Denotational Semantics

12 lectures for Part II CST 2011/12

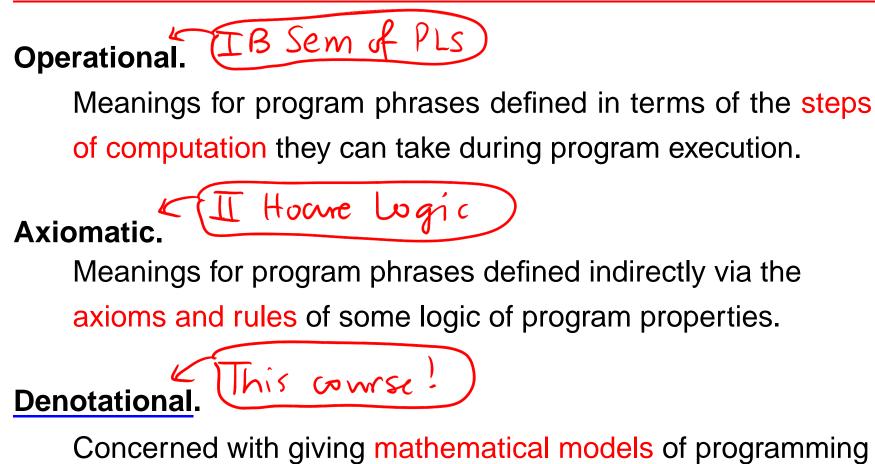
Andrew Pitts

Course web page:

http://www.cl.cam.ac.uk/teaching/1112/DenotSem/

copies of slides

Styles of formal semantics



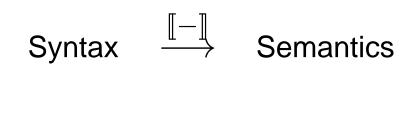
languages. Meanings for program phrases defined abstractly as elements of some suitable mathematical structure.

Why do we care?

- Rigour.
 - ... specification of programming languages
 - ... justification of program transformations
- Insight.
 - ... generalisations of notions computability
 - ... higher-order functions
 - ... data structures

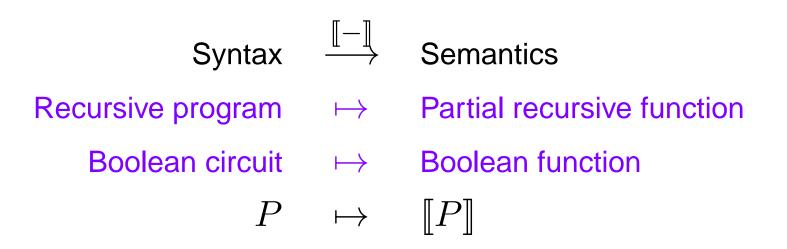
- Feedback into language design.
 - ... continuations
 - ... monads
- Reasoning principles.
 - ... Scott induction
 - ... Logical relations
 - ... Co-induction

Basic idea of denotational semantics



$P \quad \mapsto \quad \llbracket P \rrbracket$

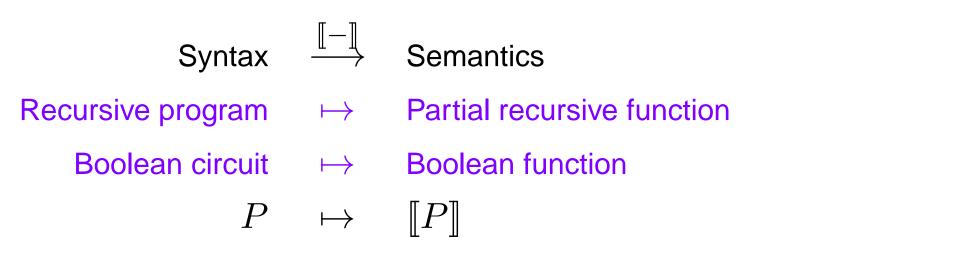
Basic idea of denotational semantics



Characteristic features of a denotational semantics

- Each phrase (= part of a program), P, is given a denotation,
 [P] a mathematical object representing the contribution of P to the meaning of any complete program in which it occurs.
- The denotation of a phrase is determined just by the denotations of its subphrases (one says that the semantics is compositional).

Basic idea of denotational semantics



Concerns:

- Abstract models (*i.e.* implementation/machine independent).
 Ist "3 rd of course" domain theory"
- Compositionality.

