

Lattices and Orders in Isabelle/HOL

Markus Wenzel
TU München

January 18, 2026

Abstract

We consider abstract structures of orders and lattices. Many fundamental concepts of lattice theory are developed, including dual structures, properties of bounds versus algebraic laws, lattice operations versus set-theoretic ones etc. We also give example instantiations of lattices and orders, such as direct products and function spaces. Well-known properties are demonstrated, like the Knaster-Tarski Theorem for complete lattices.

This formal theory development may serve as an example of applying Isabelle/HOL to the domain of mathematical reasoning about “axiomatic” structures. Apart from the simply-typed classical set-theory of HOL, we employ Isabelle’s system of axiomatic type classes for expressing structures and functors in a light-weight manner. Proofs are expressed in the Isar language for readable formal proof, while aiming at its “best-style” of representing formal reasoning.

Contents

1	Orders	3
1.1	Ordered structures	3
1.2	Duality	3
1.3	Transforming orders	4
1.3.1	Duals	4
1.3.2	Binary products	5
1.3.3	General products	5
2	Bounds	6
2.1	Infimum and supremum	6
2.2	Duality	7
2.3	Uniqueness	8
2.4	Related elements	8
2.5	General versus binary bounds	8
2.6	Connecting general bounds	9

3 Lattices	9
3.1 Lattice operations	9
3.2 Duality	10
3.3 Algebraic properties	11
3.4 Order versus algebraic structure	11
3.5 Example instances	12
3.5.1 Linear orders	12
3.5.2 Binary products	12
3.5.3 General products	13
3.6 Monotonicity and semi-morphisms	13
4 Complete lattices	14
4.1 Complete lattice operations	14
4.2 The Knaster-Tarski Theorem	15
4.3 Bottom and top elements	16
4.4 Duality	16
4.5 Complete lattices are lattices	16
4.6 Complete lattices and set-theory operations	17

1 Orders

```
theory Orders imports Main begin
```

1.1 Ordered structures

We define several classes of ordered structures over some type $'a$ with relation $\sqsubseteq :: 'a \Rightarrow 'a \Rightarrow \text{bool}$. For a *quasi-order* that relation is required to be reflexive and transitive, for a *partial order* it also has to be anti-symmetric, while for a *linear order* all elements are required to be related (in either direction).

```
class leq =
  fixes leq :: 'a ⇒ 'a ⇒ bool (infixl ‹sqsubseteq› 50)

class quasi-order = leq +
  assumes leq-refl [intro?]:  $x \sqsubseteq x$ 
  assumes leq-trans [trans]:  $x \sqsubseteq y \Rightarrow y \sqsubseteq z \Rightarrow x \sqsubseteq z$ 

class partial-order = quasi-order +
  assumes leq-antisym [trans]:  $x \sqsubseteq y \Rightarrow y \sqsubseteq x \Rightarrow x = y$ 

class linear-order = partial-order +
  assumes leq-linear:  $x \sqsubseteq y \vee y \sqsubseteq x$ 

lemma linear-order-cases:
   $((x::'a::linear-order) \sqsubseteq y \Rightarrow C) \Rightarrow (y \sqsubseteq x \Rightarrow C) \Rightarrow C$ 
  ⟨proof⟩
```

1.2 Duality

The *dual* of an ordered structure is an isomorphic copy of the underlying type, with the \sqsubseteq relation defined as the inverse of the original one.

```
datatype 'a dual = dual 'a

primrec undual :: 'a dual ⇒ 'a where
  undual-dual:  $\text{undual} (\text{dual } x) = x$ 

instantiation dual :: (leq) leq
begin

  definition
    leq-dual-def:  $x' \sqsubseteq y' \equiv \text{undual } y' \sqsubseteq \text{undual } x'$ 

  instance ⟨proof⟩

end

lemma undual-leq [iff?]:  $(\text{undual } x' \sqsubseteq \text{undual } y') = (y' \sqsubseteq x')$ 
  ⟨proof⟩
```

lemma *dual-leq* [iff?]: $(dual x \sqsubseteq dual y) = (y \sqsubseteq x)$
⟨proof⟩

Functions *dual* and *undual* are inverse to each other; this entails the following fundamental properties.

