Modal quantales, involutive Quantales, Dedekind Quantales

Cameron Calk and Georg Struth

September 13, 2023

Abstract

This AFP entry provides mathematical components for modal quantales, involutive quantales and Dedekind quantales. Modal quantales are simple extensions of modal Kleene algebras useful for the verification of recursive programs. Involutive quantales appear in the study of C^*-algebras. Dedekind quantales are relatives of Tarski's relation algebras, hence relevant to program verification and beyond that to higher rewriting. We also provide components for weaker variants such as Kleene algebras with converse and modal Kleene algebras with converse.

Contents

1 Introductory Remarks

In this AFP entry we provide mathematical components for modal quantales, involutive quantales and Dedekind quantales. Modal quantales are simple extensions of modal Kleene algebras that can be used in the verification of recursive programs [6]. Involutive quantales appear in the study of C^*-algebras [8]. Dedekind quantales, categorifications of which are known as *modular quantaloids* [9], are relatives of Tarski's relation algebras [11], and hence relevant to program verification as well. We also provide components for weaker variants such as Kleene algebras and modal Kleene algebras with converse.

Our main interest in these structures comes from recent applications in higher-dimensional rewriting [2, 3], where they are used in coherence proofs for rewriting systems based on computads or polygraphs. This includes proofs of coherent Church-Rosser theorems and coherent Newman's lemmas. A more long-term programme considers the formalisation of algebraic aspects of higher rewriting with proof assistants.

Modal quantales have previously been studied in [4], where it is shown, for instance, that any category can be lifted to a modal quantale at powerset level. Such lifting results will be formalised in a companion AFP entry.

Dedekind quantales give also rise to intuitionistic modal algebras, as the results in this AFP entry show. In particular, the set of all subidentities or coreflexives of a Dedekind quantale forms a complete Heyting algebra (aka frame or locale), on which modal box and diamond operators can be defined. A paper explaining these results is in preparation [7]. A further application of Dedekind quantales lies once again in higher-dimensional rewriting [2, 3]. Any groupoid, in particular, can be lifted to a Dedekind quantale at powerset level, a result which will once again be formalised in a companion AFP entry.

Our components build on extant AFP components for Kleene algebras [1], modal Kleene algebras [5] and quantales [10].

Georg Struth is grateful for an invited professorship at École polytechnique and a fellowship at the Collegium de Lyon, Institute of Advanced Study, during which most of this formalisation work has been done.

2 Modal Kleene algebra based on domain and range semirings

theory *Modal-Kleene-Algebra-Var*
imports *KAD.Domain-Semiring KAD.Range-Semiring*

begin

notation *domain-op* (*dom*)
notation *range-op* (*cod*)

subclass (**in** *domain-semiring*) *dioid-one-zero*..

subclass (**in** *range-semiring*) *dioid-one-zero*
by *unfold-locales simp*

2.1 Modal semirings

The following modal semirings are based on domain and range semirings instead of antidomain and antirange semirings, as in the AFP entry for Kleene algebra with domain.

class *dr-modal-semiring* = *domain-semiring* + *range-semiring* +
assumes *dc-compat* [*simp*]: *dom* (*cod x*) = *cod x*
and *cd-compat* [*simp*]: *cod* (*dom x*) = *dom x*

begin

sublocale *msrdual*: *dr-modal-semiring* (+) $\lambda x\ y.\ y \cdot x$ *1 0 cod* (≤) (<) *dom*
by *unfold-locales simp-all*

lemma *d-cod-fix*: (*dom x* = *x*) = (*x* = *cod x*)
by (*metis local.cd-compat local.dc-compat*)

lemma *local-var*: ($x \cdot y$ = *0*) = (*cod x* · *dom y* = *0*)
using *local.dom-weakly-local local.rdual.dom-weakly-local* **by** *force*

lemma *fbdia-conjugation*: (*fd x* (*dom p*) · *dom q* = *0*) = (*dom p* · *bd x* (*dom q*) = *0*)
by (*metis local.bd-def local.cd-compat local.ddual.mult-assoc local.dom-weakly-local local.fd-def local.rdual.dom-weakly-local local.rdual.dsg4*)

end

2.2 Modal Kleene algebra

class *dr-modal-kleene-algebra* = *dr-modal-semiring* + *kleene-algebra*

end

3 Kleene algebra with converse

theory *Kleene-Algebra-Converse*
imports *Kleene-Algebra.Kleene-Algebra*

begin

We start from involutive dioids and Kleene algebra and then add a so-called strong Gelfand property to obtain an operation of converse that is closer to algebras of paths and relations.

3.1 Involutive Kleene algebra

class *invol-op* =
fixes *invol* :: $'a \Rightarrow 'a$ (-° [*101*] *100*)

class *involutive-dioid* = *dioid-one-zero* + *invol-op* +
assumes *inv-invol* [*simp*]: $(x^\circ)^\circ = x$
and *inv-contrav* [*simp*]: $(x \cdot y)^\circ = y^\circ \cdot x^\circ$
and *inv-sup* [*simp*]: $(x + y)^\circ = x^\circ + y^\circ$

begin

lemma *inv-zero* [*simp*]: $0^\circ = 0$
proof−
have $0^\circ = (0^\circ \cdot 0)^\circ$
by *simp*
also have ... $= 0^\circ \cdot (0^\circ)^\circ$
using *local.inv-contrav* **by** *blast*
also have ... $= 0^\circ \cdot 0$
by *simp*
also have ... $= 0$
by *simp*
finally show *?thesis*.
qed

lemma *inv-one* [*simp*]: $1^\circ = 1$
proof−
have $1^\circ = 1^\circ \cdot (1^\circ)^\circ$
by *simp*
also have ... $= (1^\circ \cdot 1)^\circ$
using *local.inv-contrav* **by** *presburger*
also have ... $= (1^\circ)^\circ$
by *simp*
also have ... $= 1$
by *simp*
finally show *?thesis*.
qed

lemma *inv-iso*: $x \leq y \Longrightarrow x^\circ \leq y^\circ$
  by (*metis local.inv-sup local.less-eq-def*)

lemma *inv-adj*: $(x^\circ \leq y) = (x \leq y^\circ)$
  using *inv-iso* **by** *fastforce*

end

Here is an equivalent axiomatisation from Doornbos, Backhouse and van der Woude's paper on a calculational approach to mathematical induction.

class *involutive-dioid-alt* = *dioid-one-zero* +
  fixes *inv-alt* :: $'a \Rightarrow 'a$
  assumes *inv-alt*: (*inv-alt* $x \leq y$) = ($x \leq$ *inv-alt* y)
  and *inv-alt-contrav* [*simp*]: *inv-alt* $(x \cdot y)$ = *inv-alt* $y \cdot$ *inv-alt* x

begin

lemma *inv-alt-invol* [*simp*]: *inv-alt* (*inv-alt* x) = x
proof−
  have *inv-alt* (*inv-alt* x) $\leq x$
    by (*simp add*: *inv-alt*)
  thus *?thesis*
    by (*meson inv-alt order-antisym*)
qed

lemma *inv-alt-add*: *inv-alt* $(x + y)$ = *inv-alt* x + *inv-alt* y
proof−
  {**fix** z
  have (*inv-alt* $(x + y) \leq z$) = ($x + y \leq$ *inv-alt* z)
    by (*simp add*: *inv-alt*)
  also have ... = ($x \leq$ *inv-alt* $z \wedge y \leq$ *inv-alt* z)
    by *simp*
  also have ... = (*inv-alt* $x \leq z \wedge$ *inv-alt* $y \leq z$)
    by (*simp add*: *inv-alt*)
  also have ... = (*inv-alt* x + *inv-alt* $y \leq z$)
    by *force*
  finally have (*inv-alt* $(x + y) \leq z$) = (*inv-alt* x + *inv-alt* $y \leq z$)**.**}
  thus *?thesis*
    using *order-antisym* **by** *blast*
qed

sublocale *altinv*: *involutive-dioid* - - - - - - *inv-alt*
  by *unfold-locales* (*simp-all add*: *inv-alt-add*)

end

sublocale *involutive-dioid* $\subseteq$ *altinv*: *involutive-dioid-alt* - - - - - - *invol*
  by *unfold-locales* (*simp-all add*: *local.inv-adj*)

class *involutive-kleene-algebra* = *involutive-dioid* + *kleene-algebra*

begin

lemma *inv-star*: $(x^\star)^\circ = (x^\circ)^\star$
proof (*rule order.antisym*)
  have $((x^\circ)^\star)^\circ = (1 + (x^\circ)^\star \cdot x^\circ)^\circ$
    by *simp*
  also have $\ldots = 1 + (x^\circ)^\circ \cdot ((x^\circ)^\star)^\circ$
    using *local.inv-contrav local.inv-one local.inv-sup* **by** *presburger*
  finally have $1 + x \cdot ((x^\circ)^\star)^\circ \leq ((x^\circ)^\star)^\circ$
    by *simp*
  hence $x^\star \leq ((x^\circ)^\star)^\circ$
    using *local.star-inductl* **by** *force*
  thus $(x^\star)^\circ \leq (x^\circ)^\star$
    by (*simp add*: *local.inv-adj*)
next
  have $(x^\star)^\circ = (1 + x^\star \cdot x)^\circ$
    by *simp*
  also have $\ldots = 1 + x^\circ \cdot (x^\star)^\circ$
    using *local.inv-contrav local.inv-one local.inv-sup* **by** *presburger*
  finally have $1 + x^\circ \cdot (x^\star)^\circ \leq (x^\star)^\circ$
    by *simp*
  thus $(x^\circ)^\star \leq (x^\star)^\circ$
    using *local.star-inductl* **by** *force*
qed

end

3.2 Kleene algebra with converse

The name "strong Gelfand property" has been borrowed from Palmigiano and Re.

class *dioid-converse* = *involutive-dioid* +
  assumes *strong-gelfand*: $x \leq x \cdot x^\circ \cdot x$

lemma (**in** *dioid-converse*) *subid-conv*: $x \leq 1 \Longrightarrow x^\circ = x$
proof (*rule order.antisym*)
  assume *h*: $x \leq 1$
  have $x \leq x \cdot x^\circ \cdot x$
    by (*simp add*: *local.strong-gelfand*)
  also have $\ldots \leq 1 \cdot x^\circ \cdot 1$
    using *h local.mult-isol-var* **by** *blast*
  also have $\ldots = x^\circ$
    by *simp*
  finally show $x \leq x^\circ$
    by *simp*
  thus $x^\circ \leq x$
    by (*simp add*: *local.inv-adj*)

qed

class *kleene-algebra-converse* = *involutive-kleene-algebra* + *dioid-converse*

end

4 Modal Kleene algebra with converse

theory *Modal-Kleene-Algebra-Converse*
 imports *Modal-Kleene-Algebra-Var Kleene-Algebra-Converse*

begin

Here we mainly study the interaction of converse with domain and codomain.

4.1 Involutive modal Kleene algebras

class *involutive-domain-semiring* = *domain-semiring* + *involutive-dioid*

begin

notation *domain-op* (*dom*)

lemma *strong-conv-conv*: $dom\ x \leq x \cdot x^\circ \Longrightarrow x \leq x \cdot x^\circ \cdot x$
proof−
 assume *h*: $dom\ x \leq x \cdot x^\circ$
 have $x = dom\ x \cdot x$
  by *simp*
 also have $\ldots \leq x \cdot x^\circ \cdot x$
  using *h local.mult-isor* **by** *presburger*
 finally show *?thesis*.
qed

end

class *involutive-dr-modal-semiring* = *dr-modal-semiring* + *involutive-dioid*

class *involutive-dr-modal-kleene-algebra* = *involutive-dr-modal-semiring* + *kleene-algebra*

4.2 Modal semirings algebras with converse

class *dr-modal-semiring-converse* = *dr-modal-semiring* + *dioid-converse*

begin

lemma *d-conv* [*simp*]: $(dom\ x)^\circ = dom\ x$
proof−
 have $dom\ x \leq dom\ x \cdot (dom\ x)^\circ \cdot dom\ x$
  by (*simp add*: *local.strong-gelfand*)

also have $\ldots \leq 1 \cdot (dom\ x)^\circ \cdot 1$
by (*simp add*: *local.subid-conv*)
finally have *a*: $dom\ x \leq (dom\ x)^\circ$
by *simp*
hence $(dom\ x)^\circ \leq dom\ x$
by (*simp add*: *local.inv-adj*)
thus *?thesis*
using *a* **by** *auto*
qed

lemma *cod-conv*: $(cod\ x)^\circ = cod\ x$
by (*metis d-conv local.dc-compat*)

lemma *d-conv-cod* [*simp*]: $dom\ (x^\circ) = cod\ x$
proof−
have $dom\ (x^\circ) = dom\ ((x \cdot cod\ x)^\circ)$
by *simp*
also have $\ldots = dom\ ((cod\ x)^\circ \cdot x^\circ)$
using *local.inv-contrav* **by** *presburger*
also have $\ldots = dom\ (cod\ x \cdot x^\circ)$
by (*simp add*: *cod-conv*)
also have $\ldots = dom\ (dom\ (cod\ x) \cdot x^\circ)$
by *simp*
also have $\ldots = dom\ (cod\ x) \cdot dom\ (x^\circ)$
using *local.dsg3* **by** *blast*
also have $\ldots = cod\ x \cdot dom\ (x^\circ)$
by *simp*
also have $\ldots = cod\ x \cdot cod\ (dom\ (x^\circ))$
by *simp*
also have $\ldots = cod\ (x \cdot cod\ (dom\ (x^\circ)))$
using *local.rdual.dsg3* **by** *presburger*
also have $\ldots = cod\ (x \cdot dom\ (x^\circ))$
by *simp*
also have $\ldots = cod\ ((x^\circ)^\circ \cdot (dom\ (x^\circ))^\circ)$
by *simp*
also have $\ldots = cod\ ((dom\ (x^\circ) \cdot x^\circ)^\circ)$
using *local.inv-contrav* **by** *presburger*
also have $\ldots = cod\ ((x^\circ)^\circ)$
by *simp*
also have $\ldots = cod\ x$
by *simp*
finally show *?thesis*.
qed

lemma *cod-conv-d*: $cod\ (x^\circ) = dom\ x$
by (*metis d-conv-cod local.inv-invol*)

lemma $dom\ y = y \Longrightarrow fd\ (x^\circ)\ y = bd\ x\ y$
proof−

assume *h*: *dom y* = *y*
have *fd* (x°) *y* = *dom* (x° · *dom y*)
by (*simp add*: *local.fd-def*)
also have ... = *dom* ((*dom y* · *x*)$^\circ$)
by *simp*
also have ... = *cod* (*dom y* · *x*)
using *d-conv-cod* **by** *blast*
also have ... = *bd x y*
by (*simp add*: *h local.bd-def*)
finally show *?thesis*.
qed

lemma *dom y* = *y* ⟹ *bd* (x°) *y* = *fd x y*
by (*metis cod-conv-d d-conv local.bd-def local.fd-def local.inv-contrav*)

end

4.3 Modal Kleene algebras with converse

class *dr-modal-kleene-algebra-converse* = *dr-modal-semiring-converse* + *kleene-algebra*

class *dr-modal-semiring-strong-converse* = *involutive-dr-modal-semiring* +
assumes *weak-dom-def*: *dom x* ≤ *x* · x°
and *weak-cod-def*: *cod x* ≤ x° · *x*

subclass (**in** *dr-modal-semiring-strong-converse*) *dr-modal-semiring-converse*
by *unfold-locales* (*metis local.ddual.mult-isol-var local.dsg1 local.eq-refl local.weak-dom-def*)

class *dr-modal-kleene-algebra-strong-converse* = *dr-modal-semiring-strong-converse*
+ *kleene-algebra*

end

5 Modal quantales

theory *Modal-Quantale*
imports *Quantales.Quantale-Star Modal-Kleene-Algebra-Var KAD.Modal-Kleene-Algebra*

begin

5.1 Simplified modal semirings and Kleene algebras

The previous formalisation of modal Kleene algebra in the AFP adds two compatibility axioms between domain and codomain when combining an antidomain semiring with an antirange semiring. But these are unnecessary. They are derivable from the other axioms. Thus I provide a simpler axiomatisation that should eventually replace the one in the AFP.

