

Lattice Properties

Viorel Preoteasa

April 10, 2026

Abstract

This formalization introduces and collects some algebraic structures based on lattices and complete lattices for use in other developments. The structures introduced are modular, and lattice ordered groups. In addition to the results proved for the new lattices, this formalization also introduces theorems about lattices and complete lattices in general.

Contents

1	Overview	1
2	Well founded and transitive relations	2
3	Fixpoints and Complete Lattices	3
4	Conjunctive and Disjunctive Functions	5
5	Simplification Lemmas for Lattices	8
6	Modular and Distributive Lattices	9
7	Lattice Orderd Groups	13

1 Overview

Section 2 introduces well founded and transitive relations. Section 3 introduces some properties about fixpoints of monotonic application which maps monotonic functions to monotonic functions. The most important property is that such a monotonic application has the least fixpoint monotonic. Section 4 introduces conjunctive, disjunctive, universally conjunctive, and universally disjunctive functions. In section 5 some simplification lemmas for lattices are proved. Section 6 introduces modular lattices and proves some

properties about them and about distributive lattices. The main result of this section is that a lattice is distributive if and only if it satisfies

$$\forall x y z : x \sqcap z = y \sqcap z \wedge x \sqcup z = y \sqcup z \longrightarrow x = y$$

Section 7 introduces lattice ordered groups and some of their properties. The most important is that they are distributive lattices, and this property is proved using the results from Section 5.

2 Well founded and transitive relations

theory *WellFoundedTransitive*

imports *Main*

begin

class *transitive* = *ord* +

assumes *order-trans1*: $x < y \implies y < z \implies x < z$

and *less-eq-def*: $x \leq y \iff x = y \vee x < y$

begin

lemma *eq-less-eq* [*simp*]:

$x = y \implies x \leq y$

<proof>

lemma *order-trans2* [*simp*]:

$x \leq y \implies y < z \implies x < z$

<proof>

lemma *order-trans3*:

$x < y \implies y \leq z \implies x < z$

<proof>

end

class *well-founded* = *ord* +

assumes *less-induct1* [*case-names less*]: $(!!x . (!!y . y < x \implies P y) \implies P x) \implies P a$

class *well-founded-transitive* = *transitive* + *well-founded*

instantiation *prod*:: (*ord*, *ord*) *ord*

begin

definition

less-pair-def: $a < b \iff \text{fst } a < \text{fst } b \vee (\text{fst } a = \text{fst } b \wedge \text{snd } a < \text{snd } b)$

definition

less-eq-pair-def: $(a::('a::ord * 'b::ord)) <= b \iff a = b \vee a < b$

instance *<proof>*

end

```

instantiation prod:: (transitive, transitive) transitive
begin
instance ⟨proof⟩
end

```

```

instantiation prod:: (well-founded, well-founded) well-founded
begin
instance ⟨proof⟩
end

```

```

instantiation prod:: (well-founded-transitive, well-founded-transitive) well-founded-transitive
begin
instance ⟨proof⟩
end

```

```

instantiation nat :: transitive
begin

```

```

instance ⟨proof⟩
end

```

```

instantiation nat:: well-founded
begin
instance ⟨proof⟩
end

```

```

instantiation nat:: well-founded-transitive
begin
instance ⟨proof⟩
end

```

```

end

```

3 Fixpoints and Complete Lattices

```

theory Complete-Lattice-Prop
imports WellFoundedTransitive
begin

```

This theory introduces some results about fixpoints of functions on complete lattices. The main result is that a monotonic function mapping monotonic functions to monotonic functions has the least fixpoint monotonic.

```

context complete-lattice begin

```

```

lemma inf-Inf: assumes nonempty:  $A \neq \{\}$ 
shows  $\text{inf } x \ (\text{Inf } A) = \text{Inf } ((\text{inf } x) \text{ ` } A)$ 
  ⟨proof⟩

```

end

definition

$mono\text{-}mono\ F = (mono\ F \wedge (\forall f . mono\ f \longrightarrow mono\ (F\ f)))$

theorem *lfp-mono* [*simp*]:

$mono\text{-}mono\ F \Longrightarrow mono\ (lfp\ F)$
<proof>

lemma *gfp-ordinal-induct*:

fixes $f :: 'a::complete\text{-}lattice \Rightarrow 'a$

assumes $mono: mono\ f$

and $P\text{-}f: !!S. P\ S \Longrightarrow P\ (f\ S)$

and $P\text{-}Union: !!M. \forall S \in M. P\ S \Longrightarrow P\ (Inf\ M)$

shows $P\ (gfp\ f)$

<proof>

theorem *gfp-mono* [*simp*]:

$mono\text{-}mono\ F \Longrightarrow mono\ (gfp\ F)$
<proof>

context *complete-lattice* **begin**

definition

$Sup\text{-}less\ x\ (w::'b::well\text{-}founded) = Sup\ \{y :: 'a . \exists v < w . y = x\ v\}$

lemma *Sup-less-upper*:

$v < w \Longrightarrow P\ v \leq Sup\text{-}less\ P\ w$
<proof>

lemma *Sup-less-least*:

$(!!v . v < w \Longrightarrow P\ v \leq Q) \Longrightarrow Sup\text{-}less\ P\ w \leq Q$
<proof>

end

lemma *Sup-less-fun-eq*:

$((Sup\text{-}less\ P\ w)\ i) = (Sup\text{-}less\ (\lambda v . P\ v\ i))\ w$
<proof>

theorem *fp-wf-induction*:

$f\ x = x \Longrightarrow mono\ f \Longrightarrow (\forall w . (y\ w) \leq f\ (Sup\text{-}less\ y\ w)) \Longrightarrow Sup\ (range\ y) \leq x$
<proof>

end

4 Conjunctive and Disjunctive Functions

theory *Conj-Disj*
imports *Main*
begin

This theory introduces the definitions and some properties for conjunctive, disjunctive, universally conjunctive, and universally disjunctive functions.

locale *conjunctive* =
 fixes *inf-b* :: 'b \Rightarrow 'b \Rightarrow 'b
 and *inf-c* :: 'c \Rightarrow 'c \Rightarrow 'c
 and *times-abc* :: 'a \Rightarrow 'b \Rightarrow 'c
begin

definition

conjunctive = $\{x . (\forall y z . \text{times-abc } x (\text{inf-b } y z) = \text{inf-c } (\text{times-abc } x y) (\text{times-abc } x z))\}$

lemma *conjunctiveI*:

assumes $(\bigwedge b c . \text{times-abc } a (\text{inf-b } b c) = \text{inf-c } (\text{times-abc } a b) (\text{times-abc } a c))$
 shows $a \in \text{conjunctive}$
 $\langle \text{proof} \rangle$

lemma *conjunctiveD*: $x \in \text{conjunctive} \Longrightarrow \text{times-abc } x (\text{inf-b } y z) = \text{inf-c } (\text{times-abc } x y) (\text{times-abc } x z)$
 $\langle \text{proof} \rangle$

end

interpretation *Apply*: *conjunctive inf::'a::semilattice-inf \Rightarrow 'a \Rightarrow 'a*
inf::'b::semilattice-inf \Rightarrow 'b \Rightarrow 'b $\lambda f . f$
 $\langle \text{proof} \rangle$

interpretation *Comp*: *conjunctive inf::('a::lattice \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a)*
inf::('a::lattice \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) (o)
 $\langle \text{proof} \rangle$

lemma *Apply.conjunctive = Comp.conjunctive*
 $\langle \text{proof} \rangle$

locale *disjunctive* =
 fixes *sup-b* :: 'b \Rightarrow 'b \Rightarrow 'b
 and *sup-c* :: 'c \Rightarrow 'c \Rightarrow 'c
 and *times-abc* :: 'a \Rightarrow 'b \Rightarrow 'c
begin

definition

$disjunctive = \{x . (\forall y z . times-abc x (sup-b y z) = sup-c (times-abc x y) (times-abc x z))\}$

lemma *disjunctiveI*:

assumes $(\bigwedge b c . times-abc a (sup-b b c) = sup-c (times-abc a b) (times-abc a c))$
shows $a \in disjunctive$
 $\langle proof \rangle$

lemma *disjunctiveD*: $x \in disjunctive \implies times-abc x (sup-b y z) = sup-c (times-abc x y) (times-abc x z)$