lemma *dual-undual* [simp]: $dual (undual x') = x'$
⟨proof⟩

lemma *undual-dual-id* [simp]: $undual \circ dual = id$
⟨proof⟩

lemma *dual-undual-id* [simp]: $dual \circ undual = id$
⟨proof⟩

Since *dual* (and *undual*) are both injective and surjective, the basic logical connectives (equality, quantification etc.) are transferred as follows.

lemma *undual-equality* [iff?]: $(undual x' = undual y') = (x' = y')$
⟨proof⟩

lemma *dual-equality* [iff?]: $(dual x = dual y) = (x = y)$
⟨proof⟩

lemma *dual-ball* [iff?]: $(\forall x \in A. P (dual x)) = (\forall x' \in dual ` A. P x')$
⟨proof⟩

lemma *range-dual* [simp]: *surj* *dual*
⟨proof⟩

lemma *dual-all* [iff?]: $(\forall x. P (dual x)) = (\forall x'. P x')$
⟨proof⟩

lemma *dual-ex*: $(\exists x. P (dual x)) = (\exists x'. P x')$
⟨proof⟩

lemma *dual-Collect*: $\{dual x \mid x. P (dual x)\} = \{x' \mid P x'\}$
⟨proof⟩

1.3 Transforming orders

1.3.1 Duals

The classes of quasi, partial, and linear orders are all closed under formation of dual structures.

instance *dual* :: (*quasi-order*) *quasi-order*
⟨proof⟩

instance *dual* :: (*partial-order*) *partial-order*
 $\langle proof \rangle$

instance *dual* :: (*linear-order*) *linear-order*
 $\langle proof \rangle$

1.3.2 Binary products

The classes of quasi and partial orders are closed under binary products. Note that the direct product of linear orders need *not* be linear in general.

instantiation *prod* :: (*leq*, *leq*) *leq*
begin

definition

leq-prod-def: $p \sqsubseteq q \equiv \text{fst } p \sqsubseteq \text{fst } q \wedge \text{snd } p \sqsubseteq \text{snd } q$

instance $\langle proof \rangle$

end

lemma *leq-prodI* [*intro?*]:

$\text{fst } p \sqsubseteq \text{fst } q \implies \text{snd } p \sqsubseteq \text{snd } q \implies p \sqsubseteq q$
 $\langle proof \rangle$

lemma *leq-prodE* [*elim?*]:

$p \sqsubseteq q \implies (\text{fst } p \sqsubseteq \text{fst } q \implies \text{snd } p \sqsubseteq \text{snd } q \implies C) \implies C$
 $\langle proof \rangle$

instance *prod* :: (*quasi-order*, *quasi-order*) *quasi-order*
 $\langle proof \rangle$

instance *prod* :: (*partial-order*, *partial-order*) *partial-order*
 $\langle proof \rangle$

1.3.3 General products

The classes of quasi and partial orders are closed under general products (function spaces). Note that the direct product of linear orders need *not* be linear in general.

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

definition

leq-fun-def: $f \sqsubseteq g \equiv \forall x. f x \sqsubseteq g x$

instance $\langle proof \rangle$

end

lemma *leq-funI* [*intro?*]: $(\bigwedge x. f x \sqsubseteq g x) \implies f \sqsubseteq g$
⟨proof⟩

lemma *leq-funD* [*dest?*]: $f \sqsubseteq g \implies f x \sqsubseteq g x$
⟨proof⟩

instance *fun* :: (*type*, *quasi-order*) *quasi-order*
⟨proof⟩

instance *fun* :: (*type*, *partial-order*) *partial-order*
⟨proof⟩

end

2 Bounds

theory *Bounds* **imports** *Orders* **begin**

hide-const (**open**) *inf sup*

2.1 Infimum and supremum

Given a partial order, we define infimum (greatest lower bound) and supremum (least upper bound) wrt. \sqsubseteq for two and for any number of elements.

