box-hoare-valid* = *box-hoare-sound* + *box-hoare-complete* + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \longleftrightarrow p \leq |x]q$
begin

Theorem 49.2

subclass *hoare-calculus-valid*
apply *unfold-locales*
by (*simp add: hoare-triple-def pre-def*)

lemma *rule-skip-valid*:
$-p\{\!|1|\!\}-p$
by (*simp add: rule-skip*)

end

class *diamond-hoare-valid* = *diamond-hoare-sound* + *diamond-hoare-complete* + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \longleftrightarrow p \leq |x>q$
begin

lemma *circ-star-equal*:
$x^\circ = x^\star$
by (*metis aL-zero order.antisym dL-circ mult-left-one one-def star-below-circ*)

Theorem 49.1

subclass *hoare-calculus-valid*
apply *unfold-locales*

by (*simp add*: *hoare-triple-def pre-def*)

end

class *diamond-hoare-sound-2* = *diamond-hoare-calculus* +
left-kleene-conway-semiring +
assumes *diamond-circ-induct-2*: $--p*-q \leq |x>(-q) \longrightarrow\ aL*-q \leq |x^\circ>(-p)$
begin

subclass *hoare-calculus-sound*
apply *unfold-locales*
by (*smt a-export diamond-associative diamond-circ-induct-2 tests-dual.double-negation tests-dual.sup-complement-intro pre-def pre-import-equiv-mult sub-comm sub-mult-closed while-def*)

end

class *diamond-hoare-valid-2* = *diamond-hoare-sound-2* +
diamond-hoare-complete + *hoare-triple* +
assumes *hoare-triple-def*: $p\{\!|x|\!\}q \longleftrightarrow p \leq |x>q$
begin

subclass *hoare-calculus-valid*
apply *unfold-locales*
by (*simp add*: *hoare-triple-def pre-def*)

end

end

33 Pre-Post Specifications

theory *Pre-Post*

imports *Preconditions*

begin

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

class *pre-post-spec-greatest* = *bounded-idempotent-left-semiring* + *precondition* +
pre-post +
assumes *pre-post-galois*: $-p \leq x«-q \longleftrightarrow x \leq -p⊣-q$
begin

Theorem 42.1

lemma *post-pre-left-antitone*:
$x \leq y \Longrightarrow y«-q \leq x«-q$

by (*smt order-refl order-trans pre-closed pre-post-galois*)

lemma *pre-left-sub-dist*:
$x \sqcup y \ll -q \leq x \ll -q$
by (*simp add*: *post-pre-left-antitone*)

Theorem 42.2

lemma *pre-post-left-antitone*:
$-p \leq -q \Longrightarrow -q \dashv -r \leq -p \dashv -r$
using *order-lesseq-imp pre-post-galois* **by** *blast*

lemma *pre-post-left-sub-dist*:
$-p \sqcup -q \dashv -r \leq -p \dashv -r$
by (*metis sup.cobounded1 tests-dual.sba-dual.sub-sup-closed pre-post-left-antitone*)

lemma *pre-post-left-sup-dist*:
$-p \dashv -r \leq -p * -q \dashv -r$
by (*metis tests-dual.sba-dual.sub-inf-def pre-post-left-sub-dist tests-dual.inf-absorb*)

Theorem 42.5

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

Theorem 42.6

lemma *pre-post-pre*:
$-p \leq (-p \dashv -q) \ll -q$
by (*simp add*: *pre-post-galois*)

Theorem 42.8

lemma *pre-post-zero-top*:
$bot \dashv -q = top$
by (*metis order.eq-iff pre-post-galois sup.cobounded2 sup-monoid.add-0-right top-greatest tests-dual.top-double-complement*)

Theorem 42.7

lemma *pre-post-pre-one*:
$(1 \dashv -q) \ll -q = 1$
by (*metis order.eq-iff pre-below-one tests-dual.sba-dual.top-double-complement pre-post-pre*)

Theorem 42.3

lemma *pre-post-right-isotone*:
$-p \leq -q \Longrightarrow -r \dashv -p \leq -r \dashv -q$
using *order-lesseq-imp pre-iso pre-post-galois* **by** *blast*

lemma *pre-post-right-sub-dist*:

$-r\dashv -p \leq -r\dashv -p\sqcup -q$
by (*metis sup.cobounded1 tests-dual.sba-dual.sub-sup-closed pre-post-right-isotone*)

lemma *pre-post-right-sup-dist*:
$-r\dashv -p*-q \leq -r\dashv -p$
by (*metis tests-dual.sub-sup-closed pre-post-right-isotone tests-dual.upper-bound-left*)

Theorem 42.7

lemma *pre-post-reflexive*:
$1 \leq -p\dashv -p$
using *pre-one-increasing pre-post-galois* **by** *auto*

Theorem 42.9

lemma *pre-post-compose*:
$-q \leq -r \Longrightarrow (-p\dashv -q)*(-r\dashv -s) \leq -p\dashv -s$
using *order-lesseq-imp pre-compose pre-post-galois* **by** *blast*

Theorem 42.10

lemma *pre-post-compose-1*:
$(-p\dashv -q)*(-q\dashv -r) \leq -p\dashv -r$
by (*simp add*: *pre-post-compose*)

Theorem 42.11

lemma *pre-post-compose-2*:
$(-p\dashv -p)*(-p\dashv -q) = -p\dashv -q$
by (*meson case-split-left order.eq-iff le-supI1 pre-post-compose-1 pre-post-reflexive*)

Theorem 42.12

lemma *pre-post-compose-3*:
$(-p\dashv -q)*(-q\dashv -q) = -p\dashv -q$
by (*meson order.eq-iff order.trans mult-right-isotone mult-sub-right-one pre-post-compose-1 pre-post-reflexive*)

Theorem 42.13

lemma *pre-post-compose-4*:
$(-p\dashv -p)*(-p\dashv -p) = -p\dashv -p$
by (*simp add*: *pre-post-compose-3*)

Theorem 42.14

lemma *pre-post-one-one*:
$x«1 = 1 \longleftrightarrow x \leq 1\dashv 1$
by (*metis order.eq-iff one-def pre-below-one pre-post-galois*)

Theorem 42.4

lemma *post-pre-left-dist-sup*:
$x\sqcup y«-q = (x«-q)*(y«-q)$

apply (*rule order.antisym*)
apply (*metis mult-isotone pre-closed sup-commute tests-dual.sup-idempotent pre-left-sub-dist*)
by (*smt* (*z3*) *order.refl pre-closed pre-post-galois sup.boundedI tests-dual.sba-dual.greatest-lower-bound tests-dual.sub-sup-closed*)

proposition *pre-post-right-dist-sup*: $-p\dashv-q\sqcup-r = (-p\dashv-q) \sqcup (-p\dashv-r)$ **nitpick** [*expect=genuine,card=4*] **oops**

end

class *pre-post-spec-greatest-2* = *pre-post-spec-greatest* + *precondition-test-test*
begin

subclass *precondition-test-box*
apply *unfold-locales*
by (*smt* (*verit*) *sup-commute mult-1-right tests-dual.double-negation order.eq-iff mult-left-one mult-right-dist-sup one-def tests-dual.inf-complement tests-dual.inf-complement-intro pre-below-one pre-import pre-post-galois pre-test-test tests-dual.top-def bot-least*)

lemma *pre-post-seq-sub-associative*:
$(-p\dashv-q)*-r \leq -p\dashv-q*-r$
by (*smt* (*z3*) *pre-compose pre-post-galois pre-post-pre sub-comm test-below-pre-test-mult tests-dual.sub-sup-closed*)

lemma *pre-post-right-import-mult*:
$(-p\dashv-q)*-r = (-p\dashv-q*-r)*-r$
by (*metis order.antisym mult-assoc tests-dual.sup-idempotent mult-left-isotone pre-post-right-sup-dist pre-post-seq-sub-associative*)

lemma *seq-pre-post-sub-associative*:
$-r*(-p\dashv-q) \leq --r\sqcup-p\dashv-q$
by (*smt* (*z3*) *pre-compose pre-post-galois pre-post-pre pre-test tests-dual.sba-dual.reflexive tests-dual.sba-dual.sub-sup-closed*)

lemma *pre-post-left-import-sup*:
$-r*(-p\dashv-q) = -r*(--r\sqcup-p\dashv-q)$
by (*metis sup-commute order.antisym mult-assoc tests-dual.sup-idempotent mult-right-isotone pre-post-left-sub-dist seq-pre-post-sub-associative*)

lemma *pre-post-import-same*:
$-p*(-p\dashv-q) = -p*(1\dashv-q)$
using *pre-test pre-test-test-same pre-post-left-import-sup* **by** *auto*

lemma *pre-post-import-complement*:
$--p*(-p\dashv-q) = --p*top$
by (*metis tests-dual.sup-idempotent tests-dual.inf-cases tests-dual.inf-closed pre-post-left-import-sup pre-post-zero-top tests-dual.top-def*

tests-dual.top-double-complement)

lemma *pre-post-export*:
$-p \dashv -q = (1 \dashv -q) \sqcup --p*top$
proof (*rule order.antisym*)
have *1*: $-p*(-p \dashv -q) \leq (1 \dashv -q) \sqcup --p*top$
by (*metis le-supI1 pre-test pre-test-test-same seq-pre-post-sub-associative*)
have $--p*(-p \dashv -q) \leq (1 \dashv -q) \sqcup --p*top$
by (*simp add*: *pre-post-import-complement*)
thus $-p \dashv -q \leq (1 \dashv -q) \sqcup --p*top$
using *1* **by** (*smt case-split-left eq-refl tests-dual.inf-complement*)
next
show $(1 \dashv -q) \sqcup --p*top \leq -p \dashv -q$
by (*metis le-sup-iff tests-dual.double-negation tests-dual.sub-bot-least pre-neg-mult pre-post-galois pre-post-pre-one*)
qed

lemma *pre-post-left-dist-mult*:
$-p*-q \dashv -r = (-p \dashv -r) \sqcup (-q \dashv -r)$
proof −
have $\forall\, p\ q\ .\ -p*(-p*-q \dashv -r) = -p*(-q \dashv -r)$
using *sup-monoid.add-commute tests-dual.sba-dual.sub-inf-def pre-post-left-import-sup tests-dual.inf-complement-intro* **by** *auto*
hence *1*: $(-p \sqcup -q)*(-p*-q \dashv -r) \leq (-p \dashv -r) \sqcup (-q \dashv -r)$
by (*metis sup-commute le-sup-iff sup-ge2 mult-left-one mult-right-dist-sup tests-dual.inf-left-unit sub-comm*)
have $-(-p \sqcup -q)*(-p*-q \dashv -r) = -(-p \sqcup -q)*top$
by (*smt* (*z3*) *sup.left-commute sup-commute tests-dual.sba-dual.sub-sup-closed tests-dual.sub-sup-closed pre-post-import-complement pre-post-left-import-sup tests-dual.inf-absorb*)
hence $-(-p \sqcup -q)*(-p*-q \dashv -r) \leq (-p \dashv -r) \sqcup (-q \dashv -r)$
by (*smt* (*z3*) *order.trans le-supI1 pre-post-left-sub-dist tests-dual.sba-dual.sub-sup-closed tests-dual.sub-sup-closed seq-pre-post-sub-associative*)
thus *?thesis*
using *1* **by** (*smt* (*z3*) *le-sup-iff order.antisym case-split-left order-refl tests-dual.inf-closed tests-dual.inf-complement pre-post-left-sup-dist sub-comm*)
qed

lemma *pre-post-left-import-mult*:
$-r*(-p \dashv -q) = -r*(-r*-p \dashv -q)$
by (*metis sup-commute tests-dual.inf-complement-intro pre-post-left-import-sup sub-mult-closed*)

lemma *pre-post-right-import-sup*:
$(-p \dashv -q)*-r = (-p \dashv -q \sqcup --r)*-r$
by (*smt* (*z3*) *sup-monoid.add-commute tests-dual.sba-dual.inf-cases-2 tests-dual.sba-dual.inf-complement-intro tests-dual.sub-complement tests-dual.sub-inf-def pre-post-right-import-mult*)

lemma *pre-post-shunting*:
  $x \le -p{*}-q\dashv -r \longleftrightarrow -p{*}x \le -q\dashv -r$
proof −
  have $--p{*}x \le -p{*}-q\dashv -r$
    by (*metis tests-dual.double-negation order-trans pre-neg-mult pre-post-galois pre-post-left-sup-dist*)
  hence *1*: $-p{*}x \le -q\dashv -r \longrightarrow x \le -p{*}-q\dashv -r$
    by (*smt case-split-left eq-refl order-trans tests-dual.inf-complement pre-post-left-sup-dist sub-comm*)
  have $-p{*}(-p{*}-q\dashv -r) \le -q\dashv -r$
    by (*metis mult-left-isotone mult-left-one tests-dual.sub-bot-least pre-post-left-import-mult*)
  thus *?thesis*
    using *1 mult-right-isotone order-lesseq-imp* **by** *blast*
qed

proposition *pre-post-right-dist-sup*: $-p\dashv -q\sqcup -r = (-p\dashv -q) \sqcup (-p\dashv -r)$ **oops**

end

class *left-zero-pre-post-spec-greatest-2* = *pre-post-spec-greatest-2* + *bounded-idempotent-left-zero-semiring*
begin

lemma *pre-post-right-dist-sup*:
  $-p\dashv -q\sqcup -r = (-p\dashv -q) \sqcup (-p\dashv -r)$
proof −
  have *1*: $(-p\dashv -q\sqcup -r){*}-q \le (-p\dashv -q) \sqcup (-p\dashv -r)$
    by (*metis le-supI1 pre-post-seq-sub-associative tests-dual.sba-dual.inf-absorb tests-dual.sba-dual.sub-sup-closed*)
  have $(-p\dashv -q\sqcup -r){*}--q = (-p\dashv -r){*}--q$
    by (*simp add*: *pre-post-right-import-sup sup-commute*)
  hence $(-p\dashv -q\sqcup -r){*}--q \le (-p\dashv -q) \sqcup (-p\dashv -r)$
    by (*metis sup-ge2 mult-left-sub-dist-sup-right mult-1-right order-trans tests-dual.inf-left-unit*)
  thus *?thesis*
    using *1* **by** (*metis le-sup-iff order.antisym case-split-right tests-dual.sub-bot-least tests-dual.inf-commutative tests-dual.inf-complement pre-post-right-sub-dist*)
qed

end

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

class *idempotent-left-semiring-H* = *bounded-idempotent-left-semiring* + *havoc* +
  assumes *H-zero* : $H * bot = bot$

assumes *H-split*: $x \leq x * bot \sqcup H$
begin

lemma *H-galois*:
$x * bot \leq y \longleftrightarrow x \leq y \sqcup H$
apply (*rule iffI*)
using *H-split order-lesseq-imp sup-mono* **apply** *blast*
by (*smt* (*verit*, *ccfv-threshold*) *H-zero mult-right-dist-sup sup.cobounded2 sup.orderE sup-assoc sup-bot-left sup-commute zero-right-mult-decreasing*)

lemma *H-greatest-finite*:
$x * bot = bot \longleftrightarrow x \leq H$
by (*metis H-galois le-iff-sup sup-bot-left sup-monoid.add-0-right*)

lemma *H-reflexive*:
$1 \leq H$
using *H-greatest-finite mult-left-one* **by** *blast*

lemma *H-transitive*:
$H = H * H$
by (*metis H-greatest-finite H-reflexive H-zero preorder-idempotent mult-assoc*)

lemma *T-split-H*:
$top * bot \sqcup H = top$
by (*simp add*: *H-split order.antisym*)

proposition $H * (x \sqcup y) = H * x \sqcup H * y$ **nitpick** [*expect=genuine*,*card=6*]
oops

end

class *pre-post-spec-least* = *bounded-idempotent-left-semiring* + *precondition-test-test* + *precondition-promote* + *pre-post* +
assumes *test-mult-right-distr-sup*: $-p * (x \sqcup y) = -p * x \sqcup -p * y$
assumes *pre-post-galois*: $-p \leq x«-q \longleftrightarrow -p\dashv-q \leq x$
begin

lemma *shunting-top*:
$-p * x \leq y \longleftrightarrow x \leq y \sqcup --p * top$
proof
assume $-p * x \leq y$
thus $x \leq y \sqcup --p * top$
by (*smt* (*verit*, *ccfv-SIG*) *case-split-left eq-refl le-supI1 le-supI2 mult-right-isotone tests-dual.sba-dual.top-def top-greatest*)
next
assume $x \leq y \sqcup --p * top$
hence $-p * x \leq -p * y$
by (*metis sup-bot-right mult-assoc tests-dual.sup-complement mult-left-zero mult-right-isotone test-mult-right-distr-sup*)

thus $-p * x \leq y$
by (*metis mult-left-isotone mult-left-one tests-dual.sub-bot-least order-trans*)
qed

lemma *post-pre-left-isotone*:
$x \leq y \Longrightarrow x«-q \leq y«-q$
by (*smt order-refl order-trans pre-closed pre-post-galois*)

lemma *pre-left-sub-dist*:
$x«-q \leq x \sqcup y«-q$
by (*simp add*: *post-pre-left-isotone*)

lemma *pre-post-left-isotone*:
$-p \leq -q \Longrightarrow -p\dashv-r \leq -q\dashv-r$
using *order-lesseq-imp pre-post-galois* **by** *blast*

lemma *pre-post-left-sub-dist*:
$-p\dashv-r \leq -p\sqcup-q\dashv-r$
by (*metis sup-ge1 tests-dual.inf-closed pre-post-left-isotone*)

lemma *pre-post-left-sup-dist*:
$-p*-q\dashv-r \leq -p\dashv-r$
by (*metis tests-dual.upper-bound-left pre-post-left-isotone sub-mult-closed*)

