Residuated Lattices

Victor B. F. Gomes
Georg Struth
Department of Computer Science, University of Sheffield

September 13, 2023

Abstract

The theory of residuated lattices, first proposed by Ward and Dilworth [5], is formalised in Isabelle/HOL. This includes concepts of residuated functions; their adjoints and conjugates. It also contains necessary and sufficient conditions for the existence of these operations in an arbitrary lattice. The mathematical components for residuated lattices are linked to the AFP entry for relation algebra. In particular, we prove Jónsson and Tsinakis [2] conditions for a residuated boolean algebra to form a relation algebra.

Contents

1 Introduction

text * These theory files formalise algebraic residuated structures. They are briefly and sparsely commented. More information can be found in the books by Galatos and *al.* [1], or the originals papers by Ward and Dilworth [5], Jonsson and Tsinakis [2], and Maddux [3].

The mathematical components for residuated lattices are linked to the AFP entry for relation algebra. Residuated lattices are also important in the context of Pratt's action algebras, which are currently formalised whitin the AFP entry for Kleene algebra. We are planning to link Kleene algebras and action algebras with this entry in the future.

Isabelle/HOL default notation for lattices is used whenever possible. Nevertheless, we use $\cdot$ as the multiplicative symbol instead of $*$, which is the one used in Isabelle libraries.

theory *Residuated-Lattices*
imports *Kleene-Algebra.Signatures*
begin

notation
times (**infixl** $\cdot$ *70*)

unbundle *lattice-syntax*

2 Residuated Lattices

2.1 Residuated Functions on a Partial Order

We follow Galatos and *al.* to define residuated funtions on partial orders. Material from articles by Maddux, and Jósson and Tsinakis are also considered.

This development should in the future be expanded to functions or categories where the sources and targets have different type.

Let P be a partial order, or a poset. A map $f : P \to P$ is residuated if there exists a map $g : P \to P$ such that $f(x) \leq y \Leftrightarrow x \leq g(y)$ for all $x, y \in P$. That is, they are adjoints of a Galois connection.

context *order* **begin**

definition *residuated-pair* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow bool$ **where**
residuated-pair f g $\equiv \forall\, x\ y.\ f(x) \leq y \longleftrightarrow x \leq g(y)$

theorem *residuated-pairI* [*intro*]:
assumes $\forall\, x\ y.\ x \leq y \longrightarrow f\ x \leq f\ y$
and $\forall\, x\ y.\ x \leq y \longrightarrow g\ x \leq g\ y$
and $\forall\, x.\ x \leq (g\ o\ f)\ x$
and $\forall\, x.\ (f\ o\ g)\ x \leq x$
shows *residuated-pair f g*
⟨*proof*⟩

definition *residuated* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where**
residuated f $\equiv \exists\, g.$ *residuated-pair f g*

If a map f is residuated, then its residual g is unique.

lemma *residual-unique*: *residuated f* $\Longrightarrow \exists!g.$ *residuated-pair f g*
⟨*proof*⟩

Since the residual of a map f is unique, it makes sense to define a residual operator.

definition *residual* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a)$ **where**
residual f $\equiv$ *THE g. residuated-pair f g*

lemma *residual-galois*: *residuated f* $\Longrightarrow f(x) \leq y \longleftrightarrow x \leq$ *residual f y*
⟨*proof*⟩

lemma *residual-galoisI1*: *residuated f* $\Longrightarrow f(x) \leq y \Longrightarrow x \leq$ *residual f y*
⟨*proof*⟩

lemma *residual-galoisI2*: *residuated f* $\Longrightarrow x \leq$ *residual f y* $\Longrightarrow f(x) \leq y$
⟨*proof*⟩

A closure operator on a poset is a map that is expansive, isotone and idempotent. The composition of the residual of a function f with f is a closure operator.

definition *closure* :: $('a \Rightarrow 'a) \Rightarrow bool$ **where**
closure f $\equiv (\forall\, x.\ x \leq f\ x) \wedge (\forall\, x\ y.\ x \leq y \longrightarrow f\ x \leq f\ y) \wedge (\forall\, x.\ f(f\ x) = f\ x)$

lemma *res-c1*: *residuated f* $\Longrightarrow x \leq$ *residual f* $(f\ x)$
⟨*proof*⟩

lemma *res-c2*: *residuated f* $\Longrightarrow x \leq y \Longrightarrow$ *residual f* $(f\ x) \leq$ *residual f* $(f\ y)$
⟨*proof*⟩

lemma *res-c3*: *residuated f* $\Longrightarrow$ *residual f* $(f\ ($*residual f* $(f\ x))) =$ *residual f* $(f\ x)$
⟨*proof*⟩

lemma *res-closure*: *residuated f* $\Longrightarrow$ *closure* (*residual f o f*)
⟨*proof*⟩

Dually, an interior operator on a poset is a map that is contractive, isotone and idempotent. The composition of f with its residual is an interior operator.

definition *interior* :: ('*a* $\Rightarrow$ '*a*) $\Rightarrow$ *bool* **where**
interior f $\equiv$ ($\forall$ *x*. *f x* $\leq$ *x*) $\wedge$ ($\forall$ *x y*. *x* $\leq$ *y* $\longrightarrow$ *f x* $\leq$ *f y*) $\wedge$ ($\forall$ *x*. *f*(*f x*) = *f x*)

lemma *res-i1*: *residuated f* $\Longrightarrow$ *f* (*residual f x*) $\leq$ *x*
⟨*proof*⟩

lemma *res-i2*: *residuated f* $\Longrightarrow$ *x* $\leq$ *y* $\Longrightarrow$ *f* (*residual f x*) $\leq$ *f* (*residual f y*)
⟨*proof*⟩

lemma *res-i3*: *residuated f* $\Longrightarrow$ *f* (*residual f* (*f* (*residual f x*))) = *f* (*residual f x*)
⟨*proof*⟩

lemma *res-interior*: *residuated f* $\Longrightarrow$ *interior* (*f o residual f*)
⟨*proof*⟩

Next we show a few basic lemmas about residuated maps.

lemma *res-iso*: *residuated f* $\Longrightarrow$ *x* $\leq$ *y* $\Longrightarrow$ *f x* $\leq$ *f y*
⟨*proof*⟩

lemma *res-residual-iso*: *residuated f* $\Longrightarrow$ *x* $\leq$ *y* $\Longrightarrow$ *residual f x* $\leq$ *residual f y*
⟨*proof*⟩

lemma *res-comp1* [*simp*]: *residuated f* $\Longrightarrow$ *f o residual f o f* = *f*
⟨*proof*⟩

lemma *res-comp2* [*simp*]: *residuated f* $\Longrightarrow$ *residual f o f o residual f* = *residual f*
⟨*proof*⟩

end

A residuated function f preserves all existing joins. Dually, its residual preserves all existing meets. We restrict our attention to semilattices, where binary joins or meets exist, and to complete lattices, where arbitrary joins and meets exist.

lemma (**in** *semilattice-sup*) *residuated-sup*: *residuated f* $\Longrightarrow$ *f* (*x* $\sqcup$ *y*) = *f x* $\sqcup$ *f y*
⟨*proof*⟩

lemma (**in** *semilattice-inf*) *residuated-inf*: *residuated f* $\Longrightarrow$ *residual f* (*x* $\sqcap$ *y*) = *residual f x* $\sqcap$ *residual f y*
⟨*proof*⟩

context *bounded-lattice* **begin**

lemma *residuated-strict*: *residuated f* $\Longrightarrow$ $f\ \bot = \bot$
⟨*proof*⟩

lemma *res-top*: *residuated f* $\Longrightarrow$ *residual f* $\top = \top$
⟨*proof*⟩

end

context *complete-lattice* **begin**

On a complete lattice, a function f is residuated if and only if it preserves arbitrary (possibly infinite) joins. Dually, a function g is a residual of a residuated funtion f if and only if g preserver arbitrary meets.