definition

is-inf :: $'a::\text{partial-order} \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{is-inf } x \ y \ \text{inf} = (\text{inf} \sqsubseteq x \wedge \text{inf} \sqsubseteq y \wedge (\forall z. z \sqsubseteq x \wedge z \sqsubseteq y \longrightarrow z \sqsubseteq \text{inf}))$

definition

is-sup :: $'a::\text{partial-order} \Rightarrow 'a \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{is-sup } x \ y \ \text{sup} = (x \sqsubseteq \text{sup} \wedge y \sqsubseteq \text{sup} \wedge (\forall z. x \sqsubseteq z \wedge y \sqsubseteq z \longrightarrow \text{sup} \sqsubseteq z))$

definition

is-Inf :: $'a::\text{partial-order set} \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{is-Inf } A \ \text{inf} = ((\forall x \in A. \text{inf} \sqsubseteq x) \wedge (\forall z. (\forall x \in A. z \sqsubseteq x) \longrightarrow z \sqsubseteq \text{inf}))$

definition

is-Sup :: $'a::\text{partial-order set} \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{is-Sup } A \ \text{sup} = ((\forall x \in A. x \sqsubseteq \text{sup}) \wedge (\forall z. (\forall x \in A. x \sqsubseteq z) \longrightarrow \text{sup} \sqsubseteq z))$

These definitions entail the following basic properties of boundary elements.

lemma *is-infI* [*intro?*]: $\text{inf} \sqsubseteq x \implies \text{inf} \sqsubseteq y \implies$
 $(\bigwedge z. z \sqsubseteq x \implies z \sqsubseteq y \implies z \sqsubseteq \text{inf}) \implies \text{is-inf } x \ y \ \text{inf}$
⟨proof⟩

lemma *is-inf-greatest* [*elim?*]:
 $\text{is-inf } x \ y \ \text{inf} \implies z \sqsubseteq x \implies z \sqsubseteq y \implies z \sqsubseteq \text{inf}$

$\langle proof \rangle$

lemma *is-inf-lower* [*elim?*]:
 $is\text{-}inf\ x\ y\ inf \implies (inf \sqsubseteq x \implies inf \sqsubseteq y \implies C) \implies C$
 $\langle proof \rangle$

lemma *is-supI* [*intro?*]: $x \sqsubseteq sup \implies y \sqsubseteq sup \implies$
 $(\bigwedge z. x \sqsubseteq z \implies y \sqsubseteq z \implies sup \sqsubseteq z) \implies is\text{-}sup\ x\ y\ sup$
 $\langle proof \rangle$

lemma *is-sup-least* [*elim?*]:
 $is\text{-}sup\ x\ y\ sup \implies x \sqsubseteq z \implies y \sqsubseteq z \implies sup \sqsubseteq z$
 $\langle proof \rangle$

lemma *is-sup-upper* [*elim?*]:
 $is\text{-}sup\ x\ y\ sup \implies (x \sqsubseteq sup \implies y \sqsubseteq sup \implies C) \implies C$
 $\langle proof \rangle$

lemma *is-InfI* [*intro?*]: $(\bigwedge x. x \in A \implies inf \sqsubseteq x) \implies$
 $(\bigwedge z. (\forall x \in A. z \sqsubseteq x) \implies z \sqsubseteq inf) \implies is\text{-}Inf\ A\ inf$
 $\langle proof \rangle$

lemma *is-Inf-greatest* [*elim?*]:
 $is\text{-}Inf\ A\ inf \implies (\bigwedge x. x \in A \implies z \sqsubseteq x) \implies z \sqsubseteq inf$
 $\langle proof \rangle$

lemma *is-Inf-lower* [*dest?*]:
 $is\text{-}Inf\ A\ inf \implies x \in A \implies inf \sqsubseteq x$
 $\langle proof \rangle$

lemma *is-SupI* [*intro?*]: $(\bigwedge x. x \in A \implies x \sqsubseteq sup) \implies$
 $(\bigwedge z. (\forall x \in A. x \sqsubseteq z) \implies sup \sqsubseteq z) \implies is\text{-}Sup\ A\ sup$
 $\langle proof \rangle$

lemma *is-Sup-least* [*elim?*]:
 $is\text{-}Sup\ A\ sup \implies (\bigwedge x. x \in A \implies x \sqsubseteq z) \implies sup \sqsubseteq z$
 $\langle proof \rangle$

lemma *is-Sup-upper* [*dest?*]:
 $is\text{-}Sup\ A\ sup \implies x \in A \implies x \sqsubseteq sup$
 $\langle proof \rangle$

2.2 Duality

Infimum and supremum are dual to each other.