~> 2nd "3rd of course "PCF"

• Relationship to computation (*e.g.* operational semantics). \rightarrow last $\sqrt{3}$ rd of course IMP⁻ syntax

Arithmetic expressions

 $A \in \mathbf{Aexp} ::= \underline{n} \mid L \mid A + A \mid \dots$

where n ranges over *integers* and L over a specified set of *locations* L

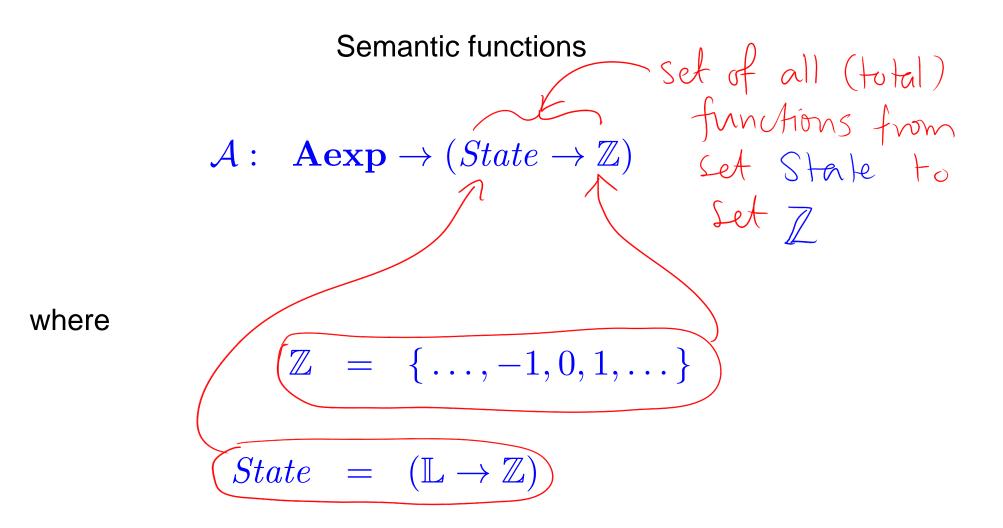
Boolean expressions

 $B \in \mathbf{Bexp} \quad ::= \quad \mathbf{true} \mid \mathbf{false} \mid A = A \mid \dots \\ \mid \quad \neg B \mid \dots$

Commands

 $C \in \mathbf{Comm} \quad ::= \quad \mathbf{skip} \quad | \quad L := A \quad | \quad C; C$ $| \quad \mathbf{if} \ B \mathbf{then} \ C \mathbf{else} \ C$

Basic example of denotational semantics (II)



Basic example of denotational semantics (II)

Semantic functions

 $\mathcal{A}: \quad \mathbf{Aexp} \to (State \to \mathbb{Z})$ $\mathcal{B}: \quad \mathbf{Bexp} \to (State \to \mathbb{B})$

where

$$\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$$
$$\mathbb{B} = \{true, false\}$$
$$State = (\mathbb{L} \to \mathbb{Z})$$

Basic example of denotational semantics (II)

Semantic functions

$$\mathcal{A}: \operatorname{Aexp} \to (\operatorname{State} \to \mathbb{Z})$$

$$\mathcal{B}: \operatorname{Bexp} \to (\operatorname{State} \to \mathbb{B})$$

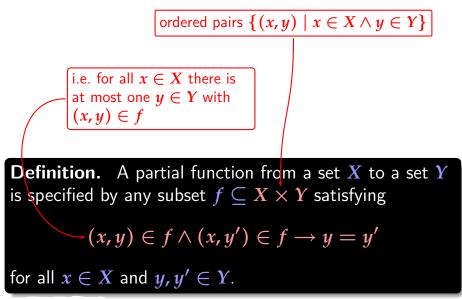
$$\mathcal{C}: \operatorname{Comm} \to (\operatorname{State} \to \operatorname{State})$$
where
$$\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$$

$$\mathbb{B} = \{\operatorname{true}, \operatorname{false}\}$$

$$\operatorname{State} = (\mathbb{L} \to \mathbb{Z})$$

$$\operatorname{State} = (\mathbb{L} \to \mathbb{Z})$$

Partial functions



Basic example of denotational semantics (III)

Semantic function \mathcal{A}

 $\mathcal{A}[\![\underline{n}]\!] = \lambda s \in State.\,n$

 $\mathcal{A}\llbracket L \rrbracket = \lambda s \in State. \, s(L)$

 $\mathcal{A}\llbracket A_1 + A_2 \rrbracket = \lambda s \in State. \, \mathcal{A}\llbracket A_1 \rrbracket(s) + \mathcal{A}\llbracket A_2 \rrbracket(s)$

Basic example of denotational semantics (IV)

Semantic function \mathcal{B}

 $\mathcal{B}\llbracket \mathbf{true} \rrbracket = \lambda s \in State. true$ $\mathcal{B}\llbracket \mathbf{false} \rrbracket = \lambda s \in State. false$ $\mathcal{B}\llbracket A_1 = A_2 \rrbracket = \lambda s \in State. eq \left(\mathcal{A}\llbracket A_1 \rrbracket (s), \mathcal{A}\llbracket A_2 \rrbracket (s) \right)$ where $eq(a, a') = \begin{cases} true & \text{if } a = a' \\ false & \text{if } a \neq a' \end{cases}$

Basic example of denotational semantics (V)

Semantic function C

 $\llbracket skip \rrbracket = \lambda s \in State.s$

NB: From now on the names of semantic functions are omitted!

