

Denotational Semantics

CS 152 (Spring 2024)

Harvard University

Thursday, February 8, 2024

Today, we learn to

- ▶ define and use denotational semantics
- ▶ model programs as partial functions from input stores to output stores
- ▶ find the fixed point of a function

Program models

- ▶ Operational model
- ▶ Denotational model

Program models

- ▶ Operational model (executions)
- ▶ Denotational model (mathematical functions)

Contrast...

$$c \sim c' \triangleq$$

$$\forall \sigma, \sigma'. \langle c, \sigma \rangle \Downarrow \sigma' \iff \langle c', \sigma \rangle \Downarrow \sigma'$$

VS

$$\{(\sigma, \sigma') \mid \langle c, \sigma \rangle \Downarrow \sigma'\} = \{(\sigma, \sigma') \mid \langle c', \sigma \rangle \Downarrow \sigma'\}$$

VS

$$\mathcal{C}[c] = \mathcal{C}[c']$$

Contrasting Meta Language

- ▶ operational semantics:
execution on an abstract machine defined by
stylized inductive sets
- ▶ denotational semantics:
language of mathematics

Compositionality

The denotation of a language construct is defined in terms of the denotations of its subphrases.

Advantages: compositionality supports

- ▶ substitution of semantically equivalent phrases,
- ▶ structural induction,
- ▶ analysis and evaluation in relative isolation.

Programs as functions

Programs as functions

For a program c , we write $\mathcal{C}[[c]]$ for the *denotation* of c , that is, the mathematical function that c represents:

$$\mathcal{C}[[c]] : \mathbf{Store} \rightarrow \mathbf{Store}.$$

Programs as functions

For a program c , we write $\mathcal{C}[[c]]$ for the *denotation* of c , that is, the mathematical function that c represents:

$$\mathcal{C}[[c]] : \mathbf{Store} \rightarrow \mathbf{Store}.$$

We write $\mathcal{C}[[c]]\sigma$ for the result of applying this function to the store σ .

Expressions as functions

Expressions as functions

Similarly, $\mathcal{A}[[a]]$ and $\mathcal{B}[[b]]$ are denotations for arithmetic and boolean expressions.

$$\mathcal{A}[[a]] : \mathbf{Store} \rightarrow \mathbf{Int}$$

$$\mathcal{B}[[b]] : \mathbf{Store} \rightarrow \{\mathbf{true}, \mathbf{false}\}$$

Expressions as functions

Note that $\mathcal{A}[[a]]$ and $\mathcal{B}[[b]]$ are total functions.

$$\mathcal{A}[[a]] : \mathbf{Store} \rightarrow \mathbf{Int}$$

$$\mathcal{B}[[b]] : \mathbf{Store} \rightarrow \{\mathbf{true}, \mathbf{false}\}$$

Arithmetic expressions

$$\mathcal{A}[[n]] = \{(\sigma, n)\}$$

Arithmetic expressions

$$\mathcal{A}[[n]] = \{(\sigma, n)\}$$

$$\mathcal{A}[[x]] = \{(\sigma, \sigma(x))\}$$

Arithmetic expressions

$$\mathcal{A}[[n]] = \{(\sigma, n)\}$$

$$\mathcal{A}[[x]] = \{(\sigma, \sigma(x))\}$$

$$\begin{aligned} \mathcal{A}[[a_1 + a_2]] = \{ & (\sigma, n) \mid (\sigma, n_1) \in \mathcal{A}[[a_1]] \\ & \wedge (\sigma, n_2) \in \mathcal{A}[[a_2]] \\ & \wedge n = n_1 + n_2 \} \end{aligned}$$

Arithmetic expressions

$$\mathcal{A}[[n]] = \{(\sigma, n)\}$$

$$\mathcal{A}[[x]] = \{(\sigma, \sigma(x))\}$$

$$\begin{aligned}\mathcal{A}[[a_1 + a_2]] = \{(\sigma, n) \mid & (\sigma, n_1) \in \mathcal{A}[[a_1]] \\ & \wedge (\sigma, n_2) \in \mathcal{A}[[a_2]] \\ & \wedge n = n_1 + n_2\}\end{aligned}$$

$$\begin{aligned}\mathcal{A}[[a_1 \times a_2]] = \{(\sigma, n) \mid & (\sigma, n_1) \in \mathcal{A}[[a_1]] \\ & \wedge (\sigma, n_2) \in \mathcal{A}[[a_2]] \\ & \wedge n = n_1 \times n_2\}\end{aligned}$$

