Powerdomains in Isabelle/HOLCF

Brian Huffman

Portland State University

Galois Technical Seminar, February 26, 2008

Outline

Background

I assume familiarity with some basic domain theory:

- Bottoms ($\perp$)
- Complete partial orders ($\sqsubseteq$)
- Limits of chains
- Monotone and continuous functions

Outline

What are Powerdomains?

A powerdomain is

- a monad
- with a nondeterministic choice operator

Powerdomains adapt the notion of powersets to work with domain theory.

What are Powerdomains?

A powerdomain is

- a monad
- with a nondeterministic choice operator

Powerdomains adapt the notion of powersets to work with domain theory.

What are Powerdomains?

A powerdomain is

- a monad
- with a nondeterministic choice operator

Powerdomains adapt the notion of powersets to work with domain theory.

What are Powerdomains?

A powerdomain is

- a monad
- with a nondeterministic choice operator

Powerdomains adapt the notion of powersets to work with domain theory.

What are Powerdomains Good For?

Powerdomains are good for reasoning about

- Nondeterministic algorithms
  - Write algorithms monadically in a powerdomain
  - Works with arbitrary recursion
- Parallel computation
  - Resumption monad transformer models interleaving
  - Powerdomain models nondeterministic scheduler

What are Powerdomains Good For?

Powerdomains are good for reasoning about

- Nondeterministic algorithms
  - Write algorithms monadically in a powerdomain
  - Works with arbitrary recursion
- Parallel computation
  - Resumption monad transformer models interleaving
  - Powerdomain models nondeterministic scheduler

What are Powerdomains Good For?

Powerdomains are good for reasoning about

- Nondeterministic algorithms
  - Write algorithms monadically in a powerdomain
  - Works with arbitrary recursion
- Parallel computation
  - Resumption monad transformer models interleaving
  - Powerdomain models nondeterministic scheduler

Outline

1. Motivation
   - Introduction to Powerdomains
   - Limitations of the List Monad for Nondeterminism
2. Properties of Powerdomains
   - Axiomatization of Powerdomains
   - Powerdomain Orderings
   - Visualizing Powerdomains
3. Formalization in Isabelle
   - Definition in Terms of Finite Elements
   - Using Powerdomains in Isabelle

Using List Monad for Nondeterminism

Good for modeling in Haskell:

- Executable

Bad for denotational semantics:

- Not abstract enough
- Problems with partial/infinite values

Using List Monad for Nondeterminism

Good for modeling in Haskell:

- Executable

Bad for denotational semantics:

- Not abstract enough
- Problems with partial/infinite values

Using List Monad for Nondeterminism

Good for modeling in Haskell:

- Executable

Bad for denotational semantics:

- Not abstract enough
- Problems with partial/infinite values

Using List Monad for Nondeterminism

Good for modeling in Haskell:

- Executable

Bad for denotational semantics:

- Not abstract enough
- Problems with partial/infinite values

Examples Using List Monad in Haskell

1. Monadic merge sort with nondeterministic comparison
2. Nondeterministically choosing a node in a tree

Monads with Nondeterministic Choice

Haskell type class for monads with nondeterministic choice operator

```
class (Monad m) => MultiMonad m where
  (+|+) :: m a -> m a -> m a
```

Haskell lists can model nondeterministic computation

```
instance MultiMonad [] where
  xs +|+ ys = xs ++ ys
```

Monadic Merge Sort

```
mergesort :: (Monad m) =>
  (a -> a -> m Bool) -> [a] -> m [a]

mergesort r [] = return []
mergesort r [x] = return [x]
mergesort r xs = do ys' <- mergesort r ys
                    zs' <- mergesort r zs
                    xs' <- merge r ys' zs'
                    return xs'
  where (ys,zs) = split xs
```

Monadic Merge Sort

```
merge :: (Monad m) =>
  (a -> a -> m Bool) -> [a] -> [a] -> m [a]

merge r xs [] = return xs
merge r [] ys = return ys
merge r (x:xs) (y:ys) =
  do b <- r x y
     if b then do zs <- merge r xs (y:ys)
                  return (x:zs)
          else do zs <- merge r (x:xs) ys
                  return (y:zs)
```