class *modal-semiring-simp* = *antidomain-semiring* + *antirange-semiring*

lemma (**in** *modal-semiring-simp*) *dr-compat* [*simp*]: $d\ (r\ x) = r\ x$
proof−
have *a*: $ar\ x \cdot d\ (r\ x) = 0$
using *local.ads-d-def local.ars-r-def local.dpdz.dom-weakly-local* **by** *auto*
have $r\ x \cdot d\ (r\ x) \cdot ar\ x \leq r\ x \cdot ar\ x$
by (*simp add*: *local.a-subid-aux2 local.ads-d-def local.mult-isor*)
hence *b*: $r\ x \cdot d\ (r\ x) \cdot ar\ x = 0$
by (*simp add*: *local.ardual.am2 local.ars-r-def local.join.bot-unique*)
have $d\ (r\ x) = (ar\ x + r\ x) \cdot d\ (r\ x)$
using *local.add-comm local.ardual.ans3 local.ars-r-def local.mult-1-left* **by** *presburger*
also have $\ldots = ar\ x \cdot d\ (r\ x) + r\ x \cdot d\ (r\ x)$
by *simp*
also have $\ldots = r\ x \cdot d\ (r\ x)$
by (*simp add*: *a*)
also have $\ldots = r\ x \cdot d\ (r\ x) \cdot (ar\ x + r\ x)$
using *local.add-comm local.ardual.ans3 local.ars-r-def* **by** *auto*
also have $\ldots = r\ x \cdot d\ (r\ x) \cdot ar\ x + r\ x \cdot d\ (r\ x) \cdot r\ x$
by *simp*
also have $\ldots = r\ x \cdot d\ (r\ x) \cdot r\ x$
using *b* **by** *auto*
also have $\ldots = r\ x$
by (*metis local.ads-d-def local.am3 local.ardual.a-mult-idem local.ars-r-def local.ds.ddual.mult-assoc*)
finally show *?thesis*
by *simp*
qed

lemma (**in** *modal-semiring-simp*) *rd-compat* [*simp*]: $r\ (d\ x) = d\ x$
by (*smt* (*verit*) *local.a-mult-idem local.ads-d-def local.am2 local.ardual.dpdz.dom-weakly-local local.ars-r-def local.dr-compat local.kat-3-equiv′*)

subclass (**in** *modal-semiring-simp*) *modal-semiring*
apply *unfold-locales* **by** *simp-all*

class *modal-kleene-algebra-simp* = *modal-semiring-simp* + *kleene-algebra*

subclass (**in** *modal-kleene-algebra-simp*) *modal-kleene-algebra*..

5.2 Domain quantales

class *domain-quantale* = *unital-quantale* + *domain-op* +
assumes *dom-absorb*: $x \leq dom\ x \cdot x$
and *dom-local*: $dom\ (x \cdot dom\ y) = dom\ (x \cdot y)$
and *dom-add*: $dom\ (x \sqcup y) = dom\ x \sqcup dom\ y$
and *dom-subid*: $dom\ x \leq 1$
and *dom-zero* [*simp*]: $dom\ \bot = \bot$

The definition is that of a domain semiring. I cannot extend the quantale class with respect to domain semirings because of different operations are used for addition/sup. The following sublocale statement brings all those properties into scope.

sublocale *domain-quantale* ⊆ *dqmsr*: *domain-semiring* (⊔) (·) *1* ⊥ *dom* (≤) (<)
 by *unfold-locales* (*simp-all add*: *dom-add dom-local dom-absorb sup.absorb2 dom-subid*)

sublocale *domain-quantale* ⊆ *dqmka*: *domain-kleene-algebra* (⊔) (·) *1* ⊥ *dom* (≤) (<) *qstar*..

typedef (**overloaded**) *′a d-element* = {*x* :: *′a* :: *domain-quantale*. *dom x* = *x*}
 using *dqmsr.dom-one* **by** *blast*

setup-lifting *type-definition-d-element*

instantiation *d-element* :: (*domain-quantale*) *bounded-lattice*

begin

lift-definition *less-eq-d-element* :: *′a d-element* ⇒ *′a d-element* ⇒ *bool* **is** (≤) **.**

lift-definition *less-d-element* :: *′a d-element* ⇒ *′a d-element* ⇒ *bool* **is** (<) **.**

lift-definition *bot-d-element* :: *′a d-element* **is** ⊥
 by *simp*

lift-definition *top-d-element* :: *′a d-element* **is** *1*
 by *simp*

lift-definition *inf-d-element* :: *′a d-element* ⇒ *′a d-element* ⇒ *′a d-element* **is** (·)
 by (*metis dqmsr.dom-mult-closed*)

lift-definition *sup-d-element* :: *′a d-element* ⇒ *′a d-element* ⇒ *′a d-element* **is** (⊔)
 by *simp*

instance
 apply (*standard*; *transfer*)
 apply (*simp-all add*: *less-le-not-le*)
 apply (*metis dqmsr.dom-subid-aux2′′*)
 apply (*metis dqmsr.dom-subid-aux1′′*)
 apply (*metis dqmsr.dom-glb-eq*)
 by (*metis dqmsr.dom-subid*)

end

instance *d-element* :: (*domain-quantale*) *distrib-lattice*
 by (*standard*, *transfer*, *metis dqmsr.dom-distrib*)

context *domain-quantale*

begin

lemma *dom-top* [*simp*]: *dom* $\top$ = *1*
proof–
  have *1* $\leq$ $\top$
    by *simp*
  hence *dom 1* $\leq$ *dom* $\top$
    using *local.dqmsr.d-iso* **by** *blast*
  thus *?thesis*
    by (*simp add*: *local.dual-order.antisym*)
qed

lemma *dom-top2*: $x \cdot \top \leq$ *dom* $x \cdot \top$
proof–
  have $x \cdot \top =$ *dom* $x \cdot x \cdot \top$
    by *simp*
  also have ... $\leq$ *dom* $x \cdot \top \cdot \top$
    using *local.dqmsr.d-restrict-iff-1 local.top-greatest local.top-times-top mult-assoc*
by *presburger*
  finally show *?thesis*
    by (*simp add*: *local.mult.semigroup-axioms semigroup.assoc*)
qed

lemma *weak-twisted*: $x \cdot$ *dom* $y \leq$ *dom* $(x \cdot y) \cdot x$
  using *local.dqmsr.fd-def local.dqmsr.fdemodalisation2 local.eq-refl* **by** *blast*

lemma *dom-meet*: *dom* $x \cdot$ *dom* $y =$ *dom* $x \sqcap$ *dom* y
  apply (*rule order.antisym*)
   apply (*simp add*: *local.dqmsr.dom-subid-aux2 local.dqmsr.dom-subid-aux2''*)
  by (*smt* (*z3*) *local.dom-absorb local.dqmsr.dom-iso local.dqmsr.dom-subid-aux2 local.dqmsr.dsg3 local.dqmsr.dsg4 local.dual-order.antisym local.inf.cobounded1 local.inf.cobounded2 local.psrpq.mult-isol-var*)

lemma *dom-meet-pres*: *dom* (*dom* $x \sqcap$ *dom* y) = *dom* $x \sqcap$ *dom* y
  using *dom-meet local.dqmsr.dom-mult-closed* **by** *presburger*

lemma *dom-meet-distl*: *dom* $x \cdot (y \sqcap z) =$ (*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$)
proof–
  have *a*: *dom* $x \cdot (y \sqcap z) \leq$ (*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$)
    using *local.mult-isol* **by** *force*
  have (*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$) = *dom* ((*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$)) $\cdot$ ((*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$))
    by *simp*
  also have ... $\leq$ *dom* ((*dom* $x \cdot y$)) $\cdot$ ((*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$))
    using *calculation local.dqmsr.dom-iso local.dqmsr.dom-llp2 local.inf.cobounded1*
by *presburger*
  also have ... $\leq$ *dom* $x \cdot$ ((*dom* $x \cdot y$) $\sqcap$ (*dom* $x \cdot z$))
    by (*metis local.dqmsr.domain-1'' local.dqmsr.domain-invol local.mult-isor*)
  also have ... $\leq$ *dom* $x \cdot (y \sqcap z)$

by (*meson local.dqmsr.dom-subid-aux2 local.inf-mono local.order-refl local.psrpq.mult-isol-var*)
finally show *?thesis*
using *a local.dual-order.antisym* **by** *blast*
qed

lemma *dom-meet-approx*: $dom\ ((dom\ x \cdot y) \sqcap (dom\ x \cdot z)) \leq dom\ x$
by (*metis dom-meet-distl local.dqmsr.domain-1'' local.dqmsr.domain-invol*)

lemma *dom-inf-pres-aux*: $Y \neq \{\} \Longrightarrow dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \leq dom\ x$
proof−
assume $Y \neq \{\}$
have $\forall z \in Y.\ dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \leq dom\ (dom\ x \cdot z)$
by (*meson local.INF-lower local.dqmsr.dom-iso*)
hence $\forall z \in Y.\ dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \leq dom\ x \cdot dom\ z$
by *fastforce*
hence $\forall z \in Y.\ dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \leq dom\ x$
using *dom-meet* **by** *fastforce*
thus $dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \leq dom\ x$
using ‹$Y \neq \{\}$› **by** *blast*
qed

lemma *dom-inf-pres-aux2*: $(\bigsqcap y \in Y.\ dom\ x \cdot y) \leq \bigsqcap Y$
by (*simp add*: *local.INF-lower2 local.dqmsr.dom-subid-aux2 local.le-Inf-iff*)

lemma *dom-inf-pres*: $Y \neq \{\} \Longrightarrow dom\ x \cdot (\bigsqcap Y) = (\bigsqcap y \in Y.\ dom\ x \cdot y)$
proof−
assume *hyp*: $Y \neq \{\}$
have *a*: $dom\ x \cdot (\bigsqcap Y) \leq (\bigsqcap y \in Y.\ dom\ x \cdot y)$
by (*simp add*: *Setcompr-eq-image local.Inf-subdistl*)
have $(\bigsqcap y \in Y.\ dom\ x \cdot y) = dom\ (\bigsqcap y \in Y.\ dom\ x \cdot y) \cdot (\bigsqcap y \in Y.\ dom\ x \cdot y)$
by *simp*
also have $\ldots \leq dom\ x \cdot (\bigsqcap y \in Y.\ dom\ x \cdot y)$
using *dom-inf-pres-aux hyp local.mult-isor* **by** *blast*
also have $\ldots \leq dom\ x \cdot \bigsqcap Y$
by (*simp add*: *dom-inf-pres-aux2 local.psrpq.mult-isol-var*)
finally show *?thesis*
using *a order.antisym* **by** *blast*
qed

lemma $dom\ (\bigsqcap X) \leq \bigsqcap\ (dom\ `\ X)$
by (*simp add*: *local.INF-greatest local.Inf-lower local.dqmsr.dom-iso*)

The domain operation need not preserve arbitrary sups, though this property holds, for instance, in quantales of binary relations. I do not aim at a stronger axiomatisation in this theory.

lemma *dom-top-pres*: $(x \leq dom\ y \cdot x) = (x \leq dom\ y \cdot \top)$
apply *standard*
apply (*meson local.dqmsr.ddual.mult-isol-var local.dual-order.eq-iff local.dual-order.trans local.top-greatest*)

using *local.dqmsr.dom-iso local.dqmsr.dom-llp* **by** *fastforce*

lemma *dom-lla-var*: $(dom\ x \leq dom\ y) = (x \leq dom\ y \cdot \top)$
using *dom-top-pres local.dqmsr.dom-llp* **by** *force*

lemma $dom\ (1 \sqcap x) = 1 \sqcap x \Longrightarrow x \leq 1 \Longrightarrow dom\ x = x$
using *local.inf-absorb2* **by** *force*

lemma *dom-meet-sub*: $dom\ (x \sqcap y) \leq dom\ x \sqcap dom\ y$
by (*simp add*: *local.dqmsr.d-iso*)

lemma *dom-dist1*: $dom\ x \sqcup (dom\ y \sqcap dom\ z) = (dom\ x \sqcup dom\ y) \sqcap (dom\ x \sqcup dom\ z)$
by (*metis dom-meet local.dom-add local.dqmsr.dom-distrib*)

lemma *dom-dist2*: $dom\ x \sqcap (dom\ y \sqcup dom\ z) = (dom\ x \sqcap dom\ y) \sqcup (dom\ x \sqcap dom\ z)$
by (*metis dom-meet local.dom-add local.sup-distl*)

abbreviation $fd' \equiv dqmsr.fd$

definition $bb\ x\ y = \bigsqcup\{dom\ z\ |z.\ fd'\ x\ z \leq dom\ y\}$

lemma *fd′-bb-galois-aux*: $fd'\ x\ (dom\ p) \leq dom\ q \Longrightarrow dom\ p \leq bb\ x\ (dom\ q)$
by (*simp add*: *bb-def local.SUP-upper setcompr-eq-image*)

lemma *dom-iso-var*: $(\bigsqcup x \in X.\ dom\ x) \leq dom\ (\bigsqcup x \in X.\ dom\ x)$
by (*meson local.SUP-le-iff local.dom-subid local.dqmsr.domain-subid*)

lemma *dom-iso-var2*: $(\bigsqcup x \in X.\ dom\ x) \leq dom\ (\bigsqcup x \in X.\ x)$
by (*simp add*: *local.SUP-le-iff local.Sup-upper local.dqmsr.dom-iso*)

end

5.3 Codomain quantales

class *codomain-quantale* = *unital-quantale* + *range-op* +
assumes *cod-absorb*: $x \leq x \cdot cod\ x$
and *cod-local*: $cod\ (cod\ x \cdot y) = cod\ (x \cdot y)$
and *cod-add*: $cod\ (x \sqcup y) = cod\ x \sqcup cod\ y$
and *cod-subid*: $cod\ x \leq 1$
and *cod-zero*: $cod\ \bot = \bot$

sublocale *codomain-quantale* ⊆ *coddual*: *domain-quantale range-op* - $\lambda x\ y.\ y \cdot x$ - - - - - - - -
by *unfold-locales* (*auto simp*: *mult-assoc cod-subid cod-zero cod-add cod-local cod-absorb Sup-distr Sup-distl*)

abbreviation (**in** *codomain-quantale*) $bd' \equiv coddual.fd'$

definition (**in** *codomain-quantale*) *fb x y* = $\bigsqcup\{$*cod z* |*z*. *bd′ x z* ≤ *cod y*$\}$

lemma (**in** *codomain-quantale*) *bd′-fb-galois-aux*: *bd′ x* (*cod p*) ≤ *cod q* ⟹ *cod p* ≤ *fb x* (*cod q*)
using *local.coddual.bb-def local.coddual.fd′-bb-galois-aux local.fb-def* **by** *auto*

5.4 Modal quantales

class *dc-modal-quantale* = *domain-quantale* + *codomain-quantale* +
assumes *dc-compat* [*simp*]: *dom* (*cod x*) = *cod x*
and *cd-compat* [*simp*]: *cod* (*dom x*) = *dom x*

sublocale *dc-modal-quantale* ⊆ *mqs*: *dr-modal-kleene-algebra* (⊔) (·) *1* ⊥ (≤) (<) *qstar dom cod*
by *unfold-locales simp-all*

sublocale *dc-modal-quantale* ⊆ *mqdual*: *dc-modal-quantale* - λ*x y*. *y* · *x* - - - - - - - - *dom cod*
by *unfold-locales simp-all*

lemma (**in** *dc-modal-quantale*) *x* · ⊤ = *dom x* · ⊤

oops

lemma (**in** *dc-modal-quantale*) ⊤ · *x* = ⊤ · *cod x*

oops

5.5 Antidomain and anticodomain quantales

notation *antidomain-op* (*adom*)

class *antidomain-quantale* = *unital-quantale* + *antidomain-op* +
assumes *as1* [*simp*]: *adom x* · *x* = ⊥
and *as2* [*simp*]: *adom* (*x* · *y*) ≤ *adom* (*x* · *adom* (*adom y*))
and *as3* [*simp*]: *adom* (*adom x*) ⊔ *adom x* = *1*

definition (**in** *antidomain-quantale*) *ddom* = *adom* ∘ *adom*

sublocale *antidomain-quantale* ⊆ *adqmsr*: *antidomain-semiring adom* (⊔) (·) *1* ⊥ (≤) (<)
by *unfold-locales* (*simp-all add*: *local.sup.absorb2*)

sublocale *antidomain-quantale* ⊆ *adqmka*: *antidomain-kleene-algebra adom* (⊔) (·) *1* ⊥ (≤) (<) *qstar*..

