

Residuated Lattices

Victor B. F. Gomes

Georg Struth

Department of Computer Science, University of Sheffield

March 17, 2025

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	2
2	Residuated Lattices	2
2.1	Residuated Functions on a Partial Order	2
2.2	Residuated Structures	5
3	Residuated Boolean Algebras	13
3.1	Conjugation on Boolean Algebras	13
3.2	Residuated Boolean Structures	16
4	Involutive Residuated Structures	21
5	Action Algebras	23
5.1	Equational Action Algebras	24
5.2	Another Variant	25
6	Models of Action Algebras	26
6.1	The Powerset Action Algebra over a Monoid	26
6.2	The Powerset Action Algebra over a Monoid	26
6.3	Language Action Algebras	27
6.4	Relation Action Algebras	27

6.5	Trace Action Algebras	27
6.6	Path Action Algebras	28
6.7	The Min-Plus Action Algebra	28
7	Residuated Relation Algebras	29

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 within 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 <..> 70)

  unbundle lattice-syntax
```

2 Residuated Lattices

2.1 Residuated Functions on a Partial Order

We follow Galatos and *al.* to define residuated functions 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 $f : P \rightarrow P$ is residuated if there exists a map $g : P \rightarrow P$ 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* :: $('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow \text{bool}$ **where**
 $\text{residuated-pair } f g \equiv \forall x y. f(x) \leq y \longleftrightarrow x \leq g(y)$

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 $\forall x. x \leq (g \circ f) x$
and $\forall x. (f \circ g) x \leq x$
shows *residuated-pair* $f g$
 $\langle \text{proof} \rangle$

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

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

lemma *residual-unique*: $\text{residuated } f \implies \exists! g. \text{residuated-pair } f g$
 $\langle \text{proof} \rangle$

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**
 $\text{residual } f \equiv \text{THE } g. \text{residuated-pair } f g$

lemma *residual-galois*: $\text{residuated } f \implies f(x) \leq y \longleftrightarrow x \leq \text{residual } f y$
 $\langle \text{proof} \rangle$

lemma *residual-galoisI1*: $\text{residuated } f \implies f(x) \leq y \implies x \leq \text{residual } f y$
 $\langle \text{proof} \rangle$

lemma *residual-galoisI2*: $\text{residuated } f \implies x \leq \text{residual } f y \implies f(x) \leq y$
 $\langle \text{proof} \rangle$

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

lemma *res-c1*: $\text{residuated } f \implies x \leq \text{residual } f (f x)$
 $\langle \text{proof} \rangle$

lemma *res-c2*: $\text{residuated } f \implies x \leq y \implies \text{residual } f (f x) \leq \text{residual } f (f y)$
 $\langle \text{proof} \rangle$

lemma *res-c3*: $\text{residuated } f \implies \text{residual } f (f (\text{residual } f (f x))) = \text{residual } f (f x)$
 $\langle \text{proof} \rangle$

lemma *res-closure*: $\text{residuated } f \implies \text{closure}(\text{residual } f \circ f)$
 $\langle \text{proof} \rangle$

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* :: $('a \Rightarrow 'a) \Rightarrow \text{bool}$ **where**
 $\text{interior } f \equiv (\forall x. f x \leq x) \wedge (\forall x y. x \leq y \rightarrow f x \leq f y) \wedge (\forall x. f(f x) = f x)$

lemma *res-i1*: $\text{residuated } f \implies f(\text{residual } f x) \leq x$
 $\langle \text{proof} \rangle$

lemma *res-i2*: $\text{residuated } f \implies x \leq y \implies f(\text{residual } f x) \leq f(\text{residual } f y)$
 $\langle \text{proof} \rangle$

lemma *res-i3*: $\text{residuated } f \implies f(\text{residual } f(f(\text{residual } f x))) = f(\text{residual } f x)$
 $\langle \text{proof} \rangle$

lemma *res-interior*: $\text{residuated } f \implies \text{interior}(f \circ \text{residual } f)$
 $\langle \text{proof} \rangle$

Next we show a few basic lemmas about residuated maps.

lemma *res-iso*: $\text{residuated } f \implies x \leq y \implies f x \leq f y$
 $\langle \text{proof} \rangle$

lemma *res-residual-iso*: $\text{residuated } f \implies x \leq y \implies \text{residual } f x \leq \text{residual } f y$
 $\langle \text{proof} \rangle$

lemma *res-comp1 [simp]*: $\text{residuated } f \implies f \circ \text{residual } f \circ f = f$
 $\langle \text{proof} \rangle$

lemma *res-comp2 [simp]*: $\text{residuated } f \implies \text{residual } f \circ f \circ \text{residual } f = \text{residual } f$
 $\langle \text{proof} \rangle$

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*: $\text{residuated } f \implies f(x \sqcup y) = f x \sqcup f y$
 $\langle \text{proof} \rangle$

lemma (in semilattice-inf) *residuated-inf*: $\text{residuated } f \implies \text{residual } f(x \sqcap y) = \text{residual } f x \sqcap \text{residual } f y$
 $\langle \text{proof} \rangle$

context *bounded-lattice* **begin**

lemma *residuated-strict*: *residuated f* $\implies f \perp = \perp$
 $\langle proof \rangle$

lemma *res-top*: *residuated f* $\implies residual f \top = \top$
 $\langle proof \rangle$

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 function f if and only if g preserves arbitrary meets.

lemma *residual-eq1*: *residuated f* $\implies residual f y = \bigsqcup \{x. f x \leq y\}$
 $\langle proof \rangle$

lemma *residual-eq2*: *residuated f* $\implies f x = \bigsqcap \{y. x \leq residual f y\}$
 $\langle proof \rangle$

lemma *residuated-Sup*: *residuated f* $\implies f(\bigsqcup X) = \bigsqcup \{f x \mid x. x \in X\}$
 $\langle proof \rangle$

lemma *residuated-Inf*: *residuated f* $\implies residual f(\bigsqcap X) = \bigsqcap \{residual f x \mid x. x \in X\}$
 $\langle proof \rangle$

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

lemma *Sup-residuatedI*: $\forall X. f(\bigsqcup X) = \bigsqcup \{f x \mid x. x \in X\} \implies residuated f$
 $\langle proof \rangle$

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

lemma *Inf-residuatedI*: $\forall X. \bigsqcap \{g x \mid x. x \in X\} = g(\bigsqcap X) \implies \exists f. residuated\text{-pair}$
 $f g$
 $\langle proof \rangle$

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:  $x \leq y \implies x \cdot z \leq y \cdot z$ 
and mult-isol:  $x \leq y \implies z \cdot x \leq z \cdot y$ 