Monadic Merge Sort

```
r1, r2 :: (MultiMonad m) =>
  (Int, a) -> (Int, a) -> m Bool

r1 (x,_) (y,_) =
  case compare x y of
    LT -> return True
    GT -> return False
    EQ -> return True +|+ return False

r2 (x,_) (y,_) =
  return (x <= y) +|+ return (x < y)
```

Monadic Merge Sort

```
dataset :: [(Int, String)]
dataset =
  [(3,"foo"),(2,"bar"),(1,"baz"),(2,"wibble")]
```

- r1 is basically equivalent to r2
- mergesort r1 dataset should be equivalent to mergesort r2 dataset
- What happens in Haskell?

Nondeterministic Choice with Binary Trees

```
data Tree a = Node (Tree a) (Tree a) | Leaf a

pick (Leaf a) = return a
pick (Node l r) = pick l +|+ pick r

mirror (Leaf a) = Leaf a
mirror (Node l r) = Node (mirror r) (mirror l)
```

- pick (mirror t) should be equivalent to pick t

Nondeterministic Choice with Binary Trees

```
tree1 = Node (Node (Leaf 1) (Leaf 2))
             (Node (Leaf 3) (Leaf 4))

tree2 = Node (Node (Leaf 1) (Leaf 2))
             (Node undefined (Leaf 4))

tree3 = Node (Node (Leaf 1) tree3)
             (Node (Leaf 3) (Leaf 4))
```

- What happens in Haskell?

Outline

Powerdomain Axioms

1. Return and bind satisfy monad laws

```
(return x) >>= f == f x
a >>= return == m
(a >>= f) >>= g == a >>= (\x -> f x >>= g)
```

2. Bind distributes over choice operator

```
(a +|+ b) >>= f == (a >>= f) +|+ (b >>= f)
```

3. Choice operator is associative, commutative, idempotent

```
(a +|+ b) +|+ c == a +|+ (b +|+ c)
a +|+ b == b +|+ a
a +|+ a == a
```

Powerdomain Axioms

1. Return and bind satisfy monad laws

```
(return x) >>= f == f x
a >>= return == m
(a >>= f) >>= g == a >>= (\x -> f x >>= g)
```

2. Bind distributes over choice operator

```
(a +|+ b) >>= f == (a >>= f) +|+ (b >>= f)
```

3. Choice operator is associative, commutative, idempotent

```
(a +|+ b) +|+ c == a +|+ (b +|+ c)
a +|+ b == b +|+ a
a +|+ a == a
```

Powerdomain Axioms

1. Return and bind satisfy monad laws

```
(return x) >>= f == f x
a >>= return == m
(a >>= f) >>= g == a >>= (\x -> f x >>= g)
```

2. Bind distributes over choice operator

```
(a +|+ b) >>= f == (a >>= f) +|+ (b >>= f)
```

3. Choice operator is associative, commutative, idempotent

```
(a +|+ b) +|+ c == a +|+ (b +|+ c)
a +|+ b == b +|+ a
a +|+ a == a
```

Powerdomain Axioms

In addition:

- All operations must be **monotone** and **continuous**

Representing Powerdomains with Sets

We can define the powerdomain operations in terms of sets:

```
return x = {x}
a >>= f = (⋃x∈a. f x)
a +|+ b = a ∪ b
```

- Set operations satisfy the monad laws
- Set union is associative, commutative, and idempotent
- What about monotonicity and continuity?
  - First we must define a complete partial order...

Representing Powerdomains with Sets

We can define the powerdomain operations in terms of sets:

```
return x = {x}
a >>= f = (⋃x∈a. f x)
a +|+ b = a ∪ b
```

- Set operations satisfy the monad laws
- Set union is associative, commutative, and idempotent
- What about monotonicity and continuity?
  - First we must define a complete partial order...