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

lemma *pre-post-pre*:
$-p \leq (-p\dashv-q)«-q$
by (*simp add*: *pre-post-galois*)

lemma *pre-post-zero-top*:
$bot\dashv-q = bot$
using *bot-least order.eq-iff pre-post-galois tests-dual.sba-dual.sub-bot-def* **by** *blast*

lemma *pre-post-pre-one*:
$(1\dashv-q)«-q = 1$
by (*metis order.eq-iff pre-below-one pre-post-pre tests-dual.sba-dual.top-double-complement*)

lemma *pre-post-right-antitone*:
$-p \leq -q \Longrightarrow -r\dashv-q \leq -r\dashv-p$
using *order-lesseq-imp pre-iso pre-post-galois* **by** *blast*

lemma *pre-post-right-sub-dist*:
$-r\dashv-p\sqcup-q \leq -r\dashv-p$
by (*metis sup-ge1 tests-dual.inf-closed pre-post-right-antitone*)

lemma *pre-post-right-sup-dist*:
$-r\dashv-p \le -r\dashv-p*-q$
by (*metis tests-dual.upper-bound-left pre-post-right-antitone sub-mult-closed*)

lemma *pre-top*:
$top«-q = 1$
using *order.eq-iff pre-below-one pre-post-galois tests-dual.sba-dual.one-def top-greatest* **by** *blast*

lemma *pre-mult-top-increasing*:
$-p \le -p*top«-q$
using *pre-import-equiv pre-top tests-dual.sub-bot-least* **by** *auto*

lemma *pre-post-below-mult-top*:
$-p\dashv-q \le -p*top$
using *pre-import-equiv pre-post-galois* **by** *auto*

lemma *pre-post-import-complement*:
$--p*(-p\dashv-q) = bot$
proof −
have $--p*(-p\dashv-q) \le --p*(-p*top)$
by (*simp add*: *mult-right-isotone pre-post-below-mult-top*)
thus *?thesis*
by (*metis mult-assoc mult-left-zero sub-comm tests-dual.top-def order.antisym bot-least*)
qed

lemma *pre-post-import-same*:
$-p*(-p\dashv-q) = -p\dashv-q$
proof −
have $-p\dashv-q = -p*(-p\dashv-q) \sqcup --p*(-p\dashv-q)$
by (*metis mult-left-one mult-right-dist-sup tests-dual.inf-complement*)
thus *?thesis*
using *pre-post-import-complement* **by** *auto*
qed

lemma *pre-post-export*:
$-p\dashv-q = -p*(1\dashv-q)$
proof (*rule order.antisym*)
show $-p\dashv-q \le -p*(1\dashv-q)$
by (*metis tests-dual.sub-bot-least pre-import-equiv pre-post-galois pre-post-pre-one*)
next
have *1*: $-p \le ((-p\dashv-q) \sqcup --p*top)«-q$
by (*simp add*: *pre-post-galois*)
have $--p \le ((-p\dashv-q) \sqcup --p*top)«-q$
by (*simp add*: *le-supI2 pre-post-galois pre-post-below-mult-top*)
hence $-p \sqcup --p \le ((-p\dashv-q) \sqcup --p*top)«-q$

using *1 le-supI* **by** *blast*
hence $1 \le ((-p\dashv-q) \sqcup --p*top)«-q$
by *simp*
hence $1\dashv-q \le (-p\dashv-q) \sqcup --p*top$
using *pre-post-galois tests-dual.sba-dual.one-def* **by** *blast*
thus $-p*(1\dashv-q) \le -p\dashv-q$
by (*simp add*: *shunting-top*)
qed

lemma *pre-post-seq-associative*:
$-r*(-p\dashv-q) = -r*-p\dashv-q$
by (*metis pre-post-export tests-dual.sub-sup-closed mult-assoc*)

lemma *pre-post-left-import-mult*:
$-r*(-p\dashv-q) = -r*(-r*-p\dashv-q)$
by (*metis mult-assoc tests-dual.sup-idempotent pre-post-seq-associative*)

lemma *seq-pre-post-sub-associative*:
$-r*(-p\dashv-q) \le --r\sqcup-p\dashv-q$
by (*metis le-supI1 pre-post-left-sub-dist sup-commute shunting-top*)

lemma *pre-post-left-import-sup*:
$-r*(-p\dashv-q) = -r*(--r\sqcup-p\dashv-q)$
by (*metis tests-dual.sba-dual.sub-sup-closed pre-post-seq-associative tests-dual.sup-complement-intro*)

lemma *pre-post-left-dist-sup*:
$-p\sqcup-q\dashv-r = (-p\dashv-r) \sqcup (-q\dashv-r)$
by (*metis mult-right-dist-sup tests-dual.inf-closed pre-post-export*)

lemma *pre-post-reflexive*:
$-p\dashv-p \le 1$
using *pre-one-increasing pre-post-galois* **by** *auto*

lemma *pre-post-compose*:
$-q \le -r \Longrightarrow -p\dashv-s \le (-p\dashv-q)*(-r\dashv-s)$
by (*meson pre-compose pre-post-galois pre-post-pre pre-post-right-antitone*)

lemma *pre-post-compose-1*:
$-p\dashv-r \le (-p\dashv-q)*(-q\dashv-r)$
by (*simp add*: *pre-post-compose*)

lemma *pre-post-compose-2*:
$(-p\dashv-p)*(-p\dashv-q) = -p\dashv-q$
using *order.eq-iff mult-left-isotone pre-post-compose-1 pre-post-reflexive* **by** *fastforce*

lemma *pre-post-compose-3*:
$(-p\dashv-q)*(-q\dashv-q) = -p\dashv-q$

by (*metis order.antisym mult-right-isotone mult-1-right pre-post-compose-1 pre-post-reflexive*)

lemma *pre-post-compose-4*:
$(-p\dashv -p) * (-p\dashv -p) = -p\dashv -p$
by (*simp add*: *pre-post-compose-3*)

lemma *pre-post-one-one*:
$x \ll 1 = 1 \longleftrightarrow 1\dashv 1 \leq x$
using *order.eq-iff pre-below-one pre-post-galois tests-dual.sub-bot-def* **by** *force*

lemma *pre-one-right*:
$-p \ll 1 = -p$
by (*metis order.antisym mult-1-right one-def tests-dual.inf-complement pre-left-sub-dist pre-mult-top-increasing pre-one pre-seq pre-test-promote pre-top*)

lemma *pre-pre-one*:
$x \ll -q = x * -q \ll 1$
by (*metis one-def pre-one-right pre-seq*)

subclass *precondition-test-diamond*
apply *unfold-locales*
using *tests-dual.sba-dual.sub-inf-def pre-one-right pre-pre-one* **by** *auto*

proposition *pre-post-shunting*: $x \leq -p * -q \dashv -r \longleftrightarrow -p * x \leq -q \dashv -r$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $(-p\dashv -q) * -r = (-p\dashv -q \sqcup -r) * -r$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $(-p\dashv -q) * -r = (-p\dashv -q \sqcup --r) * -r$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $(-p\dashv -q) * -r = (-p\dashv -q * -r) * -r$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $(-p\dashv -q) * -r = (-p\dashv -q * --r) * -r$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $-p\dashv -q \sqcup -r = (-p\dashv -q) \sqcup (-p\dashv -r)$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition $-p\dashv -q \sqcup -r = (-p\dashv -q) * (-p\dashv -r)$ **nitpick** [*expect=genuine*,*card=3*] **oops**
proposition *pre-post-right-dist-mult*: $-p\dashv -q * -r = (-p\dashv -q) * (-p\dashv -r)$ **oops**
proposition *pre-post-right-dist-mult*: $-p\dashv -q * -r = (-p\dashv -q) \sqcup (-p\dashv -r)$ **oops**
proposition *post-pre-left-dist-sup*: $x \sqcup y \ll -q = (x \ll -q) \sqcup (y \ll -q)$ **oops**

end

class *havoc-dual* =
fixes *Hd* :: *'a*

class *idempotent-left-semiring-Hd* = *bounded-idempotent-left-semiring* + *havoc-dual* +

assumes *Hd-total*: $Hd * top = top$
assumes *Hd-least*: $x * top = top \longrightarrow Hd \leq x$
begin

lemma *Hd-least-total*:
$x * top = top \longleftrightarrow Hd \leq x$
by (*metis Hd-least Hd-total order.antisym mult-left-isotone top-greatest*)

lemma *Hd-reflexive*:
$Hd \leq 1$
by (*simp add*: *Hd-least*)

lemma *Hd-transitive*:
$Hd = Hd * Hd$
by (*simp add*: *Hd-least Hd-total order.antisym coreflexive-transitive total-mult-closed*)

end

class *pre-post-spec-least-Hd* = *idempotent-left-semiring-Hd* + *pre-post-spec-least* +
assumes *pre-one-mult-top*: $(x«1)*top = x*top$
begin

lemma *Hd-pre-one*:
$Hd«1 = 1$
by (*metis Hd-total pre-seq pre-top*)

lemma *pre-post-below-Hd*:
$1 \dashv 1 \leq Hd$
using *Hd-pre-one pre-post-one-one* **by** *auto*

lemma *Hd-pre-post*:
$Hd = 1 \dashv 1$
by (*metis Hd-least Hd-pre-one Hd-total order.eq-iff pre-one-mult-top pre-post-one-one*)

lemma *top-left-zero*:
$top*x = top$
by (*metis mult-assoc mult-left-one mult-left-zero pre-closed pre-one-mult-top pre-seq pre-top*)

lemma *test-dual-test*:
$(-p \sqcup --p*top)*-p = -p \sqcup --p*top$
by (*simp add*: *top-left-zero mult-right-dist-sup mult-assoc*)

lemma *pre-zero-mult-top*:
$(x«bot)*top = x*bot$
by (*metis mult-assoc mult-left-zero one-def pre-one-mult-top pre-seq pre-bot*)

lemma *pre-one-mult-Hd*:
 $(x«1)*Hd \le x$
 by (*metis Hd-pre-post one-def pre-closed pre-post-export pre-pre-post*)

lemma *Hd-mult-pre-one*:
 $Hd*(x«1) \le x$
proof –
 have *1*: $-(x«1)*Hd*(x«1) \le x$
 by (*metis Hd-pre-post le-iff-sup mult-left-isotone pre-closed pre-one-right pre-post-export pre-pre-post sup-commute sup-monoid.add-0-right tests-dual.sba-dual.one-def tests-dual.top-def*)
 have $(x«1)*Hd*(x«1) \le x$
 by (*metis mult-isotone mult-1-right one-def pre-below-one pre-one-mult-Hd*)
 thus *?thesis*
 using *1* **by** (*metis case-split-left pre-closed reflexive-one-closed tests-dual.sba-dual.one-def tests-dual.sba-dual.top-def mult-assoc*)
qed

lemma *pre-post-one-def-1*:
 assumes $1 \le x«-q$
 shows $Hd*(-q\sqcup--q*top) \le x$
proof –
 have $Hd*(-q\sqcup--q*top) \le x*-q*(-q\sqcup--q*top)$
 by (*metis assms Hd-pre-post order.antisym pre-below-one pre-post-one-one pre-pre-one mult-left-isotone*)
 thus *?thesis*
 by (*metis mult-assoc tests-dual.sup-complement mult-left-sub-dist-sup-left mult-left-zero mult-1-right tests-dual.inf-complement test-mult-right-distr-sup order-trans*)
qed

lemma *pre-post-one-def*:
 $1\dashv-q = Hd*(-q\sqcup--q*top)$
proof (*rule order.antisym*)
 have $1 \le (1\dashv1)*(-q\sqcup--q)«1$
 by (*metis pre-post-pre one-def mult-1-right tests-dual.inf-complement*)
 also have ... $\le (1\dashv1)*(-q\sqcup--q*top)«-q$
 by (*metis sup-right-isotone mult-right-isotone mult-1-right one-def post-pre-left-isotone pre-seq pre-test-promote test-dual-test top-right-mult-increasing*)
 finally show $1\dashv-q \le Hd*(-q\sqcup--q*top)$
 using *Hd-pre-post pre-post-galois tests-dual.sub-bot-def* **by** *blast*
next
 show $Hd*(-q\sqcup--q*top) \le 1\dashv-q$
 by (*simp add*: *pre-post-pre-one pre-post-one-def-1*)
qed

lemma *pre-post-def*:

$-p\dashv-q = -p*Hd*(-q\sqcup--q*top)$
by (*simp add*: *pre-post-export mult-assoc pre-post-one-def*)

end

end

34 Pre-Post Specifications and Modal Operators

theory *Pre-Post-Modal*

imports *Pre-Post Hoare-Modal*

begin

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

lemma *nat-test-pre-post*:
$nat\text{-}test\ t\ s \Longrightarrow -q \leq s \Longrightarrow (\forall n\ .\ x \leq t\ n*-p*-q\dashv(pSum\ t\ n*-q)) \Longrightarrow -p\star x \leq -q\dashv--p*-q$
by (*smt* (*verit*, *ccfv-threshold*) *nat-test-def nat-test-pre pSum-test-nat pre-post-galois tests-dual.sub-sup-closed*)

lemma *nat-test-pre-post-2*:
$nat\text{-}test\ t\ s \Longrightarrow -r \leq s \Longrightarrow (\forall n\ .\ x \leq t\ n*-p\dashv(pSum\ t\ n)) \Longrightarrow -p\star x \leq -r\dashv 1$
by (*smt* (*verit*, *ccfv-threshold*) *nat-test-def nat-test-pre-2 one-def pSum-test-nat pre-post-galois tests-dual.sub-sup-closed*)

end

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

lemma *pre-post-while*:
$x \leq -p*-q\dashv-q \longrightarrow -p\star x \leq aL*-q\dashv-q$
by (*smt aL-test pre-post-galois sub-mult-closed while-soundness*)

Theorem 43.1

lemma *while-soundness-3*:
$test\text{-}seq\ t \Longrightarrow -q \leq Sum\ t \Longrightarrow x \leq t\ 0*-p*-q\dashv aL*-q \Longrightarrow (\forall n{>}0\ .\ x \leq t\ n*-p*-q\dashv pSum\ t\ n*-q) \Longrightarrow -p\star x \leq -q\dashv--p*-q$
by (*smt* (*verit*, *del-insts*) *aL-test pSum-test tests-dual.inf-closed pre-post-galois sub-mult-closed test-seq-def while-soundness-1*)

Theorem 43.2

lemma *while-soundness-4*:
$test\text{-}seq\ t \Longrightarrow -r \leq Sum\ t \Longrightarrow (\forall n\ .\ x \leq t\ n*-p\dashv pSum\ t\ n) \Longrightarrow -p\star x \leq -r\dashv 1$

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-2* = *pre-post-spec-hoare* + *hoare-calculus-pc-2*
begin

Theorem 43.3

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

end

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

subclass *pre-post-spec-hoare-pc-2* ..