theorem *dual-inf* [*iff?*]:

is-inf (dual x) (dual y) (dual sup) = is-sup x y sup
⟨proof⟩

theorem *dual-sup* [iff?]:

is-sup (dual x) (dual y) (dual inf) = is-inf x y inf
⟨proof⟩

theorem *dual-Inf* [iff?]:

is-Inf (dual ‘ A) (dual sup) = is-Sup A sup
⟨proof⟩

theorem *dual-Sup* [iff?]:

is-Sup (dual ‘ A) (dual inf) = is-Inf A inf
⟨proof⟩

2.3 Uniqueness

Infima and suprema on partial orders are unique; this is mainly due to anti-symmetry of the underlying relation.

theorem *is-inf-uniq*: *is-inf x y inf* \implies *is-inf x y inf'* \implies *inf = inf'*
⟨proof⟩

theorem *is-sup-uniq*: *is-sup x y sup* \implies *is-sup x y sup'* \implies *sup = sup'*
⟨proof⟩

theorem *is-Inf-uniq*: *is-Inf A inf* \implies *is-Inf A inf'* \implies *inf = inf'*
⟨proof⟩

theorem *is-Sup-uniq*: *is-Sup A sup* \implies *is-Sup A sup'* \implies *sup = sup'*
⟨proof⟩

2.4 Related elements

The binary bound of related elements is either one of the argument.

theorem *is-inf-related* [elim?]: *x ⊑ y* \implies *is-inf x y x*
⟨proof⟩

theorem *is-sup-related* [elim?]: *x ⊑ y* \implies *is-sup x y y*
⟨proof⟩

2.5 General versus binary bounds

General bounds of two-element sets coincide with binary bounds.

theorem *is-Inf-binary*: *is-Inf {x, y} inf* = *is-inf x y inf*
⟨proof⟩

theorem *is-Sup-binary*: *is-Sup {x, y} sup* = *is-sup x y sup*

$\langle proof \rangle$

2.6 Connecting general bounds

Either kind of general bounds is sufficient to express the other. The least upper bound (supremum) is the same as the the greatest lower bound of the set of all upper bounds; the dual statements holds as well; the dual statement holds as well.

theorem *Inf-Sup*: *is-Inf* { b . $\forall a \in A$. $a \sqsubseteq b$ } *sup* \implies *is-Sup A sup*
 $\langle proof \rangle$

theorem *Sup-Inf*: *is-Sup* { b . $\forall a \in A$. $b \sqsubseteq a$ } *inf* \implies *is-Inf A inf*
 $\langle proof \rangle$

end

3 Lattices

theory *Lattice imports Bounds begin*

3.1 Lattice operations

A *lattice* is a partial order with infimum and supremum of any two elements (thus any *finite* number of elements have bounds as well).

class *lattice* =
assumes *ex-inf*: $\exists inf$. *is-inf x y inf*
assumes *ex-sup*: $\exists sup$. *is-sup x y sup*

The \sqcap (meet) and \sqcup (join) operations select such infimum and supremum elements.

definition
 $meet :: 'a::lattice \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\langle \sqcap \rangle$ 70) **where**
 $x \sqcap y = (THE inf. is-inf x y inf)$
definition
 $join :: 'a::lattice \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\langle \sqcup \rangle$ 65) **where**
 $x \sqcup y = (THE sup. is-sup x y sup)$

Due to unique existence of bounds, the lattice operations may be exhibited as follows.

lemma *meet-equality [elim?]*: *is-inf x y inf* \implies $x \sqcap y = inf$
 $\langle proof \rangle$

lemma *meetI [intro?]*:
 $inf \sqsubseteq x \implies inf \sqsubseteq y \implies (\bigwedge z. z \sqsubseteq x \implies z \sqsubseteq y \implies z \sqsubseteq inf) \implies x \sqcap y = inf$
 $\langle proof \rangle$

lemma *join-equality* [elim?]: *is-sup* x y *sup* $\implies x \sqcup y = sup$
 $\langle proof \rangle$

lemma *joinI* [intro?]: $x \sqsubseteq sup \implies y \sqsubseteq sup \implies$
 $(\bigwedge z. x \sqsubseteq z \implies y \sqsubseteq z \implies sup \sqsubseteq z) \implies x \sqcup y = sup$
 $\langle proof \rangle$