Representing Powerdomains with Sets

We can define the powerdomain operations in terms of sets:

```
return x = {x}
a >>= f = (⋃x∈a. f x)
a +|+ b = a ∪ b
```

- Set operations satisfy the monad laws
- Set union is associative, commutative, and idempotent
- What about monotonicity and continuity?
    - First we must define a complete partial order...

Representing Powerdomains with Sets

We can define the powerdomain operations in terms of sets:

```
return x = {x}
a >>= f = (⋃x∈a. f x)
a +|+ b = a ∪ b
```

- Set operations satisfy the monad laws
- Set union is associative, commutative, and idempotent
- What about monotonicity and continuity?
  - First we must define a complete partial order...

Outline

The Subset Ordering Does Not Work

The subset relation $(\subseteq)$

- is a partial order on sets
- is complete (unions give least upper bounds)

But not all operations are monotone w.r.t $(\subseteq)$

- $x \sqsubseteq y$ does not imply that $\{x\} \subseteq \{y\}$

The Subset Ordering Does Not Work

The subset relation ($\subseteq$)

- is a partial order on sets
- is complete (unions give least upper bounds)

But not all operations are monotone w.r.t ($\subseteq$)

- $x \sqsubseteq y$ does not imply that $\{x\} \subseteq \{y\}$

Monotonicity Implies Certain Equivalences

Theorem

Let $x \sqsubseteq y \sqsubseteq z$.
Then as elements of a powerdomain, $\{x,y,z\} = \{x,z\}$.

Proof.

- From $y \sqsubseteq z$ have $\{x,y,z\} \sqsubseteq \{x,z,z\}$ (monotonicity)
- Hence $\{x,y,z\} \sqsubseteq \{x,z\}$ (idempotency)
- From $x \sqsubseteq y$ have $\{x,x,z\} \sqsubseteq \{x,y,z\}$ (monotonicity)
- Hence $\{x,z\} \sqsubseteq \{x,y,z\}$ (idempotency)
- Finally have $\{x,y,z\} = \{x,z\}$ (antisymmetry)

$\square$

Monotonicity Implies Certain Equivalences

Theorem

Let $x \sqsubseteq y \sqsubseteq z$.
Then as elements of a powerdomain, $\{x,y,z\} = \{x,z\}$.

Proof.

- From $y \sqsubseteq z$ have $\{x,y,z\} \sqsubseteq \{x,z,z\}$ (monotonicity)
- Hence $\{x,y,z\} \sqsubseteq \{x,z\}$ (idempotency)
- From $x \sqsubseteq y$ have $\{x,x,z\} \sqsubseteq \{x,y,z\}$ (monotonicity)
- Hence $\{x,z\} \sqsubseteq \{x,y,z\}$ (idempotency)
- Finally have $\{x,y,z\} = \{x,z\}$ (antisymmetry)

$\square$

An Ordering That Does Work: The Upper Powerdomain

Upper powerdomain ordering ($\sqsubseteq^\sharp$)

- Define $(a \sqsubseteq^\sharp b) \Longleftrightarrow (\forall y \in b. \exists x \in a. x \sqsubseteq y)$
  - "everything in b is above something in a"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its upward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \cup b \sqsubseteq^\sharp a$
  - Union is greatest lower bound (meet) w.r.t. ($\sqsubseteq^\sharp$)

An Ordering That Does Work: The Upper Powerdomain

Upper powerdomain ordering ($\sqsubseteq^\sharp$)

- Define $(a \sqsubseteq^\sharp b) \iff (\forall y \in b. \exists x \in a. x \sqsubseteq y)$
  - "everything in b is above something in a"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its upward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \cup b \sqsubseteq^\sharp a$
  - Union is greatest lower bound (meet) w.r.t. ($\sqsubseteq^\sharp$)

An Ordering That Does Work: The Upper Powerdomain

Upper powerdomain ordering ($\sqsubseteq^\sharp$)