lemma *residual-eq1*: *residuated f* $\Longrightarrow$ *residual f y* $= \bigsqcup\ \{x.\ f\ x \leq y\}$
⟨*proof*⟩

lemma *residual-eq2*: *residuated f* $\Longrightarrow$ $f\ x = \bigsqcap\ \{y.\ x \leq$ *residual f y*$\}$
⟨*proof*⟩

lemma *residuated-Sup*: *residuated f* $\Longrightarrow$ $f(\bigsqcup\ X) = \bigsqcup \{f\ x \mid x.\ x \in X\}$
⟨*proof*⟩

lemma *residuated-Inf*: *residuated f* $\Longrightarrow$ *residual f*$(\bigsqcap\ X) = \bigsqcap \{$*residual f x* $\mid x.\ x \in X\}$
⟨*proof*⟩

lemma *Sup-sup*: $\forall X.\ f(\bigsqcup\ X) = \bigsqcup \{f\ x \mid x.\ x \in X\} \Longrightarrow f\ (x \sqcup y) = f\ x \sqcup f\ y$
⟨*proof*⟩

lemma *Sup-residuatedI*: $\forall X.\ f(\bigsqcup\ X) = \bigsqcup \{f\ x \mid x.\ x \in X\} \Longrightarrow$ *residuated f*
⟨*proof*⟩

lemma *Inf-inf*: $\forall X.\ f(\bigsqcap\ X) = \bigsqcap \{f\ x \mid x.\ x \in X\} \Longrightarrow f\ (x \sqcap y) = f\ x \sqcap f\ y$
⟨*proof*⟩

lemma *Inf-residuatedI*: $\forall X. \bigsqcap \{g\ x \mid x.\ x \in X\} = g\ (\bigsqcap\ X) \Longrightarrow \exists f.$ *residuated-pair f g*
⟨*proof*⟩

end

2.2 Residuated Structures

In this section, we define residuated algebraic structures, starting from the simplest of all, a *residuated partial ordered groupoid*, to *residuated l-monoids*, which are residuated lattices where the multiplicative operator forms a monoid.

class *pogroupoid* = *order* + *times* +
  assumes *mult-isor*: $x \leq y \Longrightarrow x \cdot z \leq y \cdot z$
  and *mult-isol*: $x \leq y \Longrightarrow z \cdot x \leq z \cdot y$

A residuated partial ordered groupoid is simply a partial order and a multiplicative groupoid with two extra operators satisfying the residuation laws. It is straighforward to prove that multiplication is compatible with the order, that is, multiplication is isotone.

Most of the lemmas below come in pairs; they are related by opposition duality. Formalising this duality is left for future work.

class *residual-r-op* =
  fixes *residual-r* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\rightarrow$ *60*)

class *residual-l-op* =
  fixes *residual-l* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\leftarrow$ *60*)

class *residuated-pogroupoid* = *order* + *times* + *residual-l-op* + *residual-r-op* +
  assumes *resl-galois*: $x \leq z \leftarrow y \longleftrightarrow x \cdot y \leq z$
  and *resr-galois*: $x \cdot y \leq z \longleftrightarrow y \leq x \rightarrow z$
begin

lemma *reslI* [*intro*]: $x \cdot y \leq z \Longrightarrow x \leq z \leftarrow y$
  ⟨*proof*⟩

lemma *resrI* [*intro*]: $x \cdot y \leq z \Longrightarrow y \leq x \rightarrow z$
  ⟨*proof*⟩

lemma *residuated-pair-multl* [*simp*]: *residuated-pair* $(\lambda x.\ x \cdot y)$ $(\lambda x.\ x \leftarrow y)$
  ⟨*proof*⟩

lemma *residuated-pair-multr* [*simp*]: *residuated-pair* $(\lambda y.\ x \cdot y)$ $(\lambda y.\ x \rightarrow y)$
  ⟨*proof*⟩

Multiplication is then obviously residuated.

lemma *residuated-multl* [*simp*]: *residuated* $(\lambda x.\ x \cdot y)$
  ⟨*proof*⟩

lemma *residuated-multr* [*simp*]: *residuated* $(\lambda y.\ x \cdot y)$
  ⟨*proof*⟩

lemma *resl-eq* [*simp*]: *residual* $(\lambda x.\ x \cdot y) = (\lambda x.\ x \leftarrow y)$
  ⟨*proof*⟩

lemma *resr-eq* [*simp*]: *residual* $(\lambda y.\ x \cdot y) = (\lambda y.\ x \rightarrow y)$
  ⟨*proof*⟩

Next we prove a few lemmas, all of which are instantiation of more general facts about residuated functions.

lemma *res-lc1*: $x \leq x \cdot y \leftarrow y$
⟨*proof*⟩

lemma *res-lc2*: $x \leq y \Longrightarrow x \cdot z \leftarrow z \leq y \cdot z \leftarrow z$
⟨*proof*⟩

lemma *res-lc3* [*simp*]: $(x \cdot y \leftarrow y) \cdot y \leftarrow y = x \cdot y \leftarrow y$
⟨*proof*⟩

lemma *res-rc1*: $x \leq y \rightarrow y \cdot x$
⟨*proof*⟩

lemma *res-rc2*: $x \leq y \Longrightarrow z \rightarrow z \cdot x \leq z \rightarrow z \cdot y$
⟨*proof*⟩

lemma *res-rc3* [*simp*]: $y \rightarrow y \cdot (y \rightarrow y \cdot x) = y \rightarrow y \cdot x$
⟨*proof*⟩

lemma *res-li1*: $(x \leftarrow y) \cdot y \leq x$
⟨*proof*⟩

lemma *res-li2*: $x \leq y \Longrightarrow (x \leftarrow z) \cdot z \leq (y \leftarrow z) \cdot z$
⟨*proof*⟩

lemma *res-li3* [*simp*]: $((x \leftarrow y) \cdot y \leftarrow y) \cdot y = (x \leftarrow y) \cdot y$
⟨*proof*⟩

lemma *res-ri1*: $y \cdot (y \rightarrow x) \leq x$
⟨*proof*⟩

lemma *res-ri2*: $x \leq y \Longrightarrow z \cdot (z \rightarrow x) \leq z \cdot (z \rightarrow y)$
⟨*proof*⟩

lemma *res-ri3* [*simp*]: $y \cdot (y \rightarrow y \cdot (y \rightarrow x)) = y \cdot (y \rightarrow x)$
⟨*proof*⟩

subclass *pogroupoid*
⟨*proof*⟩

lemma *resl-iso*: $x \leq y \Longrightarrow x \leftarrow z \leq y \leftarrow z$
⟨*proof*⟩

lemma *resr-iso*: $x \leq y \Longrightarrow z \rightarrow x \leq z \rightarrow y$
⟨*proof*⟩

lemma *resl-comp1* [*simp*]: $(x \cdot y \leftarrow y) \cdot y = x \cdot y$
⟨*proof*⟩

lemma *resl-comp2* [*simp*]: $(x \leftarrow y) \cdot y \leftarrow y = x \leftarrow y$

⟨*proof*⟩

lemma *resr-comp1* [*simp*]: $y \cdot (y \rightarrow y \cdot x) = y \cdot x$
⟨*proof*⟩

lemma *resr-comp2* [*simp*]: $y \rightarrow y \cdot (y \rightarrow x) = y \rightarrow x$
⟨*proof*⟩

lemma *resl-antitoner*: $x \leq y \longrightarrow z \leftarrow y \leq z \leftarrow x$
⟨*proof*⟩

lemma *resr-antitonel*: $x \leq y \longrightarrow y \rightarrow z \leq x \rightarrow z$
⟨*proof*⟩

The following lemmas are taken from Galatos and *al.*

lemma *jipsen1l*: $x \leq y \leftarrow (x \rightarrow y)$
⟨*proof*⟩

lemma *jipsen1r*: $x \leq (y \leftarrow x) \rightarrow y$
⟨*proof*⟩

lemma *jipsen2l*: $(y \leftarrow (x \rightarrow y)) \rightarrow y = x \rightarrow y$
⟨*proof*⟩

lemma *jipsen2r*: $y \leftarrow ((y \leftarrow x) \rightarrow y) = y \leftarrow x$
⟨*proof*⟩

end

In a residuated partial ordered semigroup, the multiplicative operator is associative.