The \sqcap and \sqcup operations indeed determine bounds on a lattice structure.

lemma *is-inf-meet* [intro?]: *is-inf* x y ($x \sqcap y$)
 $\langle proof \rangle$

lemma *meet-greatest* [intro?]: $z \sqsubseteq x \implies z \sqsubseteq y \implies z \sqsubseteq x \sqcap y$
 $\langle proof \rangle$

lemma *meet-lower1* [intro?]: $x \sqcap y \sqsubseteq x$
 $\langle proof \rangle$

lemma *meet-lower2* [intro?]: $x \sqcap y \sqsubseteq y$
 $\langle proof \rangle$

lemma *is-sup-join* [intro?]: *is-sup* x y ($x \sqcup y$)
 $\langle proof \rangle$

lemma *join-least* [intro?]: $x \sqsubseteq z \implies y \sqsubseteq z \implies x \sqcup y \sqsubseteq z$
 $\langle proof \rangle$

lemma *join-upper1* [intro?]: $x \sqsubseteq x \sqcup y$
 $\langle proof \rangle$

lemma *join-upper2* [intro?]: $y \sqsubseteq x \sqcup y$
 $\langle proof \rangle$

3.2 Duality

The class of lattices is closed under formation of dual structures. This means that for any theorem of lattice theory, the dualized statement holds as well; this important fact simplifies many proofs of lattice theory.

instance *dual* :: (*lattice*) *lattice*
 $\langle proof \rangle$

Apparently, the \sqcap and \sqcup operations are dual to each other.

theorem *dual-meet* [intro?]: *dual* ($x \sqcap y$) = *dual* $x \sqcup$ *dual* y
 $\langle proof \rangle$

theorem *dual-join* [intro?]: *dual* ($x \sqcup y$) = *dual* $x \sqcap$ *dual* y
 $\langle proof \rangle$

3.3 Algebraic properties

The \sqcap and \sqcup operations have the following characteristic algebraic properties: associative (A), commutative (C), and absorptive (AB).

theorem *meet-assoc*: $(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z)$
 $\langle proof \rangle$

theorem *join-assoc*: $(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
 $\langle proof \rangle$

theorem *meet-commute*: $x \sqcap y = y \sqcap x$
 $\langle proof \rangle$

theorem *join-commute*: $x \sqcup y = y \sqcup x$
 $\langle proof \rangle$

theorem *meet-join-absorb*: $x \sqcap (x \sqcup y) = x$
 $\langle proof \rangle$

theorem *join-meet-absorb*: $x \sqcup (x \sqcap y) = x$
 $\langle proof \rangle$

Some further algebraic properties hold as well. The property idempotent (I) is a basic algebraic consequence of (AB).

theorem *meet-idem*: $x \sqcap x = x$
 $\langle proof \rangle$

theorem *join-idem*: $x \sqcup x = x$
 $\langle proof \rangle$

Meet and join are trivial for related elements.

theorem *meet-related* [elim?]: $x \sqsubseteq y \implies x \sqcap y = x$
 $\langle proof \rangle$

theorem *join-related* [elim?]: $x \sqsubseteq y \implies x \sqcup y = y$
 $\langle proof \rangle$

3.4 Order versus algebraic structure

The \sqcap and \sqcup operations are connected with the underlying \sqsubseteq relation in a canonical manner.

theorem *meet-connection*: $(x \sqsubseteq y) = (x \sqcap y = x)$
 $\langle proof \rangle$

theorem *join-connection*: $(x \sqsubseteq y) = (x \sqcup y = y)$
 $\langle proof \rangle$

The most fundamental result of the meta-theory of lattices is as follows (we do not prove it here).

Given a structure with binary operations \sqcap and \sqcup such that (A), (C), and (AB) hold (cf. §3.3). This structure represents a lattice, if the relation $x \sqsubseteq y$ is defined as $x \sqcap y = x$ (alternatively as $x \sqcup y = y$). Furthermore, infimum and supremum with respect to this ordering coincide with the original \sqcap and \sqcup operations.