$\langle proof \rangle$

end**interpretation** *Apply*: $disjunctive\ sup::'a::semilattice-sup \Rightarrow 'a \Rightarrow 'a$

$sup::'b::semilattice-sup \Rightarrow 'b \Rightarrow 'b \lambda f . f$
 $\langle proof \rangle$

interpretation *Comp*: $disjunctive\ sup::('a::lattice \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a)$

$sup::('a::lattice \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a) (o)$
 $\langle proof \rangle$

lemma *apply-comp-disjunctive*: $Apply.disjunctive = Comp.disjunctive$

$\langle proof \rangle$

locale *Conjunctive* =

fixes $Inf-b :: 'b\ set \Rightarrow 'b$
and $Inf-c :: 'c\ set \Rightarrow 'c$
and $times-abc :: 'a \Rightarrow 'b \Rightarrow 'c$

begin**definition**

$Conjunctive = \{x . (\forall X . times-abc x (Inf-b X) = Inf-c ((times-abc x) ' X))\}$

lemma *ConjunctiveI*:

assumes $\bigwedge A . times-abc a (Inf-b A) = Inf-c ((times-abc a) ' A)$
shows $a \in Conjunctive$
 $\langle proof \rangle$

lemma *ConjunctiveD*:

assumes $a \in Conjunctive$
shows $times-abc a (Inf-b A) = Inf-c ((times-abc a) ' A)$
 $\langle proof \rangle$

end

interpretation *Apply: Conjunctive Inf* $\text{Inf } \lambda f . f$
<proof>

interpretation *Comp: Conjunctive Inf*::($'a::\text{complete-lattice} \Rightarrow 'a$) *set*) $\Rightarrow ('a \Rightarrow 'a)$
Inf::($'a::\text{complete-lattice} \Rightarrow 'a$) *set*) $\Rightarrow ('a \Rightarrow 'a)$ (*o*)
<proof>

lemma *Apply.Conjunctive = Comp.Conjunctive*
<proof>

locale *Disjunctive =*
 fixes *Sup-b* :: $'b$ *set* $\Rightarrow 'b$
 and *Sup-c* :: $'c$ *set* $\Rightarrow 'c$
 and *times-abc* :: $'a \Rightarrow 'b \Rightarrow 'c$
begin

definition
Disjunctive = $\{x . (\forall X . \text{times-abc } x (\text{Sup-b } X) = \text{Sup-c } ((\text{times-abc } x) ' X))\}$

lemma *DisjunctiveI*:
 assumes $\bigwedge A . \text{times-abc } a (\text{Sup-b } A) = \text{Sup-c } ((\text{times-abc } a) ' A)$
 shows $a \in \text{Disjunctive}$
<proof>

lemma *DisjunctiveD*: $x \in \text{Disjunctive} \Longrightarrow \text{times-abc } x (\text{Sup-b } X) = \text{Sup-c } ((\text{times-abc } x) ' X)$
<proof>

end

interpretation *Apply: Disjunctive Sup* $\text{Sup } \lambda f . f$
<proof>

interpretation *Comp: Disjunctive Sup*::($'a::\text{complete-lattice} \Rightarrow 'a$) *set*) $\Rightarrow ('a \Rightarrow 'a)$
Sup::($'a::\text{complete-lattice} \Rightarrow 'a$) *set*) $\Rightarrow ('a \Rightarrow 'a)$ (*o*)
<proof>

