Complete partial orders

An (ω-*chain-*)*complete partial order*, *cpo*:

$$\boxed{\mathbf{D} = \langle D, \sqsubseteq, \perp \rangle}$$

- $\sqsubseteq \subseteq D \times D$ is a partial order on D such that each countable chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_i \sqsubseteq \ldots$ has the least upper bound $\bigsqcup_{i>0} d_i$ in D
- $\perp \in D$ is the least element w.r.t. $\sqsubseteq$

BTW: Equivalently: all countable *directed* subsets of D have lub's in D.
($\Delta \subseteq D$ is *directed* if for every $x, y \in \Delta$, there is $d \in \Delta$ with $x \sqsubseteq d$ and $y \sqsubseteq d$.)

BTW: It is *not* equivalent to require that *all* chains have lub's in D.
($C \subseteq D$ is a *chain* if for every $x, y \in C$, $x \sqsubseteq y$ or $y \sqsubseteq x$.)
But it is equivalent to require that *all* countable chains have lub's in D.

Examples

Examples	Non-examples	Comments
$\langle \mathcal{P}(X), \subseteq, \emptyset \rangle$	$\langle \mathcal{P}_{\mathit{fin}}(X), \subseteq, \emptyset \rangle$	$\mathcal{P}(X)$ is the set of all subsets, and $\mathcal{P}_{\mathit{fin}}(X)$ of all finite subsets of X
$\langle X \rightharpoonup Y, \subseteq, \emptyset_{X \rightharpoonup Y} \rangle$	$\langle X \to Y, \subseteq, ??? \rangle$	partial and total function spaces
$\langle \mathbf{Nat}^\infty, \leq, 0 \rangle$	$\langle \mathbf{Nat}, \leq, 0 \rangle$	$\mathbf{Nat}^\infty = \mathbf{Nat} \cup \{\omega\}$; $n \leq \omega$, for all $n \in \mathbf{Nat}$
$\langle (\mathbf{R}^+)^\infty, \leq, 0 \rangle$	$\langle (\mathbf{Q}^+)^\infty, \leq, 0 \rangle$	non-negative reals $\mathbf{R}^+$ and rationals $\mathbf{Q}^+$ with "infinity"
$\langle (\mathbf{R}^+)^{\leq a}, \leq, 0 \rangle$	$\langle (\mathbf{Q}^+)^{\leq a}, \leq, 0 \rangle$	their bounded versions
$\langle A^{\leq \omega}, \sqsubseteq, \varepsilon \rangle$	$\langle A^*, \sqsubseteq, \varepsilon \rangle$	$A^{\leq\omega} = A^* \cup A^\omega$ (finite and infinite strings of elements from A, including the empty string ε); $\sqsubseteq$ is the prefix ordering

Continuous functions

Given cpo's $\mathbf{D} = \langle D, \sqsubseteq, \perp \rangle$ and $\mathbf{D}' = \langle D', \sqsubseteq', \perp' \rangle$, a function $f\colon D \to D'$ is

- *monotone* if it preserves the ordering, i.e., for all $d_1, d_2 \in D$,

$$\boxed{d_1 \sqsubseteq d_2 \text{ implies } f(d_1) \sqsubseteq' f(d_2)}$$

- *continuous* if it preserves lub's of all countable chains, i.e., for each chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ in D,

$$\boxed{f(\bigsqcup_{i \geq 0} d_i) = \bigsqcup_{i \geq 0} f(d_i)}$$

- *strict* if it preserves the least element, i.e.,

$$\boxed{f(\perp) = \perp'}$$

BTW: Continuous functions are monotone; in general they need not be strict.

BTW: Monotone functions in general need not be continuous.

Some intuition?

Topology

Given a cpo $\mathbf{D} = \langle D, \sqsubseteq, \perp \rangle$, define a set $X \subseteq D$ to be *open* if

– if $d_1 \in X$ and $d_1 \sqsubseteq d_2$ then $d_2 \in X$

– if $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ is such that $\bigsqcup_{i \geq 0} d_i \in X$ then $d_i \in X$ for some $i \geq 0$.