class *residuated-posemigroup* = *semigroup-mult* + *residuated-pogroupoid*
begin

lemma *resl-trans*: $(x \leftarrow y) \cdot (y \leftarrow z) \leq x \leftarrow z$
⟨*proof*⟩

lemma *resr-trans*: $(x \rightarrow y) \cdot (y \rightarrow z) \leq x \rightarrow z$
⟨*proof*⟩

lemma *resr1*: $(x \rightarrow y) \cdot z \leq (x \rightarrow y \cdot z)$
⟨*proof*⟩

lemma *resr2*: $x \rightarrow y \leq z \cdot x \rightarrow z \cdot y$
⟨*proof*⟩

lemma *resr3*: $x \cdot y \rightarrow z = y \rightarrow (x \rightarrow z)$
⟨*proof*⟩

lemma *resl1*: $z \cdot (x \leftarrow y) \leq (z \cdot x \leftarrow y)$
⟨*proof*⟩

lemma *resl2*: $x \leftarrow y \leq x \cdot z \leftarrow y \cdot z$
⟨*proof*⟩

lemma *resl3*: $x \leftarrow y \cdot z = (x \leftarrow z) \leftarrow y$
⟨*proof*⟩

lemma *residual-assoc*: $x \rightarrow (y \leftarrow z) = (x \rightarrow y) \leftarrow z$
⟨*proof*⟩

end

In a residuated partially ordered monoid, the multiplicative operator has a unit 1; that is, its reduct forms a monoid.

class *residuated-pomonoid* = *monoid-mult* + *residuated-pogroupoid*
begin

subclass *residuated-posemigroup* ⟨*proof*⟩

lemma *resl-unit*: $x \leftarrow 1 = x$
⟨*proof*⟩

lemma *resr-unit*: $1 \rightarrow x = x$
⟨*proof*⟩

lemma *mult-one-resl*: $x \cdot (1 \leftarrow y) \leq x \leftarrow y$
⟨*proof*⟩

lemma *mult-one-resr*: $(x \rightarrow 1) \cdot y \leq x \rightarrow y$
⟨*proof*⟩

More lemmas from Galatos and *al.* follow.

lemma *jipsen3l*: $1 \leq x \leftarrow x$
⟨*proof*⟩

lemma *jipsen3r*: $1 \leq x \rightarrow x$
⟨*proof*⟩

lemma *jipsen4l* [*simp*]: $(x \leftarrow x) \cdot x = x$
⟨*proof*⟩

lemma *jipsen4r* [*simp*]: $x \cdot (x \rightarrow x) = x$
⟨*proof*⟩

lemma *jipsen5l* [*simp*]: $(x \leftarrow x) \cdot (x \leftarrow x) = x \leftarrow x$
⟨*proof*⟩

lemma *jipsen5r* [*simp*]: $(x \rightarrow x) \cdot (x \rightarrow x) = x \rightarrow x$
⟨*proof*⟩

lemma *jipsen6l*: $y \leftarrow x \leq (y \leftarrow z) \leftarrow (x \leftarrow z)$
⟨*proof*⟩

lemma *jipsen6r*: $x \rightarrow y \leq (z \rightarrow x) \rightarrow (z \rightarrow y)$
⟨*proof*⟩

lemma *jipsen7l*: $y \leftarrow x \leq (z \leftarrow y) \rightarrow (z \leftarrow x)$
⟨*proof*⟩

lemma *jipsen7r*: $x \rightarrow y \leq (x \rightarrow z) \leftarrow (y \rightarrow z)$
⟨*proof*⟩

end

Residuated partially ordered groupoids can be expanded residuated join semilattice ordered groupoids. They are used as a base for action algebras, which are expansions of Kleene algebras by operations of residuation. Action algebras can be found in the AFP entry for Kleene algebras.

class *residuated-sup-lgroupoid* = *semilattice-sup* + *residuated-pogroupoid*
begin

lemma *distl*: $x \cdot (y \sqcup z) = x \cdot y \sqcup x \cdot z$
⟨*proof*⟩

lemma *distr*: $(x \sqcup y) \cdot z = x \cdot z \sqcup y \cdot z$
⟨*proof*⟩

lemma *resl-subdist-var*: $x \leftarrow z \leq (x \sqcup y) \leftarrow z$
⟨*proof*⟩

lemma *resl-subdist*: $(x \leftarrow z) \sqcup (y \leftarrow z) \leq (x \sqcup y) \leftarrow z$
⟨*proof*⟩

lemma *resr-subdist-var*: $(x \rightarrow y) \leq x \rightarrow (y \sqcup z)$
⟨*proof*⟩

lemma *resr-subdist*: $(x \rightarrow y) \sqcup (x \rightarrow z) \leq x \rightarrow (y \sqcup z)$
⟨*proof*⟩

lemma *resl-superdist-var*: $x \leftarrow (y \sqcup z) \leq x \leftarrow y$
⟨*proof*⟩

lemma *resr-superdist-var*: $(x \sqcup y) \rightarrow z \leq x \rightarrow z$
⟨*proof*⟩

end

Similarly, semigroup and monoid variants can be defined.

class *residuated-sup-lsemigroup* = *semilattice-sup* + *residuated-posemigroup*
subclass (**in** *residuated-sup-lsemigroup*) *residuated-sup-lgroupoid* ⟨*proof*⟩

class *residuated-sup-lmonoid* = *semilattice-sup* + *residuated-posemigroup*
subclass (**in** *residuated-sup-lmonoid*) *residuated-sup-lsemigroup* ⟨*proof*⟩

By lattice duality, we define residuated meet semillatice ordered groupoid.

class *residuated-inf-lgroupoid* = *semilattice-inf* + *residuated-pogroupoid*
begin

lemma *resl-dist*: $(x \sqcap y) \leftarrow z = (x \leftarrow z) \sqcap (y \leftarrow z)$
⟨*proof*⟩

lemma *resr-dist*: $x \rightarrow (y \sqcap z) = (x \rightarrow y) \sqcap (x \rightarrow z)$
⟨*proof*⟩

lemma *resl-inf-superdist*: $x \leftarrow y \leq x \leftarrow (y \sqcap z)$
⟨*proof*⟩

lemma *resr-inf-superdist-var*: $x \rightarrow y \leq (x \sqcap z) \rightarrow y$
⟨*proof*⟩

end

class *residuated-inf-lsemigroup* = *semilattice-inf* + *residuated-posemigroup*
subclass (**in** *residuated-inf-lsemigroup*) *residuated-inf-lgroupoid* ⟨*proof*⟩

class *residuated-inf-lmonoid* = *semilattice-inf* + *residuated-posemigroup*
subclass (**in** *residuated-inf-lmonoid*) *residuated-inf-lsemigroup* ⟨*proof*⟩

Finally, we obtain residuated lattice groupoids.

class *residuated-lgroupoid* = *lattice* + *residuated-pogroupoid*
begin

subclass *residuated-sup-lgroupoid* ⟨*proof*⟩

lemma *resl-distr*: $z \leftarrow (x \sqcup y) = (z \leftarrow x) \sqcap (z \leftarrow y)$
⟨*proof*⟩

lemma *resr-distl*: $(x \sqcup y) \rightarrow z = (x \rightarrow z) \sqcap (y \rightarrow z)$
⟨*proof*⟩

end

class *residuated-lsemigroup* = *lattice* + *residuated-sup-lsemigroup*
subclass (**in** *residuated-lsemigroup*) *residuated-lgroupoid* ⟨*proof*⟩

class *residuated-lmonoid* = *lattice* + *residuated-sup-lmonoid*

subclass (**in** *residuated-lmonoid*) *residuated-lsemigroup* ⟨*proof*⟩

class *residuated-blgroupoid* = *bounded-lattice* + *residuated-pogroupoid*
begin

lemma *multl-strict* [*simp*]: $x \cdot \bot = \bot$
⟨*proof*⟩

lemma *multr-strict* [*simp*]: $\bot \cdot x = \bot$
⟨*proof*⟩

lemma *resl-top* [*simp*]: $\top \leftarrow x = \top$
⟨*proof*⟩

lemma *resl-impl* [*simp*]: $x \leftarrow \bot = \top$
⟨*proof*⟩

lemma *resr-top* [*simp*]: $x \rightarrow \top = \top$
⟨*proof*⟩

lemma *resr-impl* [*simp*]: $\bot \rightarrow x = \top$
⟨*proof*⟩

end

class *residuated-blsemigroup* = *bounded-lattice* + *residuated-sup-lsemigroup*
subclass (**in** *residuated-blsemigroup*) *residuated-blgroupoid* ⟨*proof*⟩

class *residuated-blmonoid* = *bounded-lattice* + *residuated-sup-lmonoid*
subclass (**in** *residuated-blmonoid*) *residuated-blsemigroup* ⟨*proof*⟩