sublocale *antidomain-quantale* ⊆ *addq*: *domain-quantale ddom*
by *unfold-locales* (*simp-all add*: *ddom-def local.adqmsr.ans-d-def*)

notation *antirange-op* (*acod*)

class *anticodomain-quantale* = *unital-quantale* + *antirange-op* +
assumes *ars1* [*simp*]: *x* · *acod x* = ⊥
and *ars2* [*simp*]: *acod* (*x* · *y*) ≤ *acod* (*acod* (*acod x*) · *y*)
and *ars3* [*simp*]: *acod* (*acod x*) ⊔ *acod x* = *1*

sublocale *anticodomain-quantale* ⊆ *acoddual*: *antidomain-quantale acod* - *λx y. y* · *x* - - - - - - - -
by *unfold-locales* (*auto simp*: *mult-assoc Sup-distl Sup-distr*)

definition (**in** *anticodomain-quantale*) *ccod* = *acod* ∘ *acod*

sublocale *anticodomain-quantale* ⊆ *acdqmsr*: *antirange-semiring* (⊔) (·) *1* ⊥ *acod* (≤) (<)**..**

sublocale *anticodomain-quantale* ⊆ *acdqmka*: *antirange-kleene-algebra* (⊔) (·) *1* ⊥ (≤) (<) *qstar acod***..**

sublocale *anticodomain-quantale* ⊆ *acddq*: *codomain-quantale* - - - - - - - - - - *λ x. acod* (*acod x*)
by *unfold-locales* (*simp-all add*: *local.acoddual.adqmsr.ans-d-def*)

class *modal-quantale* = *antidomain-quantale* + *anticodomain-quantale*

sublocale *modal-quantale* ⊆ *mmqs*: *modal-kleene-algebra-simp* (⊔) (·) *1* ⊥ (≤) (<) *qstar adom acod***..**

sublocale *modal-quantale* ⊆ *mmqdual*: *modal-quantale* - *λx y. y* · *x* - - - - - - - - *adom acod*
by *unfold-locales simp-all*

end

6 Quantales with converse

theory *Quantale-Converse*
imports *Modal-Quantale Modal-Kleene-Algebra-Converse*

begin

6.1 Properties of unital quantales

These properties should eventually added to the quantales AFP entry.

lemma (**in** *quantale*) *bres-bot-top* [*simp*]: ⊥ → ⊤ = ⊤
by (*simp add*: *local.bres-galois-imp local.order.antisym*)

lemma (**in** *quantale*) *fres-top-bot* [*simp*]: ⊤ ← ⊥ = ⊤

by (*meson local.fres-galois local.order-antisym local.top-greatest*)

lemma (**in** *unital-quantale*) *bres-top-top2* [*simp*]: $(x \to y \cdot \top) \cdot \top = x \to y \cdot \top$
proof−
have $(x \to y \cdot \top) \cdot \top \leq x \to y \cdot \top \cdot \top$
by (*simp add*: *local.bres-interchange*)
hence $(x \to y \cdot \top) \cdot \top \leq x \to y \cdot \top$
by (*simp add*: *mult-assoc*)
thus *?thesis*
by (*metis local.mult-1-right local.order-eq-iff local.psrpq.subdistl local.sup-top-right*)
qed

lemma (**in** *unital-quantale*) *fres-top-top2* [*simp*]: $\top \cdot (\top \cdot y \leftarrow x) = \top \cdot y \leftarrow x$
by (*metis local.dual-order.antisym local.fres-interchange local.le-top local.top-greatest mult-assoc*)

lemma (**in** *unital-quantale*) *bres-top-bot* [*simp*]: $\top \to \bot = \bot$
by (*metis local.bot-least local.bres-canc1 local.le-top local.order.antisym*)

lemma (**in** *unital-quantale*) *fres-bot-top* [*simp*]: $\bot \leftarrow \top = \bot$
by (*metis local.bot-unique local.fres-canc1 local.top-le local.uqka.independence2 local.uwqlka.star-ext*)

lemma (**in** *unital-quantale*) *top-bot-iff*: $(x \cdot \top = \bot) = (x = \bot)$
by (*metis local.fres-bot-top local.fres-canc2 local.le-bot local.mult-botl*)

6.2 Involutive quantales

The following axioms for involutive quantales are standard.

class *involutive-quantale* = *unital-quantale* + *invol-op* +
assumes *inv-invol* [*simp*]: $(x^\circ)^\circ = x$
and *inv-contrav*: $(x \cdot y)^\circ = y^\circ \cdot x^\circ$
and *inv-sup* [*simp*]: $(\bigsqcup X)^\circ = (\bigsqcup x \in X.\ x^\circ)$

context *involutive-quantale*
begin

lemma *inv-binsup* [*simp*]: $(x \sqcup y)^\circ = x^\circ \sqcup y^\circ$
proof−
have $(x \sqcup y)^\circ = (\bigsqcup z \in \{x,y\}.\ z^\circ)$
using *local.inv-sup local.sup-Sup* **by** *presburger*
also have $\ldots = (\bigsqcup z \in \{x^\circ,y^\circ\}.\ z)$
by *simp*
also have $\ldots = x^\circ \sqcup y^\circ$
by *fastforce*
finally show *?thesis*.
qed

lemma *inv-iso*: $x \leq y \Longrightarrow x^\circ \leq y^\circ$

by (*metis inv-binsup local.sup.absorb-iff1*)

lemma *inv-galois*: $(x^\circ \leq y) = (x \leq y^\circ)$
using *local.inv-iso* **by** *fastforce*

lemma *bres-fres-conv*: $(y^\circ \leftarrow x^\circ)^\circ = x \rightarrow y$
proof −
have $(y^\circ \leftarrow x^\circ)^\circ = (\bigsqcup\{z.\ z \cdot x^\circ \leq y^\circ\})^\circ$
by (*simp add*: *local.fres-def*)
also have $\ldots = \bigsqcup\{z^\circ \mid z.\ z \cdot x^\circ \leq y^\circ\}$
by (*simp add*: *image-Collect*)
also have $\ldots = \bigsqcup\{z.\ z^\circ \cdot x^\circ \leq y^\circ\}$
by (*metis local.inv-invol*)
also have $\ldots = \bigsqcup\{z.\ (x \cdot z)^\circ \leq y^\circ\}$
by (*simp add*: *local.inv-contrav*)
also have $\ldots = \bigsqcup\{z.\ x \cdot z \leq y\}$
by (*metis local.inv-invol local.inv-iso*)
also have $\ldots = x \rightarrow y$
by (*simp add*: *local.bres-def*)
finally show *?thesis*.
qed

lemma *fres-bres-conv*: $(y^\circ \rightarrow x^\circ)^\circ = x \leftarrow y$
by (*metis bres-fres-conv local.inv-invol*)

sublocale *invqka*: *involutive-kleene-algebra* $(\sqcup)$ $(\cdot)$ *1* $\bot$ $(\leq)$ $(<)$ *qstar invol*
by *unfold-locales* (*simp-all add*: *local.inv-contrav*)

lemma *inv-binf* [*simp*]: $(x \sqcap y)^\circ = x^\circ \sqcap y^\circ$
proof −
{**fix** z
have $(z \leq (x \sqcap y)^\circ) = (z^\circ \leq x \sqcap y)$
using *invqka.inv-adj* **by** *blast*
also have $\ldots = (z^\circ \leq x \wedge z^\circ \leq y)$
by *simp*
also have $\ldots = (z \leq x^\circ \wedge z \leq y^\circ)$
by (*simp add*: *invqka.inv-adj*)
also have $\ldots = (z \leq x^\circ \sqcap y^\circ)$
by *simp*
finally have $(z \leq (x \sqcap y)^\circ) = (z \leq x^\circ \sqcap y^\circ)$.}
thus *?thesis*
using *local.dual-order.antisym* **by** *blast*
qed

lemma *inv-inf* [*simp*]: $(\bigsqcap X)^\circ = (\bigsqcap x \in X.\ x^\circ)$
by (*metis invqka.inv-adj local.INF-eqI local.Inf-greatest local.Inf-lower local.inv-invol*)

lemma *inv-top* [*simp*]: $\top^\circ = \top$
proof −

have *a*: $\top^\circ \leq \top$
  by *simp*
hence $(\top^\circ)^\circ \leq \top^\circ$
  using *local.inv-iso* **by** *blast*
hence $\top \leq \top^\circ$
  by *simp*
thus *?thesis*
  by (*simp add*: *local.top-le*)
qed

lemma *inv-qstar-aux* [*simp*]: $(x \frown i)^\circ = (x^\circ) \frown i$
  by (*induct i*, *simp-all add*: *local.power-commutes*)

lemma *inv-conjugate*: $(x^\circ \sqcap y = \bot) = (x \sqcap y^\circ = \bot)$
  using *inv-binf invqka.inv-zero* **by** *fastforce*

We define domain and codomain as in relation algebra and compare with the domain and codomain axioms above.

definition *do* :: $'a \Rightarrow 'a$ **where**
  $do\ x = 1 \sqcap (x \cdot x^\circ)$

definition *cd* :: $'a \Rightarrow 'a$ **where**
  $cd\ x = 1 \sqcap (x^\circ \cdot x)$

lemma *do-inv*: $do\ (x^\circ) = cd\ x$
proof−
  have $do\ (x^\circ) = 1 \sqcap (x^\circ \cdot (x^\circ)^\circ)$
    by (*simp add*: *do-def*)
  also have $\ldots = 1 \sqcap (x^\circ \cdot x)$
    by *simp*
  also have $\ldots = cd\ x$
    by (*simp add*: *cd-def*)
  finally show *?thesis*.
qed

lemma *cd-inv*: $cd\ (x^\circ) = do\ x$
  by (*simp add*: *cd-def do-def*)

lemma *do-le-top*: $do\ x \leq 1 \sqcap (x \cdot \top)$
  by (*simp add*: *do-def local.inf.coboundedI2 local.mult-isol*)

lemma *do-subid*: $do\ x \leq 1$
  by (*simp add*: *do-def*)

lemma *cd-subid*: $cd\ x \leq 1$
  by (*simp add*: *cd-def*)

lemma *do-bot* [*simp*]: $do\ \bot = \bot$
  by (*simp add*: *do-def*)

lemma *cd-bot* [*simp*]: $cd\ \bot = \bot$
by (*simp add*: *cd-def*)

lemma *do-iso*: $x \leq y \Longrightarrow do\ x \leq do\ y$
by (*simp add*: *do-def local.inf.coboundedI2 local.inv-iso local.psrpq.mult-isol-var*)

lemma *cd-iso*: $x \leq y \Longrightarrow cd\ x \leq cd\ y$
using *cd-def local.eq-refl local.inf-mono local.inv-iso local.psrpq.mult-isol-var* **by** *presburger*

lemma *do-subdist*: $do\ x \sqcup do\ y \leq do\ (x \sqcup y)$
proof−
have $do\ x \leq do\ (x \sqcup y)$ **and** $do\ y \leq do\ (x \sqcup y)$
by (*simp-all add*: *do-iso*)
thus *?thesis*
by *simp*
qed

lemma *cd-subdist*: $cd\ x \sqcup cd\ y \leq cd\ (x \sqcup y)$
by (*simp add*: *cd-iso*)

lemma *inv-do* [*simp*]: $(do\ x)^\circ = do\ x$
by (*simp add*: *do-def*)

lemma *inv-cd* [*simp*]: $(cd\ x)^\circ = cd\ x$
by (*metis do-inv inv-do*)

lemma *dedekind-modular*:
assumes $(x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot (y \sqcap (x^\circ \cdot z))$
shows $(x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot y$
using *assms local.inf.cobounded1 local.mult-isol local.order-trans* **by** *blast*

lemma *modular-eq1*:
assumes $\forall x\ y\ z\ w.\ (y \sqcap (z \cdot x^\circ) \leq w \longrightarrow (y \cdot x) \sqcap z \leq w \cdot x)$
shows $\forall x\ y\ z.\ (x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot y$
using *assms* **by** *blast*

lemma $do\ x \cdot do\ y = do\ x \sqcap do\ y$
oops

lemma $p \leq 1 \Longrightarrow q \leq 1 \Longrightarrow p \cdot q = p \sqcap q$
oops

end

sublocale *ab-unital-quantale* ⊆ *ciq*: *involutive-quantale id* - - - - - - - - - -
by *unfold-locales* (*simp-all add*: *mult-commute*)

class *distributive-involutive-quantale* = *involutive-quantale* + *distrib-unital-quantale*

class *boolean-involutive-quantale* = *involutive-quantale* + *bool-unital-quantale*

begin

lemma *res-peirce*:
 assumes $\forall x\ y.\ x^\circ \cdot -(x \cdot y) \leq -y$
 shows $((x \cdot y) \sqcap z^\circ = \bot) = ((y \cdot z) \sqcap x^\circ = \bot)$
proof
 assume $(x \cdot y) \sqcap z^\circ = \bot$
 hence $z^\circ \leq -(x \cdot y)$
  by (*simp add*: *local.inf.commute local.inf-shunt*)
 thus $(y \cdot z) \sqcap x^\circ = \bot$
  by (*metis assms local.inf-shunt local.inv-conjugate local.inv-contrav local.inv-invol local.mult-isol local.order.trans*)
next
 assume $(y \cdot z) \sqcap x^\circ = \bot$
 hence $x^\circ \leq -(y \cdot z)$
  using *local.compl-le-swap1 local.inf-shunt* **by** *blast*
 thus $(x \cdot y) \sqcap z^\circ = \bot$
  by (*metis assms local.dual-order.trans local.inf-shunt local.inv-conjugate local.inv-contrav local.mult-isol*)
qed

lemma *res-schroeder1*:
 assumes $\forall x\ y.\ x^\circ \cdot -(x \cdot y) \leq -y$
 shows $((x \cdot y) \sqcap z = \bot) = (y \sqcap (x^\circ \cdot z) = \bot)$
proof
 assume *h*: $(x \cdot y) \sqcap z = \bot$
 hence $z \leq -(x \cdot y)$
  by (*simp add*: *local.inf.commute local.inf-shunt*)
 thus $y \sqcap (x^\circ \cdot z) = \bot$
  by (*metis assms local.dual-order.trans local.inf.commute local.inf-shunt local.mult-isol*)
next
 assume $y \sqcap (x^\circ \cdot z) = \bot$
 hence $y \leq -(x^\circ \cdot z)$
  by (*simp add*: *local.inf-shunt*)
 thus $(x \cdot y) \sqcap z = \bot$
  by (*metis assms local.inf-shunt local.inv-invol local.order-trans mult-isol*)
qed

lemma *res-schroeder2*:
 assumes $\forall x\ y.\ x^\circ \cdot -(x \cdot y) \leq -y$
 shows $((x \cdot y) \sqcap z = \bot) = (x \sqcap (z \cdot y^\circ) = \bot)$
 by (*metis assms local.inv-invol local.res-peirce local.res-schroeder1*)

lemma *res-mod*:

assumes $\forall x\ y.\ x^\circ \cdot -(x \cdot y) \leq -y$
shows $(x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot y$
proof−
have $(x \cdot y) \sqcap z = ((x \sqcap ((z \cdot y^\circ) \sqcup -(z \cdot y^\circ))) \cdot y) \sqcap z$
by *simp*
also have $\ldots = (((x \sqcap (z \cdot y^\circ)) \cdot y) \sqcap z) \sqcup (((x \sqcap -(z \cdot y^\circ)) \cdot y) \sqcap z)$
using *local.chaq.wswq.distrib-left local.inf.commute local.sup-distr* **by** *presburger*
also have $\ldots \leq ((x \sqcap (z \cdot y^\circ)) \cdot y) \sqcup ((x \cdot y) \sqcap (-(z \cdot y^\circ)) \cdot y \sqcap z)$
by (*metis assms local.inf.commute local.inf-compl-bot-right local.sup.orderI local.sup-inf-absorb res-schroeder2*)
also have $\ldots \leq ((x \sqcap (z \cdot y^\circ)) \cdot y) \sqcup ((x \cdot y) \sqcap -z \ \sqcap z)$
by (*metis assms local.dual-order.eq-iff local.inf.commute local.inf-compl-bot-right res-schroeder2*)
also have $\ldots \leq ((x \sqcap (z \cdot y^\circ)) \cdot y)$
by (*simp add*: *local.inf.commute*)
finally show *?thesis*.
qed

end

The strong Gelfand property (name by Palmigiano and Re) is important for dioids and Kleene algebras. The modular law is a convenient axiom for relational quantales, in a setting where the underlying lattice is not boolean.