lemma *pre-post-one-one-top*:
$1{\dashv}1 = top$
using *order.eq-iff pre-one-one pre-post-one-one* **by** *auto*

end

class *pre-post-spec-H* = *pre-post-spec-greatest* + *box-precondition* + *havoc* +
assumes *H-zero-2*: $H * bot = bot$
assumes *H-split-2*: $x \leq x * -q * top \sqcup H * --q$
begin

subclass *idempotent-left-semiring-H*
apply *unfold-locales*
apply (*rule H-zero-2*)
by (*smt H-split-2 tests-dual.complement-bot mult-assoc mult-left-zero mult-1-right one-def*)

lemma *pre-post-def-iff*:
$-p * x * --q \leq Z \longleftrightarrow x \leq Z \sqcup --p * top \sqcup H * -q$
proof (*rule iffI*)
assume $-p * x * --q \leq Z$
hence $x * --q * top \leq Z \sqcup --p * top$
by (*smt* (*verit, ccfv-threshold*) *Z-left-zero-above-one case-split-left-sup mult-assoc mult-left-isotone mult-right-dist-sup mult-right-isotone top-greatest top-mult-top*)
thus $x \leq Z \sqcup --p * top \sqcup H * -q$
by (*metis sup-left-isotone order-trans H-split-2 tests-dual.double-negation*)
next
assume $x \leq Z \sqcup --p * top \sqcup H * -q$

hence $-p * x * --q \leq -p * (Z * --q \sqcup --p * top * --q \sqcup H * -q * --q)$
by (*metis mult-left-isotone mult-right-dist-sup mult-right-isotone mult-assoc*)
thus $-p * x * --q \leq Z$
by (*metis H-zero-2 Z-mult-decreasing sup-commute sup-bot-left mult-assoc mult-right-dist-sup mult-right-isotone order-trans test-mult-left-dist-shunt test-mult-left-sub-dist-shunt tests-dual.top-def*)
qed

lemma *pre-post-def*:
$-p\dashv -q = Z \sqcup --p*top \sqcup H*-q$
by (*meson order.antisym order-refl pre-Z pre-post-galois pre-post-def-iff*)

end

class *pre-post-L* = *pre-post-spec-greatest* + *box-while* + *left-conway-semiring-L* + *left-kleene-conway-semiring* +
assumes *circ-below-L-add-star*: $x^\circ \leq L \sqcup x^\star$
begin

a loop does not abort if its body does not abort

this avoids abortion from all states* alternatively from states in -r if -r is an invariant

lemma *body-abort-loop*:
assumes $Z = L$
and $x \leq -p\dashv 1$
shows $-p\star x \leq 1\dashv 1$
proof −
have $-p * x * bot \leq L$
by (*metis assms pre-Z pre-post-galois tests-dual.sba-dual.one-def tests-dual.top-double-complement*)
hence $(-p * x)^\star * bot \leq L$
by (*metis L-split le-iff-sup star-left-induct sup-bot-left*)
hence $(-p * x)^\circ * bot \leq L$
by (*smt L-left-zero L-split sup-commute circ-below-L-add-star le-iff-sup mult-right-dist-sup*)
thus *?thesis*
by (*metis assms(1) a-restrict mult-isotone pre-pc-Z pre-post-compose-2 pre-post-one-one tests-dual.sba-dual.one-def while-def tests-dual.sup-right-zero*)
qed

end

class *pre-post-spec-Hd* = *pre-post-spec-least* + *diamond-precondition* + *idempotent-left-semiring-Hd* +
assumes *d-mult-top*: $d(x) * top = x * top$
begin

subclass *pre-post-spec-least-Hd*
apply *unfold-locales*

by (*simp add*: *d-mult-top diamond-x-1 pre-def*)

end

end

35 Monotonic Boolean Transformers

theory *Monotonic-Boolean-Transformers*

imports *MonoBoolTranAlgebra.Assertion-Algebra Base*

begin

no-notation *inf* (**infixl** ‹⊓› *70*)
unbundle *no uminus-syntax*

context *mbt-algebra*
begin

lemma *directed-left-mult*:
directed Y $\Longrightarrow$ *directed* ((∗) *x* ‘ *Y*)
apply (*unfold directed-def*)
using *le-comp* **by** *blast*

lemma *neg-assertion*:
neg-assert x ∈ *assertion*
by (*metis bot-comp neg-assert-def wpt-def wpt-is-assertion mult-assoc*)

lemma *assertion-neg-assert*:
x ∈ *assertion* ⟷ *x* = *neg-assert* (*neg-assert x*)
by (*metis neg-assertion uminus-uminus*)

extend and dualise part of Viorel Preoteasa's theory

definition *assumption* ≡ {*x* . *1* ≤ *x* ∧ (*x* ∗ *bot*) ⊔ (*x* ^ *o*) = *x*}

definition *neg-assume* (*x*::′*a*) ≡ (*x* ^ *o* ∗ *top*) ⊔ *1*

lemma *neg-assume-assert*:
neg-assume x = (*neg-assert* (*x* ^ *o*)) ^ *o*
using *dual-bot dual-comp dual-dual dual-inf dual-one neg-assert-def neg-assume-def* **by** *auto*

lemma *assert-iff-assume*:
x ∈ *assertion* ⟷ *x* ^ *o* ∈ *assumption*
by (*smt assertion-def assumption-def dual-bot dual-comp dual-dual dual-inf dual-le dual-one mem-Collect-eq*)

lemma *assertion-iff-assumption-subseteq*:

$X \subseteq assertion \longleftrightarrow dual \text{ ‘ } X \subseteq assumption$
using *assert-iff-assume* **by** *blast*

lemma *assumption-iff-assertion-subseteq*:
$X \subseteq assumption \longleftrightarrow dual \text{ ‘ } X \subseteq assertion$
using *assert-iff-assume* **by** *auto*

lemma *assumption-prop*:
$x \in assumption \Longrightarrow (x * bot) \sqcup 1 = x$
by (*smt assert-iff-assume assertion-prop dual-comp dual-dual dual-neg-top dual-one dual-sup dual-top*)

lemma *neg-assumption*:
$neg\text{-}assume\ x \in assumption$
using *assert-iff-assume neg-assertion neg-assume-assert* **by** *auto*

lemma *assumption-neg-assume*:
$x \in assumption \longleftrightarrow x = neg\text{-}assume\ (neg\text{-}assume\ x)$
by (*smt assert-iff-assume assertion-neg-assert dual-dual neg-assume-assert*)

lemma *assumption-sup-comp-eq*:
$x \in assumption \Longrightarrow y \in assumption \Longrightarrow x \sqcup y = x * y$
by (*smt assert-iff-assume assertion-inf-comp-eq dual-comp dual-dual dual-sup*)

lemma *sup-uminus-assume*[*simp*]:
$x \in assumption \Longrightarrow x \sqcap neg\text{-}assume\ x = 1$
by (*smt assert-iff-assume dual-dual dual-one dual-sup neg-assume-assert sup-uminus*)

lemma *inf-uminus-assume*[*simp*]:
$x \in assumption \Longrightarrow x \sqcup neg\text{-}assume\ x = top$
by (*smt assert-iff-assume dual-dual dual-sup dual-top inf-uminus neg-assume-assert sup-bot-right*)

lemma *uminus-assumption*[*simp*]:
$x \in assumption \Longrightarrow neg\text{-}assume\ x \in assumption$
by (*simp add*: *neg-assumption*)

lemma *uminus-uminus-assume*[*simp*]:
$x \in assumption \Longrightarrow neg\text{-}assume\ (neg\text{-}assume\ x) = x$
by (*simp add*: *assumption-neg-assume*)

lemma *sup-assumption*[*simp*]:
$x \in assumption \Longrightarrow y \in assumption \Longrightarrow x \sqcup y \in assumption$
by (*smt assert-iff-assume dual-dual dual-sup inf-assertion*)

lemma *comp-assumption*[*simp*]:
$x \in assumption \Longrightarrow y \in assumption \Longrightarrow x * y \in assumption$
using *assumption-sup-comp-eq sup-assumption* **by** *auto*

lemma *inf-assumption*[*simp*]:
$x \in assumption \Longrightarrow y \in assumption \Longrightarrow x \sqcap y \in assumption$
by (*smt assert-iff-assume dual-dual dual-inf sup-assertion*)

lemma *assumption-comp-idempotent*[*simp*]:
$x \in assumption \Longrightarrow x * x = x$
using *assumption-sup-comp-eq* **by** *fastforce*

lemma *assumption-comp-idempotent-dual*[*simp*]:
$x \in assumption \Longrightarrow (x \frown o) * (x \frown o) = x \frown o$
by (*metis assumption-comp-idempotent dual-comp*)

lemma *top-assumption*[*simp*]:
$top \in assumption$
by (*simp add*: *assumption-def*)

lemma *one-assumption*[*simp*]:
$1 \in assumption$
by (*simp add*: *assumption-def*)

lemma *assert-top*:
$neg\text{-}assert\ (neg\text{-}assert\ p) \frown o * bot = neg\text{-}assert\ p * top$
by (*smt bot-comp dual-comp dual-dual dual-top inf-comp inf-top-right mult.assoc mult.left-neutral neg-assert-def*)

lemma *assume-bot*:
$neg\text{-}assume\ (neg\text{-}assume\ p) \frown o * top = neg\text{-}assume\ p * bot$
by (*smt assert-top dual-bot dual-comp dual-dual neg-assume-assert*)

definition $wpb\ x \equiv (x * bot) \sqcup 1$

lemma *wpt-iff-wpb*:
$wpb\ x = wpt\ (x \frown o) \frown o$
using *dual-comp dual-dual dual-inf dual-one dual-top wpt-def wpb-def* **by** *auto*

lemma *wpb-is-assumption*[*simp*]:
$wpb\ x \in assumption$
using *assert-iff-assume wpt-is-assertion wpt-iff-wpb* **by** *auto*

lemma *wpb-comp*:
$(wpb\ x) * x = x$
by (*smt dual-comp dual-dual dual-neg-top dual-sup wpt-comp wpt-iff-wpb*)

lemma *wpb-comp-2*:
$wpb\ (x * y) = wpb\ (x * (wpb\ y))$
by (*simp add*: *sup-comp mult-assoc wpb-def*)

lemma *wpb-assumption*[*simp*]:

$x \in$ *assumption* $\Longrightarrow$ *wpb* $x = x$
by (*simp add*: *assumption-prop wpb-def*)

lemma *wpb-choice*:
wpb $(x \sqcup y) =$ *wpb* $x \sqcup$ *wpb* y
using *sup-assoc sup-commute sup-comp wpb-def* **by** *auto*

lemma *wpb-dual-assumption*:
$x \in$ *assumption* $\Longrightarrow$ *wpb* $(x \ \hat{}\ o) = 1$
by (*smt assert-iff-assume dual-dual dual-one wpt-dual-assertion wpt-iff-wpb*)

lemma *wpb-mono*:
$x \le y \Longrightarrow$ *wpb* $x \le$ *wpb* y
by (*metis le-iff-sup wpb-choice*)

lemma *assumption-disjunctive*:
$x \in$ *assumption* $\Longrightarrow x \in$ *disjunctive*
by (*smt assert-iff-assume assertion-conjunctive dual-comp dual-conjunctive dual-dual*)

lemma *assumption-conjunctive*:
$x \in$ *assumption* $\Longrightarrow x \in$ *conjunctive*
by (*smt assert-iff-assume assertion-disjunctive dual-comp dual-disjunctive dual-dual*)

lemma *wpb-le-assumption*:
$x \in$ *assumption* $\Longrightarrow x * y = y \Longrightarrow x \le$ *wpb* y
by (*metis assumption-prop bot-least le-comp sup-commute sup-right-isotone mult-assoc wpb-def*)

definition *dual-omega* :: $'a \Rightarrow 'a$ (‹(- $\hat{}\ \mho$)› [*81*] *80*)
where $(x \ \hat{}\ \mho) \equiv (((x \ \hat{}\ o) \ \hat{}\ \omega) \ \hat{}\ o)$

lemma *dual-omega-fix*:
$x\hat{}\mho = (x * (x\hat{}\mho)) \sqcup 1$
by (*smt dual-comp dual-dual dual-omega-def dual-one dual-sup omega-fix*)

lemma *dual-omega-comp-fix*:
$x\hat{}\mho * y = (x * (x\hat{}\mho) * y) \sqcup y$
by (*metis dual-omega-fix mult-1-left sup-comp*)

lemma *dual-omega-greatest*:
$z \le (x * z) \sqcup y \Longrightarrow z \le (x\hat{}\mho) * y$
by (*smt dual-comp dual-dual dual-le dual-neg-top dual-omega-def dual-sup omega-least*)

end

context *post-mbt-algebra*

begin

lemma *post-antitone*:
 assumes $x \leq y$
 shows *post* $y \leq$ *post* x
proof −
 have *post* $y \leq$ *post* $x * y * top \sqcap$ *post* y
 by (*metis assms inf-top-left post-1 inf-mono le-comp-left-right order-refl*)
 thus *?thesis*
 using *order-lesseq-imp post-2* **by** *blast*
qed

lemma *post-assumption-below-one*:
 $q \in assumption \Longrightarrow post\ q \leq post\ 1$
 by (*simp add*: *assumption-def post-antitone*)

lemma *post-assumption-above-one*:
 $q \in assumption \Longrightarrow post\ 1 \leq post\ (q\ ^\frown\ o)$
 by (*metis dual-le dual-one post-antitone sup.commute sup-ge1 wpb-assumption wpb-def*)

lemma *post-assumption-below-dual*:
 $q \in assumption \Longrightarrow post\ q \leq post\ (q\ ^\frown\ o)$
 using *order-trans post-assumption-above-one post-assumption-below-one* **by** *blast*

lemma *assumption-assertion-absorb*:
 $q \in assumption \Longrightarrow q * (q\ ^\frown\ o) = q$
 by (*smt CollectE assumption-def assumption-prop bot-comp mult.left-neutral mult-assoc sup-comp*)

lemma *post-dual-below-post-one*:
 assumes $q \in assumption$
 shows *post* $(q\ ^\frown\ o) \leq post\ 1 * q$
proof −
 have *post* $(q\ ^\frown\ o) \leq post\ 1 * q * (q\ ^\frown\ o) * top \sqcap post\ (q\ ^\frown\ o)$
 by (*metis assms assumption-assertion-absorb gt-one-comp inf-le1 inf-top-left mult-assoc order-refl post-1 sup-uminus-assume top-unique*)
 thus *?thesis*
 using *order-lesseq-imp post-2* **by** *blast*
qed

lemma *post-below-post-one*:
 $q \in assumption \Longrightarrow post\ q \leq post\ 1 * q$
 using *order.trans post-assumption-below-dual post-dual-below-post-one* **by** *blast*

end

context *complete-mbt-algebra*

begin

lemma *Inf-assumption*[*simp*]:
$X \subseteq assumption \Longrightarrow Inf\ X \in assumption$
by (*metis Sup-assertion assert-iff-assume assumption-iff-assertion-subseteq dual-Inf dual-dual*)

definition *continuous* $x \equiv (\forall\ Y\ .\ directed\ Y \longrightarrow x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y))$

definition *Continuous* $\equiv \{\ x\ .\ continuous\ x\ \}$

lemma *continuous-Continuous*:
$continuous\ x \longleftrightarrow x \in Continuous$
by (*simp add*: *Continuous-def*)

Theorem 53.1

lemma *one-continuous*:
$1 \in Continuous$
by (*simp add*: *Continuous-def continuous-def image-def*)

lemma *continuous-dist-ascending-chain*:
assumes $x \in Continuous$
and *ascending-chain f*
shows $x * (SUP\ n{::}nat\ .\ f\ n) = (SUP\ n{::}nat\ .\ x * f\ n)$
proof −
have *directed* (*range f*)
by (*simp add*: *assms*(*2*) *ascending-chain-directed*)
hence $x * (SUP\ n{::}nat\ .\ f\ n) = (SUP\ y{\in}range\ f\ .\ x * y)$
using *assms*(*1*) *continuous-Continuous continuous-def* **by** *auto*
thus *?thesis*
by (*simp add*: *range-composition*)
qed

Theorem 53.1

lemma *assertion-continuous*:
assumes $x \in assertion$
shows $x \in Continuous$
proof −
have *1*: $x = (x * top) \sqcap 1$
using *assms assertion-prop* **by** *auto*
have $\forall\ Y\ .\ directed\ Y \longrightarrow x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
proof (*rule allI*, *rule impI*)
fix *Y*
assume *directed Y*
have $x * (SUP\ y{\in}Y\ .\ y) = (x * top) \sqcap (SUP\ y{\in}Y\ .\ y)$
using *1* **by** (*smt inf-comp mult.assoc mult.left-neutral top-comp*)
also have ... $= (SUP\ y{\in}Y\ .\ (x * top) \sqcap y)$
by (*simp add*: *inf-Sup*)

finally show $x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
  using *1* **by** (*smt inf-comp mult.left-neutral mult.assoc top-comp SUP-cong*)
qed
thus *?thesis*
  by (*simp add*: *continuous-def Continuous-def*)
qed

Theorem 53.1

lemma *assumption-continuous*:
  assumes $x \in$ *assumption*
    shows $x \in$ *Continuous*
proof −
  have *1*: $x = (x * bot) \sqcup 1$
    by (*simp add*: *assms assumption-prop*)
  have $\forall\ Y\ .\ directed\ Y \longrightarrow x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
  proof (*rule allI*, *rule impI*)
    fix *Y*
    assume *2*: *directed Y*
    have $x * (SUP\ y{\in}Y\ .\ y) = (x * bot) \sqcup (SUP\ y{\in}Y\ .\ y)$
      using *1* **by** (*smt sup-comp mult.assoc mult.left-neutral bot-comp*)
    also have ... $= (SUP\ y{\in}Y\ .\ (x * bot) \sqcup y)$
      using *2* **by** (*smt* (*verit*, *ccfv-threshold*) *sup-SUP SUP-cong directed-def*)
    finally show $x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
     using *1* **by** (*metis sup-comp mult.left-neutral mult.assoc bot-comp SUP-cong*)
  qed
  thus *?thesis*
    by (*simp add*: *continuous-def Continuous-def*)
qed