```

A residuated partial ordered groupoid is simply a partial order and a multiplicative groupoid with two extra operators satisfying the residuation laws. It is straightforward 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  $\leftrightarrow$  60)

class residual-l-op =
fixes residual-l :: ' $a \Rightarrow 'a \Rightarrow 'a$  (infixl  $\leftrightarrow$  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]:  $x \cdot y \leq z \implies x \leq z \leftarrow y$ 
<proof>

lemma resrI [intro]:  $x \cdot y \leq z \implies y \leq x \rightarrow z$ 
<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*: $x \leq x \cdot y \leftarrow y$
 $\langle proof \rangle$

lemma *res-lc2*: $x \leq y \implies x \cdot z \leftarrow z \leq y \cdot z \leftarrow z$
 $\langle proof \rangle$

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

lemma *res-rc1*: $x \leq y \rightarrow y \cdot x$
 $\langle proof \rangle$

lemma *res-rc2*: $x \leq y \implies z \rightarrow z \cdot x \leq z \rightarrow z \cdot y$
 $\langle proof \rangle$

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

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

lemma *res-li2*: $x \leq y \implies (x \leftarrow z) \cdot z \leq (y \leftarrow z) \cdot z$
 $\langle proof \rangle$

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

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

lemma *res-ri2*: $x \leq y \implies z \cdot (z \rightarrow x) \leq z \cdot (z \rightarrow y)$
 $\langle proof \rangle$

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

subclass *pogroupoid*
 $\langle proof \rangle$

lemma *resl-iso*: $x \leq y \implies x \leftarrow z \leq y \leftarrow z$
 $\langle proof \rangle$

lemma *resr-iso*: $x \leq y \implies z \rightarrow x \leq z \rightarrow y$
 $\langle proof \rangle$

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

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

$\langle proof \rangle$

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

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

lemma *resl-antitoner*: $x \leq y \longrightarrow z \leftarrow y \leq z \leftarrow x$
 $\langle proof \rangle$

lemma *resr-antitonel*: $x \leq y \longrightarrow y \rightarrow z \leq x \rightarrow z$
 $\langle proof \rangle$

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

lemma *jipsen1l*: $x \leq y \leftarrow (x \rightarrow y)$
 $\langle proof \rangle$

lemma *jipsen1r*: $x \leq (y \leftarrow x) \rightarrow y$
 $\langle proof \rangle$

lemma *jipsen2l*: $(y \leftarrow (x \rightarrow y)) \rightarrow y = x \rightarrow y$
 $\langle proof \rangle$

lemma *jipsen2r*: $y \leftarrow ((y \leftarrow x) \rightarrow y) = y \leftarrow x$
 $\langle proof \rangle$

end

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

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

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

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

lemma *resr1*: $(x \rightarrow y) \cdot z \leq (x \rightarrow y \cdot z)$
 $\langle proof \rangle$

lemma *resr2*: $x \rightarrow y \leq z \cdot x \rightarrow z \cdot y$
 $\langle proof \rangle$

lemma *resr3*: $x \cdot y \rightarrow z = y \rightarrow (x \rightarrow z)$
 $\langle proof \rangle$

```

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 jipson3l:  $1 \leq x \leftarrow x$   

  ⟨proof⟩

lemma jipson3r:  $1 \leq x \rightarrow x$   

  ⟨proof⟩

lemma jipson4l [simp]:  $(x \leftarrow x) \cdot x = x$   

  ⟨proof⟩

lemma jipson4r [simp]:  $x \cdot (x \rightarrow x) = x$   

  ⟨proof⟩

lemma jipson5l [simp]:  $(x \leftarrow x) \cdot (x \leftarrow x) = x \leftarrow x$   

  ⟨proof⟩