This defines a topology on D:

– $\emptyset$ and D are open

– intersection of two open sets is open

– union of any family of open sets is open

Given two cpo's $\mathbf{D} = \langle D, \sqsubseteq, \perp \rangle$ and $\mathbf{D}' = \langle D', \sqsubseteq', \perp' \rangle$, a function $f : D \to D'$ is continuous if and only if it is continuous in the topological sense, i.e., for $X' \subseteq D'$ open in $\mathbf{D}'$, its co-image w.r.t. f, $f^{-1}(X') \subseteq D$ is open in $\mathbf{D}$.

More intuition?

Information theory

Think of a cpo $\mathbf{D} = \langle D, \sqsubseteq, \perp \rangle$ as an "information space".

- if $d_1 \sqsubseteq d_2$ then d_2 represents "more information" than d_1; $\perp$ is "no information"
- directed sets represent consistent sets of "information pieces"; their lub's represent "information" that can be derived from the "informations" in the set
- a function is monotone if it yields more information when given more information
- a function is continuous if it deals with information "bit-by-bit" very informal

For a set of elements X, consider the cpo $\langle \mathcal{P}(X), \supseteq, X \rangle$ of "informations" about the elements in X (a set $I \subseteq X$ represents the property — information — that holds for all the elements in I, and only for those elements).

Best intuition?

Partial functions

$$\langle X \rightharpoonup Y, \subseteq, \emptyset_{X \rightharpoonup Y} \rangle$$

- $\emptyset_{X \rightharpoonup Y}$ is nowhere defined
- given two partial functions $f, g \colon X \rightharpoonup Y$, $f \subseteq g$ if g is more defined than f, but when f is defined, g yields the same result
- given a directed set of partial functions $\mathcal{F} \subseteq X \rightharpoonup Y$, no two functions in $\mathcal{F}$ yield different results for the same argument; then $\bigsqcup \mathcal{F} = \bigcup \mathcal{F}$, which is a partial function in $X \rightharpoonup Y$
- a function $F \colon (X \rightharpoonup Y) \to (X' \rightharpoonup Y')$ is continuous, if $F(f)(x')$ (for $f \colon X \rightharpoonup Y$ and $x' \in X'$) depends only on a finite number of applications of f to arguments in X. Typical non-continuous functions:
  testing definedness, checking infinitely many values, ...

this is quite informal !

Fixed-point theorem

Fact: *Given a cpo* $\mathbf{D} = \langle D, \sqsubseteq, \perp\rangle$ *and a continuous function* $f: D \to D$, *there exists the least fixed-point* $\mathit{fix}(f) \in D$ *of* f, *i.e.,*

$$\mathit{fix}(\_): [D \to D] \to D$$

- $f(\mathit{fix}(f)) = \mathit{fix}(f)$
- *if* $f(d) = d$ *for some* $d \in D$ *then* $\mathit{fix}(f) \sqsubseteq d$

Proof:

Define $f^0(\perp) = \perp$, and $f^{i+1}(\perp) = f(f^i(\perp))$ for $i \geq 0$. This yields a chain:

$$f^0(\perp) \sqsubseteq f^1(\perp) \sqsubseteq \cdots \sqsubseteq f^i(\perp) \sqsubseteq f^{i+1}(\perp) \sqsubseteq \cdots$$

Put:

$$\boxed{\mathit{fix}(f) = \bigsqcup_{i\geq 0} f^i(\perp)}$$

- $f(\mathit{fix}(f)) = f(\bigsqcup_{i\geq 0} f^i(\perp)) = \perp \sqcup \bigsqcup_{i\geq 0} f(f^i(\perp)) = \bigsqcup_{i\geq 0} f^i(\perp) = \mathit{fix}(f)$
- Suppose $f(d) = d$ for some $d \in D$; then $f^i(\perp) \sqsubseteq d$ for $i \geq 0$. Thus $\mathit{fix}(f) = \bigsqcup_{i\geq 0} f^i(\perp) \sqsubseteq d$.