Theorem 53.1

lemma *mult-continuous*:
  assumes $x \in$ *Continuous*
      and $y \in$ *Continuous*
    shows $x * y \in$ *Continuous*
proof −
  have $\forall\ Y.\ directed\ Y \longrightarrow x * y * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ x * y * z)$
  proof (*rule allI*, *rule impI*)
    fix *Y*
    assume *directed Y*
    hence $x * y * (SUP\ w{\in}Y\ .\ w) = (SUP\ z{\in}Y\ .\ x * (y * z))$
      by (*metis assms continuous-Continuous continuous-def directed-left-mult image-ident image-image mult-assoc*)
    thus $x * y * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ x * y * z)$
      using *mult-assoc* **by** *auto*
  qed
  thus *?thesis*
    using *Continuous-def continuous-def* **by** *blast*
qed

Theorem 53.1

lemma *sup-continuous*:
  $x \in Continuous \Longrightarrow y \in Continuous \Longrightarrow x \sqcup y \in Continuous$
  by (*smt SUP-cong SUP-sup-distrib continuous-Continuous continuous-def sup-comp*)

Theorem 53.1

lemma *inf-continuous*:
  assumes $x \in Continuous$
    and $y \in Continuous$
  shows $x \sqcap y \in Continuous$
proof –
  have $\forall Y.\ directed\ Y \longrightarrow (x \sqcap y) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x \sqcap y) * z)$
  proof (*rule allI, rule impI*)
    fix Y
    assume *1*: *directed* Y
    have *2*: $(SUP\ w{\in}Y\ .\ SUP\ z{\in}Y\ .\ (x * w) \sqcap (y * z)) \leq (SUP\ z{\in}Y\ .\ (x * z) \sqcap (y * z))$
    proof (*intro SUP-least*)
      fix $w\ z$
      assume $w \in Y$ **and** $z \in Y$
      from *this* **obtain** v **where** *3*: $v{\in}Y \wedge w \leq v \wedge z \leq v$
        using *1* **by** (*meson directed-def*)
      hence $x * w \sqcap (y * z) \leq (x * v) \sqcap (y * v)$
        by (*meson inf.sup-mono le-comp*)
      thus $x * w \sqcap (y * z) \leq (SUP\ z{\in}Y\ .\ (x * z) \sqcap (y * z))$
        using *3* **by** (*meson SUP-upper2*)
    qed
    have $(SUP\ z{\in}Y\ .\ (x * z) \sqcap (y * z)) \leq (SUP\ w{\in}Y\ .\ SUP\ z{\in}Y\ .\ (x * w) \sqcap (y * z))$
      apply (*rule SUP-least*)
      by (*meson SUP-upper SUP-upper2*)
    hence $(SUP\ w{\in}Y\ .\ SUP\ z{\in}Y\ .\ (x * w) \sqcap (y * z)) = (SUP\ z{\in}Y\ .\ (x \sqcap y) * z)$
      using *2 order.antisym inf-comp* **by** *auto*
    thus $(x \sqcap y) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x \sqcap y) * z)$
      using *1* **by** (*metis assms inf-comp continuous-Continuous continuous-def SUP-inf-distrib2*)
  qed
  thus *?thesis*
    using *Continuous-def continuous-def* **by** *blast*
qed

Theorem 53.1

lemma *dual-star-continuous*:
  assumes $x \in Continuous$
  shows $x\ ^\frown\otimes \in Continuous$
proof –
  have $\forall Y.\ directed\ Y \longrightarrow (x\ ^\frown\otimes) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x\ ^\frown\otimes) * z)$
  proof (*rule allI, rule impI*)
    fix Y

assume *directed Y*
hence *directed* $((*)\ (x\ \widehat{}\ \otimes)$ ‘ $Y)$
by (*simp add*: *directed-left-mult*)
hence $x * (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \otimes) * y) = (SUP\ y{\in}Y\ .\ x * ((x\ \widehat{}\ \otimes) * y))$
by (*metis assms continuous-Continuous continuous-def image-ident image-image*)
also have ... $= (SUP\ y{\in}Y\ .\ x * (x\ \widehat{}\ \otimes) * y)$
using *mult-assoc* **by** *auto*
also have ... $\leq (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \otimes) * y)$
apply (*rule SUP-least*)
by (*simp add*: *SUP-upper2 dual-star-comp-fix*)
finally have $x * (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \otimes) * y) \sqcup (SUP\ y{\in}Y\ .\ y) \leq (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \otimes) * y)$
apply (*rule sup-least*)
by (*metis SUP-mono′ dual-star-comp-fix sup.cobounded1 sup-commute*)
thus $(x\ \widehat{}\ \otimes) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x\ \widehat{}\ \otimes) * z)$
by (*meson SUP-least SUP-upper order.antisym dual-star-least le-comp*)
qed
thus *?thesis*
using *Continuous-def continuous-def* **by** *blast*
qed

Theorem 53.1

lemma *omega-continuous*:
assumes $x \in Continuous$
shows $x\ \widehat{}\ \omega \in Continuous$
proof –
have $\forall Y.\ directed\ Y \longrightarrow (x\ \widehat{}\ \omega) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x\ \widehat{}\ \omega) * z)$
proof (*rule allI*, *rule impI*)
fix *Y*
assume *1*: *directed Y*
hence *directed* $((*)\ (x\ \widehat{}\ \omega)$ ‘ $Y)$
using *directed-left-mult* **by** *auto*
hence $x * (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \omega) * y) = (SUP\ y{\in}Y\ .\ x * ((x\ \widehat{}\ \omega) * y))$
by (*metis assms continuous-Continuous continuous-def image-ident image-image*)
hence *2*: $x * (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \omega) * y) = (SUP\ y{\in}Y\ .\ x * (x\ \widehat{}\ \omega) * y)$
by (*simp add*: *mult-assoc*)
have $(SUP\ y{\in}Y\ .\ x * (x\ \widehat{}\ \omega) * y) \sqcap (SUP\ y{\in}Y\ .\ y) = (SUP\ w{\in}Y\ .\ SUP\ z{\in}Y\ .\ (x * (x\ \widehat{}\ \omega) * w) \sqcap z)$
using *SUP-inf-distrib2* **by** *blast*
hence $x * (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \omega) * y) \sqcap (SUP\ y{\in}Y\ .\ y) = (SUP\ w{\in}Y\ .\ SUP\ z{\in}Y\ .\ (x * (x\ \widehat{}\ \omega) * w) \sqcap z)$
using *2* **by** *auto*
also have ... $\leq (SUP\ y{\in}Y\ .\ (x\ \widehat{}\ \omega) * y)$
proof (*intro SUP-least*)
fix *w z*
assume $w \in Y$ **and** $z \in Y$
from *this* **obtain** *v* **where** *3*: $v{\in}Y \wedge w \leq v \wedge z \leq v$

using *1* **by** (*meson directed-def*)
hence $x * x ^\frown \omega * w \sqcap z \le x ^\frown \omega * v$
using *inf.sup-mono le-comp omega-comp-fix* **by** *auto*
thus $x * x ^\frown \omega * w \sqcap z \le (SUP\ y{\in}Y\ .\ (x ^\frown \omega) * y)$
using *3* **by** (*meson SUP-upper2*)
qed
finally show $(x ^\frown \omega) * (SUP\ y{\in}Y\ .\ y) = (SUP\ z{\in}Y\ .\ (x ^\frown \omega) * z)$
by (*meson SUP-least SUP-upper order.antisym omega-least le-comp*)
qed
thus *?thesis*
using *Continuous-def continuous-def* **by** *blast*
qed

definition *co-continuous* $x \equiv (\forall\ Y\ .\ co\text{-}directed\ Y \longrightarrow x * (INF\ y{\in}Y\ .\ y) = (INF\ y{\in}Y\ .\ x * y))$

definition *Co-continuous* $\equiv \{\ x\ .\ co\text{-}continuous\ x\ \}$

lemma *directed-dual*:
directed $X \longleftrightarrow$ *co-directed* (*dual* ‘ X)
by (*simp add*: *directed-def co-directed-def dual-le*[*THEN sym*])

lemma *dual-dual-image*:
dual ‘ *dual* ‘ $X = X$
by (*simp add*: *image-comp*)

lemma *continuous-dual*:
continuous $x \longleftrightarrow$ *co-continuous* $(x ^\frown o)$
proof (*unfold continuous-def co-continuous-def*, *rule iffI*)
assume *1*: $\forall\ Y.\ directed\ Y \longrightarrow x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
show $\forall\ Y.\ co\text{-}directed\ Y \longrightarrow x ^\frown o * (INF\ y{\in}Y\ .\ y) = (INF\ y{\in}Y\ .\ x ^\frown o * y)$
proof (*rule allI*, *rule impI*)
fix Y
assume *co-directed* Y
hence $x ^\frown o * (INF\ y{\in}Y\ .\ y) = (INF\ y{\in}(dual\ ‘\ Y)\ .\ (x * y) ^\frown o)$
using *1* **by** (*metis dual-dual-image dual-SUP image-ident image-image dual-comp directed-dual*)
also have ... $= (INF\ y{\in}(dual\ ‘\ Y)\ .\ x ^\frown o * y ^\frown o)$
by (*meson dual-comp*)
also have ... $= (INF\ y{\in}Y\ .\ x ^\frown o * y)$
by (*simp add*: *image-image*)
finally show $x ^\frown o * (INF\ y{\in}Y\ .\ y) = (INF\ y{\in}Y\ .\ x ^\frown o * y)$
.
qed
next
assume *2*: $\forall\ Y.\ co\text{-}directed\ Y \longrightarrow x ^\frown o * (INF\ y{\in}Y\ .\ y) = (INF\ y{\in}Y\ .\ x ^\frown o * y)$
show $\forall\ Y.\ directed\ Y \longrightarrow x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
proof (*rule allI*, *rule impI*)

fix *Y*
assume *directed Y*
hence $x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}(dual\ `\ Y)\ .\ (x\ \hat{}\ o * y)\ \hat{}\ o)$
using *2* **by** (*metis directed-dual dual-dual-image image-ident image-image dual-SUP dual-comp dual-dual*)
also have ... $= (SUP\ y{\in}(dual\ `\ Y)\ .\ x * y\ \hat{}\ o)$
using *dual-comp dual-dual* **by** *auto*
also have ... $= (SUP\ y{\in}Y\ .\ x * y)$
by (*simp add*: *image-image*)
finally show $x * (SUP\ y{\in}Y\ .\ y) = (SUP\ y{\in}Y\ .\ x * y)$
.
qed
qed

lemma *co-continuous-Co-continuous*:
co-continuous $x \longleftrightarrow x \in$ *Co-continuous*
by (*simp add*: *Co-continuous-def*)

Theorem 53.1 and Theorem 53.2

lemma *Continuous-dual*:
$x \in$ *Continuous* $\longleftrightarrow x\ \hat{}\ o \in$ *Co-continuous*
by (*simp add*: *Co-continuous-def Continuous-def continuous-dual*)

Theorem 53.2

lemma *one-co-continuous*:
1 $\in$ *Co-continuous*
using *Continuous-dual one-continuous* **by** *auto*

lemma *ascending-chain-dual*:
ascending-chain f $\longleftrightarrow$ *descending-chain* (*dual o f*)
using *ascending-chain-def descending-chain-def dual-le* **by** *auto*

lemma *co-continuous-dist-descending-chain*:
assumes $x \in$ *Co-continuous*
and *descending-chain f*
shows $x * (INF\ n{::}nat\ .\ f\ n) = (INF\ n{::}nat\ .\ x * f\ n)$
proof –
have $x\ \hat{}\ o * (SUP\ n{::}nat\ .\ (dual\ o\ f)\ n) = (SUP\ n{::}nat\ .\ x\ \hat{}\ o * (dual\ o\ f)\ n)$
by (*smt assms Continuous-dual SUP-cong ascending-chain-dual continuous-dist-ascending-chain descending-chain-def dual-dual o-def*)
thus *?thesis*
by (*smt INF-cong dual-SUP dual-comp dual-dual o-def*)
qed

Theorem 53.2

lemma *assertion-co-continuous*:
$x \in$ *assertion* $\Longrightarrow x \in$ *Co-continuous*
by (*smt Continuous-dual assert-iff-assume assumption-continuous dual-dual*)

Theorem 53.2

lemma *assumption-co-continuous*:
 $x \in$ *assumption* $\Longrightarrow x \in$ *Co-continuous*
 by (*smt Continuous-dual assert-iff-assume assertion-continuous dual-dual*)

Theorem 53.2

lemma *mult-co-continuous*:
 $x \in$ *Co-continuous* $\Longrightarrow y \in$ *Co-continuous* $\Longrightarrow x * y \in$ *Co-continuous*
 by (*smt Continuous-dual dual-comp dual-dual mult-continuous*)

Theorem 53.2

lemma *sup-co-continuous*:
 $x \in$ *Co-continuous* $\Longrightarrow y \in$ *Co-continuous* $\Longrightarrow x \sqcup y \in$ *Co-continuous*
 by (*smt Continuous-dual dual-sup dual-dual inf-continuous*)

Theorem 53.2

lemma *inf-co-continuous*:
 $x \in$ *Co-continuous* $\Longrightarrow y \in$ *Co-continuous* $\Longrightarrow x \sqcap y \in$ *Co-continuous*
 by (*smt Continuous-dual dual-inf dual-dual sup-continuous*)

Theorem 53.2

lemma *dual-omega-co-continuous*:
 $x \in$ *Co-continuous* $\Longrightarrow x\ \hat{}\ \mho \in$ *Co-continuous*
 by (*smt Continuous-dual dual-omega-def dual-dual omega-continuous*)

Theorem 53.2

lemma *star-co-continuous*:
 $x \in$ *Co-continuous* $\Longrightarrow x\ \hat{}\ * \in$ *Co-continuous*
 by (*smt Continuous-dual dual-star-def dual-dual dual-star-continuous*)

lemma *dual-omega-iterate*:
 assumes $y \in$ *Co-continuous*
  shows $y\ \hat{}\ \mho * z = (INF\ n{::}nat\ .\ ((\lambda x\ .\ y * x \sqcup z)\ \hat{}\ n)\ top)$
proof (*rule order.antisym*)
 show $y\ \hat{}\ \mho * z \le (INF\ n{::}nat\ .\ ((\lambda x\ .\ y * x \sqcup z)\ \hat{}\ n)\ top)$
 proof (*rule INF-greatest*)
  fix n
  show $y\ \hat{}\ \mho * z \le ((\lambda x.\ y * x \sqcup z)\ \hat{}\ n)\ top$
   apply (*induct n*)
   apply (*metis power-zero-id id-def top-greatest*)
   by (*smt dual-omega-comp-fix le-comp mult-assoc order-refl sup-mono*
power-succ-unfold-ext)
 qed
next
 have *1*: *descending-chain* $(\lambda n\ .\ ((\lambda x.\ y * x \sqcup z)\ \hat{}\ n)\ top)$
 proof (*unfold descending-chain-def*, *rule allI*)
  fix n
  show $((\lambda x.\ y * x \sqcup z)\ \hat{}\ Suc\ n)\ top \le ((\lambda x.\ y * x \sqcup z)\ \hat{}\ n)\ top$
   apply (*induct n*)
   apply (*metis power-zero-id id-def top-greatest*)

by (*smt power-succ-unfold-ext sup-mono order-refl le-comp*)
qed
have (*INF n.* ((λ*x. y* ∗ *x* ⊔ *z*) ^ *n*) *top*) ≤ (*INF n.* ((λ*x. y* ∗ *x* ⊔ *z*) ^ *Suc n*) *top*)
apply (*rule INF-greatest*)
apply (*unfold power-succ-unfold-ext*)
by (*smt power-succ-unfold-ext INF-lower UNIV-I*)
thus (*INF n.* ((λ*x. y* ∗ *x* ⊔ *z*) ^ *n*) *top*) ≤ *y* ^℧ ∗ *z*
using *1* **by** (*smt assms INF-cong co-continuous-dist-descending-chain power-succ-unfold-ext sup-INF sup-commute dual-omega-greatest*)
qed

lemma *dual-omega-iterate-one*:
y ∈ *Co-continuous* ⟹ *y* ^℧ = (*INF n*::*nat* . ((λ*x* . *y* ∗ *x* ⊔ *1*) ^ *n*) *top*)
by (*metis dual-omega-iterate mult.right-neutral*)

subclass *ccpo*
apply *unfold-locales*
apply (*simp add*: *Sup-upper*)
using *Sup-least* **by** *auto*

end

class *post-mbt-algebra-ext* = *post-mbt-algebra* +
assumes *post-sub-fusion*: *post 1* ∗ *neg-assume q* ≤ *post* (*neg-assume q* ^ *o*)
begin