Boolean expressions

$$\mathcal{B}[\mathbf{true}] = \{(\sigma, \mathbf{true})\}$$

$$\mathcal{B}[\mathbf{false}] = \{(\sigma, \mathbf{false})\}$$

Boolean expressions

$$\mathcal{B}[\mathbf{true}] = \{(\sigma, \mathbf{true})\}$$

$$\mathcal{B}[\mathbf{false}] = \{(\sigma, \mathbf{false})\}$$

$$\begin{aligned} \mathcal{B}[a_1 < a_2] = & \{(\sigma, \mathbf{true}) \mid (\sigma, n_1) \in \mathcal{A}[a_1] \\ & \wedge (\sigma, n_2) \in \mathcal{A}[a_2] \wedge n_1 < n_2\} \\ & \cup \{(\sigma, \mathbf{false}) \mid (\sigma, n_1) \in \mathcal{A}[a_1] \\ & \wedge (\sigma, n_2) \in \mathcal{A}[a_2] \\ & \wedge n_1 \geq n_2\} \end{aligned}$$

Denotations of some programs

Denotations of some programs

$$\mathcal{C}[\mathbf{skip}] = \{(\sigma, \sigma)\}$$

$$\mathcal{C}[\![x := a]\!] = \{(\sigma, \sigma[x \mapsto n]) \mid (\sigma, n) \in \mathcal{A}[\![a]\!]\}$$

$$\mathcal{C}[[c_1; c_2]] = \{(\sigma, \sigma') \mid \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c_1]] \wedge (\sigma'', \sigma') \in \mathcal{C}[[c_2]])\}$$

Composition of relations

Suppose $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$.

The *composition* of relations $R_2 \circ R_1 \subseteq A \times C$ is

$$R_2 \circ R_1 = \{(a, c) \mid \exists b \in B. (a, b) \in R_1 \wedge (b, c) \in R_2\}.$$

Composition of relations

We have $\mathcal{C}[[c_1; c_2]] = \mathcal{C}[[c_2]] \circ \mathcal{C}[[c_1]]$, where \circ is the composition of relations.

Definition of a function

A function is a set of input-output pairs s.t. each input has a unique output.

if b then c_1 else c_2

The mathematical function represented by
if b then c_1 else c_2 is the set

$$\begin{aligned} \mathcal{C}[\text{if } b \text{ then } c_1 \text{ else } c_2] = & \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[b] \\ & \wedge (\sigma, \sigma') \in \mathcal{C}[c_1]\} \cup \\ & \{(\sigma, \sigma') \mid (\sigma, \mathbf{false}) \in \mathcal{B}[b] \\ & \wedge (\sigma, \sigma') \in \mathcal{C}[c_2]\} \end{aligned}$$

C [[while b do c]]

$\mathcal{C}[\text{while } b \text{ do } c]$

$$\begin{aligned} \mathcal{C}[\text{while } b \text{ do } c] = & \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[b]\} \cup \\ & \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[b] \\ & \wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[c] \\ & \wedge (\sigma'', \sigma') \in \mathcal{C}[\text{while } b \text{ do } c])\} \end{aligned}$$

Computing $f = \mathcal{C}[\mathbf{while} \ b \ \mathbf{do} \ c]$

So far we only have a recursive equation

$$f = \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[b]\} \cup \\ \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[b] \\ \wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[c] \wedge (\sigma'', \sigma') \in f)\}$$

What is f , as a subset $\mathbf{Store} \times \mathbf{Store}$?

