Domain theory
Basics
Advanced
Powerdomains
Powerset
Ordering
Varieties
Formalization
Compact basis
Defining operations
Properties
Applications

Powerdomains in Isabelle / HOLCF

Brian Huffman

Portland State University

February 8, 2008

Outline

Domain theory
Basics
Advanced
Powerdomains
Powerset
Ordering
Varieties
Formalization
Compact basis
Defining operations
Properties
Applications

1. **Overview of domain theory**
   1. Basics: partial orders, cpos, continuity, fixed points
   2. Advanced: compactness, bifinite cpos, compact bases
2. Properties of powerdomains
3. Formalization using ideal completion
4. Applications: interleaving and concurrency

Information Ordering

Information order relation: $\sqsubseteq$

- $x \sqsubseteq y$ means that x approximates y
- $\sqsubseteq$ is a partial order (reflexive, transitive, antisymmetric)
- $\perp$ denotes the least element: $\forall x.\ \perp \sqsubseteq x$

Example: Haskell List datatype

- $\perp \sqsubseteq (1 : \perp) \sqsubseteq (1 : 2 : \perp) \sqsubseteq (1 : 2 : 3 : \perp) \sqsubseteq (1 : 2 : 3 : [])$
- Constructors preserve ordering:
  $x : xs \sqsubseteq y : ys \iff x \sqsubseteq y \land xs \sqsubseteq ys$

Completeness

A **chain** is a sequence a with increasing information:

- $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq a_3 \sqsubseteq a_4 \sqsubseteq \ldots$

A **cpo** has **least upper bounds** for all chains:

- $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq a_3 \sqsubseteq a_4 \sqsubseteq \cdots \rightsquigarrow \bigsqcup_n a_n$

Example: lub of partial lists may be an infinite list

- $a_0 = \perp, a_1 = 1 : \perp, a_2 = 1 : 1 : \perp, a_3 = 1 : 1 : 1 : \perp, \ldots$
- $\bigsqcup_n a_n = \mathrm{ones} = 1 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : 1 : \ldots$

Continuity

A **continuous** function f preserves least upper bounds:

- $f(\bigsqcup_n a_n) = (\bigsqcup_n f(a_n))$

Continuous functions must also be **monotone**:

- $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$
- Monotonicity ensures that $f(a_n)$ is a chain

Every continuous f over a pointed cpo has a **least fixed point**:

- $\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq f(f(f(\bot))) \sqsubseteq \cdots \rightsquigarrow \mathrm{fix}(f)$

Admissibility

An **admissible** predicate P is preserved by lubs:

- $(\forall n.\, P(a_n)) \implies P(\bigsqcup_n a_n)$ for any chain a

Many constructions preserve admissibility:

- $(\lambda x.\, P(x) \wedge Q(x))$ is admissible if P and Q are
- $(\lambda x.\, P(x) \vee Q(x))$ is admissible if P and Q are
- $(\lambda x.\, f(x) = g(x))$ is admissible for continuous f and g
- $(\lambda x.\, f(x) \sqsubseteq g(x))$ is admissible for continuous f and g

Admissibility is important for **fixed point induction**:

- For admissible P, if $P(\bot)$ and $\forall x.\, P(x) \implies P(f(x))$, then $P(\text{fix}(f))$

Compactness

An infinite value may be the lub of a chain of strictly smaller values:

- $\bot \sqsubseteq 1 : \bot \sqsubseteq 1 : 1 : \bot \sqsubseteq 1 : 1 : 1 : \bot \sqsubseteq \cdots \rightsquigarrow$ ones

But a **compact** value k cannot be written this way:

- Any chain with lub k must contain k
- $k \sqsubseteq (\bigsqcup_n a_n) \sqsubseteq k \implies \exists n.\, k \sqsubseteq a_n$
- $(\lambda x.\, x \neq k)$ and $(\lambda x.\, x \not\sqsubseteq k)$ are admissible
- Intuitively, k has a finite amount of information

Example: ordinal numbers

- Successor ordinals are compact, limit ordinals are not

Finite approximation

Bifinite cpos admit a notion of finite approximation:

- $\mathrm{take}_0(x) \sqsubseteq \mathrm{take}_1(x) \sqsubseteq \mathrm{take}_2(x) \sqsubseteq \mathrm{take}_3(x) \sqsubseteq \cdots \rightsquigarrow x$
- $\mathrm{take}_n(\mathrm{take}_n(x)) = \mathrm{take}_n(x)$
- Each function take_n should have finite range
- $\mathrm{take}_n(x)$ is compact for any x
- x is compact iff $\exists n.\, \mathrm{take}_n(x) = x$