class *residuated-clgroupoid* = *complete-lattice* + *residuated-pogroupoid*
begin

lemma *resl-eq-def*: $y \leftarrow x = \bigsqcup \{z.\ z \cdot x \leq y\}$
⟨*proof*⟩

lemma *resr-eq-def*: $x \rightarrow y = \bigsqcup \{z.\ x \cdot z \leq y\}$
⟨*proof*⟩

lemma *multl-def*: $x \cdot y = \bigsqcap \{z.\ x \leq z \leftarrow y\}$
⟨*proof*⟩

lemma *multr-def*: $x \cdot y = \bigsqcap \{z.\ y \leq x \rightarrow z\}$
⟨*proof*⟩

end

class *residuated-clsemigroup* = *complete-lattice* + *residuated-sup-lsemigroup*

subclass (**in** *residuated-clsemigroup*) *residuated-clgroupoid* ⟨*proof*⟩

class *residuated-clmonoid* = *complete-lattice* + *residuated-sup-lmonoid*
subclass (**in** *residuated-clmonoid*) *residuated-clsemigroup* ⟨*proof*⟩

end

3 Residuated Boolean Algebras

theory *Residuated-Boolean-Algebras*
imports *Residuated-Lattices*
begin

unbundle *lattice-syntax*

3.1 Conjugation on Boolean Algebras

Similarly, as in the previous section, we define the conjugation for arbitrary residuated functions on boolean algebras.

context *boolean-algebra*
begin

lemma *inf-bot-iff-le*: $x \sqcap y = \bot \longleftrightarrow x \leq -y$
⟨*proof*⟩

lemma *le-iff-inf-bot*: $x \leq y \longleftrightarrow x \sqcap -y = \bot$
⟨*proof*⟩

lemma *indirect-eq*: $(\bigwedge z.\ x \leq z \longleftrightarrow y \leq z) \Longrightarrow x = y$
⟨*proof*⟩

Let B be a boolean algebra. The maps f and g on B are a pair of conjugates if and only if for all $x, y \in B$, $f(x) \sqcap y = \bot \Leftrightarrow x \sqcap g(t) = \bot$.

definition *conjugation-pair* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow$ *bool* **where**
conjugation-pair f g $\equiv \forall x\ y.\ f(x) \sqcap y = \bot \longleftrightarrow x \sqcap g(y) = \bot$

lemma *conjugation-pair-commute*: *conjugation-pair f g* $\Longrightarrow$ *conjugation-pair g f*
⟨*proof*⟩

lemma *conjugate-iff-residuated*: *conjugation-pair f g* = *residuated-pair f* $(\lambda x.\ -g(-x))$
⟨*proof*⟩

lemma *conjugate-residuated*: *conjugation-pair f g* $\Longrightarrow$ *residuated-pair f* $(\lambda x.\ -g(-x))$
⟨*proof*⟩

lemma *residuated-iff-conjugate*: *residuated-pair f g* = *conjugation-pair f* $(\lambda x.\ -g(-x))$
⟨*proof*⟩

A map f has a conjugate pair if and only if it is residuated.

lemma *conj-residuatedI1*: $\exists\, g.$ *conjugation-pair f g* $\Longrightarrow$ *residuated f*
⟨*proof*⟩

lemma *conj-residuatedI2*: $\exists\, g.$ *conjugation-pair g f* $\Longrightarrow$ *residuated f*
⟨*proof*⟩

lemma *exist-conjugateI*[*intro*]: *residuated f* $\Longrightarrow \exists\, g.$ *conjugation-pair f g*
⟨*proof*⟩

lemma *exist-conjugateI2*[*intro*]: *residuated f* $\Longrightarrow \exists\, g.$ *conjugation-pair g f*
⟨*proof*⟩

The conjugate of a residuated function f is unique.

lemma *unique-conjugate*[*intro*]: *residuated f* $\Longrightarrow \exists!g.$ *conjugation-pair f g*
⟨*proof*⟩

lemma *unique-conjugate2*[*intro*]: *residuated f* $\Longrightarrow \exists!g.$ *conjugation-pair g f*
⟨*proof*⟩

Since the conjugate of a residuated map is unique, we define a conjugate operation.

definition *conjugate* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a)$ **where**
conjugate f $\equiv$ *THE g. conjugation-pair g f*

lemma *conjugate-iff-def*: *residuated f* $\Longrightarrow f(x) \sqcap y = \bot \longleftrightarrow x \sqcap$ *conjugate f y* $= \bot$
⟨*proof*⟩

lemma *conjugateI1*: *residuated f* $\Longrightarrow f(x) \sqcap y = \bot \Longrightarrow x \sqcap$ *conjugate f y* $= \bot$
⟨*proof*⟩

lemma *conjugateI2*: *residuated f* $\Longrightarrow x \sqcap$ *conjugate f y* $= \bot \Longrightarrow f(x) \sqcap y = \bot$
⟨*proof*⟩

Few more lemmas about conjugation follow.

lemma *residuated-conj1*: *residuated f* $\Longrightarrow$ *conjugation-pair f* (*conjugate f*)
⟨*proof*⟩

lemma *residuated-conj2*: *residuated f* $\Longrightarrow$ *conjugation-pair* (*conjugate f*) *f*
⟨*proof*⟩

lemma *conj-residuated*: *residuated f* $\Longrightarrow$ *residuated* (*conjugate f*)
⟨*proof*⟩

lemma *conj-involution*: *residuated f* $\Longrightarrow$ *conjugate* (*conjugate f*) $= f$
⟨*proof*⟩

lemma *residual-conj-eq*: *residuated f* $\Longrightarrow$ *residual* (*conjugate f*) = ($\lambda x.\ -f(-x)$)
⟨*proof*⟩

lemma *residual-conj-eq-ext*: *residuated f* $\Longrightarrow$ *residual* (*conjugate f*) *x* = $-f(-x)$
⟨*proof*⟩

lemma *conj-iso*: *residuated f* $\Longrightarrow$ $x \leq y \Longrightarrow$ *conjugate f x* $\leq$ *conjugate f y*
⟨*proof*⟩

lemma *conjugate-strict*: *residuated f* $\Longrightarrow$ *conjugate f* $\bot = \bot$
⟨*proof*⟩

lemma *conjugate-sup*: *residuated f* $\Longrightarrow$ *conjugate f* $(x \sqcup y)$ = *conjugate f x* $\sqcup$ *conjugate f y*
⟨*proof*⟩

lemma *conjugate-subinf*: *residuated f* $\Longrightarrow$ *conjugate f* $(x \sqcap y) \leq$ *conjugate f x* $\sqcap$ *conjugate f y*
⟨*proof*⟩

Next we prove some lemmas from Maddux's article. Similar lemmas have been proved in AFP entry for relation algebras. They should be consolidated in the future.

lemma *maddux1*: *residuated f* $\Longrightarrow f(x \sqcap -$ *conjugate* $f(y)) \leq f(x) \sqcap -y$
⟨*proof*⟩

lemma *maddux1'*: *residuated f* $\Longrightarrow$ *conjugate* $f(x \sqcap -f(y)) \leq$ *conjugate* $f(x) \sqcap -y$
⟨*proof*⟩

lemma *maddux2*: *residuated f* $\Longrightarrow f(x) \sqcap y \leq f(x \sqcap$ *conjugate f y*$)$
⟨*proof*⟩

lemma *maddux2'*: *residuated f* $\Longrightarrow$ *conjugate* $f(x) \sqcap y \leq$ *conjugate* $f(x \sqcap f\ y)$
⟨*proof*⟩

lemma *residuated-conjugate-ineq*: *residuated f* $\Longrightarrow$ *conjugate f x* $\leq y \longleftrightarrow x \leq -f(-y)$
⟨*proof*⟩

lemma *residuated-comp-closed*: *residuated f* $\Longrightarrow$ *residuated g* $\Longrightarrow$ *residuated* (*f o g*)
⟨*proof*⟩

lemma *conjugate-comp*: *residuated f* $\Longrightarrow$ *residuated g* $\Longrightarrow$ *conjugate* (*f o g*) = *conjugate g o conjugate f*
⟨*proof*⟩

lemma *conjugate-comp-ext*: *residuated f* $\Longrightarrow$ *residuated g* $\Longrightarrow$ *conjugate* ($\lambda x.$ *f* (*g*