A simpler example

A simpler example

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x - 1) + 2x - 1 & \text{otherwise} \end{cases} \quad (1)$$

$$f_0 = \emptyset$$

$$f_1 = \begin{cases} 0 & \text{if } x = 0 \\ f_0(x - 1) + 2x - 1 & \text{otherwise} \end{cases}$$
$$= \{(0, 0)\}$$

$$f_2 = \begin{cases} 0 & \text{if } x = 0 \\ f_1(x - 1) + 2x - 1 & \text{otherwise} \end{cases}$$
$$= \{(0, 0), (1, 1)\}$$

$$f_3 = \begin{cases} 0 & \text{if } x = 0 \\ f_2(x - 1) + 2x - 1 & \text{otherwise} \end{cases}$$
$$= \{(0, 0), (1, 1), (2, 4)\}$$

⋮

Consider this higher-order function F :

$$F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$$
$$F(g) = g'$$

where

$$g'(x) = \begin{cases} 0 & \text{if } x = 0 \\ g(x - 1) + 2x - 1 & \text{otherwise} \end{cases}$$

$$F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$$
$$F(g) = g'$$

where

$$g'(x) = \begin{cases} 0 & \text{if } x = 0 \\ g(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

- ▶ E.g. $F(\emptyset) = \{(0, 0)\}$,
 $F(\{(1, 0)\}) = \{(0, 0), (2, 3)\}$, and
 $F(\{(3, 1), (0, 1)\}) = \{(0, 0), (4, 8), (1, 2)\}$.
- ▶ Note, however, that $F(f) = f$ for $f(x) = x^2$.
- ▶ In other words, f is a *fixed point* of F .

$$\begin{aligned} f &= \text{fix}(F) \\ &= f_0 \cup f_1 \cup f_2 \cup f_3 \cup \dots \\ &= \emptyset \cup F(\emptyset) \cup F(F(\emptyset)) \cup F(F(F(\emptyset))) \cup \dots \\ &= \bigcup_{i \geq 0} F^i(\emptyset) \end{aligned}$$

Fixed-point semantics for loops

$$F_{b,c} : (\mathbf{Store} \rightarrow \mathbf{Store}) \rightarrow (\mathbf{Store} \rightarrow \mathbf{Store})$$
$$F_{b,c}(f) = \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup$$
$$\{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]]$$
$$\wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c]]$$
$$\wedge (\sigma'', \sigma') \in f)\}$$

$$\begin{aligned} \mathcal{C}[\text{while } b \text{ do } c] &= \bigcup_{i \geq 0} F_{b,c}^i(\emptyset) \\ &= \emptyset \cup F_{b,c}(\emptyset) \cup F_{b,c}(F_{b,c}(\emptyset)) \\ &\quad \cup F_{b,c}(F_{b,c}(F_{b,c}(\emptyset))) \cup \dots \\ &= \text{fix}(F_{b,c}) \end{aligned}$$

Let's consider an example:

while $foo < bar$ **do** $foo := foo + 1$.

Here $b = foo < bar$ and $c = foo := foo + 1$.

$$\begin{aligned} F_{b,c}(\emptyset) &= \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup \\ &\quad \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\ &\quad \wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c]] \wedge (\sigma'', \sigma') \in \emptyset)\} \\ &= \{(\sigma, \sigma) \mid \sigma(\mathbf{foo}) \geq \sigma(\mathbf{bar})\} \end{aligned}$$

$$\begin{aligned}
F_{b,c}^2(\emptyset) &= \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup \\
&\quad \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\
&\quad \wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c]] \wedge (\sigma'', \sigma') \in F_{b,c}(\emptyset))\} \\
&= \{(\sigma, \sigma) \mid \sigma(\mathbf{foo}) \geq \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 1]) \mid \sigma(\mathbf{foo}) < \sigma(\mathbf{bar}) \\
&\quad \wedge \sigma(\mathbf{foo}) + 1 \geq \sigma(\mathbf{bar})\}
\end{aligned}$$

But $\sigma(\mathbf{foo}) < \sigma(\mathbf{bar}) \wedge \sigma(\mathbf{foo}) + 1 \geq \sigma(\mathbf{bar})$ if and only if $\sigma(\mathbf{foo}) + 1 = \sigma(\mathbf{bar})$ so we get:

$$\begin{aligned}
&= \{(\sigma, \sigma) \mid \sigma(\mathbf{foo}) \geq \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 1]) \mid \sigma(\mathbf{foo}) + 1 = \sigma(\mathbf{bar})\}
\end{aligned}$$

$$\begin{aligned}
F_{b,c}^3(\emptyset) &= \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup \\
&\quad \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c]] \\
&\quad \wedge (\sigma'', \sigma') \in F_{b,c}^2(\emptyset))\} \\
&= \{(\sigma, \sigma) \mid \sigma(\mathbf{foo}) \geq \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 1]) \mid \sigma(\mathbf{foo}) + 1 = \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 2]) \mid \sigma(\mathbf{foo}) + 2 = \sigma(\mathbf{bar})\}
\end{aligned}$$