Proof techniques

Given a cpo $\mathbf{D} = \langle D, \sqsubseteq, \bot \rangle$ and a continuous function $f \colon D \to D$:

Fact: *For any $d \in D$, if $f(d) \sqsubseteq d$ then $\mathit{fix}(f) \sqsubseteq d$.*

Fixed-point induction

A property $P \subseteq D$ is *admissible* if it is preserved by lub's of all countable chains: for any chain $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$, if $d_i \in P$ for all $i \geq 0$ then also $\bigsqcup_{i \geq 0} d_i \in P$, and $\bot \in P$.

Fact: *For any admissible $P \subseteq D$ that is closed under f (i.e., if $d \in P$ then $f(d) \in P$)*

$$\boxed{\mathit{fix}(f) \in P}$$

Semantics of **while**

Recall the (original direct) semantic clause for **while**:

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!] = \mathit{fix}(\Phi)$$

where $\Phi\colon \mathbf{STMT} \to \mathbf{STMT}$ is given by $\Phi(F) = \mathit{cond}(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!];F, id_{\mathbf{State}})$.

Is $\mathbf{STMT}$ a cpo?

Is Φ continuous?

In this case we can easily check that indeed $\langle \mathbf{STMT}, \subseteq, \emptyset_{\mathbf{State} \rightharpoonup \mathbf{State}} \rangle$ is a cpo and $\Phi\colon \mathbf{STMT} \to \mathbf{STMT}$ is continuous.

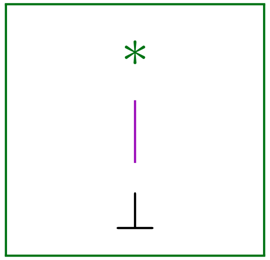
BUT: we do not want to have to check this each time we use a fixed-point definition!

Domain constructors

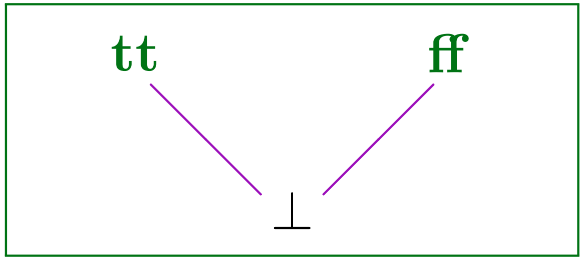
Basic domains

For any set X, $\mathbf{X}_\perp = \langle X_\perp, \sqsubseteq, \perp \rangle$ is a *flat cpo*, where $X_\perp = X \cup \{\perp\}$, $\perp$ is a new element, $\perp \sqsubseteq x$ for all $x \in X$ and otherwise $\sqsubseteq$ is trivial.

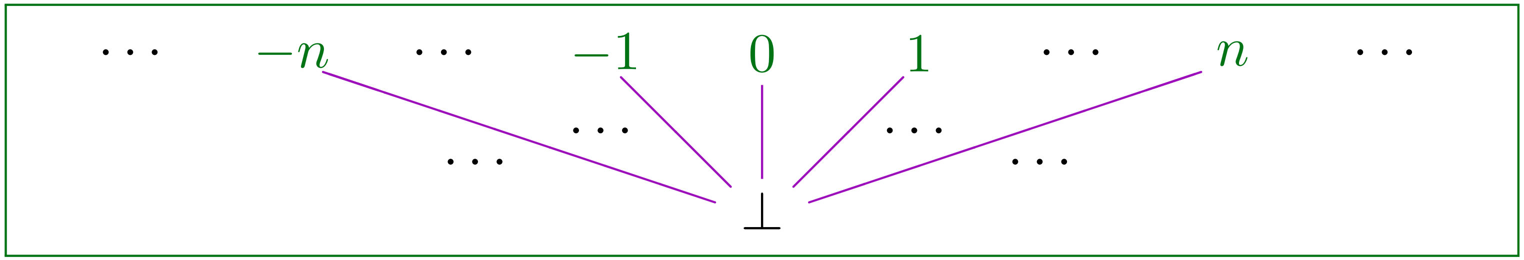
$\{*\}_\perp$:



$\mathbf{Bool}_\perp$:



$\mathbf{Int}_\perp$:



Fact: *Every monotone function defined on a flat cpo is continuous.*

For any cpo's $\mathbf{D}_1 = \langle D_1, \sqsubseteq_1, \perp_1 \rangle$ and $\mathbf{D}_2 = \langle D_2, \sqsubseteq_2, \perp_2 \rangle$:

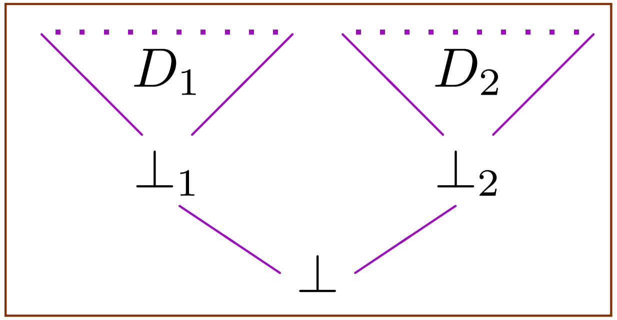
Product

Product of $\mathbf{D}_1$ and $\mathbf{D}_2$ is the following cpo:

$$\mathbf{D}_1 \times \mathbf{D}_2 = \langle D_1 \times D_2, \sqsubseteq, \langle \perp_1, \perp_2 \rangle \rangle$$

where for all $d_1, d_1' \in D_1$ and $d_2, d_2' \in D_2$, $\langle d_1, d_2 \rangle \sqsubseteq \langle d_1', d_2' \rangle$ if $d_1 \sqsubseteq_1 d_1'$ and $d_2 \sqsubseteq_2 d_2'$.

Sum



Disjoint sum of $\mathbf{D}_1$ and $\mathbf{D}_2$ is the following cpo:

$$\mathbf{D}_1 + \mathbf{D}_2 = \langle (D_1 \times \{1\}) \cup (D_2 \times \{2\}) \cup \{\perp\}, \sqsubseteq, \perp \rangle$$

where for $d_1, d_1' \in D_1$, $\langle d_1, 1 \rangle \sqsubseteq \langle d_1', 1 \rangle$ if $d_1 \sqsubseteq_1 d_1'$, for $d_2, d_2' \in D_2$, $\langle d_2, 2 \rangle \sqsubseteq \langle d_2', 2 \rangle$ if $d_2 \sqsubseteq_2 d_2'$, and for $d_1 \in D_1$, $d_2 \in D_2$, $\perp \sqsubseteq \langle d_1, 1 \rangle$ and $\perp \sqsubseteq \langle d_2, 2 \rangle$.

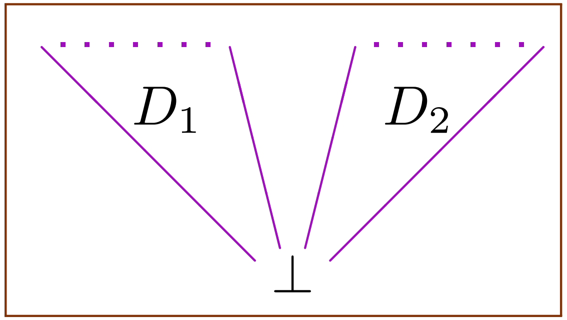
To avoid proliferation of bottoms:

Smashed product

Smashed product of $\mathbf{D}_1$ and $\mathbf{D}_2$ is the following cpo:

$$\mathbf{D}_1 \otimes \mathbf{D}_2 = \langle (D_1 \setminus \{\perp_1\}) \times (D_2 \setminus \{\perp_2\}) \cup \{\perp\}, \sqsubseteq, \perp \rangle$$

where for all non-bottom $d_1, d_1' \in D_1$ and $d_2, d_2' \in D_2$, $\langle d_1, d_2 \rangle \sqsubseteq \langle d_1', d_2' \rangle$ if $d_1 \sqsubseteq_1 d_1'$ and $d_2 \sqsubseteq_2 d_2'$, and $\perp \sqsubseteq \langle d_1, d_2 \rangle$.