$x))$ $x = conjugate\ g\ (conjugate\ f\ x)$
⟨*proof*⟩

end

context *complete-boolean-algebra* **begin**

On a complete boolean algebra, it is possible to give an explicit definition of conjugation.

lemma *conjugate-eq*: *residuated* $f \Longrightarrow$ *conjugate* $f\ y = \bigsqcap \{x.\ y \leq -f(-x)\}$
⟨*proof*⟩

end

3.2 Residuated Boolean Structures

In this section, we present various residuated structures based on boolean algebras. The left and right conjugation of the multiplicative operation is defined, and a number of facts is derived.

class *residuated-boolean-algebra* = *boolean-algebra* + *residuated-pogroupoid*
begin

subclass *residuated-lgroupoid* ⟨*proof*⟩

definition *conjugate-l* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\lhd$ *60*) **where**
$x \lhd y \equiv -(-x \leftarrow y)$

definition *conjugate-r* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\rhd$ *60*) **where**
$x \rhd y \equiv -(x \rightarrow -y)$

lemma *residual-conjugate-r*: $x \rightarrow y = -(x \rhd -y)$
⟨*proof*⟩

lemma *residual-conjugate-l*: $x \leftarrow y = -(-x \lhd y)$
⟨*proof*⟩

lemma *conjugation-multl*: $x{\cdot}y \sqcap z = \bot \longleftrightarrow x \sqcap (z \lhd y) = \bot$
⟨*proof*⟩

lemma *conjugation-multr*: $x{\cdot}y \sqcap z = \bot \longleftrightarrow y \sqcap (x \rhd z) = \bot$
⟨*proof*⟩

lemma *conjugation-conj*: $(x \lhd y) \sqcap z = \bot \longleftrightarrow y \sqcap (z \rhd x) = \bot$
⟨*proof*⟩

lemma *conjugation-pair-multl* [*simp*]: *conjugation-pair* $(\lambda x.\ x{\cdot}y)\ (\lambda x.\ x \lhd y)$
⟨*proof*⟩

lemma *conjugation-pair-multr* [*simp*]: *conjugation-pair* $(\lambda x.\ y{\cdot}x)$ $(\lambda x.\ y \rhd x)$
⟨*proof*⟩

lemma *conjugation-pair-conj* [*simp*]: *conjugation-pair* $(\lambda x.\ y \lhd x)$ $(\lambda x.\ x \rhd y)$
⟨*proof*⟩

lemma *residuated-conjl1* [*simp*]: *residuated* $(\lambda x.\ x \lhd y)$
⟨*proof*⟩

lemma *residuated-conjl2* [*simp*]: *residuated* $(\lambda x.\ y \lhd x)$
⟨*proof*⟩

lemma *residuated-conjr1* [*simp*]: *residuated* $(\lambda x.\ y \rhd x)$
⟨*proof*⟩

lemma *residuated-conjr2* [*simp*]: *residuated* $(\lambda x.\ x \rhd y)$
⟨*proof*⟩

lemma *conjugate-multr* [*simp*]: *conjugate* $(\lambda x.\ y{\cdot}x) = (\lambda x.\ y \rhd x)$
⟨*proof*⟩

lemma *conjugate-conjr1* [*simp*]: *conjugate* $(\lambda x.\ y \rhd x) = (\lambda x.\ y{\cdot}x)$
⟨*proof*⟩

lemma *conjugate-multl* [*simp*]: *conjugate* $(\lambda x.\ x{\cdot}y) = (\lambda x.\ x \lhd y)$
⟨*proof*⟩

lemma *conjugate-conjl1* [*simp*]: *conjugate* $(\lambda x.\ x \lhd y) = (\lambda x.\ x{\cdot}y)$
⟨*proof*⟩

lemma *conjugate-conjl2*[*simp*]: *conjugate* $(\lambda x.\ y \lhd x) = (\lambda x.\ x \rhd y)$
⟨*proof*⟩

lemma *conjugate-conjr2*[*simp*]: *conjugate* $(\lambda x.\ x \rhd y) = (\lambda x.\ y \lhd x)$
⟨*proof*⟩

lemma *conjl1-iso*: $x \leq y \Longrightarrow x \lhd z \leq y \lhd z$
⟨*proof*⟩

lemma *conjl2-iso*: $x \leq y \Longrightarrow z \lhd x \leq z \lhd y$
⟨*proof*⟩

lemma *conjr1-iso*: $x \leq y \Longrightarrow z \rhd x \leq z \rhd y$
⟨*proof*⟩

lemma *conjr2-iso*: $x \leq y \Longrightarrow x \rhd z \leq y \rhd z$
⟨*proof*⟩

lemma *conjl1-sup*: $z \lhd (x \sqcup y) = (z \lhd x) \sqcup (z \lhd y)$

⟨*proof*⟩

lemma *conjl2-sup*: $(x \sqcup y) \lhd z = (x \lhd z) \sqcup (y \lhd z)$
⟨*proof*⟩

lemma *conjr1-sup*: $z \rhd (x \sqcup y) = (z \rhd x) \sqcup (z \rhd y)$
⟨*proof*⟩

lemma *conjr2-sup*: $(x \sqcup y) \rhd z = (x \rhd z) \sqcup (y \rhd z)$
⟨*proof*⟩

lemma *conjl1-strict*: $\bot \lhd x = \bot$
⟨*proof*⟩

lemma *conjl2-strict*: $x \lhd \bot = \bot$
⟨*proof*⟩

lemma *conjr1-strict*: $\bot \rhd x = \bot$
⟨*proof*⟩

lemma *conjr2-strict*: $x \rhd \bot = \bot$
⟨*proof*⟩

lemma *conjl1-iff*: $x \lhd y \leq z \longleftrightarrow x \leq -(-z \cdot y)$
⟨*proof*⟩

lemma *conjl2-iff*: $x \lhd y \leq z \longleftrightarrow y \leq -(-z \rhd x)$
⟨*proof*⟩

lemma *conjr1-iff*: $x \rhd y \leq z \longleftrightarrow y \leq -(x \cdot -z)$
⟨*proof*⟩

lemma *conjr2-iff*: $x \rhd y \leq z \longleftrightarrow x \leq -(y \lhd -z)$
⟨*proof*⟩

We apply Maddux's lemmas regarding conjugation of an arbitrary residuated function for each of the 6 functions.

lemma *maddux1a*: $a \cdot (x \sqcap -(a \rhd y)) \leq a \cdot x$
⟨*proof*⟩

lemma *maddux1a′*: $a \cdot (x \sqcap -(a \rhd y)) \leq -y$
⟨*proof*⟩

lemma *maddux1b*: $(x \sqcap -(y \lhd a)) \cdot a \leq x \cdot a$
⟨*proof*⟩

lemma *maddux1b′*: $(x \sqcap -(y \lhd a)) \cdot a \leq -y$
⟨*proof*⟩

lemma *maddux1c*: $a \lhd x \sqcap -(y \rhd a) \leq a \lhd x$
⟨*proof*⟩

lemma *maddux1c'*: $a \lhd x \sqcap -(y \rhd a) \leq -y$
⟨*proof*⟩

lemma *maddux1d*: $a \rhd x \sqcap -(a \cdot y) \leq a \rhd x$
⟨*proof*⟩

lemma *maddux1d'*: $a \rhd x \sqcap -(a \cdot y) \leq -y$
⟨*proof*⟩

lemma *maddux1e*: $x \sqcap -(y \cdot a) \lhd a \leq x \lhd a$
⟨*proof*⟩

lemma *maddux1e'*: $x \sqcap -(y \cdot a) \lhd a \leq -y$
⟨*proof*⟩

lemma *maddux1f*: $x \sqcap -(a \lhd y) \rhd a \leq x \rhd a$
⟨*proof*⟩

lemma *maddux1f'*: $x \sqcap -(a \lhd y) \rhd a \leq -y$
⟨*proof*⟩

lemma *maddux2a*: $a \cdot x \sqcap y \leq a \cdot (x \sqcap (a \rhd y))$
⟨*proof*⟩

lemma *maddux2b*: $x \cdot a \sqcap y \leq (x \sqcap (y \lhd a)) \cdot a$
⟨*proof*⟩

lemma *maddux2c*: $(a \lhd x) \sqcap y \leq a \lhd (x \sqcap (y \rhd a))$
⟨*proof*⟩

lemma *maddux2d*: $(a \rhd x) \sqcap y \leq a \rhd (x \sqcap (a \cdot y))$
⟨*proof*⟩

lemma *maddux2e*: $(x \lhd a) \sqcap y \leq (x \sqcap (y \cdot a)) \lhd a$
⟨*proof*⟩

lemma *maddux2f*: $(x \rhd a) \sqcap y \leq (x \sqcap (a \lhd y)) \rhd a$
⟨*proof*⟩