3.5 Example instances

3.5.1 Linear orders

Linear orders with *minimum* and *maximum* operations are a (degenerate) example of lattice structures.

definition

```
minimum :: 'a::linear-order ⇒ 'a ⇒ 'a where
minimum x y = (if x ⊑ y then x else y)
```

definition

```
maximum :: 'a::linear-order ⇒ 'a ⇒ 'a where
maximum x y = (if x ⊑ y then y else x)
```

lemma *is-inf-minimum*: *is-inf* x y (*minimum* x y)
 $\langle proof \rangle$

lemma *is-sup-maximum*: *is-sup* x y (*maximum* x y)
 $\langle proof \rangle$

instance *linear-order* \subseteq *lattice*
 $\langle proof \rangle$

The lattice operations on linear orders indeed coincide with *minimum* and *maximum*.

theorem *meet-minimum*: $x \sqcap y = \text{minimum } x \text{ } y$
 $\langle proof \rangle$

theorem *meet-maximum*: $x \sqcup y = \text{maximum } x \text{ } y$
 $\langle proof \rangle$

3.5.2 Binary products

The class of lattices is closed under direct binary products (cf. §1.3.2).

lemma *is-inf-prod*: *is-inf* p q (*fst* $p \sqcap \text{fst } q$, *snd* $p \sqcap \text{snd } q$)
 $\langle proof \rangle$

lemma *is-sup-prod*: *is-sup* p q (*fst* $p \sqcup \text{fst } q$, *snd* $p \sqcup \text{snd } q$)
 $\langle proof \rangle$

instance *prod* :: (*lattice*, *lattice*) *lattice*
 $\langle proof \rangle$

The lattice operations on a binary product structure indeed coincide with the products of the original ones.

theorem *meet-prod*: $p \sqcap q = (\text{fst } p \sqcap \text{fst } q, \text{snd } p \sqcap \text{snd } q)$
 $\langle proof \rangle$

theorem *join-prod*: $p \sqcup q = (\text{fst } p \sqcup \text{fst } q, \text{snd } p \sqcup \text{snd } q)$
 $\langle proof \rangle$

3.5.3 General products

The class of lattices is closed under general products (function spaces) as well (cf. §1.3.3).

lemma *is-inf-fun*: $\text{is-inf } f g (\lambda x. f x \sqcap g x)$
 $\langle proof \rangle$

lemma *is-sup-fun*: $\text{is-sup } f g (\lambda x. f x \sqcup g x)$
 $\langle proof \rangle$

instance *fun* :: (*type*, *lattice*) *lattice*
 $\langle proof \rangle$

The lattice operations on a general product structure (function space) indeed emerge by point-wise lifting of the original ones.

theorem *meet-fun*: $f \sqcap g = (\lambda x. f x \sqcap g x)$
 $\langle proof \rangle$

theorem *join-fun*: $f \sqcup g = (\lambda x. f x \sqcup g x)$
 $\langle proof \rangle$

3.6 Monotonicity and semi-morphisms

The lattice operations are monotone in both argument positions. In fact, monotonicity of the second position is trivial due to commutativity.

theorem *meet-mono*: $x \sqsubseteq z \implies y \sqsubseteq w \implies x \sqcap y \sqsubseteq z \sqcap w$
 $\langle proof \rangle$

theorem *join-mono*: $x \sqsubseteq z \implies y \sqsubseteq w \implies x \sqcup y \sqsubseteq z \sqcup w$
 $\langle proof \rangle$

A semi-morphisms is a function f that preserves the lattice operations in the following manner: $f(x \sqcap y) \sqsubseteq f x \sqcap f y$ and $f x \sqcup f y \sqsubseteq f(x \sqcup y)$, respectively. Any of these properties is equivalent with monotonicity.

theorem *meet-semimorph*:

$$(\bigwedge x y. f(x \sqcap y) \sqsubseteq f x \sqcap f y) \equiv (\bigwedge x y. x \sqsubseteq y \implies f x \sqsubseteq f y)$$

$\langle proof \rangle$

lemma *join-semimorph*:

$$(\bigwedge x y. f x \sqcup f y \sqsubseteq f(x \sqcup y)) \equiv (\bigwedge x y. x \sqsubseteq y \implies f x \sqsubseteq f y)$$

$\langle proof \rangle$

end

4 Complete lattices

theory *CompleteLattice* **imports** *Lattice* **begin**

4.1 Complete lattice operations

A *complete lattice* is a partial order with general (infinitary) infimum of any set of elements. General supremum exists as well, as a consequence of the connection of infinitary bounds (see §2.6).