Smashed sum

Smashed sum of $\mathbf{D}_1$ and $\mathbf{D}_2$ is the following cpo:

$$\mathbf{D}_1 \oplus \mathbf{D}_2 = \langle ((D_1 \setminus \{\perp_1\}) \times \{1\}) \cup ((D_2 \setminus \{\perp_2\}) \times \{2\}) \cup \{\perp\}, \sqsubseteq, \perp \rangle$$

where for all non-bottom $d_1, d_1' \in D_1$, $\langle d_1, 1 \rangle \sqsubseteq \langle d_1', 1 \rangle$ if $d_1 \sqsubseteq_1 d_1'$, for $d_2, d_2' \in D_2$, $\langle d_2, 2 \rangle \sqsubseteq \langle d_2', 2 \rangle$ if $d_2 \sqsubseteq_2 d_2'$, and $\perp \sqsubseteq \langle d_1, 1 \rangle$ and $\perp \sqsubseteq \langle d_2, 2 \rangle$.

Function spaces

Continuous-function space from $\mathbf{D}_1$ to $\mathbf{D}_2$ is the following cpo:

$$[\mathbf{D}_1 \to \mathbf{D}_2] = \langle [D_1 \to D_2], \sqsubseteq, \bot \rangle$$

where

- $[D_1 \to D_2]$ is the set of all *continuous* functions from D_1 to D_2
- for functions $f, g \colon D_1 \to D_2$, $f \sqsubseteq g$ if for each $d_1 \in D_1$, $f(d_1) \sqsubseteq_2 g(d_1)$
- $\bot(d_1) = \bot_2$ for each $d_1 \in D_1$.

$\sqsubseteq$ does not depend on the ordering on D_1

For any set X, *function space* from X to $\mathbf{D}_2$ is the following cpo:

$$(X \to \mathbf{D}_2) = \langle X \to D_2, \sqsubseteq, \bot \rangle$$

where $X \to D_2$ is the set of total functions from X to D_2 ordered by $\sqsubseteq$ as above.

Domain isomorphism

Cpo's $\mathbf{D}_1$ and $\mathbf{D}_2$ are *isomorphic*

$$\boxed{\mathbf{D}_1 \cong \mathbf{D}_2}$$

if there is a bijection between D_1 and D_2 which preserves and reflects the ordering.

Examples:

$$\mathbf{Bool}_\perp \cong \{*\}_\perp \oplus \{*\}_\perp$$

$$\langle X \rightharpoonup Y, \subseteq, \emptyset_{X \rightharpoonup Y} \rangle \cong \langle X \to Y_\perp, \sqsubseteq, \perp \rangle$$

Consider semantic domains up to isomorphism only

So, we can forget (boolean values and) partial functions !

Forgeting natural numbers is more difficult – but possible as well

BTW:

Informally:

- $\mathbf{D} \otimes \mathbf{D}'$ admits only "strict" (defined) elements in the pairs
- $\mathbf{D} \times \mathbf{D}'$ admits both "strict" and "undefined" ("unknown") elements in the pairs
- $\mathbf{D}_\perp$ makes all elements in $\mathbf{D}$ "strict"

Hence:

$$(\mathbf{D} \times \mathbf{D}')_\perp \cong \mathbf{D}_\perp \otimes \mathbf{D}'_\perp$$

$$\mathbf{D} + \mathbf{D}' \cong \mathbf{D}_\perp \oplus \mathbf{D}'_\perp$$

We also define:

$$\mathbf{D} \otimes_L \mathbf{D}' \cong \mathbf{D} \otimes \mathbf{D}'_\perp$$

admitting pairs with "strict" 1st elements (and 2nd elements "strict" or "udefined")