```

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

lemma *jipsen6l*: $y \leftarrow x \leq (y \leftarrow z) \leftarrow (x \leftarrow z)$
 $\langle proof \rangle$

lemma *jipsen6r*: $x \rightarrow y \leq (z \rightarrow x) \rightarrow (z \rightarrow y)$
 $\langle proof \rangle$

lemma *jipsen7l*: $y \leftarrow x \leq (z \leftarrow y) \rightarrow (z \leftarrow x)$
 $\langle proof \rangle$

lemma *jipsen7r*: $x \rightarrow y \leq (x \rightarrow z) \leftarrow (y \rightarrow z)$
 $\langle proof \rangle$

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*: $x \cdot (y \sqcup z) = x \cdot y \sqcup x \cdot z$
 $\langle proof \rangle$

lemma *distr*: $(x \sqcup y) \cdot z = x \cdot z \sqcup y \cdot z$
 $\langle proof \rangle$

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

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

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

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

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

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

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 semilattice 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]:  $x \cdot \perp = \perp$ 
  ⟨proof⟩

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

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

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

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

lemma resr-impl [simp]:  $\perp \rightarrow x = \top$ 
  ⟨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:  $x \cdot y = \bigcap \{z. x \leq z \leftarrow y\}$ 
  ⟨proof⟩

lemma multr-def:  $x \cdot y = \bigcap \{z. y \leq x \rightarrow z\}$ 
  ⟨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: $x \sqcap y = \perp \longleftrightarrow x \leq -y$
⟨proof⟩

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

lemma indirect-eq: $(\bigwedge z. x \leq z \longleftrightarrow y \leq z) \implies x = y$
⟨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 $x, y \in B$, $f(x) \sqcap y = \perp \Leftrightarrow x \sqcap g(y) = \perp$.

definition conjugation-pair :: ('a ⇒ 'a) ⇒ ('a ⇒ 'a) ⇒ bool **where**
conjugation-pair f g ≡ ∀ x y. f(x) ∙ y = ⊥ ↔ x ∙ g(y) = ⊥

lemma conjugation-pair-commute: conjugation-pair f g ⇒ conjugation-pair g f
⟨proof⟩

lemma conjugate-iff-residuated: conjugation-pair f g = residuated-pair f (λx. -g(-x))
⟨proof⟩

lemma conjugate-residuated: conjugation-pair f g ⇒ residuated-pair f (λx. -g(-x))
⟨proof⟩

lemma residuated-iff-conjugate: residuated-pair f g = conjugation-pair f (λx. -g(-x))
⟨proof⟩

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

lemma *conj-residuatedI1*: $\exists g. \text{conjugation-pair } f g \implies \text{residuated } f$
 $\langle \text{proof} \rangle$

lemma *conj-residuatedI2*: $\exists g. \text{conjugation-pair } g f \implies \text{residuated } f$
 $\langle \text{proof} \rangle$

lemma *exist-conjugateI[intro]*: $\text{residuated } f \implies \exists g. \text{conjugation-pair } f g$
 $\langle \text{proof} \rangle$

lemma *exist-conjugateI2[intro]*: $\text{residuated } f \implies \exists g. \text{conjugation-pair } g f$
 $\langle \text{proof} \rangle$

The conjugate of a residuated function f is unique.

lemma *unique-conjugate[intro]*: $\text{residuated } f \implies \exists!g. \text{conjugation-pair } f g$
 $\langle \text{proof} \rangle$

lemma *unique-conjugate2[intro]*: $\text{residuated } f \implies \exists!g. \text{conjugation-pair } g f$
 $\langle \text{proof} \rangle$

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

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

lemma *conjugate-iff-def*: $\text{residuated } f \implies f(x) \sqcap y = \perp \longleftrightarrow x \sqcap \text{conjugate } f y = \perp$
 $\langle \text{proof} \rangle$

lemma *conjugateI1*: $\text{residuated } f \implies f(x) \sqcap y = \perp \implies x \sqcap \text{conjugate } f y = \perp$
 $\langle \text{proof} \rangle$

lemma *conjugateI2*: $\text{residuated } f \implies x \sqcap \text{conjugate } f y = \perp \implies f(x) \sqcap y = \perp$
 $\langle \text{proof} \rangle$

Few more lemmas about conjugation follow.