lemma *Apply.Disjunctive = Comp.Disjunctive*
<proof>

lemma [*simp*]: $(F::'a::\text{complete-lattice} \Rightarrow 'b::\text{complete-lattice}) \in \text{Apply.Conjunctive} \Longrightarrow F \in \text{Apply.conjunctive}$
<proof>

lemma [*simp*]: $F \in \text{Apply.conjunctive} \Longrightarrow \text{mono } F$
<proof>

lemma [*simp*]: ($F::'a::\text{complete-lattice} \Rightarrow 'b::\text{complete-lattice}$) \in *Apply.Conjunctive*
 $\Longrightarrow F \text{ top} = \text{top}$
 ⟨*proof*⟩

lemma [*simp*]: ($F::'a::\text{complete-lattice} \Rightarrow 'b::\text{complete-lattice}$) \in *Apply.Disjunctive*
 $\Longrightarrow F \in \text{Apply.disjunctive}$
 ⟨*proof*⟩

lemma [*simp*]: $F \in \text{Apply.disjunctive} \Longrightarrow \text{mono } F$
 ⟨*proof*⟩

lemma [*simp*]: ($F::'a::\text{complete-lattice} \Rightarrow 'b::\text{complete-lattice}$) \in *Apply.Disjunctive*
 $\Longrightarrow F \text{ bot} = \text{bot}$
 ⟨*proof*⟩

lemma *weak-fusion*: $h \in \text{Apply.Disjunctive} \Longrightarrow \text{mono } f \Longrightarrow \text{mono } g \Longrightarrow$
 $h \circ f \leq g \circ h \Longrightarrow h (\text{lfp } f) \leq \text{lfp } g$
 ⟨*proof*⟩

lemma *inf-Disj*: ($\lambda (x::'a::\text{complete-distrib-lattice}) . \text{inf } x \ y$) \in *Apply.Disjunctive*
 ⟨*proof*⟩

end

5 Simplification Lemmas for Lattices

theory *Lattice-Prop*
imports *Main*
begin

This theory introduces some simplification lemmas for semilattices and lattices

notation

inf (**infixl** $\langle \sqcap \rangle$ 70) **and**
sup (**infixl** $\langle \sqcup \rangle$ 65)

context *semilattice-inf* **begin**

lemma [*simp*]: $(x \sqcap y) \sqcap z \leq x$
 ⟨*proof*⟩

lemma [*simp*]: $x \sqcap y \sqcap z \leq y$
 ⟨*proof*⟩

lemma [*simp*]: $x \sqcap (y \sqcap z) \leq y$
 ⟨*proof*⟩

lemma [*simp*]: $x \sqcap (y \sqcap z) \leq z$
 ⟨*proof*⟩

end

context *semilattice-sup* **begin**

lemma [*simp*]: $x \leq x \sqcup y \sqcup z$
<proof>

lemma [*simp*]: $y \leq x \sqcup y \sqcup z$
<proof>

lemma [*simp*]: $y \leq x \sqcup (y \sqcup z)$
<proof>

lemma [*simp*]: $z \leq x \sqcup (y \sqcup z)$
<proof>

end

context *lattice* **begin**

lemma [*simp*]: $x \sqcap y \leq x \sqcup z$
<proof>

lemma [*simp*]: $y \sqcap x \leq x \sqcup z$
<proof>

lemma [*simp*]: $x \sqcap y \leq z \sqcup x$
<proof>

lemma [*simp*]: $y \sqcap x \leq z \sqcup x$
<proof>

end

end

6 Modular and Distributive Lattices

theory *Modular-Distrib-Lattice*

imports *Lattice-Prop*

begin

The main result of this theory is the fact that a lattice is distributive if and only if it satisfies the following property:

term $(\forall x y z . x \sqcap z = y \sqcap z \wedge x \sqcup z = y \sqcup z \implies x = y)$

This result was proved by Bergmann in [1]. The formalization presented here is based on [3, 4].