lemma *post-fusion*:
post (*neg-assume q* ^ *o*) = *post 1* ∗ *neg-assume q*
using *order.antisym neg-assumption post-dual-below-post-one post-sub-fusion*
by *auto*

lemma *post-dual-post-one*:
q ∈ *assumption* ⟹ *post 1* ∗ *q* ≤ *post* (*q* ^ *o*)
by (*metis assumption-neg-assume post-sub-fusion*)

end

instance *MonoTran* :: (*complete-boolean-algebra*) *post-mbt-algebra-ext*
proof
fix *q* :: *'a MonoTran*
show *post 1* ∗ *neg-assume q* ≤ *post* (*neg-assume q* ^ *o*)
proof (*unfold neg-assume-def*, *transfer*)
fix *f* :: *'a* ⇒ *'a*
assume *mono f*
have ∀ *x. top* ≤ −*f bot* ⊔ *x* ⟶ ¬ *f bot* ≤ *x* ⟶ *top* ≤ *bot*
by (*metis* (*no-types*, *lifting*) *double-compl inf.sup-bot-left inf-compl-bot sup.order-iff sup-bot-left sup-commute sup-inf-distrib1 top.extremum-uniqueI*)
hence *post-fun top* ∘ (*dual-fun f* ∘ *top*) ⊔ *id* ≤ *post-fun* (*f bot*)

by (*simp add*: *dual-fun-def le-fun-def post-fun-def*)
thus *post-fun* (*id top*) ∘ (*dual-fun f* ∘ *top*) ⊔ *id* ≤ *post-fun* (*dual-fun* ((*dual-fun f* ∘ *top*) ⊔ *id*) *top*)
by *simp*
qed
qed

class *complete-mbt-algebra-ext* = *complete-mbt-algebra* + *post-mbt-algebra-ext*

instance *MonoTran* :: (*complete-boolean-algebra*) *complete-mbt-algebra-ext* **..**

end

36 Instances of Monotonic Boolean Transformers

theory *Monotonic-Boolean-Transformers-Instances*

imports *Monotonic-Boolean-Transformers Pre-Post-Modal General-Refinement-Algebras*

begin

sublocale *mbt-algebra* < *mbta*: *bounded-idempotent-left-semiring*
apply *unfold-locales*
apply (*simp add*: *le-comp*)
apply (*simp add*: *sup-comp*)
apply *simp*
apply *simp*
apply *simp*
apply *simp*
by (*simp add*: *mult-assoc*)

sublocale *mbt-algebra* < *mbta-dual*: *bounded-idempotent-left-semiring* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *top* = *bot*
apply *unfold-locales*
using *inf.bounded-iff inf-le1 inf-le2 mbta.mult-right-isotone* **apply** *simp*
using *inf-comp* **apply** *blast*
apply *simp*
apply *simp*
apply *simp*
apply *simp*
by (*simp add*: *mult-assoc*)

sublocale *mbt-algebra* < *mbta*: *bounded-general-refinement-algebra* **where** *star* = *dual-star* **and** *Omega* = *dual-omega*
apply *unfold-locales*
using *dual-star-fix sup-commute* **apply** *force*
apply (*simp add*: *dual-star-least*)
using *dual-omega-fix sup-commute* **apply** *force*

by (*simp add*: *dual-omega-greatest sup-commute*)

sublocale *mbt-algebra* < *mbta-dual*: *bounded-general-refinement-algebra* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *Omega* = *omega* **and** *top* = *bot*
apply *unfold-locales*
using *order.eq-iff star-fix* **apply** *simp*
using *star-greatest* **apply** *simp*
using *inf-commute omega-fix* **apply** *fastforce*
by (*simp add*: *inf.sup-monoid.add-commute omega-least*)

Theorem 50.9(b)

sublocale *mbt-algebra* < *mbta*: *left-conway-semiring-L* **where** *circ* = *dual-star* **and** *L* = *bot*
apply *unfold-locales*
apply (*simp add*: *mbta.star-one*)
by *simp*

Theorem 50.8(a)

sublocale *mbt-algebra* < *mbta-dual*: *left-conway-semiring-L* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *L* = *bot*
apply *unfold-locales*
apply *simp*
by *simp*

Theorem 50.8(b)

sublocale *mbt-algebra* < *mbta-fix*: *left-conway-semiring-L* **where** *circ* = *dual-omega* **and** *L* = *top*
apply *unfold-locales*
apply (*simp add*: *mbta.Omega-one*)
by *simp*

Theorem 50.9(a)

sublocale *mbt-algebra* < *mbta-fix-dual*: *left-conway-semiring-L* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **and** *L* = *top*
apply *unfold-locales*
apply (*simp add*: *mbta-dual.star-one*)
by *simp*

sublocale *mbt-algebra* < *mbta*: *left-kleene-conway-semiring* **where** *circ* = *dual-star* **and** *star* = *dual-star* **..**

sublocale *mbt-algebra* < *mbta-dual*: *left-kleene-conway-semiring* **where** *circ* = *omega* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **..**

sublocale *mbt-algebra* < *mbta-fix*: *left-kleene-conway-semiring* **where** *circ* = *dual-omega* **and** *star* = *dual-star* **..**

sublocale *mbt-algebra* < *mbta-fix-dual*: *left-kleene-conway-semiring* **where** *circ* = *star* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *bot* = *top* **..**

sublocale *mbt-algebra* < *mbta*: *tests* **where** *uminus* = *neg-assert*
  apply *unfold-locales*
  apply (*simp add*: *mult-assoc*)
  apply (*metis neg-assertion assertion-inf-comp-eq inf-commute*)
  subgoal for *x y*
  proof −
    have $(x \,\hat{}\, o * bot \sqcup y * top) \sqcap ((x \,\hat{}\, o * bot \sqcup y \,\hat{}\, o * bot) \sqcap 1) = x \,\hat{}\, o * bot \sqcap 1$
      by (*metis inf-assoc dual-neg sup-bot-right sup-inf-distrib1*)
    thus *?thesis*
      by (*simp add*: *dual-inf dual-comp inf-comp sup-comp neg-assert-def*)
  qed
  apply (*simp add*: *neg-assertion*)
  using *assertion-inf-comp-eq inf-uminus neg-assertion* **apply** *force*
  apply (*simp add*: *neg-assert-def*)
  apply (*simp add*: *dual-inf dual-comp sup-comp neg-assert-def inf-sup-distrib2*)
  apply (*simp add*: *assertion-inf-comp-eq inf.absorb-iff1 neg-assertion*)
  using *inf.less-le-not-le* **by** *blast*

sublocale *mbt-algebra* < *mbta-dual*: *tests* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *uminus* = *neg-assume* **and** *bot* = *top*
  apply *unfold-locales*
  apply (*simp add*: *mult-assoc*)
  apply (*metis neg-assumption assumption-sup-comp-eq sup-commute*)
  subgoal for *x y*
  proof −
    have $(x \,\hat{}\, o * top \sqcap y * bot) \sqcup ((x \,\hat{}\, o * top \sqcap y \,\hat{}\, o * top) \sqcup 1) = x \,\hat{}\, o * top \sqcup 1$
      by (*metis dual-dual dual-neg-top inf-sup-distrib1 inf-top-right sup-assoc*)
    thus *?thesis*
      by (*simp add*: *dual-comp dual-sup inf-comp sup-comp neg-assume-def*)
  qed
  using *assumption-neg-assume comp-assumption neg-assumption* **apply** *blast*
  using *assumption-sup-comp-eq inf-uminus-assume neg-assumption* **apply** *fastforce*
  apply (*simp add*: *neg-assume-def*)
  apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assume-def sup-inf-distrib2*)
  apply (*simp add*: *assumption-sup-comp-eq neg-assumption sup.absorb-iff1*)
  using *inf.less-le-not-le* **by** *auto*

Theorem 51.2

sublocale *mbt-algebra* < *mbta*: *bounded-relative-antidomain-semiring* **where** $d = \lambda x\,.\,(x * top) \sqcap 1$ **and** *uminus* = *neg-assert* **and** $Z = bot$

apply *unfold-locales*
subgoal for x
proof $-$
have x ^ $o * bot \sqcap x \leq bot$
by (*metis dual-neg eq-refl inf.commute inf-mono mbta.top-right-mult-increasing*)
thus *?thesis*
by (*simp add*: *neg-assert-def inf-comp*)
qed
apply (*simp add*: *dual-comp dual-inf neg-assert-def sup-comp mult-assoc*)
apply *simp*
apply *simp*
apply (*simp add*: *dual-inf dual-comp sup-comp neg-assert-def inf-sup-distrib2*)
apply (*simp add*: *dual-sup inf-comp neg-assert-def inf.assoc*)
by (*simp add*: *dual-inf dual-comp sup-comp neg-assert-def*)

Theorem 51.1

sublocale *mbt-algebra* < *mbta-dual*: *bounded-relative-antidomain-semiring* **where** $d = \lambda x . (x * bot) \sqcup 1$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $uminus = neg\text{-}assume$ **and** $bot = top$ **and** $top = bot$ **and** $Z = top$
apply *unfold-locales*
subgoal for x
proof $-$
have $top \leq x$ ^ $o * top \sqcup x$
by (*metis dual-dual dual-neg-top mbta-dual.top-right-mult-increasing sup-commute sup-left-isotone*)
thus *?thesis*
by (*simp add*: *sup-comp neg-assume-def*)
qed
using *assume-bot dual-comp neg-assume-def sup-comp mult-assoc* **apply** *simp*
apply *simp*
apply *simp*
apply (*simp add*: *dual-inf dual-comp dual-sup inf-comp sup-comp neg-assume-def sup-inf-distrib2*)
apply (*simp add*: *dual-inf sup-comp neg-assume-def sup.assoc*)
by (*simp add*: *dual-comp dual-sup inf-comp neg-assume-def*)

sublocale *mbt-algebra* < *mbta*: *relative-domain-semiring-split* **where** $d = \lambda x . (x * top) \sqcap 1$ **and** $Z = bot$
apply *unfold-locales*
by *simp*

sublocale *mbt-algebra* < *mbta-dual*: *relative-domain-semiring-split* **where** $d = \lambda x . (x * bot) \sqcup 1$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $bot = top$ **and** $Z = top$
apply *unfold-locales*
by *simp*

sublocale *mbt-algebra* < *mbta*: *diamond-while* **where** $box = \lambda x\ y . neg\text{-}assert\ (x$

$*$ *neg-assert* y) **and** *circ* = *dual-star* **and** $d = \lambda x$. $(x * top) \sqcap 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * top) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcup$ (*neg-assert* $p * y$) **and** *pre* = $\lambda x\ y$. *wpt* $(x * y)$ **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. $((p * x)\ \widehat{}\otimes)$ $*$ *neg-assert* p **and** Z = *bot*
  apply *unfold-locales*
  apply *simp*
  apply *simp*
  apply (*rule wpt-def*)
  apply *simp*
  by *simp*

sublocale *mbt-algebra* < *mbta-dual*: *box-while* **where** *box* = $\lambda x\ y$. *neg-assume* ($x *$ *neg-assume* y) **and** *circ* = *omega* **and** $d = \lambda x$. $(x * bot) \sqcup 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * bot) \sqcup 1$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* ($x\ \widehat{}$ $o * y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. $((p * x)\ \widehat{}\omega) *$ *neg-assume* p **and** *bot* = *top* **and** *top* = *bot* **and** Z = *top*
  apply *unfold-locales*
  apply *simp*
  apply *simp*
  apply (*metis assume-bot dual-comp mbta-dual.a-mult-d-2 mbta-dual.d-def neg-assume-def wpb-def mult-assoc*)
  apply *simp*
  by *simp*

sublocale *mbt-algebra* < *mbta-fix*: *diamond-while* **where** *box* = $\lambda x\ y$. *neg-assert* ($x *$ *neg-assert* y) **and** *circ* = *dual-omega* **and** $d = \lambda x$. $(x * top) \sqcap 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * top) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcup$ (*neg-assert* $p *$ y) **and** *pre* = $\lambda x\ y$. *wpt* $(x * y)$ **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. $((p * x)\ \widehat{}\mho) *$ *neg-assert* p **and** Z = *bot*
  apply *unfold-locales*
  by *simp-all*

sublocale *mbt-algebra* < *mbta-fix-dual*: *box-while* **where** *box* = $\lambda x\ y$. *neg-assume* ($x *$ *neg-assume* y) **and** *circ* = *star* **and** $d = \lambda x$. $(x * bot) \sqcup 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * bot) \sqcup 1$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcap$ (*neg-assume* $p *$ y) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* ($x\ \widehat{}$ $o * y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. $((p * x)\ \widehat{}*) *$ *neg-assume* p **and** *bot* = *top* **and** *top* = *bot* **and** Z = *top*
  apply *unfold-locales*
  by *simp-all*

sublocale *mbt-algebra* < *mbta-pre*: *box-while* **where** *box* = $\lambda x\ y$. *neg-assert* ($x *$ *neg-assert* y) **and** *circ* = *dual-star* **and** $d = \lambda x$. $(x * top) \sqcap 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * top) \sqcap 1$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcup$ (*neg-assert* $p * y$) **and** *pre* = $\lambda x\ y$. *wpt* ($x\ \widehat{}$ $o * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. $((p * x)\ \widehat{}$ $\otimes) *$ *neg-assert* p **and** Z = *bot*
  apply *unfold-locales*
  apply (*metis dual-comp dual-dual dual-top inf-top-right*

mbta-dual.mult-right-dist-sup mult-1-left neg-assert-def top-comp wpt-def mult-assoc)
  apply *simp*
  by *simp*

sublocale *mbt-algebra* < *mbta-pre-dual*: *diamond-while* **where** *box* = λ*x y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *omega* **and** *d* = λ*x* . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λ*x y* . *wpb* (*x* ∗ *y*) **and** *uminus* = *neg-assume* **and** *while* = λ*p x* . ((*p* ∗ *x*) ⌢ω) ∗ *neg-assume p* **and** *bot* = *top* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  apply (*simp add*: *wpb-def*)
  apply *simp*
  by *simp*

sublocale *mbt-algebra* < *mbta-pre-fix*: *box-while* **where** *box* = λ*x y* . *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-omega* **and** *d* = λ*x* . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *pre* = λ*x y* . *wpt* (*x* ⌢ *o* ∗ *y*) **and** *uminus* = *neg-assert* **and** *while* = λ*p x* . ((*p* ∗ *x*) ⌢℧) ∗ *neg-assert p* **and** *Z* = *bot*
  apply *unfold-locales*
  by *simp-all*

sublocale *mbt-algebra* < *mbta-pre-fix-dual*: *diamond-while* **where** *box* = λ*x y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λ*x* . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = λ*x p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λ*x y* . *wpb* (*x* ∗ *y*) **and** *uminus* = *neg-assume* **and** *while* = λ*p x* . ((*p* ∗ *x*) ⌢∗) ∗ *neg-assume p* **and** *bot* = *top* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  by *simp-all*

sublocale *post-mbt-algebra* < *mbta*: *pre-post-spec-Hd* **where** *box* = λ*x y* . *neg-assert* (*x* ∗ *neg-assert y*) **and** *d* = λ*x* . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = λ*x y* . (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *pre* = λ*x y* . *wpt* (*x* ∗ *y*) **and** *pre-post* = λ*p q* . *p* ∗ *post q* **and** *uminus* = *neg-assert* **and** *Hd* = *post 1* **and** *Z* = *bot*
  apply *unfold-locales*
  apply (*metis mult.assoc mult.left-neutral post-1*)
  apply (*metis inf.commute inf-top-right mult.assoc mult.left-neutral post-2*)
  apply (*metis neg-assertion assertion-disjunctive disjunctiveD*)
  subgoal for *p x q*
  proof
    let *?pt* = *neg-assert p*
    let *?qt* = *neg-assert q*
    assume *?pt* ≤ *wpt* (*x* ∗ *?qt*)
    hence *?pt* ∗ *post ?qt* ≤ *x* ∗ *?qt* ∗ *top* ∗ *post ?qt* ⊓ *post ?qt*
      by (*metis mbta.mult-left-isotone wpt-def inf-comp mult.left-neutral*)
    thus *?pt* ∗ *post ?qt* ≤ *x*

by (*smt mbta.top-left-zero mult.assoc post-2 order-trans*)
next
let *?pt* = *neg-assert p*
let *?qt* = *neg-assert q*
assume *?pt* $*$ *post ?qt* $\leq$ *x*
thus *?pt* $\leq$ *wpt* (*x* $*$ *?qt*)
by (*smt mbta.a-d-closed post-1 mult-assoc mbta.diamond-left-isotone wpt-def*)
qed
by (*simp add*: *mbta-dual.mult-right-dist-sup*)