lemma *residuated-conj1*: $\text{residuated } f \implies \text{conjugation-pair } f (\text{conjugate } f)$
 $\langle \text{proof} \rangle$

lemma *residuated-conj2*: $\text{residuated } f \implies \text{conjugation-pair } (\text{conjugate } f) f$
 $\langle \text{proof} \rangle$

lemma *conj-residuated*: $\text{residuated } f \implies \text{residuated } (\text{conjugate } f)$
 $\langle \text{proof} \rangle$

lemma *conj-involution*: $\text{residuated } f \implies \text{conjugate } (\text{conjugate } f) = f$
 $\langle \text{proof} \rangle$

lemma *residual-conj-eq*: *residuated f* \implies *residual (conjugate f) = ($\lambda x. -f(-x)$)*
 $\langle proof \rangle$

lemma *residual-conj-eq-ext*: *residuated f* \implies *residual (conjugate f) x = -f(-x)*
 $\langle proof \rangle$

lemma *conj-iso*: *residuated f* \implies *x ≤ y* \implies *conjugate f x ≤ conjugate f y*
 $\langle proof \rangle$

lemma *conjugate-strict*: *residuated f* \implies *conjugate f ⊥ = ⊥*
 $\langle proof \rangle$

lemma *conjugate-sup*: *residuated f* \implies *conjugate f (x ∪ y) = conjugate f x ∪ conjugate f y*
 $\langle proof \rangle$

lemma *conjugate-subinf*: *residuated f* \implies *conjugate f (x ∩ y) ≤ conjugate f x ∩ conjugate f y*
 $\langle proof \rangle$

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* \implies *f(x ∩ -conjugate f(y)) ≤ f(x) ∩ -y*
 $\langle proof \rangle$

lemma *maddux1'*: *residuated f* \implies *conjugate f(x ∩ -f(y)) ≤ conjugate f(x) ∩ -y*
 $\langle proof \rangle$

lemma *maddux2*: *residuated f* \implies *f(x) ∩ y ≤ f(x ∩ conjugate f y)*
 $\langle proof \rangle$

lemma *maddux2'*: *residuated f* \implies *conjugate f(x) ∩ y ≤ conjugate f(x ∩ f y)*
 $\langle proof \rangle$

lemma *residuated-conjugate-ineq*: *residuated f* \implies *conjugate f x ≤ y ↔ x ≤ -f(-y)*
 $\langle proof \rangle$

lemma *residuated-comp-closed*: *residuated f* \implies *residuated g* \implies *residuated (f o g)*
 $\langle proof \rangle$

lemma *conjugate-comp*: *residuated f* \implies *residuated g* \implies *conjugate (f o g) = conjugate g o conjugate f*
 $\langle proof \rangle$

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

```

 $x))$   $x = \text{conjugate } g (\text{conjugate } f x)$   

 $\langle \text{proof} \rangle$   

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  $\implies$  conjugate f y =  $\bigcap \{x. y \leq -f(-x)\}$   

 $\langle \text{proof} \rangle$   

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  $\langle \text{proof} \rangle$   

definition conjugate-l ::  $'a \Rightarrow 'a \Rightarrow 'a$  (infixl  $\triangleleft$  60) where  

 $x \triangleleft y \equiv -(-x \leftarrow y)$   

definition conjugate-r ::  $'a \Rightarrow 'a \Rightarrow 'a$  (infixl  $\triangleright$  60) where  

 $x \triangleright y \equiv -(x \rightarrow -y)$   

lemma residual-conjugate-r:  $x \rightarrow y = -(x \triangleright -y)$   

 $\langle \text{proof} \rangle$   

lemma residual-conjugate-l:  $x \leftarrow y = -(-x \triangleleft y)$   

 $\langle \text{proof} \rangle$   

lemma conjugation-multl:  $x \cdot y \sqcap z = \perp \longleftrightarrow x \sqcap (z \triangleleft y) = \perp$   

 $\langle \text{proof} \rangle$   

lemma conjugation-multr:  $x \cdot y \sqcap z = \perp \longleftrightarrow y \sqcap (x \triangleright z) = \perp$   

 $\langle \text{proof} \rangle$   

lemma conjugation-conj:  $(x \triangleleft y) \sqcap z = \perp \longleftrightarrow y \sqcap (z \triangleright x) = \perp$   

 $\langle \text{proof} \rangle$   

lemma conjugation-pair-multl [simp]: conjugation-pair  $(\lambda x. x \cdot y)$   $(\lambda x. x \triangleleft y)$   

 $\langle \text{proof} \rangle$ 