- Define $(a \sqsubseteq^\sharp b) \iff (\forall y \in b. \exists x \in a. x \sqsubseteq y)$
  - "everything in b is above something in a"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its upward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \cup b \sqsubseteq^\sharp a$
  - Union is greatest lower bound (meet) w.r.t. ($\sqsubseteq^\sharp$)

An Ordering That Does Work: The Upper Powerdomain

Upper powerdomain ordering ($\sqsubseteq^\sharp$)

- Define $(a \sqsubseteq^\sharp b) \iff (\forall y \in b. \exists x \in a. x \sqsubseteq y)$
  - "everything in b is above something in a"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its upward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \cup b \sqsubseteq^\sharp a$
  - Union is greatest lower bound (meet) w.r.t. ($\sqsubseteq^\sharp$)

Another Ordering That Works: The Lower Powerdomain

Lower powerdomain ordering ($\sqsubseteq^\flat$)

- Define $(a \sqsubseteq^\flat b) \Longleftrightarrow (\forall x \in a. \exists y \in b. x \sqsubseteq y)$
  - "everything in a is below something in b"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its downward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \sqsubseteq^\flat a \cup b$
  - Union is least upper bound (join) w.r.t. ($\sqsubseteq^\flat$)

Another Ordering That Works: The Lower Powerdomain

Lower powerdomain ordering ($\sqsubseteq^\flat$)

- Define $(a \sqsubseteq^\flat b) \Longleftrightarrow (\forall x \in a. \exists y \in b. x \sqsubseteq y)$
  - "everything in a is below something in b"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its downward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \sqsubseteq^\flat a \cup b$
  - Union is least upper bound (join) w.r.t. ($\sqsubseteq^\flat$)

Another Ordering That Works: The Lower Powerdomain

Lower powerdomain ordering ($\sqsubseteq^\flat$)

- Define $(a \sqsubseteq^\flat b) \iff (\forall x \in a. \exists y \in b. x \sqsubseteq y)$
  - "everything in a is below something in b"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its downward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \sqsubseteq^\flat a \cup b$
  - Union is least upper bound (join) w.r.t. ($\sqsubseteq^\flat$)

Another Ordering That Works: The Lower Powerdomain

Lower powerdomain ordering ($\sqsubseteq^\flat$)

- Define $(a \sqsubseteq^\flat b) \iff (\forall x \in a. \exists y \in b. x \sqsubseteq y)$
  - "everything in a is below something in b"
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its downward-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\sharp$)
- Satisfies an additional law: $a \sqsubseteq^\flat a \cup b$
  - Union is least upper bound (join) w.r.t. ($\sqsubseteq^\flat$)

A Third Ordering That Works: The Convex Powerdomain

Convex powerdomain ordering ($\sqsubseteq^\natural$)

- Define $(a \sqsubseteq^\natural b) \iff (a \sqsubseteq^\sharp b) \wedge (a \sqsubseteq^\flat b)$
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its convex-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\natural$)
- Satisfies no additional laws
  - Is the continuous free algebra satisfying powerdomain axioms

A Third Ordering That Works: The Convex Powerdomain

Convex powerdomain ordering ($\sqsubseteq^\natural$)

- Define $(a \sqsubseteq^\natural b) \iff (a \sqsubseteq^\sharp b) \wedge (a \sqsubseteq^\flat b)$
- Partial *preorder* on sets (not antisymmetric)
    - Every set equivalent to its convex-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\natural$)
- Satisfies no additional laws
    - Is the continuous free algebra satisfying powerdomain axioms

A Third Ordering That Works: The Convex Powerdomain

Convex powerdomain ordering ($\sqsubseteq^\natural$)

- Define $(a \sqsubseteq^\natural b) \iff (a \sqsubseteq^\sharp b) \wedge (a \sqsubseteq^\flat b)$
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its convex-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\natural$)
- Satisfies no additional laws
  - Is the continuous free algebra satisfying powerdomain axioms

A Third Ordering That Works: The Convex Powerdomain

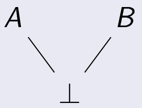
Convex powerdomain ordering ($\sqsubseteq^\natural$)