class *quantale-converse* = *involutive-quantale* +
assumes *strong-gelfand*: $x \leq x \cdot x^\circ \cdot x$

begin

lemma *do-gelfand* [*simp*]: $do\ x \cdot do\ x \cdot do\ x = do\ x$
apply (*rule order.antisym*)
using *local.do-subid local.h-seq local.mult-isol* **apply** *fastforce*
by (*metis local.inv-do local.strong-gelfand*)

lemma *cd-gelfand* [*simp*]: $cd\ x \cdot cd\ x \cdot cd\ x = cd\ x$
by (*metis do-gelfand local.do-inv*)

lemma *do-idem* [*simp*]: $do\ x \cdot do\ x\ = do\ x$
apply (*rule order.antisym*)
using *local.do-subid local.mult-isol* **apply** *fastforce*
by (*metis do-gelfand local.do-subid local.eq-refl local.nsrnqo.mult-oner local.psrpq.mult-isol-var*)

lemma *cd-idem* [*simp*]: $cd\ x \cdot cd\ x\ = cd\ x$
by (*metis do-idem local.do-inv*)

lemma *dodo* [*simp*]: $do\ (do\ x) = do\ x$
proof−
have $do\ (do\ x) = 1 \sqcap (do\ x \cdot do\ x)$
using *local.do-def local.inv-do* **by** *force*

also have ... = *1* ⊓ *do x*
  by *simp*
also have ... = *do x*
  by (*simp add*: *local.do-def*)
finally show *?thesis*.
qed

lemma *cdcd* [*simp*]: *cd* (*cd x*) = *cd x*
  using *cd-idem local.cd-def local.inv-cd* **by** *force*

lemma *docd-compat* [*simp*]: *do* (*cd x*) = *cd x*
proof−
  have *do* (*cd x*) = *do* (*do* (x°))
    by (*simp add*: *local.do-inv*)
  also have ... = *do* (x°)
    by *simp*
  also have ... = *cd x*
    by (*simp add*: *local.do-inv*)
  finally show *?thesis*.
qed

lemma *cddo-compat* [*simp*]: *cd* (*do x*) = *do x*
  by (*metis docd-compat local.cd-inv local.inv-do*)

end

sublocale *quantale-converse* ⊆ *convqka*: *kleene-algebra-converse* (⊔) (·) *1* ⊥ (≤) (<) *invol qstar*
  by *unfold-locales* (*simp add*: *local.strong-gelfand*)

6.3 Dedekind quantales

class *dedekind-quantale* = *involutive-quantale* +
  assumes *modular-law*: $(x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot y$

begin

sublocale *convdqka*: *kleene-algebra-converse* (⊔) (·) *1* ⊥ (≤) (<) *invol qstar*
  by *unfold-locales* (*metis local.inf.absorb2 local.le-top local.modular-law local.top-greatest*)

subclass *quantale-converse*
  by *unfold-locales* (*simp add*: *local.convdqka.strong-gelfand*)

lemma *modular-2* [*simp*]: $((x \sqcap (z \cdot y^\circ)) \cdot y) \sqcap z = (x \cdot y) \sqcap z$
  apply (*rule order.antisym*)
  using *local.inf.cobounded1 local.inf-mono local.mult-isor local.order-refl* **apply** *presburger*
  by (*simp add*: *local.modular-law*)

lemma *modular-1* [*simp*]: $(x \cdot (y \sqcap (x^\circ \cdot z))) \sqcap z = (x \cdot y) \sqcap z$
  by (*metis local.inv-binf local.inv-contrav local.inv-invol modular-2*)

lemma *modular3*: $(x \cdot y) \sqcap z \leq x \cdot (y \sqcap (x^\circ \cdot z))$
  by (*metis local.inf.cobounded1 modular-1*)

The name Dedekind quantale owes to the following formula, which is equivalent to the modular law. Dedekind quantales are called modular quantales in Rosenthal's book on quantaloids (to be precise: he discusses modular quantaloids, but the notion of modular quantale is then obvious).

lemma *dedekind*: $(x \cdot y) \sqcap z \leq (x \sqcap (z \cdot y^\circ)) \cdot (y \sqcap (x^\circ \cdot z))$
proof−
  have $(x \cdot y) \sqcap z = (x \cdot (y \sqcap (x^\circ \cdot z))) \sqcap z$
    by *simp*
  also have $\ldots \leq (x \sqcap (z \cdot (y \sqcap (x^\circ \cdot z))^\circ)) \cdot (y \sqcap (x^\circ \cdot z))$
    using *local.modular-law* **by** *presburger*
  also have $\ldots = (x \sqcap (z \cdot (y^\circ \sqcap (z^\circ \cdot x)))) \cdot (y \sqcap (x^\circ \cdot z))$
    by *simp*
  also have $\ldots \leq (x \sqcap (z \cdot y^\circ)) \cdot (y \sqcap (x^\circ \cdot z))$
    using *local.inf.commute modular-1* **by** *fastforce*
  finally show *?thesis***.**
qed

lemma *peirce*: $((x \cdot y) \sqcap z^\circ = \bot) = ((y \cdot z) \sqcap x^\circ = \bot)$
proof
  assume $(x \cdot y) \sqcap z^\circ = \bot$
  hence $((x \cdot y) \sqcap z^\circ) \cdot y^\circ = \bot$
    by *simp*
  hence $(z^\circ \cdot y^\circ) \sqcap x = \bot$
    by (*metis local.inf.commute local.inv-invol local.le-bot local.modular-law*)
  hence $((y \cdot z) \sqcap x^\circ)^\circ = \bot^\circ$
    by *simp*
  thus $(y \cdot z) \sqcap x^\circ = \bot$
    by (*metis local.inv-invol*)
next
  assume *h*: $(y \cdot z) \sqcap x^\circ = \bot$
  hence $z^\circ \cdot ((y \cdot z) \sqcap x^\circ) = \bot$
    by *simp*
  hence $(y^\circ \cdot x^\circ) \sqcap z = \bot$
    by (*metis h local.inf.commute local.inv-invol local.le-bot local.mult-botr modular3*)
  hence $((x \cdot y) \sqcap z^\circ)^\circ = \bot^\circ$
    by *simp*
  thus $(x \cdot y) \sqcap z^\circ = \bot$
    by (*metis local.inv-invol*)
qed

lemma *schroeder-1*: $((x \cdot y) \sqcap z = \bot) = (y \sqcap (x^\circ \cdot z) = \bot)$
  by (*metis local.inf.commute local.inf-bot-right local.inv-invol local.mult-botr mod-*

ular-1)

lemma *schroeder-2*: $((x \cdot y) \sqcap z = \bot) = (x \sqcap (z \cdot y^\circ) = \bot)$
  by (*metis local.inv-invol peirce schroeder-1*)

lemma *modular-eq2*: $y \sqcap (z \cdot x^\circ) \leq w \Longrightarrow (y \cdot x) \sqcap z \leq w \cdot x$
  by (*meson local.dual-order.trans local.eq-refl local.h-w1 local.modular-law*)

lemma *lla-top-aux*: $p \leq 1 \Longrightarrow ((x \leq p \cdot x) = (x \leq p \cdot \top))$
proof
  assume *h*: $p \leq 1$
  and *h1*: $x \leq p \cdot x$
  thus $x \leq p \cdot \top$
    by (*meson local.mult-isol local.order-trans local.top-greatest*)
next
  assume *h*: $p \leq 1$
  and $x \leq p \cdot \top$
  hence $x = (p \cdot \top) \sqcap x$
    using *local.inf.absorb-iff2* **by** *auto*
  also have $\ldots \leq p \cdot (\top \sqcap (p^\circ \cdot x))$
    using *modular3* **by** *blast*
  also have $\ldots = p \cdot p \cdot x$
    by (*simp add*: *h local.convdqka.subid-conv mult-assoc*)
  finally show $x \leq p \cdot x$
    by (*metis h local.dual-order.trans local.mult-isor local.nsrnqo.mult-onel*)
qed

Next we turn to properties of domain and codomain in Dedekind quantales.

lemma *lra-top-aux*: $p \leq 1 \Longrightarrow ((x \leq x \cdot p\ ) = (x \leq \top \cdot p))$
  by (*metis convdqka.subid-conv local.inf.absorb-iff2 local.mult-1-right local.psrpq.subdistl local.sup.absorb-iff2 local.top-greatest modular-eq2*)

lemma *lla*: $p \leq 1 \Longrightarrow ((do\ x \leq p) = (x \leq p \cdot \top))$
proof
  assume *a1*: $x \leq p \cdot \top$
  assume *a2*: $p \leq 1$
  have *f3*: $x \cdot \top \leq p \cdot \top \cdot \top$
    by (*simp add*: *a1 local.mult-isor*)
  have *f4*: $p \cdot do\ x \leq p$
   by (*simp add*: *local.do-subid local.uqka.star-inductr-var-equiv local.uwqlka.star-subid*)
  have $x \cdot \top \leq p \cdot \top$
    using *f3* **by** (*simp add*: *local.mult.semigroup-axioms semigroup.assoc*)
  thus $do\ x \leq p$
    using *f4 a2 lla-top-aux local.do-le-top local.inf.bounded-iff local.order-trans* **by** *blast*
next
  assume *a1*: $do\ x \leq p$
  assume *a2*: $p \leq 1$
  hence $do\ x \cdot x \leq p \cdot x$

by (*simp add*: *a1 local.mult-isor*)
hence $x \leq p \cdot x$
using *a1 local.do-def modular-eq2* **by** *fastforce*
thus $x \leq p \cdot \top$
by (*simp add*: *a2 lla-top-aux*)
qed

lemma *lla-Inf*: $do\ x = \bigsqcap\{p.\ x \leq p \cdot \top \wedge p \leq 1\}$
apply (*rule order.antisym*)
using *lla local.Inf-greatest* **apply** *fastforce*
by (*metis CollectI lla local.Inf-lower local.do-subid local.order.refl*)

lemma *lra*: $p \leq 1 \Longrightarrow ((cd\ x \leq p) = (x \leq \top \cdot p))$
by (*metis invqka.inv-adj lla local.convdqka.subid-conv local.do-inv local.inv-contrav local.inv-top*)

lemma *lra-Inf*: $cd\ x = \bigsqcap\{p.\ x \leq \top \cdot p \wedge p \leq 1\}$
apply (*rule order.antisym*)
using *local.Inf-greatest lra* **apply** *fastforce*
by (*metis CollectI local.Inf-lower local.cd-subid local.order.refl lra*)

lemma *lla-var*: $p \leq 1 \Longrightarrow ((do\ x \leq p) = (x \leq p \cdot x))$
by (*simp add*: *lla lla-top-aux*)

lemma *lla-Inf-var*: $do\ x = \bigsqcap\{p.\ x \leq p \cdot x \wedge p \leq 1\}$
apply (*rule order.antisym*)
using *lla-var local.Inf-greatest* **apply** *fastforce*
by (*metis CollectI lla-var local.Inf-lower local.do-subid local.order.refl*)

lemma *lra-var*: $p \leq 1 \Longrightarrow ((cd\ x \leq p) = (x \leq x \cdot p))$
by (*simp add*: *lra lra-top-aux*)

lemma *lra-Inf-var*: $cd\ x = \bigsqcap\{p.\ x \leq x \cdot p \wedge p \leq 1\}$
apply (*rule order.antisym*)
using *local.Inf-greatest lra-var* **apply** *fastforce*
by (*metis CollectI local.Inf-lower local.cd-subid local.order.refl lra-var*)

lemma *do-top*: $do\ x = 1 \sqcap (x \cdot \top)$
proof−
have $1 \sqcap (x \cdot \top) = 1 \sqcap (x \cdot (\top \sqcap x^\circ \cdot 1))$
by (*metis local.inf.commute local.inf-top.left-neutral modular-1*)
also have $\ldots = do\ x$
by (*simp add*: *local.do-def*)
finally show *?thesis*..
qed

lemma *cd-top*: $cd\ x = 1 \sqcap (\top \cdot x)$
by (*metis do-top invqka.inv-one local.do-inv local.inv-binf local.inv-cd local.inv-contrav local.inv-invol local.inv-top*)

We start deriving the axioms of modal semirings and modal quantales.

lemma *do-absorp*: $x \leq do\ x \cdot x$
  using *lla-var local.do-subid* **by** *blast*

lemma *cd-absorp*: $x \leq x \cdot cd\ x$
  using *local.cd-subid lra-var* **by** *blast*

lemma *do-absorp-eq* [*simp*]: $do\ x \cdot x = x$
  using *do-absorp local.do-subid local.dual-order.antisym local.h-w1* **by** *fastforce*

lemma *cd-absorp-eq* [*simp*]: $x \cdot cd\ x = x$
  by (*metis do-top local.do-inv local.inf-top.right-neutral local.nsrnqo.mult-oner modular-1*)

lemma *do-top2*: $x \cdot \top = do\ x \cdot \top$
proof (*rule order.antisym*)
  have $x \cdot \top = do\ x \cdot x \cdot \top$
    by *simp*
  also have $\ldots \leq do\ x \cdot \top \cdot \top$
    using *local.psrpq.mult-double-iso local.top-greatest* **by** *presburger*
  also have $\ldots = do\ x \cdot \top$
    by (*simp add*: *local.mult.semigroup-axioms semigroup.assoc*)
  finally show $x \cdot \top \leq do\ x \cdot \top$**.**
  have $do\ x \cdot \top = (1 \sqcap (x \cdot \top)) \cdot \top$
    by (*simp add*: *do-top*)
  also have $\ldots \leq (1 \cdot \top) \sqcap (x \cdot \top \cdot \top)$
    by (*simp add*: *local.mult-isor*)
  also have $\ldots = x \cdot \top$
    by (*simp add*: *mult-assoc*)
  finally show $do\ x \cdot \top \leq x \cdot \top$**.**
qed

lemma *cd-top2*: $\top \cdot x = \top \cdot cd\ x$
  by (*metis do-top2 local.do-inv local.inv-cd local.inv-contrav local.inv-invol local.inv-top*)

lemma *do-local* [*simp*]: $do\ (x \cdot do\ y) = do\ (x \cdot y)$
proof−
  have $do\ (x \cdot do\ y) = 1 \sqcap (x \cdot do\ y \cdot \top)$
    using *do-top* **by** *force*
  also have $\ldots = 1 \sqcap (x \cdot y \cdot \top)$
    using *do-top2 mult-assoc* **by** *force*
  also have $\ldots = do\ (x \cdot y)$
    by (*simp add*: *do-top*)
  finally show *?thesis*
    by *force*
qed

lemma *cd-local* [*simp*]: $cd\ (cd\ x \cdot y) = cd\ (x \cdot y)$

by (*metis cd-top cd-top2 mult-assoc*)