```

lemma *conjugation-pair-multr* [*simp*]: *conjugation-pair* $(\lambda x. y \cdot x) (\lambda x. y \triangleright x)$
 $\langle proof \rangle$

lemma *conjugation-pair-conj* [*simp*]: *conjugation-pair* $(\lambda x. y \triangleleft x) (\lambda x. x \triangleright y)$
 $\langle proof \rangle$

lemma *residuated-conjl1* [*simp*]: *residuated* $(\lambda x. x \triangleleft y)$
 $\langle proof \rangle$

lemma *residuated-conjl2* [*simp*]: *residuated* $(\lambda x. y \triangleleft x)$
 $\langle proof \rangle$

lemma *residuated-conjr1* [*simp*]: *residuated* $(\lambda x. y \triangleright x)$
 $\langle proof \rangle$

lemma *residuated-conjr2* [*simp*]: *residuated* $(\lambda x. x \triangleright y)$
 $\langle proof \rangle$

lemma *conjugate-multr* [*simp*]: *conjugate* $(\lambda x. y \cdot x) = (\lambda x. y \triangleright x)$
 $\langle proof \rangle$

lemma *conjugate-conjr1* [*simp*]: *conjugate* $(\lambda x. y \triangleright x) = (\lambda x. y \cdot x)$
 $\langle proof \rangle$

lemma *conjugate-multl* [*simp*]: *conjugate* $(\lambda x. x \cdot y) = (\lambda x. x \triangleleft y)$
 $\langle proof \rangle$

lemma *conjugate-conjl1* [*simp*]: *conjugate* $(\lambda x. x \triangleleft y) = (\lambda x. x \cdot y)$
 $\langle proof \rangle$

lemma *conjugate-conjl2* [*simp*]: *conjugate* $(\lambda x. y \triangleleft x) = (\lambda x. x \triangleright y)$
 $\langle proof \rangle$

lemma *conjugate-conjr2* [*simp*]: *conjugate* $(\lambda x. x \triangleright y) = (\lambda x. y \triangleleft x)$
 $\langle proof \rangle$

lemma *conj1-iso*: $x \leq y \implies x \triangleleft z \leq y \triangleleft z$
 $\langle proof \rangle$

lemma *conj2-iso*: $x \leq y \implies z \triangleleft x \leq z \triangleleft y$
 $\langle proof \rangle$

lemma *conj1-iso*: $x \leq y \implies z \triangleright x \leq z \triangleright y$
 $\langle proof \rangle$

lemma *conj2-iso*: $x \leq y \implies x \triangleright z \leq y \triangleright z$
 $\langle proof \rangle$

lemma *conj1-sup*: $z \triangleleft (x \sqcup y) = (z \triangleleft x) \sqcup (z \triangleleft y)$

$\langle proof \rangle$

lemma *conj1-sup*: $(x \sqcup y) \triangleleft z = (x \triangleleft z) \sqcup (y \triangleleft z)$
 $\langle proof \rangle$

lemma *conj1-sup*: $z \triangleright (x \sqcup y) = (z \triangleright x) \sqcup (z \triangleright y)$
 $\langle proof \rangle$

lemma *conj1-sup*: $(x \sqcup y) \triangleright z = (x \triangleright z) \sqcup (y \triangleright z)$
 $\langle proof \rangle$

lemma *conj1-strict*: $\perp \triangleleft x = \perp$
 $\langle proof \rangle$

lemma *conj1-strict*: $x \triangleleft \perp = \perp$
 $\langle proof \rangle$

lemma *conj1-strict*: $\perp \triangleright x = \perp$
 $\langle proof \rangle$

lemma *conj1-strict*: $x \triangleright \perp = \perp$
 $\langle proof \rangle$

lemma *conj1-iff*: $x \triangleleft y \leq z \longleftrightarrow x \leq -(z \cdot y)$
 $\langle proof \rangle$

lemma *conj1-iff*: $x \triangleleft y \leq z \longleftrightarrow y \leq -(z \triangleright x)$
 $\langle proof \rangle$

lemma *conj1-iff*: $x \triangleright y \leq z \longleftrightarrow y \leq -(x \cdot z)$
 $\langle proof \rangle$

lemma *conj1-iff*: $x \triangleright y \leq z \longleftrightarrow x \leq -(y \triangleleft z)$
 $\langle proof \rangle$

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

lemma *maddux1a*: $a \cdot (x \sqcap -(a \triangleright y)) \leq a \cdot x$
 $\langle proof \rangle$

lemma *maddux1a'*: $a \cdot (x \sqcap -(a \triangleright y)) \leq -y$
 $\langle proof \rangle$

lemma *maddux1b*: $(x \sqcap -(y \triangleleft a)) \cdot a \leq x \cdot a$
 $\langle proof \rangle$

lemma *maddux1b'*: $(x \sqcap -(y \triangleleft a)) \cdot a \leq -y$
 $\langle proof \rangle$

lemma *maddux1c*: $a \triangleleft x \sqcap -(y \triangleright a) \leq a \triangleleft x$
(proof)

lemma *maddux1c'*: $a \triangleleft x \sqcap -(y \triangleright a) \leq -y$
(proof)