class *complete-lattice* =
assumes *ex-Inf*: $\exists \text{inf}$. *is-Inf* *A inf*

theorem *ex-Sup*: $\exists \text{sup} : 'a :: \text{complete-lattice}$. *is-Sup* *A sup*
 $\langle proof \rangle$

The general \sqcap (meet) and \sqcup (join) operations select such infimum and supremum elements.

definition

Meet :: $'a :: \text{complete-lattice}$ set $\Rightarrow 'a (\langle \sqcap \rangle [90] 90)$ **where**
 $\sqcap A = (\text{THE inf. is-Inf } A \text{ inf})$

definition

Join :: $'a :: \text{complete-lattice}$ set $\Rightarrow 'a (\langle \sqcup \rangle [90] 90)$ **where**
 $\sqcup A = (\text{THE sup. is-Sup } A \text{ sup})$

Due to unique existence of bounds, the complete lattice operations may be exhibited as follows.

lemma *Meet-equality* [*elim?*]: *is-Inf* *A inf* $\implies \sqcap A = \text{inf}$
 $\langle proof \rangle$

lemma *MeetI* [*intro?*]:

$$(\bigwedge a. a \in A \implies \text{inf} \sqsubseteq a) \implies$$

$$(\bigwedge b. \forall a \in A. b \sqsubseteq a \implies b \sqsubseteq \text{inf}) \implies$$

$$\sqcap A = \text{inf}$$

$\langle proof \rangle$

lemma *Join-equality* [*elim?*]: *is-Sup* *A sup* $\implies \sqcup A = \text{sup}$
 $\langle proof \rangle$

lemma *JoinI* [*intro?*]:

$$\begin{aligned} (\bigwedge a. a \in A \implies a \sqsubseteq \text{sup}) &\implies \\ (\bigwedge b. \forall a \in A. a \sqsubseteq b \implies \text{sup} \sqsubseteq b) &\implies \\ \bigsqcup A &= \text{sup} \end{aligned}$$

{proof}

The \sqcap and \sqcup operations indeed determine bounds on a complete lattice structure.

lemma *is-Inf-Meet* [*intro?*]: *is-Inf A* ($\sqcap A$)

{proof}

lemma *Meet-greatest* [*intro?*]: $(\bigwedge a. a \in A \implies x \sqsubseteq a) \implies x \sqsubseteq \sqcap A$

{proof}

lemma *Meet-lower* [*intro?*]: $a \in A \implies \sqcap A \sqsubseteq a$

{proof}

lemma *is-Sup-Join* [*intro?*]: *is-Sup A* ($\sqcup A$)

{proof}

lemma *Join-least* [*intro?*]: $(\bigwedge a. a \in A \implies a \sqsubseteq x) \implies \bigsqcup A \sqsubseteq x$

{proof}

lemma *Join-lower* [*intro?*]: $a \in A \implies a \sqsubseteq \bigsqcup A$

{proof}

4.2 The Knaster-Tarski Theorem

The Knaster-Tarski Theorem (in its simplest formulation) states that any monotone function on a complete lattice has a least fixed-point (see [2, pages 93–94] for example). This is a consequence of the basic boundary properties of the complete lattice operations.

theorem *Knaster-Tarski*:

assumes *mono*: $\bigwedge x y. x \sqsubseteq y \implies f x \sqsubseteq f y$

obtains *a* :: *'a::complete-lattice* **where**

$f a = a$ **and** $\bigwedge a'. f a' = a' \implies a \sqsubseteq a'$

{proof}

theorem *Knaster-Tarski-dual*:

assumes *mono*: $\bigwedge x y. x \sqsubseteq y \implies f x \sqsubseteq f y$

obtains *a* :: *'a::complete-lattice* **where**

$f a = a$ **and** $\bigwedge a'. f a' = a' \implies a' \sqsubseteq a$

{proof}

4.3 Bottom and top elements

With general bounds available, complete lattices also have least and greatest elements.