The multiplicative operation $\cdot$ on a residuated boolean algebra is generally not associative. We prove some equivalences related to associativity.

lemma *res-assoc-iff1*: $(\forall\, x\ y\ z.\ x \cdot (y \cdot z) = (x \cdot y) \cdot z) \longleftrightarrow (\forall\, x\ y\ z.\ x \rhd (y \rhd z) = y \cdot x \rhd z)$
⟨*proof*⟩

lemma *res-assoc-iff2*: $(\forall\, x\ y\ z.\ x \cdot (y \cdot z) = (x \cdot y) \cdot z) \longleftrightarrow (\forall\, x\ y\ z.\ x \lhd (y \cdot z) = (x$

$\lhd\ z) \lhd\ y)$
⟨*proof*⟩

lemma *res-assoc-iff3*: $(\forall\, x\ y\ z.\ x{\cdot}(y{\cdot}z) = (x{\cdot}y){\cdot}z) \longleftrightarrow (\forall\, x\ y\ z.\ (x \rhd y) \lhd z = x \rhd (y \lhd z))$
⟨*proof*⟩

end

class *unital-residuated-boolean* = *residuated-boolean-algebra* + *one* +
  assumes *mult-onel* [*simp*]: $x{\cdot}1 = x$
  and *mult-oner* [*simp*]: $1{\cdot}x = x$
begin

The following equivalences are taken from Jósson and Tsinakis.

lemma *jonsson1a*: $(\exists f.\ \forall\, x\ y.\ x \rhd y = f(x){\cdot}y) \longleftrightarrow (\forall\, x\ y.\ x \rhd y = (x \rhd 1){\cdot}y)$
⟨*proof*⟩

lemma *jonsson1b*: $(\forall\, x\ y.\ x \rhd y = (x \rhd 1){\cdot}y) \longleftrightarrow (\forall\, x\ y.\ x{\cdot}y = (x \rhd 1) \rhd y)$
⟨*proof*⟩

lemma *jonsson1c*: $(\forall\, x\ y.\ x \rhd y = (x \rhd 1){\cdot}y) \longleftrightarrow (\forall\, x\ y.\ y \lhd x = 1 \lhd (x \lhd y))$
⟨*proof*⟩

lemma *jonsson2a*: $(\exists\, g.\ \forall\, x\ y.\ x \lhd y = x{\cdot}g(y)) \longleftrightarrow (\forall\, x\ y.\ x \lhd y = x{\cdot}(1 \lhd y))$
⟨*proof*⟩

lemma *jonsson2b*: $(\forall\, x\ y.\ x \lhd y = x{\cdot}(1 \lhd y)) \longleftrightarrow (\forall\, x\ y.\ x{\cdot}y = x \lhd (1 \lhd y))$
⟨*proof*⟩

lemma *jonsson2c*: $(\forall\, x\ y.\ x \lhd y = x{\cdot}(1 \lhd y)) \longleftrightarrow (\forall\, x\ y.\ y \rhd x = (x \rhd y) \rhd 1)$
⟨*proof*⟩

lemma *jonsson3a*: $(\forall\, x.\ (x \rhd 1) \rhd 1 = x) \longleftrightarrow (\forall\, x.\ 1 \lhd (1 \lhd x) = x)$
⟨*proof*⟩

lemma *jonsson3b*: $(\forall\, x.\ (x \rhd 1) \rhd 1 = x) \Longrightarrow (x \sqcap y) \rhd 1 = (x \rhd 1) \sqcap (y \rhd 1)$
⟨*proof*⟩

lemma *jonsson3c*: $\forall\, x.\ (x \rhd 1) \rhd 1 = x \Longrightarrow x \rhd 1 = 1 \lhd x$
⟨*proof*⟩

end

class *residuated-boolean-semigroup* = *residuated-boolean-algebra* + *semigroup-mult*
begin

subclass *residuated-boolean-algebra* ⟨*proof*⟩

The following lemmas hold trivially, since they are equivalent to associativ-

ity.

lemma *res-assoc1*: $x \rhd (y \rhd z) = y \cdot x \rhd z$
⟨*proof*⟩

lemma *res-assoc2*: $x \lhd (y \cdot z) = (x \lhd z) \lhd y$
⟨*proof*⟩

lemma *res-assoc3*: $(x \rhd y) \lhd z = x \rhd (y \lhd z)$
⟨*proof*⟩

end

class *residuated-boolean-monoid* = *residuated-boolean-algebra* + *monoid-mult*
begin

subclass *unital-residuated-boolean*
⟨*proof*⟩

subclass *residuated-lmonoid* ⟨*proof*⟩

lemma *jonsson4*: $(\forall\, x\ y.\ x \lhd y = x \cdot (1 \lhd y)) \longleftrightarrow (\forall\, x\ y.\ x \rhd y = (x \rhd 1) \cdot y)$
⟨*proof*⟩

end

end

4 Involutive Residuated Structures

theory *Involutive-Residuated*
imports *Residuated-Lattices*
begin

class *uminus'* =
fixes *uminus'* :: $'a \Rightarrow 'a$ $(-'' \text{ - } [81]\ 80)$

Involutive posets is a structure where the double negation property holds for the negation operations, and a Galois connection for negations exists.

class *involutive-order* = *order* + *uminus* + *uminus'* +
assumes *gn*: $x \leq -'y \longleftrightarrow y \leq -x$
and *dn1*[*simp*]: $-'(-x) = x$
and *dn2*[*simp*]: $-(-'x) = x$

class *involutive-pogroupoid* = *order* + *times* + *involutive-order* +
assumes *ipg1*: $x \cdot y \leq z \longleftrightarrow (-z) \cdot x \leq -y$
and *ipg2*: $x \cdot y \leq z \longleftrightarrow y \cdot (-'z) \leq -'x$
begin

lemma *neg-antitone*: $x \leq y \Longrightarrow -y \leq -x$
⟨*proof*⟩

lemma *neg′-antitone*: $x \leq y \Longrightarrow -'y \leq -'x$
⟨*proof*⟩

subclass *pogroupoid*
⟨*proof*⟩

abbreviation *inv-resl* :: $'a \Rightarrow 'a \Rightarrow 'a$ **where**
$\mathit{inv\text{-}resl}\ y\ x \equiv -(x \cdot (-'y))$

abbreviation *inv-resr* :: $'a \Rightarrow 'a \Rightarrow 'a$ **where**
$\mathit{inv\text{-}resr}\ x\ y \equiv -'((-y) \cdot x)$

sublocale *residuated-pogroupoid* - - - *inv-resl inv-resr*
⟨*proof*⟩

end

class *division-order* = *order* + *residual-l-op* + *residual-r-op* +
assumes *div-galois*: $x \leq z \leftarrow y \longleftrightarrow y \leq x \rightarrow z$

class *involutive-division-order* = *division-order* + *involutive-order* +
assumes *contraposition*: $y \rightarrow -x = -'y \leftarrow x$

context *involutive-pogroupoid* **begin**

sublocale *involutive-division-order* - - *inv-resl inv-resr*
⟨*proof*⟩

lemma *inv-resr-neg* [*simp*]: $\mathit{inv\text{-}resr}\ (-x)\ (-y) = \mathit{inv\text{-}resl}\ x\ y$
⟨*proof*⟩

lemma *inv-resl-neg′* [*simp*]: $\mathit{inv\text{-}resl}\ (-'x)\ (-'y) = \mathit{inv\text{-}resr}\ x\ y$
⟨*proof*⟩

lemma *neg′-mult-resl*: $-'((-y) \cdot (-x)) = \mathit{inv\text{-}resl}\ x\ (-'y)$
⟨*proof*⟩

lemma *neg-mult-resr*: $-((-'y) \cdot (-'x)) = \mathit{inv\text{-}resr}\ (-x)\ y$
⟨*proof*⟩

lemma *resr-de-morgan1*: $-'(\mathit{inv\text{-}resr}\ (-y)\ (-x)) = -'(\mathit{inv\text{-}resl}\ y\ x)$
⟨*proof*⟩