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

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

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

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

lemma *maddux1f*: $x \sqcap -(a \triangleleft y) \triangleright a \leq x \triangleright a$
(proof)

lemma *maddux1f'*: $x \sqcap -(a \triangleleft y) \triangleright a \leq -y$
(proof)

lemma *maddux2a*: $a \cdot x \sqcap y \leq a \cdot (x \sqcap (a \triangleright y))$
(proof)

lemma *maddux2b*: $x \cdot a \sqcap y \leq (x \sqcap (y \triangleleft a)) \cdot a$
(proof)

lemma *maddux2c*: $(a \triangleleft x) \sqcap y \leq a \triangleleft (x \sqcap (y \triangleright a))$
(proof)

lemma *maddux2d*: $(a \triangleright x) \sqcap y \leq a \triangleright (x \sqcap (a \cdot y))$
(proof)

lemma *maddux2e*: $(x \triangleleft a) \sqcap y \leq (x \sqcap (y \cdot a)) \triangleleft a$
(proof)

lemma *maddux2f*: $(x \triangleright a) \sqcap y \leq (x \sqcap (a \triangleleft y)) \triangleright a$
(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*: $(\forall x y z. x \cdot (y \cdot z) = (x \cdot y) \cdot z) \longleftrightarrow (\forall x y z. x \triangleright (y \triangleright z) = y \cdot x \triangleright z)$
(proof)

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

```

 $\triangleleft z) \triangleleft y)$ 
 $\langle proof \rangle$ 

lemma res-assoc-iff3:  $(\forall x y z. x \cdot (y \cdot z) = (x \cdot y) \cdot z) \longleftrightarrow (\forall x y z. (x \triangleright y) \triangleleft z = x$ 
 $\triangleright (y \triangleleft z))$ 
 $\langle proof \rangle$ 

```

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:  $(\exists f. \forall x y. x \triangleright y = f(x) \cdot y) \longleftrightarrow (\forall x y. x \triangleright y = (x \triangleright 1) \cdot y)$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson1b:  $(\forall x y. x \triangleright y = (x \triangleright 1) \cdot y) \longleftrightarrow (\forall x y. x \cdot y = (x \triangleright 1) \triangleright y)$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson1c:  $(\forall x y. x \triangleright y = (x \triangleright 1) \cdot y) \longleftrightarrow (\forall x y. y \triangleleft x = 1 \triangleleft (x \triangleleft y))$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson2a:  $(\exists g. \forall x y. x \triangleleft y = x \cdot g(y)) \longleftrightarrow (\forall x y. x \triangleleft y = x \cdot (1 \triangleleft y))$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson2b:  $(\forall x y. x \triangleleft y = x \cdot (1 \triangleleft y)) \longleftrightarrow (\forall x y. x \cdot y = x \triangleleft (1 \triangleleft y))$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson2c:  $(\forall x y. x \triangleleft y = x \cdot (1 \triangleleft y)) \longleftrightarrow (\forall x y. y \triangleright x = (x \triangleright y) \triangleright 1)$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson3a:  $(\forall x. (x \triangleright 1) \triangleright 1 = x) \longleftrightarrow (\forall x. 1 \triangleleft (1 \triangleleft x) = x)$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson3b:  $(\forall x. (x \triangleright 1) \triangleright 1 = x) \implies (x \sqcap y) \triangleright 1 = (x \triangleright 1) \sqcap (y \triangleright 1)$ 
 $\langle proof \rangle$ 

```

```

lemma jonsson3c:  $\forall x. (x \triangleright 1) \triangleright 1 = x \implies x \triangleright 1 = 1 \triangleleft x$ 
 $\langle proof \rangle$ 

```

end

```

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

```

```

subclass residuated-boolean-algebra  $\langle proof \rangle$ 

```

The following lemmas hold trivially, since they are equivalent to associativity:

ity.

```

lemma res-assoc1:  $x \triangleright (y \triangleright z) = y \cdot x \triangleright z$ 
   $\langle proof \rangle$ 

lemma res-assoc2:  $x \triangleleft (y \cdot z) = (x \triangleleft z) \triangleleft y$ 
   $\langle proof \rangle$ 

lemma res-assoc3:  $(x \triangleright y) \triangleleft z = x \triangleright (y \triangleleft z)$ 
   $\langle proof \rangle$ 

end

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

subclass unital-residuated-boolean
   $\langle proof \rangle$ 

subclass residuated-lmonoid  $\langle proof \rangle$ 

lemma jonsson4:  $(\forall x y. x \triangleleft y = x \cdot (1 \triangleleft y)) \longleftrightarrow (\forall x y. x \triangleright y = (x \triangleright 1) \cdot y)$ 
   $\langle proof \rangle$ 

end

end

```

4 Involutive Residuated Structures

```

theory Involutive-Residuated
  imports Residuated-Lattices
begin

class uminus' =
  fixes uminus' :: ' $a \Rightarrow 'a$  ( $\dashv\dashv$ )  $\rightarrow [81] 80$ )

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:  $x \leq y \implies -y \leq -x$ 
   $\langle proof \rangle$ 

lemma neg'-antitone:  $x \leq y \implies -'y \leq -'x$ 
   $\langle proof \rangle$ 

subclass pogroupoid
   $\langle proof \rangle$ 

abbreviation inv-resl ::  $'a \Rightarrow 'a \Rightarrow 'a$  where
  inv-resl  $y\ x \equiv -(x \cdot (-'y))$ 

abbreviation inv-resr ::  $'a \Rightarrow 'a \Rightarrow 'a$  where
  inv-resr  $x\ y \equiv -'((-y) \cdot x)$ 

sublocale residuated-pogroupoid - - - inv-resl inv-resr
   $\langle proof \rangle$ 