definition

```
bottom :: 'a::complete-lattice (⊥) where
  ⊥ = ⋂ UNIV
```

definition

```
top :: 'a::complete-lattice (⊤) where
  ⊤ = ⋃ UNIV
```

lemma *bottom-least* [intro?]: $\perp \sqsubseteq x$
 $\langle \text{proof} \rangle$

lemma *bottomI* [intro?]: $(\bigwedge a. x \sqsubseteq a) \implies \perp = x$
 $\langle \text{proof} \rangle$

lemma *top-greatest* [intro?]: $x \sqsubseteq \top$
 $\langle \text{proof} \rangle$

lemma *topI* [intro?]: $(\bigwedge a. a \sqsubseteq x) \implies \top = x$
 $\langle \text{proof} \rangle$

4.4 Duality

The class of complete lattices is closed under formation of dual structures.

instance *dual* :: (*complete-lattice*) *complete-lattice*
 $\langle \text{proof} \rangle$

Apparently, the \sqcap and \sqcup operations are dual to each other.

theorem *dual-Meet* [intro?]: $\text{dual} (\sqcap A) = \sqcup (\text{dual} ` A)$
 $\langle \text{proof} \rangle$

theorem *dual-Join* [intro?]: $\text{dual} (\sqcup A) = \sqcap (\text{dual} ` A)$
 $\langle \text{proof} \rangle$

Likewise are \perp and \top duals of each other.

theorem *dual-bottom* [intro?]: $\text{dual} \perp = \top$
 $\langle \text{proof} \rangle$

theorem *dual-top* [intro?]: $\text{dual} \top = \perp$
 $\langle \text{proof} \rangle$

4.5 Complete lattices are lattices

Complete lattices (with general bounds available) are indeed plain lattices as well. This holds due to the connection of general versus binary bounds

that has been formally established in §2.5.

lemma *is-inf-binary*: *is-inf* x y ($\sqcap \{x, y\}$)
 $\langle proof \rangle$

lemma *is-sup-binary*: *is-sup* x y ($\sqcup \{x, y\}$)
 $\langle proof \rangle$

instance *complete-lattice* \subseteq *lattice*
 $\langle proof \rangle$

theorem *meet-binary*: $x \sqcap y = \sqcap \{x, y\}$
 $\langle proof \rangle$

theorem *join-binary*: $x \sqcup y = \sqcup \{x, y\}$
 $\langle proof \rangle$

4.6 Complete lattices and set-theory operations

The complete lattice operations are (anti) monotone wrt. set inclusion.

theorem *Meet-subset-antimono*: $A \subseteq B \implies \sqcap B \sqsubseteq \sqcap A$
 $\langle proof \rangle$

theorem *Join-subset-mono*: $A \subseteq B \implies \sqcup A \sqsubseteq \sqcup B$
 $\langle proof \rangle$

Bounds over unions of sets may be obtained separately.

theorem *Meet-Un*: $\sqcap(A \cup B) = \sqcap A \sqcap \sqcap B$
 $\langle proof \rangle$

theorem *Join-Un*: $\sqcup(A \cup B) = \sqcup A \sqcup \sqcup B$
 $\langle proof \rangle$

Bounds over singleton sets are trivial.

theorem *Meet-singleton*: $\sqcap \{x\} = x$
 $\langle proof \rangle$

theorem *Join-singleton*: $\sqcup \{x\} = x$
 $\langle proof \rangle$

Bounds over the empty and universal set correspond to each other.

theorem *Meet-empty*: $\sqcap \{\} = \sqcup \text{UNIV}$
 $\langle proof \rangle$

theorem *Join-empty*: $\sqcup \{\} = \sqcap \text{UNIV}$
 $\langle proof \rangle$

end

References

- [1] G. Bauer and M. Wenzel. Computer-assisted mathematics at work — the Hahn-Banach theorem in Isabelle/Isar. In T. Coquand, P. Dybjer, B. Nordström, and J. Smith, editors, *Types for Proofs and Programs: TYPES'99*, LNCS, 2000.
- [2] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.
- [3] M. Wenzel. Isar — a generic interpretative approach to readable formal proof documents. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher Order Logics: TPHOLs '99*, volume 1690 of *LNCS*, 1999.
- [4] M. Wenzel. *The Isabelle/Isar Reference Manual*, 2000. <https://isabelle.in.tum.de/doc/isar-ref.pdf>.
- [5] M. Wenzel. *Using Axiomatic Type Classes in Isabelle*, 2000. <https://isabelle.in.tum.de/doc/axclass.pdf>.