A simple example of compositionality

Given partial functions $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ and a function $\llbracket B \rrbracket : State \rightarrow \{true, false\}$, we can define

$\llbracket \mathbf{if} \ B \ \mathbf{then} \ C \ \mathbf{else} \ C' \rrbracket = \\\lambda s \in State. \ if \left(\llbracket B \rrbracket(s), \llbracket C \rrbracket(s), \llbracket C' \rrbracket(s) \right)$

where

$$if(b, x, x') = \begin{cases} x & \text{if } b = true \\ x' & \text{if } b = false \end{cases}$$

Basic example of denotational semantics (VI)

Semantic function \mathcal{C}

$\llbracket L := A \rrbracket = \lambda s \in State. \ \lambda \ell \in \mathbb{L}. \ if \left(\ell = L, \llbracket A \rrbracket(s), s(\ell) \right)$

Denotational semantics of sequential composition

Denotation of sequential composition C; C' of two commands

 $\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$

given by composition of the partial functions from states to states $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \longrightarrow State$ which are the denotations of the commands.

Denotational semantics of sequential composition

Denotation of sequential composition C; C' of two commands

 $\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$

given by composition of the partial functions from states to states $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ which are the depotations of the commands.

[[c']([c](s)) is undefined if • either [c](s) is undefined

• or $\mathbb{C}\mathcal{I}(s) = s'$, say, and $\mathbb{C}'\mathcal{I}(s')$ is undefined.

Denotational semantics of sequential composition

Denotation of sequential composition C; C' of two commands

 $\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$

given by composition of the partial functions from states to states $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ which are the denotations of the commands.

Cf. operational semantics of sequential composition:

$$\frac{C, s \Downarrow s' \quad C', s' \Downarrow s''}{C; C', s \Downarrow s''}$$

$\llbracket \mathbf{while} \ B \ \mathbf{do} \ C \rrbracket$

CE Comm ::= | while B do C

$\llbracket \mathbf{while} \ B \ \mathbf{do} \ C \rrbracket$

Operational semantics of while-loops (while B doc, s) -> < if B then C; (while B do C) else skip, S> Suggests looking for a denotation [[while 13 do C]] Satistying [[while B do c]] = [[if B then C; (while B doc) else Skip]

Fixed point property of [while B do C]

 $\begin{bmatrix} \mathbf{while} \ B \ \mathbf{do} \ C \end{bmatrix} = f_{\llbracket B \rrbracket, \llbracket C \rrbracket}(\llbracket \mathbf{while} \ B \ \mathbf{do} \ C \rrbracket)$ where, for each $b : State \to \{true, false\}$ and $c : State \to State$, we define $f_{b,c} : (State \to State) \to (State \to State)$ as $f_{b,c} = \lambda w \in (State \to State). \ \lambda s \in State.$ if(b(s), w(c(s)), s).

Fixed point property of [while B do C]

 $\begin{bmatrix} \mathbf{while} \ B \ \mathbf{do} \ C \end{bmatrix} = f_{\llbracket B \rrbracket, \llbracket C \rrbracket}(\llbracket \mathbf{while} \ B \ \mathbf{do} \ C \rrbracket)$ where, for each $b : State \to \{true, false\}$ and $c : State \to State$, we define $f_{b,c} : (State \to State) \to (State \to State)$ as $f_{b,c} = \lambda w \in (State \to State) . \lambda s \in State.$ if(b(s), w(c(s)), s).

- Why does $w = f_{\llbracket B \rrbracket, \llbracket C \rrbracket}(w)$ have a solution?
- What if it has several solutions—which one do we take to be $\llbracket while B do C \rrbracket$?

$\llbracket while X > 0 do (Y := X * Y; X := X - 1) \rrbracket$

Let

- State $\stackrel{\text{def}}{=} (\mathbb{L} \to \mathbb{Z})$ integer assignments to locations $D \stackrel{\text{def}}{=} (State \rightharpoonup State)$ partial functions on states
- For $[\![while X > 0 \text{ do } Y := X * Y ; X := X 1]\!] \in D$ we seek a minimal solution to w = f(w), where $f : D \to D$ is defined by:

$$\begin{split} f(w) \big([X \mapsto x, Y \mapsto y] \big) & \text{if } x \leq 0 \\ &= \begin{cases} [X \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w \big([X \mapsto x - 1, Y \mapsto x * y] \big) & \text{if } x > 0. \end{cases} \end{split}$$

$$D \stackrel{\mathrm{def}}{=} (State \rightharpoonup State)$$

- Partial order \sqsubseteq on D:
 - $w \sqsubseteq w'$ iff for all $s \in State$, if w is defined at s then so is w' and moreover w(s) = w'(s).
 - iff the graph of w is included in the graph of w'.
- Least element $\bot \in D$ w.r.t. \sqsubseteq :
 - \perp = totally undefined partial function
 - = partial function with empty graph