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
   $\langle proof \rangle$ 

lemma inv-resr-neg [simp]: inv-resr  $(-x)\ (-y) = inv-resl\ x\ y$ 
   $\langle proof \rangle$ 

lemma inv-resl-neg' [simp]: inv-resl  $(-'x)\ (-'y) = inv-resr\ x\ y$ 
   $\langle proof \rangle$ 

lemma neg'-mult-resl:  $-'((-y) \cdot (-x)) = inv-resl\ x\ (-'y)$ 
   $\langle proof \rangle$ 

lemma neg-mult-resr:  $-((-'y) \cdot (-'x)) = inv-resr\ (-x)\ y$ 
   $\langle proof \rangle$ 

lemma resr-de-morgan1:  $-'(inv-resr\ (-y)\ (-x)) = -'(inv-resl\ y\ x)$ 
   $\langle proof \rangle$ 

lemma resr-de-morgan2:  $-(inv-resl\ (-'x)\ (-'y)) = -(inv-resr\ x\ y)$ 
   $\langle proof \rangle$ 

```

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' ( $\leq$ ) ( $<$ )
   $\langle proof \rangle$ 

context involutive-pogroupoid begin

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

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

lemma neg-iff-div-unit:  $(\exists a. \text{negation-constant } a) \longleftrightarrow (\exists b. \text{division-unit } b)$ 
   $\langle proof \rangle$ 

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:  $1 + x^* \cdot x^* + x \leq x^*$ 
and star-rtc2:  $1 + y \cdot y + x \leq y \implies x^* \leq y$ 
begin

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

```

$\langle proof \rangle$

We first prove a reflexivity property for residuals.

lemma *residual-r-refl*: $1 \leq x \rightarrow x$
 $\langle proof \rangle$

lemma *residual-l-refl*: $1 \leq x \leftarrow x$
 $\langle proof \rangle$

We now derive pure induction laws for residuals.

lemma *residual-l-pure-induction*: $(x \leftarrow x)^* \leq x \leftarrow x$
 $\langle proof \rangle$

lemma *residual-r-pure-induction*: $(x \rightarrow x)^* \leq x \rightarrow x$
 $\langle proof \rangle$

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*: $1 + x \cdot x^* \leq x^*$
 $\langle proof \rangle$

lemma *star-mon* [intro]: $x \leq y \implies x^* \leq y^*$
 $\langle proof \rangle$

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

lemma *star-inductl*: $z + x \cdot y \leq y \implies x^* \cdot z \leq y$
 $\langle proof \rangle$

lemma *star-inductr*: $z + y \cdot x \leq y \implies z \cdot x^* \leq y$
 $\langle proof \rangle$

subclass *kleene-algebra*
 $\langle proof \rangle$

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:  $1 + x^* \cdot x^* + x \leq x^*$ 
and star-subdist:  $x^* \leq (x + y)^*$ 
and right-pure-induction:  $(x \rightarrow x)^* \leq x \rightarrow x$ 
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:  $1 + y \cdot y + x \leq y \Rightarrow x^* \leq y$ 
⟨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':  $1 + x \cdot x^* \leq x^*$ 
and star-inductl':  $z + x \cdot y \leq y \Rightarrow x^* \cdot z \leq y$ 
and star-inductr':  $z + y \cdot x \leq y \Rightarrow z \cdot x^* \leq y$ 
begin
```

```
subclass kleene-algebra
⟨proof⟩
```

```
subclass action-algebra
⟨proof⟩
```

```

end

sublocale action-algebra ⊆ 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* \Rightarrow '*a lan* \Rightarrow '*a lan* **where**
 $limp\text{-}lan Z Y = \{x. \forall y \in Y. x @ y \in Z\}$

definition *rimp-lan* :: '*a lan* \Rightarrow '*a lan* \Rightarrow '*a lan* **where**
 $rimp\text{-}lan X Z = \{y. \forall x \in X. x @ y \in Z\}$

interpretation *lan-residuated-join-semilattice*: *residuated-sup-lgroupoid* (+) (\subseteq)
 (\subset) (\cdot) *limp-lan rimp-lan*
 $\langle proof \rangle$

interpretation *lan-action-algebra*: *action-algebra* (+) (\cdot) 1 0 (\subseteq) (\subset) (+) *limp-lan rimp-lan star*
 $\langle proof \rangle$