lemma *do-fix-subid*: (*do x* = *x*) = (*x* ≤ *1*)
proof
  assume *do x* = *x*
  thus *x* ≤ *1*
    by (*metis local.do-subid*)
next
  assume *a*: *x* ≤ *1*
  hence *x* = *do x* · *x*
    by *simp*
  hence *b*: *x* ≤ *do x*
    by (*metis a local.mult-isol local.nsrnqo.mult-oner*)
  have *x* · *x* ≤ *x*
    using *a local.mult-isor* **by** *fastforce*
  hence *do x* ≤ *x*
    by (*simp add*: *a lla-var local.le-top lra-top-aux*)
  thus *do x* = *x*
    by (*simp add*: *b local.dual-order.antisym*)
qed

lemma *cd-fix-subid*: (*cd x* = *x*) = (*x* ≤ *1*)
 by (*metis local.convdqka.subid-conv local.do-inv local.do-fix-subid local.docd-compat*)

lemma *do-inf2*: *do* (*do x* ⊓ *do y*) = *do x* ⊓ *do y*
  using *do-top local.do-fix-subid local.inf.assoc* **by** *auto*

lemma *do-inf-comp*: *do x* · *do y* = *do x* ⊓ *do y*
  by (*smt* (*verit*, *best*) *local.do-def local.do-idem local.do-fix-subid local.dodo local.dual-order.trans local.inf-commute local.inf-idem local.inv-contrav local.inv-do local.mult-1-right local.mult-isol modular-1 mult-assoc*)

lemma *cd-inf-comp*: *cd x* · *cd y* = *cd x* ⊓ *cd y*
  by (*metis do-inf-comp local.docd-compat*)

lemma *subid-mult-meet*: *p* ≤ *1* ⟹ *q* ≤ *1* ⟹ *p* · *q* = *p* ⊓ *q*
  by (*metis do-inf-comp local.do-fix-subid*)

lemma *dodo-sup*: *do* (*do x* ⊔ *do y*) = *do x* ⊔ *do y*
proof−
  have *do* (*do x* ⊔ *do y*) = *1* ⊓ ((*do x* ⊔ *do y*) · (*do x* ⊔ *do y*)°)
    using *local.do-def* **by** *blast*
  also have . . . = *1* ⊓ ((*do x* ⊔ *do y*) · (*do x* ⊔ *do y*))
    by *simp*
  also have . . . = *1* ⊓ (*do x* ⊔ *do y*)
    using *local.do-subid local.dodo local.inf.idem local.le-supI subid-mult-meet* **by** *presburger*
  also have . . . = *do x* ⊔ *do y*
    by (*simp add*: *local.do-def local.inf-absorb2*)

finally show *?thesis*.
qed

lemma *do-sup* [*simp*]: *do* $(x \sqcup y) = do\ x \sqcup do\ y$
proof –
  have *do* $(x \sqcup y) = 1 \sqcap ((x \sqcup y) \cdot \top)$
    by (*simp add*: *do-top*)
  also have $\ldots = 1 \sqcap (x \cdot \top \sqcup y \cdot \top)$
    by *simp*
  also have $\ldots = 1 \sqcap (do\ x \cdot \top \sqcup do\ y \cdot \top)$
    using *do-top2* **by** *force*
  also have $\ldots = 1 \sqcap ((do\ x \sqcup do\ y) \cdot \top)$
    by *simp*
  also have $\ldots = do\ (do\ x \sqcup do\ y)$
    by (*simp add*: *do-top*)
  finally show *?thesis*
    by (*simp add*: *dodo-sup*)
qed

lemma *cdcd-sup*: $cd\ (cd\ x \sqcup cd\ y) = cd\ x \sqcup cd\ y$
  using *local.cd-fix-subid* **by** *fastforce*

lemma *cd-sup* [*simp*]: $cd\ (x \sqcup y) = cd\ x \sqcup cd\ y$
  by (*metis do-sup local.do-inv local.inv-binsup*)

Next we show that Dedekind quantales are modal quantales, hence also modal semirings.

sublocale *dmq*: *dc-modal-quantale 1* $(\cdot)$ *Inf Sup* $(\sqcap)$ $(\leq)$ $(<)$ $(\sqcup)$ $\bot$ $\top$ *cd do*
proof
  show $\bigwedge x.\ cd\ x \leq 1$
    by (*simp add*: *cd-top*)
  show $\bigwedge x.\ do\ x \leq 1$
    by (*simp add*: *do-top*)
qed *simp-all*

lemma *do-top3* [*simp*]: $do\ (x \cdot \top) = do\ x$
  using *dmq.coddual.le-top dmq.dqmsr.domain-1″ local.do-iso local.order.antisym*
by *presburger*

lemma *cd-top3* [*simp*]: $cd\ (\top \cdot x) = cd\ x$
  by (*simp add*: *cd-top dmq.coddual.mult-assoc*)

lemma *dodo-Sup-pres*: $do\ (\bigsqcup x \in X.\ do\ x) = (\bigsqcup x \in X.\ do\ x)$
  by (*simp add*: *local.SUP-least local.do-fix-subid*)

The domain elements form a complete Heyting algebra.

lemma *do-complete-heyting*: $do\ x \sqcap (\bigsqcup y \in Y.\ do\ y) = (\bigsqcup y \in Y.\ do\ x \sqcap do\ y)$
proof –
  have $do\ x \sqcap (\bigsqcup y \in Y.\ do\ y) = do\ x \cdot (\bigsqcup y \in Y.\ do\ y)$

by (*metis do-inf-comp dodo-Sup-pres*)
also have ... = ($\bigsqcup y \in Y.\ do\ x \cdot do\ y$)
by (*simp add*: *dmq.coddual.Sup-distr image-image*)
also have ... = ($\bigsqcup y \in Y.\ do\ x \sqcap do\ y$)
using *do-inf-comp* **by** *force*
finally show *?thesis*.
qed

lemma *cdcd-Sup-pres*: $cd\ (\bigsqcup x \in X.\ cd\ x) = (\bigsqcup x \in X.\ cd\ x)$
by (*simp add*: *local.SUP-least local.cd-fix-subid*)

lemma *cd-complete-heyting*: $cd\ x \sqcap (\bigsqcup y \in Y.\ cd\ y) = (\bigsqcup y \in Y.\ cd\ x \sqcap cd\ y)$
proof−
have $cd\ x \sqcap (\bigsqcup y \in Y.\ cd\ y) = cd\ x \cdot (\bigsqcup y \in Y.\ cd\ y)$
by (*metis cd-inf-comp cdcd-Sup-pres*)
also have ... = ($\bigsqcup y \in Y.\ cd\ x \cdot cd\ y$)
by (*simp add*: *dmq.coddual.Sup-distr image-image*)
also have ... = ($\bigsqcup y \in Y.\ cd\ x \sqcap cd\ y$)
using *cd-inf-comp* **by** *force*
finally show *?thesis*.
qed

lemma *subid-complete-heyting*:
assumes $p \leq 1$
and $\forall q \in Q.\ q \leq 1$
shows $p \sqcap (\bigsqcup Q) = (\bigsqcup q \in Q.\ p \sqcap q)$
proof−
have *a*: $do\ p = p$
by (*simp add*: *assms*(*1*) *local.do-fix-subid*)
have *b*: $\forall q \in Q.\ do\ q = q$
using *assms*(*2*) *local.do-fix-subid* **by** *presburger*
hence $p \sqcap (\bigsqcup Q) = do\ p \sqcap (\bigsqcup q \in Q.\ do\ q)$
by (*simp add*: *a*)
also have ... = ($\bigsqcup q \in Q.\ do\ p \sqcap do\ q$)
using *do-complete-heyting* **by** *blast*
also have ... = ($\bigsqcup q \in Q.\ p \sqcap q$)
using *a b* **by** *force*
finally show *?thesis*.
qed

Next we show that domain and codomain preserve arbitrary Sups.

lemma *do-Sup-pres-aux*: $(\bigsqcup x \in X.\ do\ x \cdot \top) = (\bigsqcup x \in do\ `\ X.\ x \cdot \top)$
by (*smt* (*verit*, *best*) *Sup.SUP-cong image-image*)

lemma *do-Sup-pres*: $do\ (\bigsqcup x \in X.\ x) = (\bigsqcup x \in X.\ do\ x)$
proof−
have $do\ (\bigsqcup x \in X.\ x) = 1 \sqcap ((\bigsqcup x \in X.\ x) \cdot \top)$
by (*simp add*: *do-top*)
also have ... = $1 \sqcap (\bigsqcup x \in X.\ x \cdot \top)$

by (*simp add*: *dmq.coddual.Sup-distl*)
also have ... = $1 \sqcap (\bigsqcup x \in X.\ do\ x \cdot \top)$
using *do-top2* **by** *force*
also have ... = $1 \sqcap (\bigsqcup x \in do\ `\ X.\ x \cdot \top)$
using *do-Sup-pres-aux* **by** *presburger*
also have ... = $1 \sqcap ((\bigsqcup x \in X.\ do\ x) \cdot \top)$
using *dmq.coddual.Sup-distl* **by** *presburger*
also have ... = $do\ (\bigsqcup x \in X.\ do\ x)$
by (*simp add*: *do-top*)
finally show *?thesis*
using *dodo-Sup-pres* **by** *presburger*
qed

lemma *cd-Sup-pres*: $cd\ (\bigsqcup x \in X.\ x) = (\bigsqcup x \in X.\ cd\ x)$
proof−
have $cd\ (\bigsqcup x \in X.\ x) = do\ ((\bigsqcup x \in X.\ x)^\circ)$
using *local.do-inv* **by** *presburger*
also have ... = $do\ (\bigsqcup x \in X.\ x^\circ)$
by *simp*
also have ... = $(\bigsqcup x \in X.\ do\ (x^\circ))$
by (*metis do-Sup-pres image-ident image-image*)
also have ... = $(\bigsqcup x \in X.\ cd\ x)$
using *local.do-inv* **by** *presburger*
finally show *?thesis*.
qed

lemma *do-inf*: $do\ (x \sqcap y) = 1 \sqcap (y \cdot x^\circ)$
by (*smt* (*z3*) *do-absorp-eq invqka.inv-one local.do-def local.inf-commute local.inv-binf local.inv-contrav local.inv-invol local.mult-1-right modular-1 modular-2 mult-assoc*)

lemma *cd-inf*: $cd\ (x \sqcap y) = 1 \sqcap (y^\circ \cdot x)$
by (*metis do-inf local.do-inv local.inv-binf local.inv-invol*)

lemma *do-bres-prop*: $p \le 1 \Longrightarrow do\ (x \rightarrow p \cdot \top) = 1 \sqcap (x \rightarrow p \cdot \top)$
by (*simp add*: *do-top*)

lemma *cd-fres-prop*: $p \le 1 \Longrightarrow cd\ (\top \cdot p \leftarrow x) = 1 \sqcap (\top \cdot p \leftarrow x)$
by (*simp add*: *cd-top*)

lemma *do-meet-prop*: $(do\ p \cdot x) \sqcap (x \cdot do\ q) = do\ p \cdot x \cdot do\ q$
proof (*rule order.antisym*)
have $x \sqcap (do\ p \cdot x \cdot do\ q) \le do\ p \cdot x$
by (*simp add*: *dmq.dqmsr.dom-subid-aux2″ local.inf.coboundedI2*)
thus $(do\ p \cdot x) \sqcap (x \cdot do\ q) \le do\ p \cdot x \cdot do\ q$
by (*simp add*: *local.inf.commute modular-eq2*)
next
have $do\ p \cdot x \cdot do\ q = (do\ p \cdot x \cdot do\ q) \sqcap (do\ p \cdot x \cdot do\ q)$
by *simp*
also have ... $\le (do\ p \cdot x) \sqcap (x \cdot do\ q)$

using *dmq.dqmsr.dom-subid-aux2 dmq.dqmsr.dom-subid-aux2'' local.psrpq.mult-isol-var*
by *auto*
finally show $do\ p \cdot x \cdot do\ q \leq (do\ p \cdot x) \sqcap (x \cdot do\ q)$**.**
qed

lemma *subid-meet-prop*: $p \leq 1 \Longrightarrow q \leq 1 \Longrightarrow (p \cdot x) \sqcap (x \cdot q) = p \cdot x \cdot q$
by (*metis do-fix-subid do-meet-prop*)

Next we consider box and diamond operators like in modal semirings and modal quantales. These are inherited from domain quantales. Diamonds are defined with respect to domain and codomain. The box operators are defined as Sups and hence right adjoints of diamonds.

abbreviation *do-dia* ≡ *dmq.fd'*

abbreviation *cd-dia* ≡ *dmq.bd'*

abbreviation *do-box* ≡ *dmq.bb*

abbreviation *cd-box* ≡ *dmq.fb*

In the sense of modal logic, the domain-based diamond is a backward operator, the codomain-based one a forward operator. These are related by opposition/converse.

lemma *do-dia-cd-dia-conv*: $p \leq 1 \Longrightarrow$ *do-dia* $(x^\circ)\ p =$ *cd-dia* $x\ p$
by (*simp add*: *convdqka.subid-conv dmq.coddual.dqmsr.fd-def dmq.dqmsr.fd-def local.cd-def local.do-def*)

lemma *cd-dia-do-dia-conv*: $p \leq 1 \Longrightarrow$ *cd-dia* $(x^\circ)\ p =$ *do-dia* $x\ p$
by (*metis do-dia-cd-dia-conv local.inv-invol*)

Diamonds preserve sups in both arguments.

lemma *do-dia-Sup*: *do-dia* $(\bigsqcup X)\ p = (\bigsqcup x \in X.$ *do-dia* $x\ p)$
proof−
have *do-dia* $(\bigsqcup X)\ p = do\ ((\bigsqcup X) \cdot p)$
by (*simp add*: *dmq.dqmsr.fd-def*)
also have $\ldots = do\ (\bigsqcup x \in X.\ x \cdot p)$
using *local.Sup-distr* **by** *fastforce*
also have $\ldots = (\bigsqcup x \in X.\ do\ (x \cdot p))$
by (*metis do-Sup-pres image-ident image-image*)
also have $\ldots = (\bigsqcup x \in X.$ *do-dia* $x\ p)$
using *dmq.dqmsr.fd-def* **by** *fastforce*
finally show *?thesis***.**
qed

lemma *do-dia-Sup2*: *do-dia* $x\ (\bigsqcup P) = (\bigsqcup p \in P.$ *do-dia* $x\ p)$
proof−
have *do-dia* $x\ (\bigsqcup P) = do\ (x \cdot (\bigsqcup P))$
by (*simp add*: *dmq.dqmsr.fd-def*)

also have ... = $(\bigsqcup p \in P.\ do\ (x \cdot p))$
by (*metis dmq.coddual.Sup-distr do-Sup-pres image-ident image-image*)
also have ... = $(\bigsqcup p \in P.\ do\text{-}dia\ x\ p)$
using *dmq.dqmsr.fd-def* **by** *auto*
finally show *?thesis*.
qed

lemma *cd-dia-Sup*: $cd\text{-}dia\ (\bigsqcup X)\ p = (\bigsqcup x \in X.\ cd\text{-}dia\ x\ p)$
proof−
have $cd\text{-}dia\ (\bigsqcup X)\ p = cd\ (p \cdot (\bigsqcup X))$
by (*simp add*: *dmq.coddual.dqmsr.fd-def*)
also have ... = $cd\ (\bigsqcup x \in X.\ p \cdot x)$
using *dmq.coddual.Sup-distr* **by** *auto*
also have ... = $(\bigsqcup x \in X.\ cd\ (p \cdot x))$
by (*metis cd-Sup-pres image-ident image-image*)
also have ... = $(\bigsqcup x \in X.\ cd\text{-}dia\ x\ p)$
using *dmq.coddual.dqmsr.fd-def* **by** *force*
finally show *?thesis*.
qed

lemma *cd-dia-Sup2*: $cd\text{-}dia\ x\ (\bigsqcup P) = (\bigsqcup p \in P.\ cd\text{-}dia\ x\ p)$
proof−
have $cd\text{-}dia\ x\ (\bigsqcup P) = cd\ ((\bigsqcup P) \cdot x)$
by (*simp add*: *dmq.coddual.dqmsr.fd-def*)
also have ... = $(\bigsqcup p \in P.\ cd\ (p \cdot x))$
by (*metis cd-Sup-pres dmq.coddual.Sup-distl image-ident image-image*)
also have ... = $(\bigsqcup p \in P.\ cd\text{-}dia\ x\ p)$
using *dmq.coddual.dqmsr.fd-def* **by** *auto*
finally show *?thesis*.
qed