sublocale *post-mbt-algebra* < *mbta-dual*: *pre-post-spec-H* **where** *box* = λ*x y* . *neg-assume* (*x* $*$ *neg-assume y*) **and** *d* = λ*x* . (*x* $*$ *bot*) $\sqcup$ *1* **and** *diamond* = λ*x y* . (*x* $*$ *y* $*$ *bot*) $\sqcup$ *1* **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λ*x y* . *wpb* (*x* $\frown$ *o* $*$ *y*) **and** *pre-post* = λ*p q* . (*p* $\frown$ *o*) $*$ *post* (*q* $\frown$ *o*) **and** *uminus* = *neg-assume* **and** *bot* = *top* **and** *H* = *post 1* **and** *top* = *bot* **and** *Z* = *top*
proof
fix *p x q*
let *?pt* = *neg-assume p*
let *?qt* = *neg-assume q*
show *wpb* (*x* $\frown$ *o* $*$ *?qt*) $\leq$ *?pt* $\longleftrightarrow$ *?pt* $\frown$ *o* $*$ *post* (*?qt* $\frown$ *o*) $\leq$ *x*
proof
assume *wpb* (*x* $\frown$ *o* $*$ *?qt*) $\leq$ *?pt*
hence *?pt* $\frown$ *o* $*$ *post* (*?qt* $\frown$ *o*) $\leq$ (*x* $*$ (*?qt* $\frown$ *o*) $*$ *top* $\sqcap$ *1*) $*$ *post* (*?qt* $\frown$ *o*)
by (*smt wpb-def dual-le dual-comp dual-dual dual-one dual-sup dual-top mbta.mult-left-isotone*)
thus *?pt* $\frown$ *o* $*$ *post* (*?qt* $\frown$ *o*) $\leq$ *x*
by (*smt inf-comp mult-assoc top-comp mult.left-neutral post-2 order-trans*)
next
assume *1*: *?pt* $\frown$ *o* $*$ *post* (*?qt* $\frown$ *o*) $\leq$ *x*
have *?pt* $\frown$ *o* = *?pt* $\frown$ *o* $*$ *post* (*?qt* $\frown$ *o*) $*$ (*?qt* $\frown$ *o*) $*$ *top* $\sqcap$ *1*
by (*metis assert-iff-assume assertion-prop dual-dual mult-assoc neg-assumption post-1*)
thus *wpb* (*x* $\frown$ *o* $*$ *?qt*) $\leq$ *?pt*
using *1* **by** (*smt dual-comp dual-dual dual-le dual-one dual-sup dual-top wpb-def mbta.diamond-left-isotone*)
qed
show *post 1* $*$ *top* = *top*
by (*simp add*: *mbta.Hd-total*)
have *x* $*$ *?qt* $*$ *bot* $\sqcap$ (*post 1* $*$ *neg-assume ?qt*) = (*x* $*$ *neg-assume ?qt* $\frown$ *o* $*$ *top* $\sqcap$ *post 1*) $*$ *neg-assume ?qt*
by (*simp add*: *assume-bot mbta-dual.mult-right-dist-sup mult-assoc*)
also have ... $\leq$ *x* $*$ *neg-assume ?qt* $\frown$ *o*
by (*smt assumption-assertion-absorb dual-comp dual-dual mbta.mult-left-isotone mult.right-neutral mult-assoc neg-assumption post-2*)
also have ... $\leq$ *x*
by (*metis dual-comp dual-dual dual-le mbta.mult-left-sub-dist-sup-left mult.right-neutral neg-assume-def sup.commute*)
finally show *x* $*$ *?qt* $*$ *bot* $\sqcap$ (*post 1* $*$ *neg-assume ?qt*) $\leq$ *x*

.
qed

sublocale *post-mbt-algebra* $<$ *mbta-pre*: *pre-post-spec-H* **where** $box = \lambda x\ y\ .$ *neg-assert* $(x * \textit{neg-assert}\ y)$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .$ $(x * y * top) \sqcap 1$ **and** $pre = \lambda x\ y\ .\ wpt\ (x\ \hat{}\ o * y)$ **and** $\textit{pre-post} = \lambda p\ q\ .\ p\ \hat{}\ o *$ $(post\ q\ \hat{}\ o)$ **and** $uminus = \textit{neg-assert}$ **and** $H = post\ 1\ \hat{}\ o$ **and** $Z = bot$
proof
  fix $p\ x\ q$
  let $?pt = \textit{neg-assert}\ p$
  let $?qt = \textit{neg-assert}\ q$
  show $?pt \leq wpt\ (x\ \hat{}\ o * ?qt) \longleftrightarrow x \leq ?pt\ \hat{}\ o * (post\ ?qt\ \hat{}\ o)$
  proof
    assume $?pt \leq wpt\ (x\ \hat{}\ o * ?qt)$
    hence $?pt * post\ ?qt \leq (x\ \hat{}\ o * ?qt * top \sqcap 1) * post\ ?qt$
      by (*simp add*: *mbta-dual.mult-left-isotone wpt-def*)
    also have ... $\leq x\ \hat{}\ o$
      using *mbta.pre-pre-post wpt-def* **by** *auto*
    finally show $x \leq ?pt\ \hat{}\ o * (post\ ?qt\ \hat{}\ o)$
      by (*metis dual-le dual-comp dual-dual*)
  next
    assume $x \leq ?pt\ \hat{}\ o * (post\ ?qt\ \hat{}\ o)$
    hence $x * ?qt\ \hat{}\ o * bot \sqcup 1 \leq (?pt * post\ ?qt * ?qt * top \sqcap 1)\ \hat{}\ o$
      by (*smt* (*z3*) *inf.absorb-iff1 sup-inf-distrib2 dual-comp dual-inf dual-one dual-top mbta.mult-left-isotone*)
    also have ... $= ?pt\ \hat{}\ o$
      by (*simp add*: *mbta.diamond-a-export post-1*)
    finally show $?pt \leq wpt\ (x\ \hat{}\ o * ?qt)$
      by (*smt dual-comp dual-dual dual-le dual-neg-top dual-one dual-sup dual-top wpt-def*)
  qed
  show $post\ 1\ \hat{}\ o * bot = bot$
    by (*metis dual-comp dual-top mbta.Hd-total*)
  have $x\ \hat{}\ o * ?qt\ \hat{}\ o * bot \sqcap (post\ 1 * \textit{neg-assert}\ ?qt\ \hat{}\ o) \leq x\ \hat{}\ o * \textit{neg-assert}\ ?qt * \textit{neg-assert}\ ?qt\ \hat{}\ o$
    by (*smt* (*verit*, *del-insts*) *bot-comp inf.commute inf-comp inf-top-left mbta.mult-left-isotone mult.left-neutral mult-assoc neg-assert-def post-2*)
  also have ... $\leq x\ \hat{}\ o$
    by (*smt assert-iff-assume assumption-assertion-absorb dual-comp dual-dual le-comp mbta.a-below-one mult-assoc neg-assertion mult-1-right*)
  finally show $x \leq x * ?qt * top \sqcup post\ 1\ \hat{}\ o * \textit{neg-assert}\ ?qt$
    by (*smt dual-comp dual-dual dual-inf dual-le dual-top*)
qed

sublocale *post-mbt-algebra* $<$ *mbta-pre-dual*: *pre-post-spec-Hd* **where** $box = \lambda x\ y\ .\ \textit{neg-assume}\ (x * \textit{neg-assume}\ y)$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $less = greater$ **and** $\textit{less-eq} = \textit{greater-eq}$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x * y)$ **and** $\textit{pre-post} = \lambda p\ q\ .\ p * (post\ (q\ \hat{}\ o)\ \hat{}\ o)$ **and** $uminus = \textit{neg-assume}$ **and** $bot = top$ **and** $Hd = post\ 1\ \hat{}\ o$ **and** $top = bot$ **and** Z

$=$ *top*
apply *unfold-locales*
apply (*simp add*: *mbta-pre.H-zero-2*)
apply (*simp add*: *mbta-pre.H-greatest-finite*)
apply (*metis* (*no-types*, *lifting*) *dual-comp dual-dual dual-inf dual-top mbta-dual.mult-L-circ-mult mult-1-right neg-assume-def sup-commute sup-inf-distrib2*)
subgoal for $p\ x\ q$
proof
let $?pt = neg\text{-}assume\ p$
let $?qt = neg\text{-}assume\ q$
assume $wpb\ (x * ?qt) \leq ?pt$
hence $?pt\ \hat{}\ o * post\ (?qt\ \hat{}\ o) \leq (x\ \hat{}\ o * ?qt\ \hat{}\ o * top \sqcap 1) * post\ (?qt\ \hat{}\ o)$
by (*smt dual-comp dual-dual dual-le dual-one dual-sup dual-top le-comp-right wpb-def*)
also have $\ldots \leq x\ \hat{}\ o$
using *mbta-dual.mult-right-dist-sup post-2* **by** *force*
finally show $x \leq ?pt * post\ (?qt\ \hat{}\ o)\ \hat{}\ o$
by (*smt dual-comp dual-dual dual-le*)
next
let $?pt = neg\text{-}assume\ p$
let $?qt = neg\text{-}assume\ q$
assume $x \leq ?pt * post\ (?qt\ \hat{}\ o)\ \hat{}\ o$
thus $wpb\ (x * ?qt) \leq ?pt$
by (*metis dual-comp dual-dual dual-le mbta-dual.pre-post-galois*)
qed
by (*simp add*: *sup-comp*)

sublocale *post-mbt-algebra* $<$ *mbta-dual*: *pre-post-spec-whiledo* **where** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x\ \hat{}\ o * y)$ **and** $pre\text{-}post = \lambda p\ q\ .\ (p\ \hat{}\ o) * post\ (q\ \hat{}\ o)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $top = bot$ **..**

sublocale *post-mbt-algebra* $<$ *mbta-fix-dual*: *pre-post-spec-whiledo* **where** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x\ \hat{}\ o * y)$ **and** $pre\text{-}post = \lambda p\ q\ .\ (p\ \hat{}\ o) * post\ (q\ \hat{}\ o)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $top = bot$ **..**

sublocale *post-mbt-algebra* $<$ *mbta-pre*: *pre-post-spec-whiledo* **where** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x\ \hat{}\ o * y)$ **and** $pre\text{-}post = \lambda p\ q\ .\ p\ \hat{}\ o * (post\ q\ \hat{}\ o)$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$ **..**

sublocale *post-mbt-algebra* $<$ *mbta-pre-fix*: *pre-post-spec-whiledo* **where** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x\ \hat{}\ o * y)$ **and** $pre\text{-}post = \lambda p\ q\ .\ p\ \hat{}\ o * (post\ q\ \hat{}\ o)$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * neg\text{-}assert\ p$ **..**

sublocale *post-mbt-algebra* < *mbta-dual*: *pre-post-L* **where** *box* = λx *y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *omega* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λx *y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = λx *p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λx *y* . *wpb* (*x* ⌢ *o* ∗ *y*) **and** *pre-post* = λp *q* . (*p* ⌢ *o*) ∗ *post* (*q* ⌢ *o*) **and** *uminus* = *neg-assume* **and** *while* = λp *x* . ((*p* ∗ *x*) ⌢ ω) ∗ *neg-assume p* **and** *bot* = *top* **and** *L* = *bot* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  by *simp*

sublocale *post-mbt-algebra* < *mbta-fix-dual*: *pre-post-L* **where** *box* = λx *y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λx *y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *ite* = λx *p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λx *y* . *wpb* (*x* ⌢ *o* ∗ *y*) **and** *pre-post* = λp *q* . (*p* ⌢ *o*) ∗ *post* (*q* ⌢ *o*) **and** *uminus* = *neg-assume* **and** *while* = λp *x* . ((*p* ∗ *x*) ⌢ ∗) ∗ *neg-assume p* **and** *bot* = *top* **and** *L* = *top* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  by *simp*

sublocale *post-mbt-algebra* < *mbta-pre*: *pre-post-L* **where** *box* = λx *y* . *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-star* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = λx *y* . (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = λx *p y* . (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *pre* = λx *y* . *wpt* (*x* ⌢ *o* ∗ *y*) **and** *pre-post* = λp *q* . *p* ⌢ *o* ∗ (*post q* ⌢ *o*) **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = λp *x* . ((*p* ∗ *x*) ⌢ ⊗) ∗ *neg-assert p* **and** *L* = *bot* **and** *Z* = *bot*
  apply *unfold-locales*
  by *simp*

sublocale *post-mbt-algebra* < *mbta-pre-fix*: *pre-post-L* **where** *box* = λx *y* . *neg-assert* (*x* ∗ *neg-assert y*) **and** *circ* = *dual-omega* **and** *d* = λx . (*x* ∗ *top*) ⊓ *1* **and** *diamond* = λx *y* . (*x* ∗ *y* ∗ *top*) ⊓ *1* **and** *ite* = λx *p y* . (*p* ∗ *x*) ⊔ (*neg-assert p* ∗ *y*) **and** *pre* = λx *y* . *wpt* (*x* ⌢ *o* ∗ *y*) **and** *pre-post* = λp *q* . *p* ⌢ *o* ∗ (*post q* ⌢ *o*) **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* = λp *x* . ((*p* ∗ *x*) ⌢ ℧) ∗ *neg-assert p* **and** *L* = *top* **and** *Z* = *bot*
  apply *unfold-locales*
  by *simp*

sublocale *complete-mbt-algebra* < *mbta*: *complete-tests* **where** *uminus* = *neg-assert*
  apply *unfold-locales*
  apply (*smt mbta.test-set-def neg-assertion subset-eq Sup-assertion assertion-neg-assert*)
  apply (*simp add*: *Sup-upper*)
  by (*simp add*: *Sup-least*)

sublocale *complete-mbt-algebra* < *mbta-dual*: *complete-tests* **where** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *uminus* = *neg-assume* **and**

$bot = top$ **and** $Inf = Sup$ **and** $Sup = Inf$
  apply *unfold-locales*
  apply (*smt mbta-dual.test-set-def neg-assumption subset-eq Inf-assumption assumption-neg-assume*)
  apply (*simp add*: *Inf-lower*)
  by (*simp add*: *Inf-greatest*)

sublocale *complete-mbt-algebra* < *mbta*: *complete-antidomain-semiring* **where** $d = \lambda x . (x * top) \sqcap 1$ **and** $uminus = neg\text{-}assert$ **and** $Z = bot$
proof
  fix $f :: nat \Rightarrow 'a$
  let $?F = dual$ ‘ $\{f\ n \mid n . True\}$
  show *ascending-chain f* $\longrightarrow$ *neg-assert* (*complete-tests.Sum Sup f*) = *complete-tests.Prod Inf* ($\lambda n.$ *neg-assert* (*f n*))
  proof
    have *neg-assert* (*complete-tests.Sum Sup f*) $= 1 \sqcap (\bigsqcap x \in ?F . x * bot)$
      using *Inf-comp dual-Sup mbta.Sum-def neg-assert-def inf-commute* **by** *auto*
    also have ... $= (\bigsqcap x \in ?F . 1 \sqcap x * bot)$
      apply (*subst inf-Inf*)
      apply *blast*
      by (*simp add*: *image-image*)
    also have ... $= \bigsqcap \{f\ n\ \hat{}\ o * bot \sqcap 1 \mid n . True\}$
      apply (*rule arg-cong*[**where** *f*=*Inf*])
      using *inf-commute* **by** *auto*
    also have ... = *complete-tests.Prod Inf* ($\lambda n.$ *neg-assert* (*f n*))
      using *mbta.Prod-def neg-assert-def* **by** *auto*
    finally show *neg-assert* (*complete-tests.Sum Sup f*) = *complete-tests.Prod Inf* ($\lambda n.$ *neg-assert* (*f n*))
      .
  qed
  show *descending-chain f* $\longrightarrow$ *neg-assert* (*complete-tests.Prod Inf f*) = *complete-tests.Sum Sup* ($\lambda n.$ *neg-assert* (*f n*))
  proof
    have *neg-assert* (*complete-tests.Prod Inf f*) $= 1 \sqcap (\bigsqcup x \in ?F . x * bot)$
      using *Sup-comp dual-Inf mbta.Prod-def neg-assert-def inf-commute* **by** *auto*
    also have ... $= (\bigsqcup x \in ?F . 1 \sqcap x * bot)$
      by (*simp add*: *inf-Sup image-image*)
    also have ... $= \bigsqcup \{f\ n\ \hat{}\ o * bot \sqcap 1 \mid n. True\}$
      apply (*rule arg-cong*[**where** *f*=*Sup*])
      using *inf-commute* **by** *auto*
    also have ... = *complete-tests.Sum Sup* ($\lambda n.$ *neg-assert* (*f n*))
      using *mbta.Sum-def neg-assert-def* **by** *auto*
    finally show *neg-assert* (*complete-tests.Prod Inf f*) = *complete-tests.Sum Sup* ($\lambda n.$ *neg-assert* (*f n*))
      .
  qed
qed