class *modular-lattice* = *lattice* +

assumes *modular*: $x \leq y \implies x \sqcup (y \sqcap z) = y \sqcap (x \sqcup z)$

context *distrib-lattice* **begin**

subclass *modular-lattice*

<proof>

end

context *lattice* **begin**

definition

$$d\text{-aux } a \ b \ c = (a \sqcap b) \sqcup (b \sqcap c) \sqcup (c \sqcap a)$$

lemma *d-b-c-a*: $d\text{-aux } b \ c \ a = d\text{-aux } a \ b \ c$

<proof>

lemma *d-c-a-b*: $d\text{-aux } c \ a \ b = d\text{-aux } a \ b \ c$

<proof>

definition

$$e\text{-aux } a \ b \ c = (a \sqcup b) \sqcap (b \sqcup c) \sqcap (c \sqcup a)$$

lemma *e-b-c-a*: $e\text{-aux } b \ c \ a = e\text{-aux } a \ b \ c$

<proof>

lemma *e-c-a-b*: $e\text{-aux } c \ a \ b = e\text{-aux } a \ b \ c$

<proof>

definition

$$a\text{-aux } a \ b \ c = (a \sqcap (e\text{-aux } a \ b \ c)) \sqcup (d\text{-aux } a \ b \ c)$$

definition

$$b\text{-aux } a \ b \ c = (b \sqcap (e\text{-aux } a \ b \ c)) \sqcup (d\text{-aux } a \ b \ c)$$

definition

$$c\text{-aux } a \ b \ c = (c \sqcap (e\text{-aux } a \ b \ c)) \sqcup (d\text{-aux } a \ b \ c)$$

lemma *b-a*: $b\text{-aux } a \ b \ c = a\text{-aux } b \ c \ a$

<proof>

lemma *c-a*: $c\text{-aux } a \ b \ c = a\text{-aux } c \ a \ b$

<proof>

lemma [*simp*]: $a\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c$

<proof>

lemma [*simp*]: $b\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c$

<proof>

lemma [*simp*]: $c\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c$

<proof>

lemma [*simp*]: $d\text{-aux } a \ b \ c \leq a\text{-aux } a \ b \ c$
<proof>

lemma [*simp*]: $d\text{-aux } a \ b \ c \leq b\text{-aux } a \ b \ c$
<proof>

lemma [*simp*]: $d\text{-aux } a \ b \ c \leq c\text{-aux } a \ b \ c$
<proof>

lemma *a-meet-e*: $a \sqcap (e\text{-aux } a \ b \ c) = a \sqcap (b \sqcup c)$
<proof>

lemma *b-meet-e*: $b \sqcap (e\text{-aux } a \ b \ c) = b \sqcap (c \sqcup a)$
<proof>

lemma *c-meet-e*: $c \sqcap (e\text{-aux } a \ b \ c) = c \sqcap (a \sqcup b)$
<proof>

lemma *a-join-d*: $a \sqcup d\text{-aux } a \ b \ c = a \sqcup (b \sqcap c)$
<proof>

lemma *b-join-d*: $b \sqcup d\text{-aux } a \ b \ c = b \sqcup (c \sqcap a)$
<proof>

end

context *lattice* **begin**

definition

no-distrib $a \ b \ c = (a \sqcap b \sqcup c \sqcap a < a \sqcap (b \sqcup c))$

definition

incomp $x \ y = (\neg x \leq y \wedge \neg y \leq x)$

definition

N5-lattice $a \ b \ c = (a \sqcap c = b \sqcap c \wedge a < b \wedge a \sqcup c = b \sqcup c)$

definition

M5-lattice $a \ b \ c = (a \sqcap b = b \sqcap c \wedge c \sqcap a = b \sqcap c \wedge a \sqcup b = b \sqcup c \wedge c \sqcup a = b \sqcup c \wedge a \sqcap b < a \sqcup b)$

lemma *M5-lattice-incomp*: $M5\text{-lattice } a \ b \ c \implies \text{incomp } a \ b$
<proof>

end

context *modular-lattice* **begin**

lemma *a-meet-d*: $a \sqcap (d\text{-aux } a \ b \ c) = (a \sqcap b) \sqcup (c \sqcap a)$
<proof>