The domain-based box is a forward operator, the codomain-based on a backward one. These interact again with respect to converse.

lemma *do-box-var*: $p \leq 1 \Longrightarrow do\text{-}box\ x\ p = \bigsqcup\{q.\ do\text{-}dia\ x\ q \leq p \wedge q \leq 1\}$
by (*smt* (*verit*, *best*) *Collect-cong dmq.bb-def dmq.dqmsr.fdia-d-simp local.do-fix-subid local.dodo*)

lemma *cd-box-var*: $p \leq 1 \Longrightarrow cd\text{-}box\ x\ p = \bigsqcup\{q.\ cd\text{-}dia\ x\ q \leq p \wedge q \leq 1\}$
by (*smt* (*verit*, *best*) *Collect-cong dmq.coddual.dqmsr.fdia-d-simp dmq.fb-def local.cd-fix-subid local.cdcd*)

lemma *do-box-cd-box-conv*: $p \leq 1 \Longrightarrow do\text{-}box\ (x^\circ)\ p = cd\text{-}box\ x\ p$
proof−
assume *a*: $p \leq 1$
hence $do\text{-}box\ (x^\circ)\ p = \bigsqcup\{q.\ do\text{-}dia\ (x^\circ)\ q \leq p \wedge q \leq 1\}$
by (*simp add*: *do-box-var*)
also have ... = $\bigsqcup\{q.\ cd\text{-}dia\ x\ q \leq p \wedge q \leq 1\}$
by (*metis do-dia-cd-dia-conv*)
also have ... = $cd\text{-}box\ x\ p$

using *a cd-box-var* **by** *auto*
finally show *?thesis*.
qed

lemma *cd-box-do-box-conv*: $p \leq 1 \Longrightarrow$ *cd-box* (x°) $p =$ *do-box* x p
by (*metis do-box-cd-box-conv local.inv-invol*)

lemma *do-box-subid*: *do-box* x $p \leq 1$
using *dmq.bb-def local.Sup-le-iff* **by** *force*

lemma *cd-box-subid*: $p \leq 1 \Longrightarrow$ *cd-box* x $p \leq 1$
by (*metis do-box-subid local.do-box-cd-box-conv*)

Next we prove that boxes and diamonds are adjoints, and then demodalisation laws known from modal semirings.

lemma *do-dia-do-box-galois*:
assumes $p \leq 1$
and $q \leq 1$
shows (*do-dia* x $p \leq q$) $=$ ($p \leq$ *do-box* x q)
proof
show *do-dia* x $p \leq q \Longrightarrow p \leq$ *do-box* x q
by (*simp add*: *assms do-box-var local.Sup-upper*)
next
assume $p \leq$ *do-box* x q
hence *do-dia* x $p \leq$ *do* $(x \cdot \bigsqcup\{t.$ *do-dia* x $t \leq q \wedge t \leq 1\})$
by (*simp add*: *assms*(*2*) *local.dmq.dqmsr.fd-def local.do-box-var local.do-iso local.mult-isol*)
also have $\ldots = \bigsqcup\{$*do* $(x \cdot t)$ $|t.$ *do-dia* x $t \leq q \wedge t \leq 1\}$
by (*metis do-Sup-pres image-ident image-image local.Sup-distl setcompr-eq-image*)
also have $\ldots = \bigsqcup\{$*do-dia* x t $|t.$ *do-dia* x $t \leq q \wedge t \leq 1\}$
using *local.dmq.dqmsr.fd-def* **by** *fastforce*
also have $\ldots \leq q$
using *local.Sup-le-iff* **by** *blast*
finally have *do-dia* x $p \leq q$.
thus $p \leq$ *do-box* x $q \Longrightarrow$ *do-dia* x $p \leq q$.
qed

lemma *cd-dia-cd-box-galois*:
assumes $p \leq 1$
and $q \leq 1$
shows (*cd-dia* x $p \leq q$) $=$ ($p \leq$ *cd-box* x q)
by (*metis assms do-box-cd-box-conv do-dia-cd-dia-conv do-dia-do-box-galois*)

lemma *do-dia-demod-subid*:
assumes $p \leq 1$
and $q \leq 1$
shows (*do-dia* x $p \leq q$) $=$ ($x \cdot p \leq q \cdot x$)
by (*metis assms dmq.dqmsr.fdemodalisation2 local.do-fix-subid*)

The demodalisation laws have variants based on residuals.

lemma *do-dia-demod-subid-fres*:
  assumes $p \leq 1$
  and $q \leq 1$
  shows $(do\text{-}dia\ x\ p \leq q) = (p \leq x \rightarrow q \cdot x)$
  by (*simp add*: *assms do-dia-demod-subid local.bres-galois*)

lemma *do-dia-demod-subid-var*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(do\text{-}dia\ x\ p \leq q) = (x \cdot p \leq q \cdot \top)$
  by (*simp add*: *assms(2) dmq.dqmsr.fd-def lla*)

lemma *do-dia-demod-subid-var-fres*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(do\text{-}dia\ x\ p \leq q) = (p \leq x \rightarrow q \cdot \top)$
  by (*simp add*: *assms do-dia-demod-subid-var local.bres-galois*)

lemma *cd-dia-demod-subid*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(cd\text{-}dia\ x\ p \leq q) = (p \cdot x \leq x \cdot q)$
  by (*metis assms dmq.coddual.dqmsr.fdemodalisation2 local.cd-fix-subid*)

lemma *cd-dia-demod-subid-fres*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(cd\text{-}dia\ x\ p \leq q) = (p \leq x \cdot q \leftarrow x)$
  by (*simp add*: *assms cd-dia-demod-subid local.fres-galois*)

lemma *cd-dia-demod-subid-var*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(cd\text{-}dia\ x\ p \leq q) = (p \cdot x \leq \top \cdot q)$
  by (*simp add*: *assms(2) dmq.coddual.dqmsr.fd-def lra*)

lemma *cd-dia-demod-subid-var-fres*:
  assumes $p \leq 1$
  and $q \leq 1$
shows $(cd\text{-}dia\ x\ p \leq q) = (p \leq \top \cdot q \leftarrow x)$
  by (*simp add*: *assms cd-dia-demod-subid-var local.fres-galois*)

lemma *do-box-iso*:
  assumes $p \leq 1$
  and $q \leq 1$
  and $p \leq q$
shows $do\text{-}box\ x\ p \leq do\text{-}box\ x\ q$
  by (*meson assms do-box-subid do-dia-do-box-galois local.order.refl local.order.trans*)

lemma *cd-box-iso*:
assumes $p \leq 1$
and $q \leq 1$
and $p \leq q$
shows *cd-box* $x\ p \leq$ *cd-box* $x\ q$
by (*metis assms do-box-cd-box-conv do-box-iso*)

lemma *do-box-demod-subid*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *do-box* $x\ q) = (x \cdot p \leq q \cdot x)$
using *assms do-dia-do-box-galois local.do-dia-demod-subid* **by** *force*

lemma *do-box-demod-subid-bres*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *do-box* $x\ q) = (p \leq x \to q \cdot x)$
by (*simp add*: *assms do-box-demod-subid local.bres-galois*)

lemma *do-box-demod-subid-var*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *do-box* $x\ q) = (x \cdot p \leq q \cdot \top)$
using *assms do-dia-demod-subid-var do-dia-do-box-galois* **by** *auto*

lemma *do-box-demod-subid-var-bres*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *do-box* $x\ q) = (p \leq x \to q \cdot \top)$
by (*simp add*: *assms do-box-demod-subid-var local.bres-galois*)

lemma *do-box-demod-subid-var-bres-do*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *do-box* $x\ q) = (p \leq$ *do* $(x \to q \cdot \top))$
by (*simp add*: *assms do-box-demod-subid-var-bres do-top*)

lemma *cd-box-demod-subid*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *cd-box* $x\ q) = (p \cdot x \leq x \cdot q)$
using *assms local.cd-dia-cd-box-galois local.cd-dia-demod-subid* **by** *force*

lemma *cd-box-demod-subid-fres*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq$ *cd-box* $x\ q) = (p \leq x \cdot q \leftarrow x)$
by (*simp add*: *assms cd-box-demod-subid local.fres-galois*)

lemma *cd-box-demod-subid-var*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq \textit{cd-box}\ x\ q) = (p \cdot x \leq \top \cdot q)$
using *assms cd-dia-cd-box-galois cd-dia-demod-subid-var* **by** *force*

lemma *cd-box-demod-subid-var-fres*:
assumes $p \leq 1$
and $q \leq 1$
shows $(p \leq \textit{cd-box}\ x\ q) = (p \leq \top \cdot q \leftarrow x)$
by (*simp add*: *assms cd-box-demod-subid-var local.fres-galois*)

We substitute demodalisation inequalities for diamonds in the definitions of boxes.

lemma *do-box-var2*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = \bigsqcup\{q.\ x \cdot q \leq p \cdot x \wedge q \leq 1\}$
unfolding *do-box-var* **by** (*meson do-dia-demod-subid*)

lemma *do-box-bres1*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = \bigsqcup\{q.\ q \leq x \rightarrow p \cdot x \wedge q \leq 1\}$
unfolding *do-box-var* **by** (*meson do-dia-demod-subid-fres*)

lemma *do-box-bres2*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = \bigsqcup\{q.\ q \leq x \rightarrow p \cdot \top \wedge q \leq 1\}$
unfolding *do-box-var* **by** (*simp add*: *dmq.dqmsr.fd-def lla local.bres-galois*)

lemma *do-box-var3*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = \bigsqcup\{q.\ x \cdot q \leq p \cdot \top \wedge q \leq 1\}$
unfolding *do-box-var* **using** *dmq.dqmsr.fd-def lla* **by** *force*

lemma *cd-box-var2*: $p \leq 1 \Longrightarrow \textit{cd-box}\ x\ p = \bigsqcup\{q.\ q \cdot x \leq x \cdot p \wedge q \leq 1\}$
unfolding *cd-box-var* **by** (*metis cd-dia-demod-subid*)

lemma *cd-box-var3*: $p \leq 1 \Longrightarrow \textit{cd-box}\ x\ p = \bigsqcup\{q.\ q \cdot x \leq \top \cdot p \wedge q \leq 1\}$
unfolding *cd-box-var* **by** (*simp add*: *dmq.coddual.dqmsr.fd-def lra*)

Using these results we get a simple characterisation of boxes via domain and codomain. Similar laws can be found implicitly in Doornbos, Backhouse and van der Woude's paper on a calculational approach to mathematical induction, which speaks about wlp operators instead modal operators.

lemma *bres-do-box*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = \textit{do}\ (x \rightarrow p \cdot \top)$
by (*meson do-box-demod-subid-var-bres-do do-box-subid local.cd-fix-subid local.cddo-compat local.dual-order.antisym local.eq-refl*)

lemma *bres-do-box-var*: $p \leq 1 \Longrightarrow \textit{do-box}\ x\ p = 1 \sqcap (x \rightarrow p \cdot \top)$
using *bres-do-box do-bres-prop* **by** *force*

lemma *bres-do-box-top*: $p \leq 1 \Longrightarrow (\textit{do-box}\ x\ p) \cdot \top = x \rightarrow p \cdot \top$
using *bres-do-box do-top2* **by** *auto*

lemma *fres-cd-box*: $p \leq 1 \Longrightarrow \textit{cd-box}\ x\ p = \textit{cd}\ (\top \cdot p \leftarrow x)$
proof–

assume *h0*: $p \leq 1$
{fix q
assume *h1*: $q \leq 1$
have $(q \leq cd\text{-}box\ x\ p) = (q \cdot x \leq \top \cdot p)$
  by (*simp add*: *cd-box-demod-subid-var h0 h1*)
also have $\ldots = (q \leq \top \cdot p \leftarrow x)$
  by (*simp add*: *local.fres-galois*)
also have $\ldots = (q \leq 1 \sqcap (\top \cdot p \leftarrow x))$
  by (*simp add*: *h1*)
also have $\ldots = (q \leq cd\ (\top \cdot p \leftarrow x))$
  by (*simp add*: *cd-fres-prop h0*)
finally have $(q \leq cd\text{-}box\ x\ p) = (q \leq cd\ (\top \cdot p \leftarrow x))$**.}**
hence $\forall y.\ y \leq cd\text{-}box\ x\ p \longleftrightarrow y \leq cd\ (\top \cdot p \leftarrow x)$
  by (*meson cd-box-subid dmq.coddual.dqmsr.dpd3 h0 local.dual-order.trans*)
thus *?thesis*
  using *local.dual-order.antisym* **by** *blast*
qed

lemma *fres-cd-box-var*: $p \leq 1 \Longrightarrow cd\text{-}box\ x\ p = 1 \sqcap (\top \cdot p \leftarrow x)$
  by (*simp add*: *cd-fres-prop fres-cd-box*)

lemma *fres-cd-box-top*: $p \leq 1 \Longrightarrow \top \cdot cd\text{-}box\ x\ p = \top \cdot p \leftarrow x$
  using *cd-top2 fres-cd-box* **by** *auto*

Next we show that the box operators act on the complete Heyting algebra of subidentities.

lemma *cd-box-act*:
  assumes $p \leq 1$
  shows $cd\text{-}box\ (x \cdot y)\ p = cd\text{-}box\ x\ (cd\text{-}box\ y\ p)$
proof−
  {fix q
    assume *a*: $q \leq 1$
    hence $(q \leq cd\text{-}box\ (x \cdot y)\ p) = (cd\text{-}dia\ (x \cdot y)\ q \leq p)$
      by (*simp add*: *assms cd-dia-cd-box-galois*)
    also have $\ldots = (cd\text{-}dia\ y\ (cd\text{-}dia\ x\ q) \leq p)$
      by (*simp add*: *local.dmq.coddual.dqmsr.fdia-mult*)
    also have $\ldots = (cd\text{-}dia\ x\ q \leq cd\text{-}box\ y\ p)$
      using *assms cd-dia-cd-box-galois cd-top dmq.coddual.dqmsr.fd-def* **by** *force*
    also have $\ldots = (q \leq cd\text{-}box\ x\ (cd\text{-}box\ y\ p))$
      by (*simp add*: *a assms cd-dia-cd-box-galois local.cd-box-subid*)
    finally have $(q \leq cd\text{-}box\ (x \cdot y)\ p) = (q \leq cd\text{-}box\ x\ (cd\text{-}box\ y\ p))$**.}**
  thus *?thesis*
    by (*meson assms local.cd-box-subid local.dual-order.eq-iff*)
qed

lemma *do-box-act*:
  assumes $p \leq 1$
  shows $do\text{-}box\ (x \cdot y)\ p = do\text{-}box\ y\ (do\text{-}box\ x\ p)$
  by (*smt* (*verit*) *assms cd-box-act local.cd-box-do-box-conv local.do-box-subid lo-*

cal.inv-contrav)

Next we show that the box operators are Sup reversing in the first and Inf preserving in the second argument.