6.4 Relation Action Algebras

definition *limp-rel* :: '*a rel* \Rightarrow '*a rel* \Rightarrow '*a rel* **where**
 $limp\text{-}rel T S = \{(y,x) \mid y x. \forall z. (x,z) \in S \rightarrow (y,z) \in T\}$

definition *rimp-rel* :: '*a rel* \Rightarrow '*a rel* \Rightarrow '*a rel* **where**
 $rimp\text{-}rel R T = \{(y,z) \mid y z. \forall x. (x,y) \in R \rightarrow (x,z) \in T\}$

interpretation *rel-residuated-join-semilattice*: *residuated-sup-lgroupoid* (\cup) (\subseteq) (\subset)
 (O) *limp-rel rimp-rel*
 $\langle proof \rangle$

interpretation *rel-action-algebra*: *action-algebra* (\cup) (O) *Id* {} (\subseteq) (\subset) (\cup) *limp-rel rimp-rel rtranc*
 $\langle proof \rangle$

6.5 Trace Action Algebras

definition *limp-trace* :: ('*p*, '*a*) *trace set* \Rightarrow ('*p*, '*a*) *trace set* \Rightarrow ('*p*, '*a*) *trace set*
where
 $limp\text{-}trace Z Y = \bigcup \{X. t\text{-}prod X Y \subseteq Z\}$

definition *rimp-trace* :: ('*p*, '*a*) *trace set* \Rightarrow ('*p*, '*a*) *trace set* \Rightarrow ('*p*, '*a*) *trace set*
where
 $rimp\text{-}trace X Z = \bigcup \{Y. t\text{-}prod X Y \subseteq Z\}$

interpretation *trace-residuated-join-semilattice*: *residuated-sup-lgroupoid* (\cup) (\subseteq)
 (\subset) *t-prod limp-trace rimp-trace*
 $\langle proof \rangle$

interpretation *trace-action-algebra*: *action-algebra* (\cup) *t-prod t-one t-zero* (\subseteq) (\subset)
 (\cup) *limp-trace rimp-trace t-star*
 $\langle proof \rangle$

6.6 Path Action Algebras

We start with paths that include the empty path.

definition *limp-path* :: 'a path set \Rightarrow 'a path set \Rightarrow 'a path set **where**
 $limp\text{-path } Z Y = \bigcup \{X. p\text{-prod } X Y \subseteq Z\}$

definition *rimp-path* :: 'a path set \Rightarrow 'a path set \Rightarrow 'a path set **where**
 $rimp\text{-path } X Z = \bigcup \{Y. p\text{-prod } X Y \subseteq Z\}$

interpretation *path-residuated-join-semilattice*: residuated-sup-lgroupoid (\cup) (\subseteq)
 (\subset) *p-prod* *limp-path* *rimp-path*
 $\langle proof \rangle$

interpretation *path-action-algebra*: action-algebra (\cup) *p-prod* *p-one* $\{\}$ (\subseteq) (\subset)
 (\cup) *limp-path* *rimp-path* *p-star*
 $\langle proof \rangle$

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

definition *limp-ppath* :: 'a ppath set \Rightarrow 'a ppath set \Rightarrow 'a ppath set **where**
 $limp\text{-ppath } Z Y = \bigcup \{X. pp\text{-prod } X Y \subseteq Z\}$

definition *rimp-ppath* :: 'a ppath set \Rightarrow 'a ppath set \Rightarrow 'a ppath set **where**
 $rimp\text{-ppath } X Z = \bigcup \{Y. pp\text{-prod } X Y \subseteq Z\}$

interpretation *ppath-residuated-join-semilattice*: residuated-sup-lgroupoid (\cup) (\subseteq)
 (\subset) *pp-prod* *limp-ppath* *rimp-ppath*
 $\langle proof \rangle$

interpretation *ppath-action-algebra*: action-algebra (\cup) *pp-prod* *pp-one* $\{\}$ (\subseteq) (\subset)
 (\cup) *limp-ppath* *rimp-ppath* *pp-star*
 $\langle proof \rangle$

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 $\langle proof \rangle$

end

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

```

instance
  ⟨proof⟩

end

instantiation pnat :: residuated-sup-lgroupoid
begin

  definition residual-r-pnat where
    (x::pnat) → y ≡ y – x

  definition residual-l-pnat where
    (y::pnat) ← x ≡ 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:  $x \triangleright y = (x \triangleright 1) \cdot y$ 
and conv2:  $x \triangleleft y = x \cdot (1 \triangleleft y)$ 
begin

```

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

```

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

```

```

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

```

```

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

```

```

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

```

Similarly, for $x^\sim = 1 \triangleleft x$, since $x \triangleright 1 = 1 \triangleleft x$ when $x \triangleright 1 \triangleright 1 = x$ holds.

```

lemma conv-invol':  $1 \triangleleft (1 \triangleleft x) = x$ 
⟨proof⟩

```

```

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

```

```

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

```

```

lemma conv-res':  $(1 \triangleleft x) \cdot - (x \cdot y) \leq - y$ 
⟨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:  $x \triangleright y = (x \triangleright 1) \cdot y$ 
begin

sublocale nra
⟨proof⟩

sublocale relation-algebra where
  composition = (·) and unit = 1 and
  converse =  $\lambda x. x \triangleright 1$ 
⟨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.