lemma *b-meet-d*: $b \sqcap (d\text{-aux } a \ b \ c) = (b \sqcap c) \sqcup (a \sqcap b)$
 ⟨proof⟩

lemma *c-meet-d*: $c \sqcap (d\text{-aux } a \ b \ c) = (c \sqcap a) \sqcup (b \sqcap c)$
 ⟨proof⟩

lemma *d-less-e*: *no-distrib* $a \ b \ c \implies d\text{-aux } a \ b \ c < e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *a-meet-b-eg-d*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies a\text{-aux } a \ b \ c \sqcap b\text{-aux } a \ b \ c = d\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *b-meet-c-eg-d*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies b\text{-aux } a \ b \ c \sqcap c\text{-aux } a \ b \ c = d\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *c-meet-a-eg-d*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies c\text{-aux } a \ b \ c \sqcap a\text{-aux } a \ b \ c = d\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *a-def-equiv*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies a\text{-aux } a \ b \ c = (a \sqcup d\text{-aux } a \ b \ c) \sqcap e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *b-def-equiv*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies b\text{-aux } a \ b \ c = (b \sqcup d\text{-aux } a \ b \ c) \sqcap e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *c-def-equiv*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies c\text{-aux } a \ b \ c = (c \sqcup d\text{-aux } a \ b \ c) \sqcap e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *a-join-b-eg-e*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies a\text{-aux } a \ b \ c \sqcup b\text{-aux } a \ b \ c = e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *b-join-c-eg-e*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies b\text{-aux } a \ b \ c \sqcup c\text{-aux } a \ b \ c = e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *c-join-a-eg-e*: $d\text{-aux } a \ b \ c \leq e\text{-aux } a \ b \ c \implies c\text{-aux } a \ b \ c \sqcup a\text{-aux } a \ b \ c = e\text{-aux } a \ b \ c$
 ⟨proof⟩

lemma *no-distrib* $a \ b \ c \implies \text{incomp } a \ b$
 ⟨proof⟩

lemma *M5-modular: no-distrib* $a b c \implies M5\text{-lattice } (a\text{-aux } a b c) (b\text{-aux } a b c)$
 $(c\text{-aux } a b c)$
 $\langle proof \rangle$

lemma *M5-modular-def*: $M5\text{-lattice } a b c = (a \sqcap b = b \sqcap c \wedge c \sqcap a = b \sqcap c \wedge a \sqcup b = b \sqcup c \wedge c \sqcup a = b \sqcup c \wedge a \sqcap b < a \sqcup b)$
 $\langle proof \rangle$

end

context *lattice* **begin**

lemma *not-modular-N5*: $(\neg \text{class.modular-lattice inf } ((\leq)::'a \Rightarrow 'a \Rightarrow \text{bool}) (<) \text{sup}) =$
 $(\exists a b c::'a . N5\text{-lattice } a b c)$
 $\langle proof \rangle$

lemma *not-distrib-N5-M5*: $(\neg \text{class.distrib-lattice } (\sqcap) ((\leq)::'a \Rightarrow 'a \Rightarrow \text{bool}) (<) (\sqcup)) =$
 $(\exists a b c::'a . N5\text{-lattice } a b c) \vee (\exists a b c::'a . M5\text{-lattice } a b c)$
 $\langle proof \rangle$

lemma *distrib-not-N5-M5*: $(\text{class.distrib-lattice } (\sqcap) ((\leq)::'a \Rightarrow 'a \Rightarrow \text{bool}) (<) (\sqcup)) =$
 $(\forall a b c::'a . \neg N5\text{-lattice } a b c) \wedge (\forall a b c::'a . \neg M5\text{-lattice } a b c)$
 $\langle proof \rangle$