lemma *resr-de-morgan2*: $-(\mathit{inv\text{-}resl}\ (-'x)\ (-'y)) = -(\mathit{inv\text{-}resr}\ x\ y)$
⟨*proof*⟩

end

We prove that an involutive division poset is equivalent to an involutive po-groupoid by a lemma to avoid cyclic definitions

lemma (**in** *involutive-division-order*) *inv-pogroupoid*:
class.involutive-pogroupoid ($\lambda x\ y.\ -(y \rightarrow -'x)$) *uminus uminus'* ($\leq$) ($<$)
⟨*proof*⟩

context *involutive-pogroupoid* **begin**

definition *negation-constant* :: $'a \Rightarrow$ *bool* **where**
negation-constant a $\equiv \forall\, x.\ -'x = \mathit{inv\text{-}resr}\ x\ a \wedge -x = \mathit{inv\text{-}resl}\ a\ x$

definition *division-unit* :: $'a \Rightarrow$ *bool* **where**
division-unit a $\equiv \forall\, x.\ x = \mathit{inv\text{-}resr}\ a\ x \wedge x = \mathit{inv\text{-}resl}\ x\ a$

lemma *neg-iff-div-unit*: $(\exists\, a.\ \mathit{negation\text{-}constant}\ a) \longleftrightarrow (\exists\, b.\ \mathit{division\text{-}unit}\ b)$
⟨*proof*⟩

end

end

5 Action Algebras

theory *Action-Algebra*
imports *../Residuated-Lattices/Residuated-Lattices Kleene-Algebra.Kleene-Algebra*
begin

Action algebras have been defined and discussed in Pratt's paper on *Action Logic and Pure Induction* [4]. They are expansions of Kleene algebras by operations of left and right residuation. They are interesting, first because most models of Kleene algebras, e.g. relations, traces, paths and languages, possess the residuated structure, and second because, in this setting, the Kleene star can be defined equationally.

Action algebras can be based on residuated semilattices. Many important properties of action algebras already arise at this level.

We can define an action algebra as a residuated join semilattice that is also a dioid. Following Pratt, we also add a star operation that is axiomatised as a reflexive transitive closure operation.

class *action-algebra* = *residuated-sup-lgroupoid* + *dioid-one-zero* + *star-op* +
assumes *star-rtc1*: $1 + x^\star \cdot x^\star + x \leq x^\star$
and *star-rtc2*: $1 + y \cdot y + x \leq y \Longrightarrow x^\star \leq y$
begin

lemma *plus-sup*: $(+) = (\sqcup)$

⟨*proof*⟩

We first prove a reflexivity property for residuals.

lemma *residual-r-refl*: $1 \leq x \rightarrow x$
⟨*proof*⟩

lemma *residual-l-refl*: $1 \leq x \leftarrow x$
⟨*proof*⟩

We now derive pure induction laws for residuals.

lemma *residual-l-pure-induction*: $(x \leftarrow x)^\star \leq x \leftarrow x$
⟨*proof*⟩

lemma *residual-r-pure-induction*: $(x \rightarrow x)^\star \leq x \rightarrow x$
⟨*proof*⟩

Next we show that every action algebra is a Kleene algebra. First, we derive the star unfold law and the star induction laws in action algebra. Then we prove a subclass statement.

lemma *star-unfoldl*: $1 + x \cdot x^\star \leq x^\star$
⟨*proof*⟩

lemma *star-mon* [*intro*]: $x \leq y \Longrightarrow x^\star \leq y^\star$
⟨*proof*⟩

lemma *star-subdist'*: $x^\star \leq (x + y)^\star$
⟨*proof*⟩

lemma *star-inductl*: $z + x \cdot y \leq y \Longrightarrow x^\star \cdot z \leq y$
⟨*proof*⟩

lemma *star-inductr*: $z + y \cdot x \leq y \Longrightarrow z \cdot x^\star \leq y$
⟨*proof*⟩

subclass *kleene-algebra*
⟨*proof*⟩

end

5.1 Equational Action Algebras

The induction axioms of Kleene algebras are universal Horn formulas. This is unavoidable, because due to a well known result of Redko, there is no finite equational axiomatisation for the equational theory of regular expressions.

Action algebras, in contrast, admit a finite equational axiomatization, as Pratt has shown. We now formalise this result. Consequently, the equational action algebra axioms, which imply those based on Galois connections, which

in turn imply those of Kleene algebras, are complete with respect to the equational theory of regular expressions. However, this completeness result does not account for residuation.

class *equational-action-algebra* = *residuated-sup-lgroupoid* + *dioid-one-zero* + *star-op* +
  assumes *star-ax*: $1 + x^\star \cdot x^\star + x \leq x^\star$
  and *star-subdist*: $x^\star \leq (x + y)^\star$
  and *right-pure-induction*: $(x \rightarrow x)^\star \leq x \rightarrow x$
begin

We now show that the equational axioms of action algebra satisfy those based on the Galois connections. Since we can use our correspondence between the two variants of residuated semilattice, it remains to derive the second reflexive transitive closure axiom for the star, essentially copying Pratt's proof step by step. We then prove a subclass statement.

lemma *star-rtc-2*: $1 + y \cdot y + x \leq y \Longrightarrow x^\star \leq y$
⟨*proof*⟩

subclass *action-algebra*
  ⟨*proof*⟩

end

Conversely, every action algebra satisfies the equational axioms of equational action algebras.

Because the subclass relation must be acyclic in Isabelle, we can only establish this for the corresponding locales. Again this proof is based on the residuated semilattice result.

sublocale *action-algebra* ⊆ *equational-action-algebra*
  ⟨*proof*⟩

5.2 Another Variant

Finally we show that Pratt and Kozen's star axioms generate precisely the same theory.

class *action-algebra-var* = *residuated-sup-lgroupoid* + *dioid-one-zero* + *star-op* +
  assumes *star-unfold′*: $1 + x \cdot x^\star \leq x^\star$
  and *star-inductl′*: $z + x \cdot y \leq y \Longrightarrow x^\star \cdot z \leq y$
  and *star-inductr′*: $z + y \cdot x \leq y \Longrightarrow z \cdot x^\star \leq y$
begin

subclass *kleene-algebra*
  ⟨*proof*⟩

subclass *action-algebra*
  ⟨*proof*⟩

end

sublocale *action-algebra* $\subseteq$ *action-algebra-var*
⟨*proof*⟩

end

6 Models of Action Algebras

theory *Action-Algebra-Models*
imports *Action-Algebra Kleene-Algebra.Kleene-Algebra-Models*
begin

6.1 The Powerset Action Algebra over a Monoid

Here we show that various models of Kleene algebras are also residuated; hence they form action algebras. In each case the main work is to establish the residuated lattice structure.

The interpretation proofs for some of the following models are quite similar. One could, perhaps, abstract out common reasoning.

6.2 The Powerset Action Algebra over a Monoid

instantiation *set* :: (*monoid-mult*) *residuated-sup-lgroupoid*
begin

definition *residual-r-set* **where**
$X \rightarrow Z = \bigcup \{Y.\ X \cdot Y \subseteq Z\}$

definition *residual-l-set* **where**
$Z \leftarrow Y = \bigcup \{X.\ X \cdot Y \subseteq Z\}$

instance
⟨*proof*⟩

end

instantiation *set* :: (*monoid-mult*) *action-algebra*
begin

instance
⟨*proof*⟩

end

6.3 Language Action Algebras

definition *limp-lan* :: *'a lan* ⇒ *'a lan* ⇒ *'a lan* **where**
limp-lan Z Y = {*x*. ∀ *y* ∈ *Y*. *x* @ *y* ∈ *Z*}

definition *rimp-lan* :: *'a lan* ⇒ *'a lan* ⇒ *'a lan* **where**
rimp-lan X Z = {*y*. ∀ *x* ∈ *X*. *x* @ *y* ∈ *Z*}

interpretation *lan-residuated-join-semilattice*: *residuated-sup-lgroupoid* (+) (⊆) (⊂) (·) *limp-lan rimp-lan*
⟨*proof*⟩

interpretation *lan-action-algebra*: *action-algebra* (+) (·) *1 0* (⊆) (⊂) (+) *limp-lan rimp-lan star*
⟨*proof*⟩