Define: $\mathbf{D} \oplus_L \mathbf{D}'$, $\mathbf{D} \otimes_R \mathbf{D}'$, $\mathbf{D} \oplus_R \mathbf{D}'$

Building continuous functions

- Every constant function is continuous
- Partial functions on sets, as used so far, can be replaced by (strict) continuous functions between flat domains; for instance, with a bit of abuse of notation:
  - $\mathit{ifte}_{\mathbf{D}} \in [\mathbf{Bool}_\perp \times \mathbf{D} \times \mathbf{D} \to \mathbf{D}]$ is given by:

$$\mathit{ifte}_{\mathbf{D}}(c,d,d') = \begin{cases} \mathit{ifte}_D(c,d,d') & \text{if } c \neq \perp \\ \perp_{\mathbf{D}} & \text{if } c = \perp \end{cases}$$

  - $\_ + \_ \in [\mathbf{Int}_\perp \times \mathbf{Int}_\perp \to \mathbf{Int}_\perp]$ is given by:

$$n + n' = \begin{cases} n + n' & \text{if } n \neq \perp \text{ and } n' \neq \perp \\ \perp & \text{if } n = \perp \text{ or } n' = \perp \end{cases}$$

More constructs

- function composition: $\_;\_ \in [[\mathbf{D}_1 \to \mathbf{D}_2] \times [\mathbf{D}_2 \to \mathbf{D}_3] \to [\mathbf{D}_1 \to \mathbf{D}_3]]$, i.e.:
  - composition of continuous functions is continuous
  - the composition function is continuous
- indexing:
  $lift^{\mathbf{I}} \in [[\mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}] \to [[\mathbf{I} \to \mathbf{D}_1] \times \ldots \times [\mathbf{I} \to \mathbf{D}_n] \to [\mathbf{I} \to \mathbf{D}]]]$, i.e.:
  - indexing a continuous function yields a continuous function
  - the indexing function is continuous
- Given a function $f\colon D_1 \times \ldots \times D_n \to D$, f is a continuous function from the product domain $\mathbf{D}_1 \times \ldots \times \mathbf{D}_n$ to $\mathbf{D}$ if and only if it is continuous w.r.t. each argument separately
  - this justifies the use of lambda-notation to build continuous functions:
    $\Lambda \in [[\mathbf{D}_0 \times \mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}] \to [\mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to [\mathbf{D}_0 \to \mathbf{D}]]]$

. . . and more

- continuous-function application is continuous: $\_(\_) \in [[\mathbf{D}_1 \to \mathbf{D}_2] \times \mathbf{D}_1 \to \mathbf{D}_2]$
- projections: $\pi_1 \in [\mathbf{D}_1 \times \mathbf{D}_2 \to \mathbf{D}_1]$ and $\pi_2 \in [\mathbf{D}_1 \times \mathbf{D}_2 \to \mathbf{D}_2]$
- (two-argument pairing, but how to write this sensibly?)
- injections: $\iota_1 \in [\mathbf{D}_1 \to \mathbf{D}_1 + \mathbf{D}_2]$ and $\iota_2 \in [\mathbf{D}_2 \to \mathbf{D}_1 + \mathbf{D}_2]$,
- domain checks: $\mathit{is\_in}_1 \in [\mathbf{D}_1 + \mathbf{D}_2 \to \mathbf{Bool}_\perp]$ and $\mathit{is\_in}_2 \in [\mathbf{D}_1 + \mathbf{D}_2 \to \mathbf{Bool}_\perp]$