$$\begin{aligned}
F_{b,c}^4(\emptyset) &= \{(\sigma, \sigma) \mid \sigma(\mathbf{foo}) \geq \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 1]) \mid \sigma(\mathbf{foo}) + 1 = \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 2]) \mid \sigma(\mathbf{foo}) + 2 = \sigma(\mathbf{bar})\} \cup \\
&\quad \{(\sigma, \sigma[\mathbf{foo} \mapsto \sigma(\mathbf{foo}) + 3]) \mid \sigma(\mathbf{foo}) + 3 = \sigma(\mathbf{bar})\}
\end{aligned}$$

$$\begin{aligned} \mathcal{C}[\text{while } \text{foo} < \text{bar} \text{ do } \text{foo} := \text{foo} + 1] = \\ \{(\sigma, \sigma) \mid \sigma(\text{foo}) \geq \sigma(\text{bar})\} \cup \\ \{(\sigma, \sigma[\text{foo} \mapsto \sigma(\text{foo}) + n]) \mid \sigma(\text{foo}) + n = \sigma(\text{bar}) \wedge n \geq 1\} \end{aligned}$$

Exercise

Let $w = \mathbf{while\ } b \mathbf{\ do\ } c$. Prove

$$\mathcal{C}[[w]] = \mathcal{C}[[\mathbf{if\ } b \mathbf{\ then\ } c; w \mathbf{\ else\ skip}]]$$

.

Unfolding

Let $w = \mathbf{while } b \mathbf{ do } c$. Prove

$$\mathcal{C}[[w]] = \mathcal{C}[[\mathbf{if } b \mathbf{ then } c; w \mathbf{ else skip}]]$$

.

Unfolding

$$F_{b,c} : (\mathbf{Store} \rightarrow \mathbf{Store}) \rightarrow (\mathbf{Store} \rightarrow \mathbf{Store})$$
$$F_{b,c}(f) = \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup$$
$$\{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]]$$
$$\wedge \exists \sigma''. ((\sigma, \sigma'') \in \mathcal{C}[[c]]$$
$$\wedge (\sigma'', \sigma') \in f)\}$$

Unfolding

$$F_{b,c} : (\mathbf{Store} \rightarrow \mathbf{Store}) \rightarrow (\mathbf{Store} \rightarrow \mathbf{Store})$$
$$F_{b,c}(f) = \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup$$
$$\{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]]$$
$$\wedge (\sigma, \sigma') \in f \circ \mathcal{C}[[c]]\}$$

Unfolding

$$F_{b,c}(\mathcal{C}[[w]]) = \{(\sigma, \sigma) \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]]\} \cup \\ \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\ \wedge (\sigma, \sigma') \in \mathcal{C}[[w]] \circ \mathcal{C}[[c]]\}$$

Unfolding

$$F_{b,c}(\mathcal{C}[[w]]) = \{(\sigma, \sigma') \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]] \\ \wedge (\sigma, \sigma') \in \mathcal{C}[[\mathbf{skip}]]\} \cup \\ \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\ \wedge (\sigma, \sigma') \in \mathcal{C}[[w]] \circ \mathcal{C}[[c]]\}$$

Unfolding

$$F_{b,c}(\mathcal{C}[[w]]) = \{(\sigma, \sigma') \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]] \\ \wedge (\sigma, \sigma') \in \mathcal{C}[[\mathbf{skip}]]\} \cup \\ \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\ \wedge (\sigma, \sigma') \in \mathcal{C}[[c; w]]\}$$

Unfolding

$$\begin{aligned} F_{b,c}(\mathcal{C}[[w]]) &= \{(\sigma, \sigma') \mid (\sigma, \mathbf{false}) \in \mathcal{B}[[b]] \\ &\quad \wedge (\sigma, \sigma') \in \mathcal{C}[[\mathbf{skip}]]\} \cup \\ &\quad \{(\sigma, \sigma') \mid (\sigma, \mathbf{true}) \in \mathcal{B}[[b]] \\ &\quad \wedge (\sigma, \sigma') \in \mathcal{C}[[c; w]]\} \\ &= \mathcal{C}[[\mathbf{if } b \mathbf{ then } c; w \mathbf{ else skip}]] \end{aligned}$$

Unfolding

$$\begin{aligned} C[[w]] &= \\ F_{b,c}(C[[w]]) &= \\ C[[\mathbf{if } b \mathbf{ then } c; w \mathbf{ else skip}]] & \end{aligned}$$

Why/when can we take the least fixed point?