Example: Haskell list type

- $\mathrm{take}_0(xs) = \bot$
- $\mathrm{take}_{n+1}(x : xs) = \mathrm{take}_n(x) : \mathrm{take}_n(xs)$
- $\mathrm{take}_3(\mathrm{ones}) = 1 : 1 : 1 : \bot$

Directed sets and Ideals

A **directed set** S contains an upper bound for every finite subset:

- $\forall \text{finite } M \subseteq S.\ \exists z \in S.\ \forall x \in M.\ x \sqsubseteq z$

Equivalently, S is nonempty and has an upper bound for every pair of elements:

- $\exists x \in S$ and $\forall x \in S.\ \forall y \in S.\ \exists z \in S.\ x \sqsubseteq z \wedge y \sqsubseteq z$

A directed set S is an **ideal** if it is also downward-closed:

- $\forall x\, y.\, x \sqsubseteq y \implies y \in S \implies x \in S$
- For any x, the set $x{\downarrow} = \{y.\, y \sqsubseteq x\}$ is an ideal, called a principal ideal

Ideal Completion

Let $(A, \sqsubseteq)$ be a partial order, but not a cpo

- Some chains do not have least upper bounds
- Can we extend it to a cpo by adding the missing lubs?

Yes! The set $\mathsf{Ideal}(A)$ of ideals over A is a cpo:

- Let $\sqsubseteq$ be the subset ordering on ideals
- The union of a chain of ideals is an ideal
- Principal ideals are exactly the compact elements

We say that A is a **compact basis** for $\mathsf{Ideal}(A)$

- $\mathsf{K}(\mathsf{Ideal}(A)) \simeq A$
- $\mathsf{Ideal}(\mathsf{K}(B)) \simeq B$ for a bifinite cpo B

Examples of Ideal Completion

Natural numbers with $\leq$:

- The naturals with $\leq$ form a partial order, but not a cpo (some chains like 1,2,3,4,5... have no least upper bound)
- Ideals over nat are nonempty, downward-closed sets
- Most ideals are principal, like $\{0\}$, $\{0,1\}$, $\{0,1,2,3\}$
- There is one non-principal ideal — ω. The addition of ω at the top of the ordering turns the naturals into a cpo.

Lists with the prefix ordering:

- Infinite chains like [], [1], [1,2], [1,2,3],... are unbounded
- Non-principal ideals represent infinite lists

Compact Bases

Some more notes about compact bases and bifinite cpos:

- The set $K(x)$ of compact approximations to x is an ideal
- x is the least upper bound of $K(x)$
- A continuous function is completely determined by its values on compact inputs
- If admissible predicate P holds for all compact x, then it holds for all x

Outline

1. Overview of domain theory
2. **Properties of powerdomains**
    1. Powerset monad
    2. Information ordering on powerdomains
    3. Varieties: convex, upper, and lower
3. Formalization using ideal completion
4. Applications: interleaving and concurrency

Powerset monad

A nondeterministic function can be modeled as a function that returns a set of possible results.

Powerset monad

Operations:

- unit :: $A \to \wp(A)$
  $\text{unit}(x) = \{x\}$
- bind :: $\wp(A) \to (A \to \wp(B)) \to \wp(B)$
  $\text{bind}(m, f) = \bigcup_{x \in m} f(x)$
- mplus :: $\wp(A) \to \wp(A) \to \wp(A)$
  $\text{mplus}(m, n) = m \cup n$

Properties:

- unit and bind satisfy the monad laws
- mplus is associative, commutative, and idempotent

Powerset monad

Can we make the powerset monad into a cpo?

- This would let us combine nondeterminism with recursion

Possibility: use subset ordering

- Ordering is complete (use set-union for lubs)
- However, unit operation $x \mapsto \{x\}$ is not monotone!

Information ordering on $\wp(A)$ must respect ordering on A

Powerdomain ordering

Requirements for a powerdomain ordering:

- ordering must give a cpo
- unit, bind, and mplus must be monotone and continuous

Examples:

- $\{\bot, 1\} \sqsubseteq \{1, 1\} = \{1\}$
- $\{\bot\} = \{\bot, \bot\} \sqsubseteq \{1, \bot\} = \{1, \bot, \bot\} \sqsubseteq \{1, 2, \bot\} \ldots$

Powerdomain ordering identifies sets with the same convex closure:

- assume $x \sqsubseteq y \sqsubseteq z$
- $\{x, z\} = \{x, x, z\} \sqsubseteq \{x, y, z\}$
- $\{x, y, z\} \sqsubseteq \{x, z, z\} = \{x, z\}$

Free Continuous Algebras

Consider the free continuous algebra over $\{-\}$ and $\uplus$, modulo these laws:

- $\sqsubseteq$ is a partial order
- $x \sqsubseteq y \implies \{x\} \sqsubseteq \{y\}$
- $a \sqsubseteq a' \wedge b \sqsubseteq b' \implies a \uplus b \sqsubseteq a' \uplus b'$
- $(a \uplus (b \uplus c)) = ((a \uplus b) \uplus c)$
- $a \uplus b = b \uplus a$
- $a \uplus a = a$

(bind can be defined uniquely in terms of unit and mplus)
This specifies a powerdomain called the **convex powerdomain**.