lemma *do-box-sup-inf*: $p \leq 1 \Longrightarrow$ *do-box* $(x \sqcup y)$ $p =$ *do-box* x $p \cdot$ *do-box* y p
proof−
  assume *h1*: $p \leq 1$
  {**fix** q
  assume *h2*: $q \leq 1$
  hence $(q \leq$ *do-box* $(x \sqcup y)$ $p) = ($*do-dia* $(x \sqcup y)$ $q \leq p)$
    by (*simp add*: *do-dia-do-box-galois h1*)
  also have … $= ($*do-dia* x $q \leq p \wedge$ *do-dia* y $q \leq p)$
    by (*simp add*: *dmq.dqmsr.fdia-add2*)
  also have … $= (q \leq$ *do-box* x $p \wedge q \leq$ *do-box* y $p)$
    by (*simp add*: *do-dia-do-box-galois h1 h2*)
  also have … $= (q \leq$ *do-box* x $p \cdot$ *do-box* y $p)$
    by (*simp add*: *do-box-subid subid-mult-meet*)
  finally have $(q \leq$ *do-box* $(x \sqcup y)$ $p) = (q \leq$ *do-box* x $p \cdot$ *do-box* y $p)$.}
  hence $\forall z.\ z \leq$ *do-box* $(x \sqcup y)$ $p \longleftrightarrow z \leq$ *do-box* x $p \cdot$ *do-box* y p
   by (*metis do-box-subid local.dual-order.trans local.inf.boundedE subid-mult-meet*)
  thus *?thesis*
    using *local.dual-order.antisym* **by** *blast*
qed

lemma *do-box-sup-inf-var*: $p \leq 1 \Longrightarrow$ *do-box* $(x \sqcup y)$ $p =$ *do-box* x $p \sqcap$ *do-box* y p
  by (*simp add*: *do-box-subid do-box-sup-inf subid-mult-meet*)

lemma *do-box-Sup-Inf*:
  assumes $X \neq \{\}$
  and $p \leq 1$
  shows *do-box* $(\bigsqcup X)$ $p = (\bigsqcap x \in X.$ *do-box* x $p)$
proof−
  {**fix** q
    assume *h*: $q \leq 1$
    hence $(q \leq$ *do-box* $(\bigsqcup X)$ $p) = ($*do-dia* $(\bigsqcup X)$ $q \leq p)$
    by (*simp add*: *do-dia-do-box-galois assms*(*2*))
  also have … $= ((\bigsqcup x \in X.$ *do-dia* x $q) \leq p)$
    using *do-dia-Sup* **by** *force*
  also have … $= (\forall x \in X.$ *do-dia* x $q \leq p)$
    by (*simp add*: *local.SUP-le-iff*)
  also have … $= (\forall x \in X.\ q \leq$ *do-box* x $p)$
    using *do-dia-do-box-galois assms*(*2*) *h* **by** *blast*
  also have … $= (q \leq (\bigsqcap x \in X.$ *do-box* x $p))$
    by (*simp add*: *local.le-INF-iff*)
  finally have $(q \leq$ *do-box* $(\bigsqcup X)$ $p) = (q \leq (\bigsqcap x \in X.$ *do-box* x $p))$.}
  hence $\forall y.\ y \leq$ *do-box* $(\bigsqcup X)$ $p \longleftrightarrow y \leq (\bigsqcap x \in X.$ *do-box* x $p)$
      by (*smt* (*verit*, *ccfv-threshold*) *assms*(*1*) *do-box-subid local.INF-le-SUP local.SUP-least local.order-trans*)
  thus *?thesis*

using *local.dual-order.antisym* **by** *blast*
qed

lemma *do-box-Sup-Inf2*:
assumes $P \neq \{\}$
and $\forall p \in P.\ p \leq 1$
shows $do\text{-}box\ x\ (\bigsqcap P) = (\bigsqcap p \in P.\ do\text{-}box\ x\ p)$
proof−
have *h0*: $(\bigsqcap p \in P.\ do\text{-}box\ x\ p) \leq 1$
by (*meson all-not-in-conv assms*(*1*) *do-box-subid local.INF-lower2*)
{fix *q*
assume *h1*: $q \leq 1$
hence $(q \leq do\text{-}box\ x\ (\bigsqcap P)) = (do\text{-}dia\ x\ q \leq \bigsqcap P)$
by (*simp add*: *assms do-dia-do-box-galois local.Inf-less-eq*)
also have $\ldots = (\forall p \in P.\ do\text{-}dia\ x\ q \leq p)$
using *local.le-Inf-iff* **by** *blast*
also have $\ldots = (\forall p \in P.\ q \leq do\text{-}box\ x\ p)$
using *assms*(*2*) *do-dia-do-box-galois h1* **by** *auto*
also have $\ldots = (q \leq (\bigsqcap p \in P.\ do\text{-}box\ x\ p))$
by (*simp add*: *local.le-INF-iff*)
finally have $(q \leq do\text{-}box\ x\ (\bigsqcap P)) = (q \leq (\bigsqcap p \in P.\ do\text{-}box\ x\ p))$**.}**
thus *?thesis*
using *do-box-subid h0 local.dual-order.antisym* **by** *blast*
qed

lemma *cd-box-sup-inf*: $p \leq 1 \Longrightarrow cd\text{-}box\ (x \sqcup y)\ p = cd\text{-}box\ x\ p \cdot cd\text{-}box\ y\ p$
by (*metis do-box-cd-box-conv do-box-sup-inf local.inv-binsup*)

lemma *cd-box-sup-inf-var*: $p \leq 1 \Longrightarrow cd\text{-}box\ (x \sqcup y)\ p = cd\text{-}box\ x\ p \sqcap cd\text{-}box\ y\ p$
by (*simp add*: *cd-box-subid cd-box-sup-inf subid-mult-meet*)

lemma *cd-box-Sup-Inf*:
assumes $X \neq \{\}$
and $p \leq 1$
shows $cd\text{-}box\ (\bigsqcup X)\ p = (\bigsqcap x \in X.\ cd\text{-}box\ x\ p)$
proof−
have $cd\text{-}box\ (\bigsqcup X)\ p = do\text{-}box\ ((\bigsqcup X)^\circ)\ p$
using *assms*(*2*) *do-box-cd-box-conv* **by** *presburger*
also have $\ldots = (\bigsqcap y \in \{x^\circ | x.\ x \in X\}.\ do\text{-}box\ y\ p)$
by (*simp add*: *Setcompr-eq-image assms do-box-Sup-Inf*)
also have $\ldots = (\bigsqcap x \in X.\ do\text{-}box\ (x^\circ)\ p)$
by (*simp add*: *Setcompr-eq-image image-image*)
also have $\ldots = (\bigsqcap x \in X.\ cd\text{-}box\ x\ p)$
using *assms*(*2*) *do-box-cd-box-conv* **by** *force*
finally show *?thesis***.**
qed

lemma *cd-box-Sup-Inf2*:
assumes $P \neq \{\}$

and $\forall p \in P.\ p \leq 1$
shows $\mathit{cd\text{-}box}\ x\ (\bigsqcap P) = (\bigsqcap p \in P.\ \mathit{cd\text{-}box}\ x\ p)$
proof−
  have $\mathit{cd\text{-}box}\ x\ (\bigsqcap P) = \mathit{do\text{-}box}\ (x^\circ)\ (\bigsqcap P)$
    by (*simp add*: *assms do-box-cd-box-conv local.Inf-less-eq*)
  also have $\ldots = (\bigsqcap p \in P.\ \mathit{do\text{-}box}\ (x^\circ)\ p)$
    using *assms do-box-Sup-Inf2* **by** *presburger*
  also have $\ldots = (\bigsqcap p \in P.\ \mathit{cd\text{-}box}\ x\ p)$
    using *assms*(*2*) *do-box-cd-box-conv* **by** *force*
  finally show *?thesis*.
qed

Next we define an antidomain operation in the style of modal semirings. A natural condition is that the antidomain of an element is the greatest test that cannot be left-composed with that elements, and hence a greatest left annihilator. The definition of anticodomain is similar. As we are not in a boolean domain algebra, we cannot expect that the antidomain of the antidomain yields the domain or that the union of a domain element with the corresponding antidomain element equals one.

definition $\mathit{ado}\ x = \bigsqcup\{p.\ p \cdot x = \bot \wedge p \leq 1\}$

definition $\mathit{acd}\ x = \bigsqcup\{p.\ x \cdot p = \bot \wedge p \leq 1\}$

lemma *ado-acd*: $\mathit{ado}\ (x^\circ) = \mathit{acd}\ x$
  unfolding *ado-def acd-def*
  by (*metis convdqka.subid-conv invqka.inv-zero local.inv-contrav local.inv-invol*)

lemma *acd-ado*: $\mathit{acd}\ (x^\circ) = \mathit{ado}\ x$
  unfolding *ado-def acd-def*
  by (*metis acd-def ado-acd ado-def local.inv-invol*)

lemma *ado-left-zero* [*simp*]: $\mathit{ado}\ x \cdot x = \bot$
  unfolding *ado-def*
  using *dmq.coddual.Sup-distl* **by** *auto*

lemma *acd-right-zero* [*simp*]: $x \cdot \mathit{acd}\ x = \bot$
  unfolding *acd-def*
  by (*simp add*: *dmq.coddual.h-Sup local.dual-order.antisym*)

lemma *ado-greatest*: $p \leq 1 \Longrightarrow p \cdot x = \bot \Longrightarrow p \leq \mathit{ado}\ x$
  by (*simp add*: *ado-def local.Sup-upper*)

lemma *acd-greatest*: $p \leq 1 \Longrightarrow x \cdot p = \bot \Longrightarrow p \leq \mathit{acd}\ x$
  by (*simp add*: *acd-def local.Sup-upper*)

lemma *ado-subid*: $\mathit{ado}\ x \leq 1$
  using *ado-def local.Sup-le-iff* **by** *force*

lemma *acd-subid*: $acd\ x \leq 1$
by (*simp add*: *acd-def local.Sup-le-iff*)

lemma *ado-left-zero-iff*: $p \leq 1 \Longrightarrow (p \leq ado\ x) = (p \cdot x = \bot)$
by (*metis ado-greatest ado-left-zero local.bot-unique local.mult-isor*)

lemma *acd-right-zero-iff*: $p \leq 1 \Longrightarrow (p \leq acd\ x) = (x \cdot p = \bot)$
by (*metis acd-greatest acd-right-zero local.bot-unique local.mult-isol*)

This gives an eqational characterisation of antidomain and anticodomain.

lemma *ado-cd-bot*: $ado\ x = cd\ (\bot \leftarrow x)$
proof−
{**fix** p
assume h: $p \leq 1$
hence $(p \leq ado\ x) = (p \cdot x = \bot)$
by (*simp add*: *ado-left-zero-iff*)
also have $\ldots = (p \leq \bot \leftarrow x)$
using *local.bot-unique local.fres-galois* **by** *blast*
also have $\ldots = (p \leq 1 \sqcap (\bot \leftarrow x))$
by (*simp add*: *h*)
also have $\ldots = (p \leq cd\ (\bot \leftarrow x))$
by (*metis cd-fres-prop local.bot-least local.mult-botr*)
finally have $(p \leq ado\ x) = (p \leq cd\ (\bot \leftarrow x))$**.**}
hence $\forall y.\ y \leq ado\ x \longleftrightarrow y \leq cd\ (\bot \leftarrow x)$
using *ado-subid local.cd-subid local.dual-order.trans* **by** *blast*
thus *?thesis*
using *local.dual-order.antisym* **by** *blast*
qed

lemma *acd-do-bot*: $acd\ x = do\ (x \rightarrow \bot)$
by (*metis ado-acd ado-cd-bot invqka.inv-zero local.bres-fres-conv local.cd-inv local.inv-invol*)

lemma *ado-cd-bot-id*: $ado\ x = 1 \sqcap (\bot \leftarrow x)$
by (*metis ado-cd-bot cd-fres-prop local.cd-bot local.cd-subid local.mult-botr*)

lemma *acd-do-bot-id*: $acd\ x = 1 \sqcap (x \rightarrow \bot)$
by (*metis acd-do-bot do-bres-prop local.bot-least local.mult-botl*)

lemma *ado-cd-bot-var*: $ado\ x = cd\ (\bot \leftarrow do\ x)$
by (*metis ado-cd-bot do-top2 local.fres-bot-top local.fres-curry*)

lemma *acd-do-bot-var*: $acd\ x = do\ (cd\ x \rightarrow \bot)$
by (*metis acd-do-bot cd-top2 local.bres-curry local.bres-top-bot*)

lemma *ado-do-bot*: $ado\ x = do\ (do\ x \rightarrow \bot)$
using *acd-ado acd-do-bot-var local.cd-inv* **by** *auto*

lemma $do\ x = ado\ (ado\ x)$

oops

lemma *acd-cd-bot*: *acd x* = *cd* (⊥ ← *cd x*)
by (*metis ado-acd ado-cd-bot-var local.cd-inv local.inv-invol*)

lemma *ado-do-bot-var*: *ado x* = *1* ⊓ (*do x* → ⊥)
by (*metis ado-do-bot do-bres-prop local.bot-unique local.bres-bot-bot local.bres-canc1 local.do-bot local.do-subid*)

lemma *acd-cd-bot-var*: *acd x* = *1* ⊓ (⊥ ← *cd x*)
by (*metis acd-cd-bot acd-right-zero cd-top local.fres-top-top2*)

Domain and codomain are compatible with the boxes.

lemma *cd-box-ado*: *cd-box x* ⊥ = *ado x*
by (*simp add*: *ado-cd-bot fres-cd-box*)

lemma *do-box-acd*: *do-box x* ⊥ = *acd x*
by (*simp add*: *acd-do-bot bres-do-box*)

lemma *ado-subid-prop*: *p* ≤ *1* ⟹ *ado p* = *1* ⊓ (*p* → ⊥)
by (*metis ado-do-bot-var do-fix-subid*)

lemma *ado-do*: *p* ≤ *1* ⟹ *ado p* = *do* (*p* → ⊥)
using *ado-do-bot do-fix-subid* **by** *force*

lemma *ado-do-compl*: *ado x* · *do x* = ⊥
using *dmq.dqmsr.dom-weakly-local* **by** *force*

lemma *ado x* ⊔ *do x* = ⊤
oops

lemma ∀ *x p*. ∃ *f*. *1* ⊓ (⊤ · *p* ← *x*) = *1* ⊓ (⊥ ← (*x* → *p* · ⊤))
oops

lemma *cd-box x p* = *ado* (*x* · *ado p*)
oops

lemma *ad-do-bot* [*simp*]: (*1* ⊓ (*do x* → ⊥)) · *do x* = ⊥
using *ado-do-bot-var ado-left-zero dmq.dqmsr.dom-weakly-local* **by** *presburger*

lemma *do-heyting-galois*: (*do x* · *do y* ≤ *do z*) = (*do x* ≤ *1* ⊓ (*do y* → *do z*))
by (*metis dmq.dqmsr.dsg4 dmq.mqdual.cod-subid local.bres-galois local.le-inf-iff*)

lemma *do-heyting-galois-var*: (*do x* · *do y* ≤ *do z*) = (*do x* ≤ *cd-box* (*do y*) (*do z*))
by (*metis cd-dia-cd-box-galois cd-fix-subid dmq.coddual.dqmsr.fd-def dmq.dqmsr.dom-mult-closed local.do-subid*)

Antidomain is therefore Heyting negation.

lemma *ado-heyting-negation*: *ado* (*do x*) = *cd-box* (*do x*) ⊥
  by (*simp add*: *cd-box-ado*)

lemma *ad-ax1* [*simp*]: (*1* ⊓ (*do x* → ⊥)) · *x* = ⊥
  by (*simp add*: *local.dmq.dqmsr.dom-weakly-local*)

lemma *1* ⊓ (*do* (*1* ⊓ (*do x* → ⊥)) → ⊥) = *do x*
  oops

lemma *p* ≤ *1* ⟹ *do-dia x p* = *1* ⊓ (*cd-box x* (*1* ⊓ (*p* → ⊥)) → ⊥)
  oops

lemma *p* ≤ *1* ⟹ *cd-box x p* = *1* ⊓ (*do-dia x* (*1* ⊓ (*p* → ⊥)) → ⊥)
  oops

lemma *p* ≤ *1* ⟹ *cd-dia x p* = *1* ⊓ (*do-box x* (*1* ⊓ (*p* → ⊥)) → ⊥)
  oops

lemma *p* ≤ *1* ⟹ *do-box x p* = *1* ⊓ (*cd-dia x* (*1* ⊓ (*p* → ⊥)) → ⊥)
  oops

end

6.4 Boolean Dedekind quantales

class *distributive-dedekind-quantale* = *distrib-unital-quantale* + *dedekind-quantale*

class *boolean-dedekind-quantale* = *bool-unital-quantale* + *distributive-dedekind-quantale*

begin

lemma *ad-do-bot* [*simp*]: (*1* − *do x*) · *do x* = ⊥
  by (*simp add*: *local.diff-eq local.inf-shunt local.subid-mult-meet*)

lemma *ad-ax1* [*simp*]: (*1* − *do x*) · *x* = ⊥
  by (*simp add*: *local.dmq.dqmsr.dom-weakly-local*)

lemma *ad-do* [*simp*]: *1* − *do* (*1* − *do x*) = *do x*
proof−
  have *1* − *do* (*1* − *do x*) = *1* − (*1* − *do x*)
    by (*metis local.diff-eq local.do-fix-subid local.inf.cobounded1*)
  also have ... = *1* ⊓ −(*1* ⊓ − *do x*)
    by (*simp add*: *local.diff-eq*)
  also have ... = *do x*
    by (*simp add*: *local.chaq.wswq.distrib-left local.do-top*)
  finally show *?thesis*.
qed

lemma *ad-ax2*: *1* − *do* (*x* · *y*) ⊔ (*1* − *do* (*x* · (*1* − *do* (*1* − *do y*)))) = *1* − *do*