lemma *distrib-inf-sup-eq*:
 $(\text{class.distrib-lattice } (\sqcap) ((\leq)::'a \Rightarrow 'a \Rightarrow \text{bool}) (<) (\sqcup)) =$
 $(\forall x y z::'a . x \sqcap z = y \sqcap z \wedge x \sqcup z = y \sqcup z \longrightarrow x = y)$
 $\langle proof \rangle$

end

class *inf-sup-eq-lattice* = *lattice* +
assumes *inf-sup-eq*: $x \sqcap z = y \sqcap z \implies x \sqcup z = y \sqcup z \implies x = y$
begin
subclass *distrib-lattice*
 $\langle proof \rangle$

end

end

7 Lattice Orderd Groups

theory *Lattice-Ordered-Group*
imports *Modular-Distrib-Lattice*
begin

This theory introduces lattice ordered groups [2] and proves some results about them. The most important result is that a lattice ordered group is also a distributive lattice.

```
class lgroup = group-add + lattice +
assumes add-order-preserving:  $a \leq b \implies u + a + v \leq u + b + v$ 
begin
```

```
lemma add-order-preserving-left:  $a \leq b \implies u + a \leq u + b$ 
  <proof>
```

```
lemma add-order-preserving-right:  $a \leq b \implies a + v \leq b + v$ 
  <proof>
```

```
lemma minus-order:  $-a \leq -b \implies b \leq a$ 
  <proof>
```

```
lemma right-move-to-left:  $a + -c \leq b \implies a \leq b + c$ 
  <proof>
```

```
lemma right-move-to-right:  $a \leq b + -c \implies a + c \leq b$ 
  <proof>
```

```
lemma [simp]:  $(a \sqcap b) + c = (a + c) \sqcap (b + c)$ 
  <proof>
```

```
lemma [simp]:  $(a \sqcap b) - c = (a - c) \sqcap (b - c)$ 
  <proof>
```

```
lemma left-move-to-left:  $-c + a \leq b \implies a \leq c + b$ 
  <proof>
```

```
lemma left-move-to-right:  $a \leq -c + b \implies c + a \leq b$ 
  <proof>
```

```
lemma [simp]:  $c + (a \sqcap b) = (c + a) \sqcap (c + b)$ 
  <proof>
```

```
lemma [simp]:  $-(a \sqcap b) = (-a) \sqcup (-b)$ 
  <proof>
```

```
lemma [simp]:  $(a \sqcup b) + c = (a + c) \sqcup (b + c)$ 
  <proof>
```

```
lemma [simp]:  $c + (a \sqcup b) = (c + a) \sqcup (c + b)$ 
  <proof>
```

lemma [*simp*]: $c - (a \sqcap b) = (c - a) \sqcup (c - b)$
<proof>

lemma [*simp*]: $(a \sqcup b) - c = (a - c) \sqcup (b - c)$
<proof>

lemma [*simp*]: $-(a \sqcup b) = (-a) \sqcap (-b)$
<proof>

lemma [*simp*]: $c - (a \sqcup b) = (c - a) \sqcap (c - b)$
<proof>

lemma *add-pos*: $0 \leq a \implies b \leq b + a$
<proof>

lemma *add-pos-left*: $0 \leq a \implies b \leq a + b$
<proof>

lemma *inf-sup*: $a - (a \sqcap b) + b = a \sqcup b$
<proof>

lemma *inf-sup-2*: $b = (a \sqcap b) - a + (a \sqcup b)$
<proof>

subclass *inf-sup-eq-lattice*
<proof>

end

end

References

- [1] G. Bergmann. Zur axiomatik der elementargeometrie. *Monatshefte für Mathematik*, 36:269–284, 1929. 10.1007/BF02307616.
- [2] G. Birkhoff. Lattice, ordered groups. *Ann. of Math. (2)*, 43:298–331, 1942.
- [3] G. Birkhoff. *Lattice theory*. Third edition. American Mathematical Society Colloquium Publications, Vol. XXV. American Mathematical Society, Providence, R.I., 1967.
- [4] S. Burris and H. P. Sankappanavar. *A course in universal algebra*, volume 78 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1981.