Varieties of powerdomains

We can add extra laws to create more varieties:

- Adding $a \uplus b \sqsubseteq a$ gives the **upper powerdomain**
- Adding $a \sqsubseteq a \uplus b$ gives the **lower powerdomain**

If a and b are finite (i.e. finite combinations of $\{-\}$ and $\uplus$) then:

- $a \sqsubseteq b$ in upper powerdomain iff $\forall y \in b.\, \exists x \in a.\, x \sqsubseteq y$
- $a \sqsubseteq b$ in lower powerdomain iff $\forall x \in a.\, \exists y \in b.\, x \sqsubseteq y$
- $a \sqsubseteq b$ in convex powerdomain iff $\forall y \in b.\, \exists x \in a.\, x \sqsubseteq y$ and $\forall x \in a.\, \exists y \in b.\, x \sqsubseteq y$

Outline

1. Overview of domain theory
2. Specification of powerdomains
3. **Formalization using ideal completion**
    1. Compact basis for powerdomains
    2. Defining operations using compact basis
    3. Transferring properties from basis
4. Applications: interleaving and concurrency

Compact Basis for Powerdomains

Compact elements of the powerdomain are finite, nonempty sets of compact elements:

```
typedef 'a pd_basis =
  {S::K('a) set. finite S ∧ nonempty S}
```

We also define the unit and mplus operations on the basis:

```
basis_unit x = {x}
basis_mplus t u = t ∪ u
```

The ordering for the convex powerdomain is defined thus:

```
t⊑u =
  (∀x∈t. ∃y∈u. x⊑y) ∧ (∀y∈u. ∃x∈t. x⊑y)
```

The ordering is not antisymmetric, but we can quotient by equivalence.

Defining Powerdomain by Ideal Completion

Now we can define the convex powerdomain type constructor:

```
typedef 'a convex_pd =
 {S::'a pd_basis set. ideal S}
```

We can also define a function for embedding basis elements:

```
principal :: 'a pd_basis => 'a convex_pd
principal x = Abs_convex_pd {y. y ⊑ x}
```

Defining Functions on a Basis

To define a function from A to B, if A has compact basis C:

- First define a monotone function from C to B
- Uniquely extend that function to all of A

Definition:

$$\texttt{basis\_fun f x} = (\bigsqcup a{\in}\texttt{K(x)}.\ \texttt{f(x)})$$

Desired property:

```
basis_fun f (principal a) = f a
```

How do we know the lub exists? This is equivalent:

$$\texttt{basis\_fun f x} = $$
$$(\bigsqcup \texttt{n}.\ \bigsqcup a{\in}\texttt{take n ` K(x)}.\ \texttt{f(x)})$$

- inner lub exists because the set is finite and directed
- outer lub exists because it is a chain

Defining Powerdomain Operations

To define unit, mplus:

```
unit = basis_fun (λx.
  principal (basis_unit x))

mplus = basis_fun (λt.
  basis_fun (λu.
  principal (basis_mplus t u)))
```

Defining Powerdomain Operations

To define bind:

```
basis_bind (basis_unit x) = (λf. f x)
basis_bind (basis_mplus a b) =
 (λf. mplus (basis_bind a f)
  (basis_bind b f))

bind = basis_fun basis_bind
```

Proving Properties of Operations

We can lift many properties from pd_basis to convex_pd using this rule:

```
lemma principal_induct:
  adm P ==>
  (∀t. P (principal t)) ==>
  (∀a. P a)
```

For example:

```
∀a b. mplus a b = mplus b a
∀t b. mplus (principal t) b
  = mplus b (principal t)
∀t u. mplus (principal t) (principal u)
  = mplus (principal u) (principal t)
```

Applications: Interleaving and Concurrency

The convex powerdomain constructor can model nondeterminism directly.
It can also model concurrency, when used together with a resumption monad transformer:

- Use resumptions to model waiting threads
- Use mplus operator to nondeterministically choose which thread to run
- Powerdomain models the set of possible interleavings

Applications: Interleaving and Concurrency

I formalized a resumption monad transformer in HOLCF a while back

- Combined with powerdomains, we can now formalize interesting results about monads with concurrency, state, nondeterminism, etc.
- Having an axiomatic constructor class for powerdomains means that it is easy to try different varieties of powerdomain