(satisfies $\perp \sqsubseteq w$, for all $w \in D$).

$$f: D \rightarrow D$$
 is given by
 $f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & if x \leq 0 \\ w [x \mapsto x - 1, Y \mapsto x \neq y] & if x > 0 \end{cases}$
Want to find weD s.t. $w = f(w)$

Define
$$W_0 = 1$$
, $W_1 = f(1)$, $W_2 = f(f(1))$, etc.

$$f: D \rightarrow D \text{ is given by} \qquad f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w [x \mapsto x, Y \mapsto y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $w = f(w)$
Define $w_0 = \bot$, $w_1 = f(\bot)$, $w_2 = f(f(\bot))$, etc.
$$w_1[x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w_1[x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \end{cases}$$
undefined if $x \geq 1$

$$f: D \rightarrow D \text{ is given by} \qquad f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w[x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $w = f(w)$
Define $w_0 = \bot$, $w_1 = f(\bot)$, $w_2 = f(f(\bot))$, etc.
$$w_2[x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto 0, Y \mapsto y] & \text{if } x = 1 \\ undefined & \text{if } x > 2 \end{cases}$$

$$f: D \rightarrow D \text{ is given by} \qquad f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w[x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $w = f(w)$
Define $w_0 = \bot$, $w_1 = f(\bot)$, $w_2 = f(f(\bot))$, etc.
$$w_3[x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto 0, Y \mapsto y] & \text{if } x = 1 \\ [x \mapsto 0, Y \mapsto 2y] & \text{if } x = 2 \\ undefined & \text{if } x \geq 3 \end{cases}$$

$$f: D \rightarrow D \text{ is given by} f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w[x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $w = f(w)$
Define $w_0 = 1$, $w_1 = f(1)$, $w_2 = f(f(1))$, etc.
$$\begin{bmatrix} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto 0, Y \mapsto y] & \text{if } x = 1 \\ [x \mapsto 0, Y \mapsto 2y] & \text{if } x = 2 \\ [x \mapsto 0, Y \mapsto 6y] & \text{if } x = 3 \\ wdefine & \text{if } x \geq 4 \end{cases}$$

$$f: D \rightarrow D \text{ is given by} \qquad f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w[x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $w = f(w)$
Define $w_0 = \bot$, $w_1 = f(\bot)$, $w_2 = f(f(\bot))$, etc.
Union $w_{\infty} = w_0 \cup w_1 \cup w_2 \cup \cdots$ is the function
$$w_{\infty}[x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x < 0 \\ [x \mapsto 0, Y \mapsto [x \neq y] & \text{if } x > 0 \end{cases}$$

$$f: D \rightarrow D \text{ is given by} f(\omega) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ \omega [x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find weD s.t. $\omega = f(\omega)$
Define $\omega_0 = \bot$, $\omega_1 = f(\bot)$, $\omega_2 = f(f(\bot))$, etc.
Union $\omega_{\infty} = \omega_0 \cup \omega_1 \cup \omega_2 \cup \dots$ is the function
$$\omega_{\infty} [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto 0, Y \mapsto [x \neq y]] & \text{if } x > 0 \end{cases}$$
It satisfies $\omega_{\infty} = f(\omega_{\infty}) - f(\omega_{\infty}) - f(\omega_{\infty}) = \sum_{\alpha \in X} (y = y + x) = x = x = 1) \end{cases}$

$$f: D \rightarrow D \text{ is given by} \qquad f(w) [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ w [x \mapsto x - 1, Y \mapsto x \neq y] & \text{if } x > 0 \end{cases}$$
Want to find $w \in D$ s.t. $w = f(w)$
Define $w_0 = \bot$, $w_1 = f(\bot)$, $w_2 = f(f(\bot))$, etc.
Union $w_{\infty} = w_0 \cup w_1 \cup w_2 \cup \dots$ is the function
$$w_{\infty} [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto x, Y \mapsto y] = \begin{cases} [x \mapsto x, Y \mapsto y] & \text{if } x \geq 0 \\ [x \mapsto x, Y \mapsto y] = \end{cases}$$
It satisfies $w_{\infty} = f(w_{\infty})$ and
 $\forall w \} w = f(w) \Rightarrow w_{\infty} \subseteq w \qquad -w_{\infty} \text{ is a least fixed point for } f$