6.4 Relation Action Algebras

definition *limp-rel* :: *'a rel* ⇒ *'a rel* ⇒ *'a rel* **where**
limp-rel T S = {(*y*,*x*) | *y x*. ∀ *z*. (*x*,*z*) ∈ *S* ⟶ (*y*,*z*) ∈ *T*}

definition *rimp-rel* :: *'a rel* ⇒ *'a rel* ⇒ *'a rel* **where**
rimp-rel R T = {(*y*,*z*) | *y z*. ∀ *x*. (*x*,*y*) ∈ *R* ⟶ (*x*,*z*) ∈ *T*}

interpretation *rel-residuated-join-semilattice*: *residuated-sup-lgroupoid* (∪) (⊆) (⊂) (*O*) *limp-rel rimp-rel*
⟨*proof*⟩

interpretation *rel-action-algebra*: *action-algebra* (∪) (*O*) *Id* {} (⊆) (⊂) (∪) *limp-rel rimp-rel rtrancl*
⟨*proof*⟩

6.5 Trace Action Algebras

definition *limp-trace* :: (*'p*, *'a*) *trace set* ⇒ (*'p*, *'a*) *trace set* ⇒ (*'p*, *'a*) *trace set* **where**
limp-trace Z Y = ⋃ {*X*. *t-prod X Y* ⊆ *Z*}

definition *rimp-trace* :: (*'p*, *'a*) *trace set* ⇒ (*'p*, *'a*) *trace set* ⇒ (*'p*, *'a*) *trace set* **where**
rimp-trace X Z = ⋃ {*Y*. *t-prod X Y* ⊆ *Z*}

interpretation *trace-residuated-join-semilattice*: *residuated-sup-lgroupoid* (∪) (⊆) (⊂) *t-prod limp-trace rimp-trace*
⟨*proof*⟩

interpretation *trace-action-algebra*: *action-algebra* (∪) *t-prod t-one t-zero* (⊆) (⊂) (∪) *limp-trace rimp-trace t-star*
⟨*proof*⟩

6.6 Path Action Algebras

We start with paths that include the empty path.

definition *limp-path* :: *'a path set* ⇒ *'a path set* ⇒ *'a path set* **where**
limp-path Z Y = ⋃ {*X*. *p-prod X Y* ⊆ *Z*}

definition *rimp-path* :: *'a path set* ⇒ *'a path set* ⇒ *'a path set* **where**
rimp-path X Z = ⋃ {*Y*. *p-prod X Y* ⊆ *Z*}

interpretation *path-residuated-join-semilattice*: *residuated-sup-lgroupoid* (∪) (⊆) (⊂) *p-prod limp-path rimp-path*
⟨*proof*⟩

interpretation *path-action-algebra*: *action-algebra* (∪) *p-prod p-one* {} (⊆) (⊂) (∪) *limp-path rimp-path p-star*
⟨*proof*⟩

We now consider a notion of paths that does not include the empty path.

definition *limp-ppath* :: *'a ppath set* ⇒ *'a ppath set* ⇒ *'a ppath set* **where**
limp-ppath Z Y = ⋃ {*X*. *pp-prod X Y* ⊆ *Z*}

definition *rimp-ppath* :: *'a ppath set* ⇒ *'a ppath set* ⇒ *'a ppath set* **where**
rimp-ppath X Z = ⋃ {*Y*. *pp-prod X Y* ⊆ *Z*}

interpretation *ppath-residuated-join-semilattice*: *residuated-sup-lgroupoid* (∪) (⊆) (⊂) *pp-prod limp-ppath rimp-ppath*
⟨*proof*⟩

interpretation *ppath-action-algebra*: *action-algebra* (∪) *pp-prod pp-one* {} (⊆) (⊂) (∪) *limp-ppath rimp-ppath pp-star*
⟨*proof*⟩

6.7 The Min-Plus Action Algebra

instantiation *pnat* :: *minus*
begin

fun *minus-pnat* **where**
(*pnat x*) − (*pnat y*) = *pnat* (*x* − *y*)
| *x* − *PInfty* = *1*
| *PInfty* − (*pnat x*) = *0*

instance ⟨*proof*⟩

end

instantiation *pnat* :: *semilattice-sup*
begin
definition *sup-pnat*: *sup-pnat x y* ≡ *pnat-min x y*

instance
⟨*proof*⟩

end

instantiation *pnat* :: *residuated-sup-lgroupoid*
begin

definition *residual-r-pnat* **where**
$(x::pnat) \rightarrow y \equiv y - x$

definition *residual-l-pnat* **where**
$(y::pnat) \leftarrow x \equiv y - x$

instance
⟨*proof*⟩

end

instantiation *pnat* :: *action-algebra*
begin

The Kleene star for type *pnat* has already been defined in theory *Kleene-Algebra.Kleene-Algebra-Model*

instance
⟨*proof*⟩

end

end

7 Residuated Relation Algebras

theory *Residuated-Relation-Algebra*
imports *Residuated-Boolean-Algebras Relation-Algebra.Relation-Algebra*
begin

context *boolean-algebra* **begin**

The notation used in the relation algebra AFP entry differs a little from ours.

notation
times (**infixl** · *70*)
and *plus* (**infixl** + *65*)
and *Groups.zero-class.zero* (*0*)
and *Groups.one-class.one* (*1*)

no-notation

inf (**infixl** · *70*)
and *sup* (**infixl** + *65*)
and *bot* (*0*)
and *top* (*1*)

end

We prove that a unital residuated boolean algebra enriched with two simple equalities form a non-associative relation algebra, that is, a relation algebra where the associativity law does not hold.

class *nra* = *unital-residuated-boolean* +
assumes *conv1*: $x \rhd y = (x \rhd 1)\cdot y$
and *conv2*: $x \lhd y = x\cdot(1 \lhd y)$
begin

When the converse operation is set to be $\lambda x.\ x \rhd 1$, a unital residuated boolean algebra forms a non associative relation algebra.

lemma *conv-invol*: $x \rhd 1 \rhd 1 = x$
⟨*proof*⟩

lemma *conv-add*: $x \sqcup y \rhd 1 = (x \rhd 1) \sqcup (y \rhd 1)$
⟨*proof*⟩

lemma *conv-contrav*: $x\cdot y \rhd 1 = (y \rhd 1)\cdot(x \rhd 1)$
⟨*proof*⟩

lemma *conv-res*: $(x \rhd 1)\cdot - (x\cdot y) \leq - y$
⟨*proof*⟩

Similarly, for $x^\smile = 1 \lhd x$, since $x \rhd 1 = 1 \lhd x$ when $x \rhd 1 \rhd 1 = x$ holds.

lemma *conv-invol′*: $1 \lhd (1 \lhd x) = x$
⟨*proof*⟩

lemma *conv-add′*: $1 \lhd (x \sqcup y) = (1 \lhd x) \sqcup (1 \lhd y)$
⟨*proof*⟩

lemma *conv-contrav′*: $1 \lhd x\cdot y = (1 \lhd y)\cdot(1 \lhd x)$
⟨*proof*⟩

lemma *conv-res′*: $(1 \lhd x)\cdot - (x\cdot y) \leq - y$
⟨*proof*⟩

end

Since the previous axioms are equivalent when multiplication is associative in a residuated boolean monoid, one of them are sufficient to derive a relation algebra.

class *residuated-ra* = *residuated-boolean-monoid* +

assumes *conv*: $x \rhd y = (x \rhd 1) \cdot y$
begin

subclass *nra*
⟨*proof*⟩

sublocale *relation-algebra* **where**
composition $= (\cdot)$ **and** *unit* $= 1$ **and**
converse $= \lambda x.\ x \rhd 1$
⟨*proof*⟩

end

end

References

[1] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics: An Algebraic Glimpse at Substructural Logics*, volume 151. Elsevier, 2007.

[2] B. Jónsson and C. Tsinakis. Relation algebras as residuated boolean algebras. *Algebra Universalis*, 30(4):469–478, 1993.

[3] R. D. Maddux. Relation-algebraic semantics. *Theoretical Computer Sciences*, 160(1&2):1–85, 1996.

[4] V. R. Pratt. Action logic and pure induction. In J. van Eijck, editor, *Logics in AI, European Workshop, JELIA '90*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120. Springer, 1991.

[5] M. Ward and R. P. Dilworth. Residuated lattices. *Transactions of the American Mathematical Society*, 45(3):335–354, 1939.