sublocale *complete-mbt-algebra* < *mbta-dual*: *complete-antidomain-semiring*

where $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $uminus = neg\text{-}assume$ **and** $bot = top$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $Z = top$
proof
  fix $f :: nat \Rightarrow 'a$
  let $?F = dual\ `\ \{f\ n \mid n\ .\ True\}$
  show $ord.ascending\text{-}chain\ greater\text{-}eq\ f \longrightarrow neg\text{-}assume\ (complete\text{-}tests.Sum\ Inf\ f) = complete\text{-}tests.Prod\ Sup\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
  proof
    have $neg\text{-}assume\ (complete\text{-}tests.Sum\ Inf\ f) = 1 \sqcup (\bigsqcup x \in ?F\ .\ x * top)$
      using *mbta-dual.Sum-def neg-assume-def dual-Inf Sup-comp sup-commute* **by** *auto*
    also have ... $= (\bigsqcup x \in ?F\ .\ 1 \sqcup x * top)$
      apply (*subst sup-Sup*)
      apply *blast*
      by (*simp add*: *image-image*)
    also have ... $= \bigsqcup\{f\ n\ ^\frown o * top \sqcup 1 \mid n\ .\ True\}$
      apply (*rule arg-cong*[**where** *f*=*Sup*])
      using *sup-commute* **by** *auto*
    also have ... $= complete\text{-}tests.Prod\ Sup\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
      using *mbta-dual.Prod-def neg-assume-def* **by** *auto*
    finally show $neg\text{-}assume\ (complete\text{-}tests.Sum\ Inf\ f) = complete\text{-}tests.Prod\ Sup\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
      .
  qed
  show $ord.descending\text{-}chain\ greater\text{-}eq\ f \longrightarrow neg\text{-}assume\ (complete\text{-}tests.Prod\ Sup\ f) = complete\text{-}tests.Sum\ Inf\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
  proof
    have $neg\text{-}assume\ (complete\text{-}tests.Prod\ Sup\ f) = 1 \sqcup (\bigsqcap x \in ?F\ .\ x * top)$
      using *mbta-dual.Prod-def neg-assume-def dual-Inf dual-Sup Inf-comp sup-commute* **by** *auto*
    also have ... $= (\bigsqcap x \in ?F\ .\ 1 \sqcup x * top)$
      by (*simp add*: *sup-Inf image-image*)
    also have ... $= \bigsqcap\{f\ n\ ^\frown o * top \sqcup 1 \mid n.\ True\}$
      apply (*rule arg-cong*[**where** *f*=*Inf*])
      using *sup-commute* **by** *auto*
    also have ... $= complete\text{-}tests.Sum\ Inf\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
      using *mbta-dual.Sum-def neg-assume-def* **by** *auto*
    finally show $neg\text{-}assume\ (complete\text{-}tests.Prod\ Sup\ f) = complete\text{-}tests.Sum\ Inf\ (\lambda n.\ neg\text{-}assume\ (f\ n))$
      .
  qed
qed

sublocale *complete-mbt-algebra* < *mbta*: *diamond-while-program* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}star$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x * y)$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x)\ ^\frown\otimes) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Continuous$

and *Atomic-test* = *assertion* **and** *Z* = *bot*
  apply *unfold-locales*
  apply (*simp add*: *one-continuous*)
  by *simp-all*

sublocale *complete-mbt-algebra* < *mbta-dual*: *box-while-program* **where** *box* = $\lambda x\ y$. *neg-assume* ($x * $ *neg-assume* y) **and** *circ* = *omega* **and** $d = \lambda x$. ($x * $ *bot*) $\sqcup$ *1* **and** *diamond* = $\lambda x\ y$. ($x * y * $ *bot*) $\sqcup$ *1* **and** *ite* = $\lambda x\ p\ y$. ($p * x$) $\sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* (x ^ $o * y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. (($p * x$) ^ ω) $*$ *neg-assume* p **and** *bot* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  apply (*simp add*: *one-continuous*)
  by *simp-all*

sublocale *complete-mbt-algebra* < *mbta-fix*: *diamond-while-program* **where** *box* = $\lambda x\ y$. *neg-assert* ($x * $ *neg-assert* y) **and** *circ* = *dual-omega* **and** $d = \lambda x$. ($x * $ *top*) $\sqcap$ *1* **and** *diamond* = $\lambda x\ y$. ($x * y * $ *top*) $\sqcap$ *1* **and** *ite* = $\lambda x\ p\ y$. ($p * x$) $\sqcup$ (*neg-assert* $p * y$) **and** *pre* = $\lambda x\ y$. *wpt* ($x * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. (($p * x$) ^ $\mho$) $*$ *neg-assert* p **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assertion* **and** *Z* = *bot*
  apply *unfold-locales*
  apply (*simp add*: *one-co-continuous*)
  by *simp-all*

sublocale *complete-mbt-algebra* < *mbta-fix-dual*: *box-while-program* **where** *box* = $\lambda x\ y$. *neg-assume* ($x * $ *neg-assume* y) **and** *circ* = *star* **and** $d = \lambda x$. ($x * $ *bot*) $\sqcup$ *1* **and** *diamond* = $\lambda x\ y$. ($x * y * $ *bot*) $\sqcup$ *1* **and** *ite* = $\lambda x\ p\ y$. ($p * x$) $\sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* (x ^ $o * y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. (($p * x$) ^ $*$) $*$ *neg-assume* p **and** *bot* = *top* **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assumption* **and** *top* = *bot* **and** *Z* = *top*
  apply *unfold-locales*
  apply (*simp add*: *one-co-continuous*)
  by *simp-all*

sublocale *complete-mbt-algebra* < *mbta-pre*: *box-while-program* **where** *box* = $\lambda x\ y$. *neg-assert* ($x * $ *neg-assert* y) **and** *circ* = *dual-star* **and** $d = \lambda x$. ($x * $ *top*) $\sqcap$ *1* **and** *diamond* = $\lambda x\ y$. ($x * y * $ *top*) $\sqcap$ *1* **and** *ite* = $\lambda x\ p\ y$. ($p * x$) $\sqcup$ (*neg-assert* $p * y$) **and** *pre* = $\lambda x\ y$. *wpt* (x ^ $o * y$) **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x$. (($p * x$) ^ $\otimes$) $*$ *neg-assert* p **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **and** *Z* = *bot* **..**

sublocale *complete-mbt-algebra* < *mbta-pre-dual*: *diamond-while-program* **where** *box* = $\lambda x\ y$. *neg-assume* ($x * $ *neg-assume* y) **and** *circ* = *omega* **and** $d = \lambda x$. ($x * $ *bot*) $\sqcup$ *1* **and** *diamond* = $\lambda x\ y$. ($x * y * $ *bot*) $\sqcup$ *1* **and** *ite* = $\lambda x\ p\ y$. ($p * x$) $\sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* ($x * y$) **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. ((p

$* x) \ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assumption$ **and** $top = bot$ **and** $Z = top$ **..**

sublocale *complete-mbt-algebra* < *mbta-pre-fix*: *box-while-program* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}omega$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x \ \hat{}\ o * y)$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x) \ \hat{}\ \mho) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$ **..**

sublocale *complete-mbt-algebra* < *mbta-pre-fix-dual*: *diamond-while-program* **where** $box = \lambda x\ y\ .\ neg\text{-}assume\ (x * neg\text{-}assume\ y)$ **and** $circ = star$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x * y)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x) \ \hat{}\ *) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assumption$ **and** $top = bot$ **and** $Z = top$ **..**

Theorem 52

sublocale *complete-mbt-algebra* < *mbta*: *diamond-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}star$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x * y)$ **and** $star = dual\text{-}star$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x) \ \hat{}\ \otimes) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$
  apply *unfold-locales*
  by (*simp add*: *mbta.aL-one-circ mbta.star-one*)

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-dual*: *box-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assume\ (x * neg\text{-}assume\ y)$ **and** $circ = omega$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x \ \hat{}\ o * y)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x) \ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assumption$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $top = bot$ **and** $Z = top$
  apply *unfold-locales*
  using *mbta.top-greatest mbta.vector-bot-closed mbta-dual.aL-one-circ mbta-dual.a-top omega-one top-comp* **by** *auto*

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-fix*: *diamond-hoare-sound-2* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}omega$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x * y)$ **and** $star = dual\text{-}star$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x) \ \hat{}\ \mho) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$
proof (*unfold-locales*, *rule impI*)

fix $p\ q\ x$
let $?pt = neg\text{-}assert\ p$
let $?qt = neg\text{-}assert\ q$
assume $neg\text{-}assert\ ?pt * ?qt \leq x * ?qt * top \sqcap 1$
hence $?qt * top \leq x\ \hat{}\ \mho * ?pt * top$
by (*smt mbta.Omega-induct mbta.d-def mbta.d-mult-top mbta.mult-left-isotone mbta.shunting-top-1 mult.assoc*)
thus $mbta\text{-}fix.aL * ?qt \leq x\ \hat{}\ \mho * ?pt * top \sqcap 1$
by (*smt* (*z3*) *inf.absorb-iff1 inf.sup-monoid.add-commute inf-comp inf-le2 inf-left-commute inf-top-left mbta-fix.aL-one-circ mbta-pre-dual.top-left-zero mult-1-left neg-assert-def mult.assoc*)
qed

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-fix-dual*: *box-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assume\ (x * neg\text{-}assume\ y)$ **and** $circ = star$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x\ \hat{}\ o * y)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assumption$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $top = bot$ **and** $Z = top$
apply *unfold-locales*
by (*simp add*: *mbta-dual.star-one mbta-fix-dual.aL-one-circ*)

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre*: *box-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}star$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x\ \hat{}\ o * y)$ **and** $star = dual\text{-}star$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$
apply *unfold-locales*
using *mbta.star-one mbta-pre.aL-one-circ* **by** *auto*

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre-dual*: *diamond-hoare-sound-2* **where** $box = \lambda x\ y\ .\ neg\text{-}assume\ (x * neg\text{-}assume\ y)$ **and** $circ = omega$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x * y)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assumption$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $top = bot$ **and** $Z = top$
proof (*unfold-locales*, *rule impI*)
fix $p\ q\ x$
let $?pt = neg\text{-}assume\ p$
let $?qt = neg\text{-}assume\ q$
assume $x * ?qt * bot \sqcup 1 \leq neg\text{-}assume\ ?pt * ?qt$

hence $x * ?qt * bot \sqcap ?pt \leq ?qt$
by (*smt* (*z3*) *inf.absorb-iff1 inf-left-commute inf-commute inf-le1 le-supE mbta-dual.a-compl-intro mbta-dual.d-def order-trans*)
hence $(x * ?qt * bot \sqcap ?pt) * bot \leq ?qt * bot$
using *mbta.mult-left-isotone* **by** *blast*
hence $x \, ^\frown\omega * ?pt * bot \sqcup 1 \leq ?qt$
by (*smt bot-comp inf-comp sup-left-isotone mbta-dual.a-d-closed mult-assoc omega-least*)
thus $x \, ^\frown\omega * ?pt * bot \sqcup 1 \leq mbta\text{-}pre\text{-}dual.aL * ?qt$
by (*simp add*: *mbta-pre-dual.aL-one-circ*)
qed

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre-fix*: *box-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}omega$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x \, ^\frown o * y)$ **and** $star = dual\text{-}star$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x) \, ^\frown\mho) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$
apply *unfold-locales*
using *mbta.Omega-one mbta-pre-fix.aL-def* **by** *auto*

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre-fix-dual*: *diamond-hoare-sound* **where** $box = \lambda x\ y\ .\ neg\text{-}assume\ (x * neg\text{-}assume\ y)$ **and** $circ = star$ **and** $d = \lambda x\ .\ (x * bot) \sqcup 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * bot) \sqcup 1$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcap (neg\text{-}assume\ p * y)$ **and** $less = greater$ **and** $less\text{-}eq = greater\text{-}eq$ **and** $sup = inf$ **and** $pre = \lambda x\ y\ .\ wpb\ (x * y)$ **and** $uminus = neg\text{-}assume$ **and** $while = \lambda p\ x\ .\ ((p * x) \, ^\frown *) * neg\text{-}assume\ p$ **and** $bot = top$ **and** $Atomic\text{-}program = Co\text{-}continuous$ **and** $Atomic\text{-}test = assumption$ **and** $Inf = Sup$ **and** $Sup = Inf$ **and** $top = bot$ **and** $Z = top$
apply *unfold-locales*
by (*simp add*: *mbta-dual.star-one mbta-pre-fix-dual.aL-one-circ*)

Theorem 52

sublocale *complete-mbt-algebra* < *mbta*: *diamond-hoare-valid* **where** $box = \lambda x\ y\ .\ neg\text{-}assert\ (x * neg\text{-}assert\ y)$ **and** $circ = dual\text{-}star$ **and** $d = \lambda x\ .\ (x * top) \sqcap 1$ **and** $diamond = \lambda x\ y\ .\ (x * y * top) \sqcap 1$ **and** $hoare\text{-}triple = \lambda p\ x\ q\ .\ p \leq wpt(x * q)$ **and** $ite = \lambda x\ p\ y\ .\ (p * x) \sqcup (neg\text{-}assert\ p * y)$ **and** $pre = \lambda x\ y\ .\ wpt\ (x * y)$ **and** $star = dual\text{-}star$ **and** $uminus = neg\text{-}assert$ **and** $while = \lambda p\ x\ .\ ((p * x) \, ^\frown\otimes) * neg\text{-}assert\ p$ **and** $Atomic\text{-}program = Continuous$ **and** $Atomic\text{-}test = assertion$ **and** $Z = bot$
apply *unfold-locales*
apply (*simp add*: *mbta.aL-zero*)
using *mbta.aL-zero* **apply** *blast*
subgoal for $x\ t$
proof
assume *1*: $x \in while\text{-}program.While\text{-}program\ (*)\ neg\text{-}assert\ Continuous\ assertion\ (\lambda p\ x\ .\ (p * x) \, ^\frown\otimes * neg\text{-}assert\ p)\ (\lambda x\ p\ y\ .\ p * x \sqcup neg\text{-}assert\ p * y) \wedge ascending\text{-}chain\ t \wedge tests.test\text{-}seq\ neg\text{-}assert\ t$

have $x \in$ *Continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=$\lambda x\ y$. *wpt* $(x * y)$ **and** *while*=$\lambda p\ x$. $((p * x)\ ^\frown\otimes) *$ *neg-assert p*])
apply *unfold-locales*
using *1* **apply** *blast*
apply *simp*
using *mult-continuous* **apply** *blast*
apply (*metis assertion-continuous mbta.test-expression-test mult-continuous neg-assertion sup-continuous*)
by (*metis assertion-continuous dual-star-continuous mbta.test-expression-test mult-continuous neg-assertion*)
thus $x *$ *complete-tests.Sum Sup t* = *complete-tests.Sum Sup* $(\lambda n.\ x * t\ n)$
using *1* **by** (*smt continuous-dist-ascending-chain SUP-cong mbta.Sum-range*)
qed
using *wpt-def* **by** *auto*