$(x \cdot (1 - do\ (1 - do\ y)))$
by *simp*

lemma *ad-ax3* [*simp*]: $do\ x \sqcup (1 - do\ x) = 1$
proof−
have $do\ x \sqcup (1 - do\ x) = do\ x \sqcup (1 \sqcap -do\ x)$
using *local.diff-eq* **by** *force*
also have $\ldots = 1 \sqcap (do\ x \sqcup -do\ x)$
by (*simp add*: *local.chaq.wswq.distrib-left local.do-top*)
also have $\ldots = 1$
by *simp*
finally show *?thesis*.
qed

sublocale *bdad*: *antidomain-semiring* $\lambda x.\ 1 - do\ x\ (\sqcup)\ (\cdot)\ 1\ \bot$ - -
by *unfold-locales simp-all*

sublocale *bdadka*: *antidomain-kleene-algebra* $\lambda x.\ 1 - do\ x\ (\sqcup)\ (\cdot)\ 1\ \bot$ - - *qstar*..

sublocale *bdar*: *antirange-semiring* $(\sqcup)\ (\cdot)\ 1\ \bot\ \lambda x.\ 1 - cd\ x$ - -
apply *unfold-locales*
apply (*metis ad-ax1 ad-do dmq.mqs.local-var local.docd-compat*)
apply (*metis ad-do local.cddo-compat local.dmq.coddual.dqmsr.dsr2 local.docd-compat local.sup.idem*)
by (*metis bdad.a-subid′ bdad.as3 local.convdqka.subid-conv local.do-inv*)

sublocale *bdaka*: *antirange-kleene-algebra* $(\sqcup)\ (\cdot)\ 1\ \bot$ - - *qstar* $\lambda x.\ 1 - cd\ x$..

sublocale *bmod*: *modal-semiring-simp* $\lambda x.\ 1 - do\ x\ (\sqcup)\ (\cdot)\ 1\ \bot$ - - $\lambda x.\ 1 - cd\ x$..

sublocale *bmod*: *modal-kleene-algebra-simp* $(\sqcup)\ (\cdot)\ 1\ \bot$ - - *qstar* $\lambda x.\ 1 - do\ x\ \lambda x.$
$1 - cd\ x$..

lemma *inv-neg*: $(-x)^\circ = -(x^\circ)$
by (*metis local.diff-eq local.diff-shunt-var local.dual-order.eq-iff local.inf.commute local.inv-conjugate local.inv-galois*)

lemma *residuation*: $x^\circ \cdot -(x \cdot y) \leq -y$
by (*metis local.inf.commute local.inf-compl-bot local.inf-shunt local.schroeder-1*)

lemma *bres-prop*: $x \to y = -(x^\circ \cdot -y)$
by (*metis local.ba-dual.dual-iff local.bres-canc1 local.bres-galois-imp local.compl-le-swap1 local.dmq.coddual.h-w1 local.dual-order.antisym local.inv-invol residuation*)

lemma *fres-prop*: $x \leftarrow y = -(-x \cdot y^\circ)$
using *bres-prop inv-neg local.fres-bres-conv* **by** *auto*

lemma *do-dia-fdia*: $do\text{-}dia\ x\ p = bdad.fdia\ x\ p$
by (*simp add*: *local.bdad.fdia-def local.dmq.dqmsr.fd-def*)

lemma *cd-dia-bdia*: *cd-dia x p* = *bdar.bdia x p*
by (*metis ad-do bdar.ardual.a-subid′ bdar.bdia-def local.cd-fix-subid local.dmq.coddual.dqmsr.fd-def local.docd-compat*)

lemma *do-dia-fbox-de-morgan*: $p \leq 1 \Longrightarrow$ *do-dia x p* = 1 − *bdad.fbox x* (1 − *p*)
by (*smt* (*verit*, *ccfv-SIG*) *ad-do bdad.a-closure bdad.am-d-def bdad.fbox-def local.dmq.dqmsr.fd-def local.do-fix-subid*)

lemma *fbox-do-dia-de-morgan*: $p \leq 1 \Longrightarrow$ *bdad.fbox x p* = 1 − *do-dia x* (1 − *p*)
using *bdad.fbox-def local.dmq.dqmsr.fd-def local.do-fix-subid* **by** *force*

lemma *cd-dia-bbox-de-morgan*: $p \leq 1 \Longrightarrow$ *cd-dia x p* = 1 − *bdar.bbox x* (1 − *p*)
by (*smt* (*verit*, *best*) *ad-do bdar.bbox-def bdar.bdia-def cd-dia-bdia local.cd-fix-subid local.do-subid local.docd-compat*)

lemma *bbox-cd-dia-de-morgan*: $p \leq 1 \Longrightarrow$ *bdar.bbox x p* = 1 − *cd-dia x* (1 − *p*)
using *bdar.bbox-def local.cd-fix-subid local.dmq.coddual.dqmsr.fd-def* **by** *force*

lemma *do-box-bbox*: $p \leq 1 \Longrightarrow$ *do-box x p* = *bdar.bbox x p*
proof−
assume *a*: $p \leq 1$
{fix *q*
assume *b*: $q \leq 1$
hence ($q \leq$ *do-box x p*) = ($x \cdot q \leq p \cdot x$)
by (*simp add*: *a local.do-box-demod-subid*)
also have ... = ($x \cdot$ *cd q* $\leq$ *cd p* $\cdot x$)
by (*metis a b local.cd-fix-subid*)
also have ... = (*cd q* $\leq$ *bdar.bbox x p*)
by (*metis bdar.bbox-def bdar.bdia-def cd-dia-bdia local.bdar.ardual.a-closure′ local.bdar.ardual.ans-d-def local.bdar.ardual.dnsz.dsg1 local.bdar.ardual.fbox-demodalisation3 local.bdar.ardual.fbox-one local.dmq.coddual.dqmsr.fd-def local.nsrnqo.mult-oner*)
also have ... = ($q \leq$ *bdar.bbox x p*)
using *b local.cd-fix-subid* **by** *force*
finally have ($q \leq$ *do-box x p*) = ($q \leq$ *bdar.bbox x p*)**.}**
thus *?thesis*
by (*metis bdar.bbox-def local.bdar.ardual.a-subid′ local.do-box-subid local.dual-order.antisym local.eq-refl*)
qed

lemma *cd-box-fbox*: $p \leq 1 \Longrightarrow$ *cd-box x p* = *bdad.fbox x p*
by (*smt* (*verit*, *ccfv-SIG*) *bdar.bbox-def do-box-bbox local.bdad.fbox-def local.cd-dia-do-dia-conv local.cd-inv local.cd-fix-subid local.cd-top local.diff-eq local.dmq.bb-def local.dmq.coddual.dqmsr.fd-def local.dmq.coddual.mult-assoc local.dmq.dqmsr.fd-def local.dmq.fb-def local.do-box-cd-box-conv local.do-fix-subid local.do-top local.inf.cobounded1*)

lemma *do-dia-cd-box-de-morgan*: $p \leq 1 \Longrightarrow$ *do-dia x p* = 1 − *cd-box x* (1 − *p*)
by (*simp add*: *cd-box-fbox local.diff-eq local.do-dia-fbox-de-morgan*)

lemma *cd-box-do-dia-de-morgan*: $p \leq 1 \Longrightarrow$ *cd-box* $x\ p = 1 - $ *do-dia* $x\ (1 - p)$
  by (*simp add*: *cd-box-fbox local.fbox-do-dia-de-morgan*)

lemma *cd-dia-do-box-de-morgan*: $p \leq 1 \Longrightarrow$ *cd-dia* $x\ p = 1 - $ *do-box* $x\ (1 - p)$
  by (*simp add*: *do-box-bbox local.cd-dia-bbox-de-morgan local.diff-eq*)

lemma *do-box-cd-dia-de-morgan*: $p \leq 1 \Longrightarrow$ *do-box* $x\ p = 1 - $ *cd-dia* $x\ (1 - p)$
  by (*simp add*: *do-box-bbox local.bbox-cd-dia-de-morgan*)

end

class *dc-involutive-modal-quantale* = *dc-modal-quantale* + *involutive-quantale*

begin

sublocale *invmqmka*: *involutive-dr-modal-kleene-algebra* $(\sqcup)\ (\cdot)\ 1\ \bot\ (\leq)\ (<)$ *qstar invol dom cod*..

lemma *do-approx-dom*: *do* $x \leq$ *dom* x
proof −
  have *do* $x =$ *dom* (*do* x) $\cdot$ *do* x
    by *simp*
  also have $\ldots \leq$ *dom* $(1 \sqcap (x \cdot x^\circ))$
    by (*simp add*: *local.do-def local.dqmsr.domain-subid*)
  also have $\ldots \leq$ *dom* $1 \sqcap$ *dom* $(x \cdot x^\circ)$
    using *local.dom-meet-sub* **by** *presburger*
  also have $\ldots \leq$ *dom* $(x \cdot$ *dom* $(x^\circ))$
    by *simp*
  also have $\ldots \leq$ *dom* x
    by (*simp add*: *local.dqmsr.domain-1''*)
  finally show *?thesis*.
qed

end

class *dc-modal-quantale-converse* = *dc-involutive-modal-quantale* + *quantale-converse*

sublocale *dc-modal-quantale-converse* ⊆ *invmqmka*: *dr-modal-kleene-algebra-converse* $(\sqcup)\ (\cdot)\ 1\ \bot\ (\leq)\ (<)$ *qstar invol dom cod*..

class *dc-modal-quantale-strong-converse* = *dc-involutive-modal-quantale* +
  assumes *weak-dom-def*: *dom* $x \leq x \cdot x^\circ$
  and *weak-cod-def*: *cod* $x \leq x^\circ \cdot x$

begin

sublocale *invmqmka*: *dr-modal-kleene-algebra-strong-converse* $(\sqcup)\ (\cdot)\ 1\ \bot\ (\leq)\ (<)$ *qstar invol dom cod*
  by (*unfold-locales*, *simp-all add*: *local.weak-dom-def local.weak-cod-def*)

lemma *dom-def*: *dom* $x = 1 \sqcap (x \cdot x^{\circ})$
  using *local.do-approx-dom local.do-def local.dual-order.eq-iff local.weak-dom-def*
by *force*

lemma *cod-def*: *cod* $x = 1 \sqcap (x^{\circ} \cdot x)$
  using *local.dom-def local.invmqmka.d-conv-cod* **by** *auto*

lemma *do-dom*: *do* $x =$ *dom* x
  by (*simp add*: *local.do-def local.dom-def*)

lemma *cd-cod*: *cd* $x =$ *cod* x
  by (*simp add*: *local.cd-def local.cod-def*)

end

class *dc-modal-dedekind-quantale* = *dc-involutive-modal-quantale* + *dedekind-quantale*

class *cd-distributive-modal-dedekind-quantale* = *dc-modal-dedekind-quantale* + *distrib-unital-quantale*

class *dc-boolean-modal-dedekind-quantale* = *dc-modal-dedekind-quantale* + *bool-unital-quantale*

begin

lemma *subid-idem*: $p \leq 1 \Longrightarrow p \cdot p = p$
  by (*simp add*: *local.subid-mult-meet*)

lemma *subid-comm*: $p \leq 1 \Longrightarrow q \leq 1 \Longrightarrow p \cdot q = q \cdot p$
  using *local.inf.commute local.subid-mult-meet* **by** *force*

lemma *subid-meet-comp*: $p \leq 1 \Longrightarrow q \leq 1 \Longrightarrow p \sqcap q = p \cdot q$
  by (*simp add*: *local.subid-mult-meet*)

lemma *subid-dom*: $p \leq 1 \Longrightarrow$ *dom* $p = p$
proof−
  assume *h*: $p \leq 1$
  have *a*: $p \sqcup (1 \sqcap -p) = 1$
   by (*metis h local.inf-sup-absorb local.sup.commute local.sup.orderE local.sup-compl-top local.sup-inf-distrib1*)
  have *b*: $(1 \sqcap -p) \sqcap p = \bot$
    by (*simp add*: *local.inf.commute*)
  hence *dom* $p = (p \sqcup (1 \sqcap -p)) \cdot$ *dom* p
    by (*simp add*: *a*)
  also have $\ldots = p \cdot$ *dom* $p \sqcup$ *dom* $((1 \sqcap -p) \cdot$ *dom* $p) \cdot (1 \sqcap -p) \cdot$ *dom* p
    using *local.dqmsr.dsg1 local.wswq.distrib-right mult-assoc* **by** *presburger*
  also have $\ldots \leq p \sqcup$ *dom* $((1 \sqcap -p) \cdot$ *dom* $p)$
   by (*metis b h local.dom-subid local.dom-zero local.inf.cobounded1 local.mqdual.cod-local local.mult-botr local.sup.mono subid-comm subid-meet-comp*)

also have ... = $p \sqcup dom\ ((1 \sqcap -p) \cdot p)$
by *simp*
also have ... = $p \sqcup dom\ \bot$
using *b h subid-meet-comp* **by** *fastforce*
also have ... = p
by *simp*
finally have $dom\ p \le p$.
thus *?thesis*
using *h local.dqmsr.domain-subid local.dual-order.antisym* **by** *presburger*
qed

lemma *do-prop*: $(do\ x \le do\ y) = (x \le do\ y \cdot \top)$
by (*simp add*: *local.lla*)

lemma *do-lla*: $(do\ x \le do\ y) = (x \le do\ y \cdot x)$
by (*simp add*: *local.lla-var*)

lemma *lla-subid*: $p \le 1 \Longrightarrow ((dom\ x \le p) = (x \le p \cdot x))$
by (*metis local.dqmsr.dom-llp subid-dom*)

lemma *dom-do*: $dom\ x = do\ x$
proof−
have $x \le do\ x \cdot x$
by *simp*
hence $dom\ x \le do\ x$
using *lla-subid local.do-subid local.dodo* **by** *presburger*
thus *?thesis*
by (*simp add*: *local.antisym-conv local.do-approx-dom*)
qed

end

end

References

[1] A. Armstrong, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2013.

[2] C. Calk, E. Goubault, P. Malbos, and G. Struth. Algebraic coherent confluence and higher globular Kleene algebras. *Logical Methods in Computer Science*, 18(4), 2022.

[3] C. Calk, P. Malbos, D. Pous, and G. Struth. Higher catoids, higher quantales and their correspondences. *arXiv*, 2307.09253, 2023.

[4] U. Fahrenberg, C. Johansen, G. Struth, and K. Ziemiański. Catoids and modal convolution algebras. *Algebra Universalis*, 84:10, 2023.

[5] V. B. F. Gomes, W. Guttmann, P. Höfner, G. Struth, and T. Weber. Kleene algebras with domain. *Archive of Formal Proofs*, 2016.

[6] V. B. F. Gomes and G. Struth. Modal Kleene algebra applied to program correctness. In *FM 2016*, volume 9995 of *LNCS*, pages 310–325, 2016.

[7] D. Pous and G. Struth. Dedekind quantaloids as intuitionistic modal algebras. In preparation, 2023.

[8] P. Resende. Open maps of involutive quantales. *Applied Categorical Structures*, 26:631–644, 2018.

[9] K. I. Rosenthal. *The Theory of Quantaloids*. Addison Wesley Longman Limited, 1996.

[10] G. Struth. Quantales. *Archive of Formal Proofs*, 2018.

[11] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.