- Define $(a \sqsubseteq^\natural b) \iff (a \sqsubseteq^\sharp b) \wedge (a \sqsubseteq^\flat b)$
- Partial *preorder* on sets (not antisymmetric)
  - Every set equivalent to its convex-closure
- All operations are monotone w.r.t. ($\sqsubseteq^\natural$)
- Satisfies no additional laws
  - Is the continuous free algebra satisfying powerdomain axioms

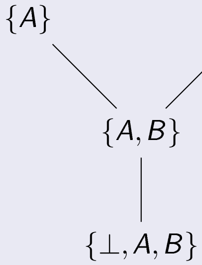
Outline

Powerdomains of Lifted 2-Element Type

Argument Type

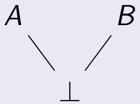


Upper Powerdomain

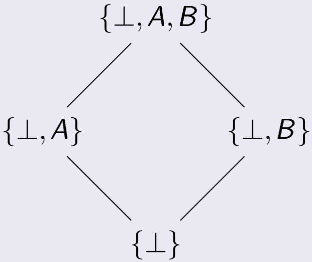


Powerdomains of Lifted 2-Element Type

Argument Type

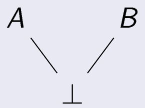


Lower Powerdomain

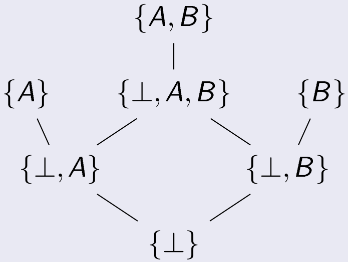


Powerdomains of Lifted 2-Element Type

Argument Type

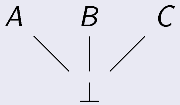


Convex Powerdomain

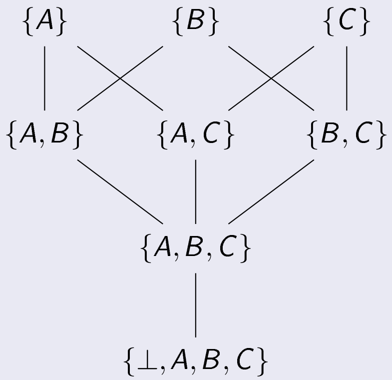


Powerdomains of Lifted 3-Element Type

Argument Type

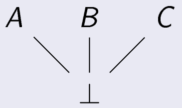


Upper Powerdomain

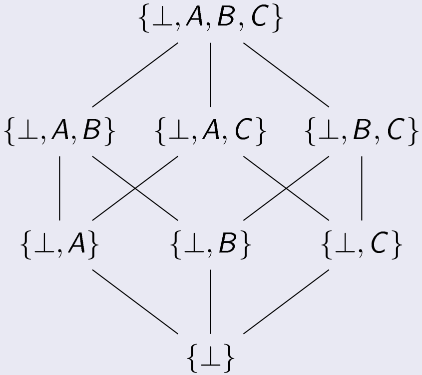


Powerdomains of Lifted 3-Element Type

Argument Type

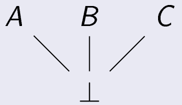


Lower Powerdomain

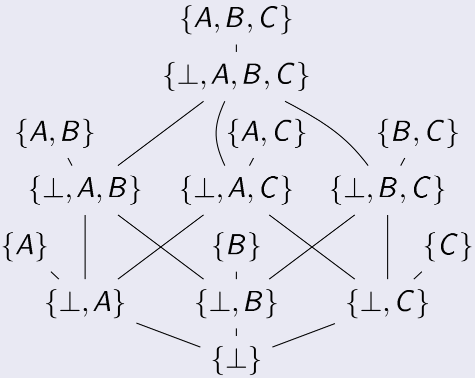


Powerdomains of Lifted 3-Element Type

Argument Type

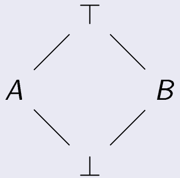


Convex Powerdomain

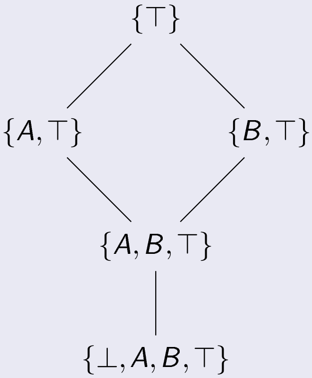


Powerdomains of 4-Element Lattice

Argument Type



Upper Powerdomain



Powerdomains of 4-Element Lattice

Argument Type

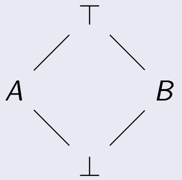
$$
\begin{tikzcd}
 & \top & \\
A \arrow[ur, dash] & & B \arrow[ul, dash] \\
 & \bot \arrow[ul, dash] \arrow[ur, dash] &
\end{tikzcd}
$$

Lower Powerdomain

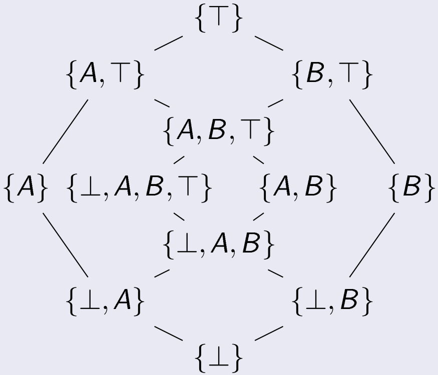
$$
\begin{tikzcd}
 & \{\bot, A, B, \top\} & \\
 & \{\bot, A, B\} \arrow[u, dash] & \\
\{\bot, A\} \arrow[ur, dash] & & \{\bot, B\} \arrow[ul, dash] \\
 & \{\bot\} \arrow[ul, dash] \arrow[ur, dash] &
\end{tikzcd}
$$

Powerdomains of 4-Element Lattice

Argument Type