also their smashed versions

- function pairing: $\langle \_, \_ \rangle \colon [[\mathbf{D} \to \mathbf{D}_1] \times [\mathbf{D} \to \mathbf{D}_2] \to [\mathbf{D} \to \mathbf{D}_1 \times \mathbf{D}_2]]$, where for $f \in [\mathbf{D} \to \mathbf{D}_1]$ and $g \in [\mathbf{D} \to \mathbf{D}_2]$, $\langle f, g \rangle = \lambda d{:}D.\langle f(d), g(d) \rangle$.
- function sum: $[\_, \_] \colon [[\mathbf{D}_1 \to \mathbf{D}] \times [\mathbf{D}_2 \to \mathbf{D}] \to [\mathbf{D}_1 + \mathbf{D}_2 \to \mathbf{D}]]$, where for $f \in [\mathbf{D}_1 \to \mathbf{D}]$ and $g \in [\mathbf{D}_2 \to \mathbf{D}]$, $[f, g](d) = \mathit{ifte}_{\mathbf{D}}(\mathit{is\_in}_1(d), f(d), g(d))$

- the least fixed-point operation $\mathit{fix}(\_) \in [[\mathbf{D} \to \mathbf{D}] \to \mathbf{D}]$
  - for $\mathbf{D} = [\mathbf{D}_1 \to \mathbf{D}_2]$, it follows that the least fixed-point of a continuous function on continuous functions is a continuous function...

Enough is enough...

Not all functions are continuous...

Enough functions are continuous...

Fixed-point equations

Elements of cpo's $d_1 \in D_1$, ..., $d_n \in D_n$ can be defined by writing (sets of) *fixed-point equations*

$$\begin{aligned} d_1 &= \Phi_1(d_1, \ldots, d_n) \\ &\cdots \\ d_n &= \Phi_n(d_1, \ldots, d_n) \end{aligned}$$

where $\Phi_1 \in [\mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}_1]$, ..., $\Phi_n \in [\mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}_n]$.

This defines $\langle d_1, \ldots, d_n \rangle$ as the least fixed-point of

$$\langle \Phi_1, \ldots, \Phi_n \rangle \in [\mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}_1 \times \ldots \times \mathbf{D}_n]$$

The continuous functions used in such definitions may be build using the basic functions and the ways of their composition as discussed so far.

Domain equations

$$\begin{aligned}
\mathbf{Int} &= \{0, 1, -1, 2, -2, \ldots\}_\perp \\
\mathbf{Bool} &= \{\mathbf{tt}, \mathbf{ff}\}_\perp \\
\mathbf{State} &= \mathbf{Var} \to \mathbf{Int} \\
\mathbf{EXP} &= [\mathbf{State} \to \mathbf{Int}] \\
\mathbf{BEXP} &= [\mathbf{State} \to \mathbf{Bool}] \\
\mathbf{STMT} &= [\mathbf{State} \to \mathbf{State}]
\end{aligned}$$

No problem!

Just use the operators to build cpo's as discussed above

BTW: $\mathbf{Int}^{??} = \mathbf{Int} \oplus \{??\}_\perp$

$\mathbf{Bool}^{??} = \mathbf{Bool} \oplus \{??\}_\perp$

If definitions of domains turn out to be recursive, use the successive approximation technique, as above for domain elements

Recursive domain equations

$$\boxed{\textbf{Stream} = A_\perp \otimes_L \textbf{Stream}}$$

$$\textbf{Stream} = (A_\perp \otimes_L \textbf{Stream}) \oplus \{\textbf{eof}\}_\perp$$

adds $\textbf{eof}, \langle a_1, \textbf{eof}\rangle, \langle a_1, \langle a_2, \textbf{eof}\rangle\rangle, \ldots$

$$\begin{aligned}
\textbf{Stream}^0 &= \{\perp\} \\
\textbf{Stream}^1 &= \{\perp \sqsubseteq \langle a_1, \perp\rangle\} \\
\textbf{Stream}^2 &= \{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, \langle a_2, \perp\rangle\rangle\} \\
&\ldots \\
\textbf{Stream}^n &= \{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, \langle a_2, \perp\rangle\rangle \sqsubseteq \cdots \sqsubseteq \langle a_1, \langle a_2, \langle \ldots, \langle a_n, \perp\rangle \ldots\rangle\rangle\rangle\} \\
&\ldots \\
\textbf{Stream} &= \textstyle\bigsqcup_{n \geq 0} \textbf{Stream}^n \\
&= \{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, \langle a_2, \perp\rangle\rangle \sqsubseteq \cdots \sqsubseteq \langle a_1, \langle a_2, \langle \ldots, \langle a_n, \perp\rangle \ldots\rangle\rangle\rangle \\
&\qquad \sqsubseteq \cdots \sqsubseteq \langle a_1, \langle a_2, \langle \ldots, \langle a_n, \langle \ldots\rangle\rangle \ldots\rangle\rangle\rangle\}
\end{aligned}$$

where all $a_1, a_2, \ldots, a_n, \ldots \in A$.