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-dual*: *box-hoare-valid* **where** *box* = $\lambda x\ y$. *neg-assume* $(x *$ *neg-assume* $y)$ **and** *circ* = *omega* **and** $d = \lambda x$. $(x * bot) \sqcup 1$ **and** *diamond* = $\lambda x\ y$. $(x * y * bot) \sqcup 1$ **and** *hoare-triple* = $\lambda p\ x\ q$. *wpb*$(x\ ^\frown o * q) \le p$ **and** *ite* = $\lambda x\ p\ y$. $(p * x) \sqcap$ (*neg-assume* $p * y$) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y$. *wpb* $(x\ ^\frown o * y)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x$. $((p * x)\ ^\frown\omega) *$ *neg-assume p* **and** *bot* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *top* = *bot* **and** *Z* = *top*
proof
fix *p x q t*
show *neg-assume* $q \le$ *neg-assume* $p *$ *neg-assume* $(x *$ *neg-assume* (*neg-assume* q)) $\longrightarrow$ *neg-assume* $q \sqcap$ *whiledo.aL* $(\lambda p\ x.\ (p * x)\ ^\frown\omega *$ *neg-assume* p) $(\lambda x\ y.$ *wpb* $(x\ ^\frown o * y))$ $1 \le$ *neg-assume* $(x\ ^\frown\omega *$ *neg-assume* (*neg-assume* p))
proof
let *?pt* = *neg-assume p*
let *?qt* = *neg-assume q*
assume *?qt* $\le$ *?pt* $*$ *neg-assume* $(x *$ *neg-assume ?qt*)
also have ... $\le x\ ^\frown o *$ *?qt* $\sqcup$ *?pt*
by (*smt assumption-sup-comp-eq sup-left-isotone mbta.zero-right-mult-decreasing mbta-dual.pre-def neg-assume-def neg-assumption sup.commute sup.left-commute sup.left-idem wpb-def*)
finally show *?qt* $\sqcap$ *mbta-dual.aL* $\le$ *neg-assume* $(x\ ^\frown\omega *$ *neg-assume ?pt*)
by (*smt dual-dual dual-omega-def dual-omega-greatest le-infI1 mbta-dual.a-d-closed mbta-dual.d-isotone mbta-dual.pre-def wpb-def*)
qed
show *whiledo.aL* $(\lambda p\ x.\ (p * x)\ ^\frown\omega *$ *neg-assume p*) $(\lambda x\ y.$ *wpb* $(x\ ^\frown o * y))$ 1 = *top* $\vee$ *whiledo.aL* $(\lambda p\ x.\ (p * x)\ ^\frown\omega *$ *neg-assume p*) $(\lambda x\ y.$ *wpb* $(x\ ^\frown o * y))$ $1 = 1$
using *mbta-dual.L-def mbta-dual.aL-one-circ mbta-dual.a-top* **by** *auto*
show $x \in$ *while-program.While-program* $(*)$ *neg-assume Continuous assumption* $(\lambda p\ x.\ (p * x)\ ^\frown\omega *$ *neg-assume p*) $(\lambda x\ p\ y.\ p * x \sqcap$ *neg-assume* $p * y) \wedge$ *ord.descending-chain* $(\lambda x\ y.\ y \le x)$ $t \wedge$ *tests.test-seq neg-assume t* $\longrightarrow x *$

complete-tests.Prod Sup t = *complete-tests.Prod Sup* ($\lambda n.$ *x* ∗ *t n*)
proof
assume *1*: *x* ∈ *while-program.While-program* (∗) *neg-assume Continuous assumption* (λp *x* . (*p* ∗ *x*) ⌢ω ∗ *neg-assume p*) (λx *p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*)) ∧ *ord.descending-chain greater-eq t* ∧ *tests.test-seq neg-assume t*
have *x* ∈ *Continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=λx *y* . *wpb* (*x* ⌢ *o* ∗ *y*) **and** *while*=λp *x* . ((*p* ∗ *x*) ⌢ω) ∗ *neg-assume p*])
apply *unfold-locales*
using *1* **apply** *blast*
apply *simp*
apply (*simp add*: *mult-continuous*)
apply (*metis assumption-continuous mbta-dual.test-expression-test mult-continuous neg-assumption inf-continuous*)
by (*metis assumption-continuous omega-continuous mbta-dual.test-expression-test mult-continuous neg-assumption*)
thus *x* ∗ *complete-tests.Prod Sup t* = *complete-tests.Prod Sup* ($\lambda n.$ *x* ∗ *t n*)
using *1* **by** (*smt ord.descending-chain-def ascending-chain-def continuous-dist-ascending-chain SUP-cong mbta-dual.Prod-range*)
qed
show (*wpb* (*x* ⌢ *o* ∗ *q*) ≤ *p*) = (*neg-assume* (*x* ∗ *neg-assume q*) ≤ *p*)
by (*simp add*: *mbta-dual.pre-def*)
qed

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre-fix-dual*: *diamond-hoare-valid* **where** *box* = λx *y* . *neg-assume* (*x* ∗ *neg-assume y*) **and** *circ* = *star* **and** *d* = λx . (*x* ∗ *bot*) ⊔ *1* **and** *diamond* = λx *y* . (*x* ∗ *y* ∗ *bot*) ⊔ *1* **and** *hoare-triple* = λp *x q* . *wpb*(*x* ∗ *q*) ≤ *p* **and** *ite* = λx *p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*) **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = λx *y* . *wpb* (*x* ∗ *y*) **and** *uminus* = *neg-assume* **and** *while* = λp *x* . ((*p* ∗ *x*) ⌢∗) ∗ *neg-assume p* **and** *bot* = *top* **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *top* = *bot* **and** *Z* = *top*
apply *unfold-locales*
using *mbta-dual.star-one mbta-pre-fix-dual.aL-one-circ* **apply** *simp*
using *mbta-pre-fix-dual.aL-zero* **apply** *blast*
subgoal for *x t*
proof
assume *1*: *x* ∈ *while-program.While-program* (∗) *neg-assume Co-continuous assumption* (λp *x* . (*p* ∗ *x*) ⌢∗ ∗ *neg-assume p*) (λx *p y* . (*p* ∗ *x*) ⊓ (*neg-assume p* ∗ *y*)) ∧ *ord.ascending-chain greater-eq t* ∧ *tests.test-seq neg-assume t*
have *x* ∈ *Co-continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=λx *y* . *wpb* (*x* ∗ *y*) **and** *while*=λp *x* . ((*p* ∗ *x*) ⌢∗) ∗ *neg-assume p*])
apply *unfold-locales*
using *1* **apply** *blast*
apply *simp*
apply (*simp add*: *mult-co-continuous*)
apply (*metis assumption-co-continuous mbta-dual.test-expression-test*

mult-co-continuous neg-assumption inf-co-continuous)
by (*metis assumption-co-continuous star-co-continuous mbta-dual.test-expression-test mult-co-continuous neg-assumption*)
thus $x * $ *complete-tests.Sum Inf* $t =$ *complete-tests.Sum Inf* $(\lambda n.\ x * t\ n)$
using *1* **by** (*smt descending-chain-def ord.ascending-chain-def co-continuous-dist-descending-chain INF-cong mbta-dual.Sum-range*)
qed
using *wpb-def* **by** *auto*

Theorem 52

sublocale *complete-mbt-algebra* < *mbta-pre-fix*: *box-hoare-valid* **where** *box* $= \lambda x\ y$. *neg-assert* $(x *$ *neg-assert* $y)$ **and** *circ* = *dual-omega* **and** $d = \lambda x$. $(x *$ *top*$) \sqcap 1$ **and** *diamond* $= \lambda x\ y$. $(x * y *$ *top*$) \sqcap 1$ **and** *hoare-triple* $= \lambda p\ x\ q$. $p \leq$ *wpt*$(x \,\hat{}\, o * q)$ **and** *ite* $= \lambda x\ p\ y$. $(p * x) \sqcup$ (*neg-assert* $p * y$) **and** *pre* $= \lambda x\ y$. *wpt* $(x \,\hat{}\, o * y)$ **and** *star* = *dual-star* **and** *uminus* = *neg-assert* **and** *while* $= \lambda p\ x$. $((p * x)\ \hat{}\mho) *$ *neg-assert* p **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assertion* **and** Z = *bot*
proof
fix $p\ x\ q\ t$
show *neg-assert* $p *$ *neg-assert* $(x *$ *neg-assert* (*neg-assert* q)) $\leq$ *neg-assert* q $\longrightarrow$ *neg-assert* $(x\ \hat{}\mho *$ *neg-assert* (*neg-assert* p)) $\leq$ *neg-assert* $q \sqcup$ *whiledo.aL* $(\lambda p\ x.\ (p * x)\ \hat{}\mho *$ *neg-assert* $p)$ $(\lambda x\ y.$ *wpt* $(x \,\hat{}\, o * y))$ *1*
proof
let *?pt* = *neg-assert* p
let *?qt* = *neg-assert* q
assume *1*: *?pt* $*$ *neg-assert* $(x *$ *?qt*$) \leq$ *?qt*
have $x \,\hat{}\, o *$ *?qt* $\sqcap$ *?pt* $\leq$ *?pt* $*$ *neg-assert* $(x *$ *neg-assert* *?qt*)
by (*smt* (*z3*) *inf.boundedI inf.coboundedI inf.sup-monoid.add-commute le-infI2 inf-comp mbta.tests-dual.sub-commutative mbta.top-right-mult-increasing mbta-pre.pre-def mult.left-neutral mult-assoc top-comp wpt-def*)
also have ... $\leq$ *?qt*
using *1* **by** *simp*
finally have $(x \,\hat{}\, o)\ \hat{}\,\omega *$ *?pt* $*$ *top* $\leq$ *?qt* $*$ *top*
using *mbta.mult-left-isotone omega-least* **by** *blast*
hence *neg-assert* $(x\ \hat{}\mho *$ *neg-assert* *?pt*) $\leq$ *?qt*
by (*smt dual-omega-def inf-mono mbta.d-a-closed mbta.d-def mbta-pre.pre-def order-refl wpt-def mbta.a-d-closed*)
thus *neg-assert* $(x\ \hat{}\mho *$ *neg-assert* *?pt*) $\leq$ *?qt* $\sqcup$ *mbta-pre-fix.aL*
using *le-supI1* **by** *blast*
qed
show *whiledo.aL* $(\lambda p\ x.\ (p * x)\ \hat{}\mho *$ *neg-assert* $p)$ $(\lambda x\ y.$ *wpt* $(x \,\hat{}\, o * y))$ *1* = *bot* $\vee$ *whiledo.aL* $(\lambda p\ x.\ (p * x)\ \hat{}\mho *$ *neg-assert* $p)$ $(\lambda x\ y.$ *wpt* $(x \,\hat{}\, o * y))$ *1* = *1*
using *mbta.Omega-one mbta.a-top mbta-dual.vector-bot-closed mbta-pre-fix.aL-one-circ* **by** *auto*
show $x \in$ *while-program.While-program* $(*)$ *neg-assert Co-continuous assertion* $(\lambda p\ x.\ (p * x)\ \hat{}\mho *$ *neg-assert* $p)$ $(\lambda x\ p\ y.\ p * x \sqcup$ *neg-assert* $p * y) \wedge$ *descending-chain* $t \wedge$ *tests.test-seq neg-assert* $t \longrightarrow x *$ *complete-tests.Prod Inf* t = *complete-tests.Prod Inf* $(\lambda n.\ x * t\ n)$
proof

assume *1*: $x \in$ *while-program.While-program* $(*)$ *neg-assert Co-continuous assertion* $(\lambda p\ x\ .\ (p * x)\ \hat{}\ \mho * \textit{neg-assert}\ p)$ $(\lambda x\ p\ y\ .\ p * x \sqcup \textit{neg-assert}\ p * y) \wedge$ *descending-chain* $t \wedge$ *tests.test-seq neg-assert* t
have $x \in$ *Co-continuous*
apply (*induct x rule*: *while-program.While-program.induct*[**where** *pre*=$\lambda x\ y$. *wpt* $(x\ \hat{}\ o * y)$ **and** *while*=$\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * \textit{neg-assert}\ p$])
apply *unfold-locales*
using *1* **apply** *blast*
apply *simp*
apply (*simp add*: *mult-co-continuous*)
apply (*metis assertion-co-continuous mbta.test-expression-test mult-co-continuous neg-assertion sup-co-continuous*)
by (*metis assertion-co-continuous dual-omega-co-continuous mbta.test-expression-test mult-co-continuous neg-assertion*)
thus $x *$ *complete-tests.Prod Inf* $t =$ *complete-tests.Prod Inf* $(\lambda n.\ x * t\ n)$
using *1* **by** (*smt descending-chain-def co-continuous-dist-descending-chain INF-cong mbta.Prod-range*)
qed
show $(p \leq \textit{wpt}\ (x\ \hat{}\ o * q)) = (p \leq \textit{neg-assert}\ (x * \textit{neg-assert}\ q))$
by (*simp add*: *mbta-pre.pre-def*)
qed

sublocale *complete-mbt-algebra* < *mbta-dual*: *pre-post-spec-hoare* **where** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcap (\textit{neg-assume}\ p * y)$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y\ .\ \textit{wpb}\ (x\ \hat{}\ o * y)$ **and** *pre-post* = $\lambda p\ q\ .\ (p\ \hat{}\ o) * \textit{post}\ (q\ \hat{}\ o)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \omega) * \textit{neg-assume}\ p$ **and** *bot* = *top* **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *top* = *bot* **..**

sublocale *complete-mbt-algebra* < *mbta-fix-dual*: *pre-post-spec-hoare* **where** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcap (\textit{neg-assume}\ p * y)$ **and** *less* = *greater* **and** *less-eq* = *greater-eq* **and** *sup* = *inf* **and** *pre* = $\lambda x\ y\ .\ \textit{wpb}\ (x\ \hat{}\ o * y)$ **and** *pre-post* = $\lambda p\ q\ .\ (p\ \hat{}\ o) * \textit{post}\ (q\ \hat{}\ o)$ **and** *uminus* = *neg-assume* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ *) * \textit{neg-assume}\ p$ **and** *bot* = *top* **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assumption* **and** *Inf* = *Sup* **and** *Sup* = *Inf* **and** *top* = *bot* **..**

sublocale *complete-mbt-algebra* < *mbta-pre*: *pre-post-spec-hoare* **where** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (\textit{neg-assert}\ p * y)$ **and** *pre* = $\lambda x\ y\ .\ \textit{wpt}\ (x\ \hat{}\ o * y)$ **and** *pre-post* = $\lambda p\ q\ .\ p\ \hat{}\ o * (\textit{post}\ q\ \hat{}\ o)$ **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \otimes) * \textit{neg-assert}\ p$ **and** *Atomic-program* = *Continuous* **and** *Atomic-test* = *assertion* **..**

sublocale *complete-mbt-algebra* < *mbta-pre-fix*: *pre-post-spec-hoare* **where** *ite* = $\lambda x\ p\ y\ .\ (p * x) \sqcup (\textit{neg-assert}\ p * y)$ **and** *pre* = $\lambda x\ y\ .\ \textit{wpt}\ (x\ \hat{}\ o * y)$ **and** *pre-post* = $\lambda p\ q\ .\ p\ \hat{}\ o * (\textit{post}\ q\ \hat{}\ o)$ **and** *uminus* = *neg-assert* **and** *while* = $\lambda p\ x\ .\ ((p * x)\ \hat{}\ \mho) * \textit{neg-assert}\ p$ **and** *Atomic-program* = *Co-continuous* **and** *Atomic-test* = *assertion* **..**

end

References

[1] R. Berghammer and W. Guttmann. Closure, properties and closure properties of multirelations. In W. Kahl, M. Winter, and J. N. Oliveira, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2015)*, volume 9348 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2015.

[2] R. Berghammer and W. Guttmann. An algebraic approach to multirelations and their properties. *Journal of Logical and Algebraic Methods in Programming*, 88:45–63, 2017.

[3] W. Guttmann. General correctness algebra. In R. Berghammer, A. M. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science (RelMiCS/AKA 2009)*, volume 5827 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2009.

[4] W. Guttmann. Partial, total and general correctness. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction (MPC 2010)*, volume 6120 of *Lecture Notes in Computer Science*, pages 157–177. Springer, 2010.

[5] W. Guttmann. Unifying recursion in partial, total and general correctness. In S. Qin, editor, *Unifying Theories of Programming, Third International Symposium (UTP 2010)*, volume 6445 of *Lecture Notes in Computer Science*, pages 207–225. Springer, 2010.

[6] W. Guttmann. Fixpoints for general correctness. *Journal of Logic and Algebraic Programming*, 80(6):248–265, 2011.

[7] W. Guttmann. Towards a typed omega algebra. In H. de Swart, editor, *Relational and Algebraic Methods in Computer Science (RAMiCS 2011)*, volume 6663 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2011.

[8] W. Guttmann. Algebras for iteration and infinite computations. *Acta Inf.*, 49(5):343–359, 2012.

[9] W. Guttmann. Typing theorems of omega algebra. *Journal of Logic and Algebraic Programming*, 81(6):643–659, 2012.

[10] W. Guttmann. Unifying correctness statements. In J. Gibbons and P. Nogueira, editors, *Mathematics of Program Construction (MPC 2012)*, volume 7342 of *Lecture Notes in Computer Science*, pages 198–219. Springer, 2012.

[11] W. Guttmann. Unifying lazy and strict computations. In W. Kahl and T. G. Griffin, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2012)*, volume 7560 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2012.

[12] W. Guttmann. Extended designs algebraically. *Sci. Comput. Programming*, 78(11):2064–2085, 2013.

[13] W. Guttmann. Algebras for correctness of sequential computations. *Sci. Comput. Programming*, 85(Part B):224–240, 2014.

[14] W. Guttmann. Extended conscriptions algebraically. In P. Höfner, P. Jipsen, W. Kahl, and M. E. Müller, editors, *Relational and Algebraic Methods in Computer Science (RAMiCS 2014)*, volume 8428 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2014.

[15] W. Guttmann. Multirelations with infinite computations. *Journal of Logical and Algebraic Methods in Programming*, 83(2):194–211, 2014.

[16] W. Guttmann. *Algebras for Iteration, Infinite Executions and Correctness of Sequential Computations.* Habilitationsschrift, Universität Ulm, 2015.

[17] W. Guttmann. Infinite executions of lazy and strict computations. *Journal of Logical and Algebraic Methods in Programming*, 84(3):326–340, 2015.

[18] W. Guttmann. Isabelle/HOL theories of algebras for iteration, infinite executions and correctness of sequential computations. Technical Report TR-COSC 02/15, University of Canterbury, 2015.

[19] W. Guttmann. An algebraic approach to computations with progress. *Journal of Logical and Algebraic Methods in Programming*, 85(4):520–539, 2016.

[20] W. Guttmann, G. Struth, and T. Weber. Automating algebraic methods in Isabelle. In S. Qin and Z. Qiu, editors, *Formal Methods and Software Engineering (ICFEM 2011)*, volume 6991 of *Lecture Notes in Computer Science*, pages 617–632. Springer, 2011.