Convex Powerdomain



Notes on Different Kinds of Powerdomains

- Upper
  - $a \sqsubseteq b \Longleftarrow a$ has *more* possible outcomes than b
  - Union is strict: $\bot \cup a = \bot$
  - "Possibly not terminating is just as bad as never terminating"
  - Good for modeling total correctness
- Lower
  - $a \sqsubseteq b \Longleftarrow a$ has *fewer* possible outcomes than b
  - Bottom is identity for union: $\bot \cup a = a$
  - "I don't care about execution paths that don't terminate"
  - Good for modeling partial correctness
- Convex
  - Distinguishes more values than upper or lower
  - Agnostic on total vs. partial correctness

Notes on Different Kinds of Powerdomains

- Upper
  - $a \sqsubseteq b \Longleftarrow a$ has *more* possible outcomes than b
  - Union is strict: $\bot \cup a = \bot$
  - "Possibly not terminating is just as bad as never terminating"
  - Good for modeling total correctness
- Lower
  - $a \sqsubseteq b \Longleftarrow a$ has *fewer* possible outcomes than b
  - Bottom is identity for union: $\bot \cup a = a$
  - "I don't care about execution paths that don't terminate"
  - Good for modeling partial correctness
- Convex
  - Distinguishes more values than upper or lower
  - Agnostic on total vs. partial correctness

Notes on Different Kinds of Powerdomains

- Upper
  - $a \sqsubseteq b \Longleftarrow a$ has *more* possible outcomes than b
  - Union is strict: $\bot \cup a = \bot$
  - "Possibly not terminating is just as bad as never terminating"
  - Good for modeling total correctness
- Lower
  - $a \sqsubseteq b \Longleftarrow a$ has *fewer* possible outcomes than b
  - Bottom is identity for union: $\bot \cup a = a$
  - "I don't care about execution paths that don't terminate"
  - Good for modeling partial correctness
- Convex
  - Distinguishes more values than upper or lower
  - Agnostic on total vs. partial correctness

Outline

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

What Sets Are Included in a Powerdomain?