Recursive domain equations

$$\boxed{\mathbf{Stream} = A_\perp \otimes_L \mathbf{Stream}}$$

$\mathbf{Stream} = (A_\perp \otimes_L \mathbf{Stream}) \oplus \{\mathbf{eof}\}_\perp$
adds $\mathbf{eof}, \langle a_1, \mathbf{eof}\rangle, \langle a_1, a_2, \mathbf{eof}\rangle, \ldots$

$$\mathbf{Stream}^0 = \{\perp\}$$
$$\mathbf{Stream}^1 = \{\perp \sqsubseteq \langle a_1, \perp\rangle\}$$
$$\mathbf{Stream}^2 = \{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, a_2, \perp\rangle\}$$
$$\ldots$$
$$\mathbf{Stream}^n = \{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, a_2, \perp\rangle \sqsubseteq \cdots \sqsubseteq \langle a_1, a_2, \ldots, a_n, \perp\rangle\}$$
$$\ldots$$
$$\mathbf{Stream} = \bigsqcup_{n\geq 0} \mathbf{Stream}^n =$$
$$\{\perp \sqsubseteq \langle a_1, \perp\rangle \sqsubseteq \langle a_1, a_2, \perp\rangle \sqsubseteq \cdots \sqsubseteq \langle a_1, a_2, \ldots, a_n, \perp\rangle \sqsubseteq \cdots \sqsubseteq \langle a_1, a_2, \ldots, a_n, \ldots\rangle\}$$

where all $a_1, a_2, \ldots, a_n, \ldots \in A$.

Problems?

If definitions of domains turn out to be recursive,
use the successive approximation technique,
as above for domain elements

Really?

No problem?

Suppose we want to add (parameterless) procedures, which are named statements to be stored in states and used in call statements:

$$\mathbf{State} = \mathbf{Var} \to \mathbf{VAL} \quad \mathbf{VAL} = \mathbf{Int} + \mathbf{PROC} \quad \mathbf{PROC} = [\mathbf{State} \to \mathbf{State}]$$

Do such domains exist?

Reflexive domains

same trouble already with $\mathbf{D} \cong \mathbf{D} \to \mathbf{Bool}_\perp$

There is no (non-trivial) *set* that satisfies

$$\boxed{\mathbf{D} \cong \mathbf{D} \to \mathbf{D}}$$

Yet, any form of *self-application* (untyped procedure parameters, dynamic binding, etc) requires a semantic domain of this or similar form.

Models for untyped λ-calculus

In particular, this is necessary to model untyped λ-calculus, a formal untyped calculus where every term may be applied to any argument.

History, late 60s, 70s:

- Dana Scott (PRG/Oxford), 1971, 1976
- then others: Plotkin, Smyth, Stoy ...

Good naive solution

Naive denotational semantics

- Use standard set-theoretic domain constructors
- Never use "heavy" recursion, as involved in the reflexive domain definition.
- Use naive set-theoretic approximations and set-theoretic unions to solve domain equations.
- This works for well-typed langauges with a hierarchy of concepts and domains.

Solution

Scott-ery

- Limit the size of domains: require countable basis plus some technical conditions
- Use continuous functions only
- Define "domain of all domains" where all such domains can be interpreted
- Define continuous functions on this domain to interpret each of the domain constructors
- Write and solve domain equations as fixed-point equations in this domain

Models: $\mathbf{P}\omega$, $\mathbf{T}\omega$, *information systems, ...*