Domain Theory

Complete Partial Order (cpo)

A partial order which has a least upper bound $\bigsqcup_{n \in \omega} d_n$ in D of any ω -chain (infinite increasing chain) $d_0 \sqsubseteq d_1 \sqsubseteq \dots \sqsubseteq d_n \sqsubseteq \dots$ of elements of D .

The partial function **Store** \rightarrow **Store** is a cpo, where the order is defined by as “less defined/partial or equal to” or “approximates”.

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$ (yes)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$ (yes)
- ▶ $([0, 1), \leq)$

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$ (yes)
- ▶ $([0, 1), \leq)$ (no)

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$ (yes)
- ▶ $([0, 1), \leq)$ (no)
- ▶ $(S, =)$

cpo Examples and non-Examples

- ▶ an arbitrary finite partial order (yes)
- ▶ $(\mathcal{P}^S, \subseteq)$ (yes)
- ▶ (\mathbb{N}, \leq) (no)
- ▶ $(\mathbb{N} \cup \infty, \leq)$ (yes)
- ▶ $([0, 1], \leq)$ (yes)
- ▶ $([0, 1), \leq)$ (no)
- ▶ $(S, =)$ (yes)

Store \rightarrow **Store cpo**

Store \rightarrow **Store**

or

Store \rightarrow **Store _{\perp}**

(notion of “less partial”)

Monotonicity

A function $f : D \rightarrow E$ between cpo's D and E is monotonic iff

$$\forall d, d' \in D. d \sqsubseteq d' \longrightarrow f(d) \sqsubseteq f(d').$$

The higher-order function $F_{b,c}$ defining **while** b **do** c is monotonic.

Continuity

A function is *continuous* iff it is monotonic and for all chains $d_0 \sqsubseteq d_1 \sqsubseteq \dots \sqsubseteq d_n \sqsubseteq \dots$ in D we have

$$\bigsqcup_{n \in \omega} f(d_n) = f(\bigsqcup_{n \in \omega} d_n).$$

The higher-order function $F_{b,c}$ defining **while** b **do** c is continuous.

.

cpo with bottom

A cpo which has a least element \perp , that is an element which is less than every other element.

The cpo **Store** \rightarrow **Store** has a least element \perp : the partial function defined nowhere.

Fixed Point Theorem

Let $f : D \rightarrow D$ be a continuous function on D a cpo with bottom \perp . Define

$$\text{fix}(f) = \bigsqcup_{n \in \omega} f^n(\perp).$$

Then $\text{fix}(f)$ is a least fixed point of f .

Proof $\text{fix}(f)$ is a fixed point of f

$$\begin{aligned} f\left(\bigsqcup_{n \in \omega} f^n(\perp)\right) &= \bigsqcup_{n \in \omega} f(f^n(\perp)) \text{ by continuity} \\ &= \bigsqcup_{n \in \omega} f^{n+1}(\perp) \text{ applying } f \\ &= \bigsqcup_{n=1,2,\dots} f^n(\perp) \text{ reindexing} \\ &= \perp \sqcup \bigsqcup_{n=1,2,\dots} f^n(\perp) \text{ by def. of } \perp \\ &= \bigsqcup_{n \in \omega} f^n(\perp) \text{ shows } f(\text{fix}(f)) = \text{fix}(f) \end{aligned}$$

Proof $\text{fix}(f)$ is the least prefixed point of f

Suppose y a prefixed point of f , def. $f(y) \sqsubseteq y$.

$\perp \sqsubseteq y$ by def. of \perp

$f(\perp) \sqsubseteq f(y) \sqsubseteq y$ taking f 's by monotonicity

$f^n(\perp) \sqsubseteq y$ inductively, for all $n \geq 0$

$\bigsqcup_{n \in \omega} f^n(\perp) \sqsubseteq y$ y must be at least as large