A powerdomain type contains

- singletons
- binary unions
- limits of chains
- nothing else

Each powerdomain includes

- All finite nonempty sets
- Only those infinite sets needed for completeness

Defining CPOs Using Ideal Completion

- Let $(\preceq)$ be a reflexive, transitive relation
- An ideal A is a set that is
  - nonempty $(\exists x. x \in A)$
  - downward-closed $(\forall y \in A. \forall x \preceq y. x \in A)$
  - directed $(\forall x, y \in A. \exists z \in A. x \preceq z \wedge y \preceq z)$
- Principal ideals have a maximum element
  - $\{a \mid a \preceq x\}$ is the principal ideal generated by x
  - $x \preceq y$ implies $\{a \mid a \preceq x\} \subseteq \{a \mid a \preceq y\}$
- The set of ideals over $(\preceq)$ is a CPO
  - The union of a chain of ideals is an ideal

Defining CPOs Using Ideal Completion

- Let $(\preceq)$ be a reflexive, transitive relation
- An ideal A is a set that is
  - nonempty $(\exists x.\, x \in A)$
  - downward-closed $(\forall y \in A.\, \forall x \preceq y.\, x \in A)$
  - directed $(\forall x, y \in A.\, \exists z \in A.\, x \preceq z \wedge y \preceq z)$
- Principal ideals have a maximum element
  - $\{a \mid a \preceq x\}$ is the principal ideal generated by x
  - $x \preceq y$ implies $\{a \mid a \preceq x\} \subseteq \{a \mid a \preceq y\}$
- The set of ideals over $(\preceq)$ is a CPO
  - The union of a chain of ideals is an ideal

Defining CPOs Using Ideal Completion

- Let $(\preceq)$ be a reflexive, transitive relation
- An ideal A is a set that is
  - nonempty $(\exists x.\, x \in A)$
  - downward-closed $(\forall y \in A.\, \forall x \preceq y.\, x \in A)$
  - directed $(\forall x, y \in A.\, \exists z \in A.\, x \preceq z \wedge y \preceq z)$
- Principal ideals have a maximum element
  - $\{a \mid a \preceq x\}$ is the principal ideal generated by x
  - $x \preceq y$ implies $\{a \mid a \preceq x\} \subseteq \{a \mid a \preceq y\}$
- The set of ideals over $(\preceq)$ is a CPO
  - The union of a chain of ideals is an ideal

Defining CPOs Using Ideal Completion

- Let $(\preceq)$ be a reflexive, transitive relation
- An ideal A is a set that is
  - nonempty $(\exists x. x \in A)$
  - downward-closed $(\forall y \in A. \forall x \preceq y. x \in A)$
  - directed $(\forall x, y \in A. \exists z \in A. x \preceq z \wedge y \preceq z)$
- Principal ideals have a maximum element
  - $\{a \mid a \preceq x\}$ is the principal ideal generated by x
  - $x \preceq y$ implies $\{a \mid a \preceq x\} \subseteq \{a \mid a \preceq y\}$
- The set of ideals over $(\preceq)$ is a CPO
  - The union of a chain of ideals is an ideal

Examples of Ideal Completion

Example

Naturals with $(\leq)$ ordering

- $(\leq)$ is not a complete ordering

In ideal completion

- Finite n represented by $\{a \mid a \leq n\}$
- Infinite value ω represented by $\mathbb{N}$

Examples of Ideal Completion

Example

Pairs of naturals with $(a,b) \preceq (c,d)$ iff $a \leq c$ and $b \leq d$

- Example: $(3,5) \preceq (4,7)$, but $(3,5) \not\preceq (7,4)$

In ideal completion

- Finite (m,n) represented by $\{(a,b) \mid a \leq m \wedge b \leq n\}$
- (ω, n) represented by $\{(a,b) \mid b \leq n\}$
- (ω, ω) represented by $\mathbb{N} \times \mathbb{N}$

Examples of Ideal Completion

Example

Lists with $xs \preceq ys$ iff xs is a prefix of ys

- Example: $(3,5) \preceq (4,7)$, but $(3,5) \not\preceq (7,4)$

In ideal completion

- Finite $[1,2,3]$ represented by $\{[], [1], [1,2], [1,2,3]\}$
- Infinite $[1,1,1...]$ represented by $\{xs \mid xs \text{ contains all 1s}\}$

What About Antisymmetry?

- $(\preceq)$ does not need to be antisymmetric
- If x and y are equivalent w.r.t. $(\preceq)$
  - i.e. $x \preceq y$ and $y \preceq x$
  - then $\{a \mid a \preceq x\} = \{a \mid a \preceq y\}$
- Ideal completion handles equivalence classes automatically

Defining Powerdomains by Ideal Completion

- Define type constructor for nonempty finite sets
  - Basis of finite elements for powerdomains
- Define partial preorder relations $(\sqsubseteq^\sharp)$, $(\sqsubseteq^\flat)$, $(\sqsubseteq^\natural)$
  - All 3 powerdomains have the same abstract basis
- Define each powerdomain using ideal completion
  - Upper $=$ Ideal$(\sqsubseteq^\sharp)$
  - Lower $=$ Ideal$(\sqsubseteq^\flat)$
  - Convex $=$ Ideal$(\sqsubseteq^\natural)$

Defining Continuous Functions on Ideal Completions

- Let f be a monotone function on an abstract basis
  - For all x and y, $x \preceq y$ implies $f(x) \sqsubseteq f(y)$
- There is a unique function g on the ideal completion such that
  - g is continuous
  - $g(\{a \mid a \preceq x\}) = f(x)$ for all x
- Function g is given by $g(A) = \bigsqcup_{x \in A} f(x)$
  - Proving that this limit exists is not easy!
- All powerdomain operations are defined using this method

Defining Continuous Functions on Ideal Completions

- Let f be a monotone function on an abstract basis
  - For all x and y, $x \preceq y$ implies $f(x) \sqsubseteq f(y)$
- There is a unique function g on the ideal completion such that
  - g is continuous
  - $g(\{a \,|\, a \preceq x\}) = f(x)$ for all x
- Function g is given by $g(A) = \bigsqcup_{x \in A} f(x)$
  - Proving that this limit exists is not easy!
- All powerdomain operations are defined using this method

Defining Continuous Functions on Ideal Completions

- Let f be a monotone function on an abstract basis
  - For all x and y, $x \preceq y$ implies $f(x) \sqsubseteq f(y)$
- There is a unique function g on the ideal completion such that
  - g is continuous
  - $g(\{a \mid a \preceq x\}) = f(x)$ for all x
- Function g is given by $g(A) = \bigsqcup_{x \in A} f(x)$
  - Proving that this limit exists is not easy!
- All powerdomain operations are defined using this method

Defining Continuous Functions on Ideal Completions

- Let f be a monotone function on an abstract basis
  - For all x and y, $x \preceq y$ implies $f(x) \sqsubseteq f(y)$
- There is a unique function g on the ideal completion such that
  - g is continuous
  - $g(\{a \mid a \preceq x\}) = f(x)$ for all x
- Function g is given by $g(A) = \bigsqcup_{x \in A} f(x)$
  - Proving that this limit exists is not easy!
- All powerdomain operations are defined using this method

Outline

Integration with Axiomatic Constructor Classes

- Axiomatic Constructor Classes in Isabelle/HOLCF
  - Joint work with John Matthews & Peter White, 2005
  - Formalized axiomatic classes for Functor and Monad
  - Defined resumption monad transformer
- Now extended to support powerdomains
  - Axiomatic class for powerdomains
  - Powerdomain operations use overloaded syntax
  - Can apply resumption monad transformer to powerdomains

Example Proofs in Isabelle

- The Haskell examples shown earlier have been formalized
  - With powerdomains, the properties are actually true!
- See it in action

Summary

- Powerdomains are well-suited for reasoning about nondeterminism in functional programs.
- You can do proofs about powerdomains right now in Isabelle/HOLCF.

- Future work
  - Proofs about parallel code, using monad transformers.