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

Operational.
Meanings for program phrases defined in terms of the steps of computation they can take during program execution.

Axiomatic.
Meanings for program phrases defined indirectly via the axioms and rules of some logic of program properties.

Denotational.
Concerned with giving mathematical models of programming 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

Syntax $\left[\rightarrow\right]$ \quad Semantics

\[ P \quad \rightarrow \quad \left[ P \right] \]
Basic idea of denotational semantics

<table>
<thead>
<tr>
<th>Syntax</th>
<th>→</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive program</td>
<td>↦</td>
<td>Partial recursive function</td>
</tr>
<tr>
<td>Boolean circuit</td>
<td>↦</td>
<td>Boolean function</td>
</tr>
</tbody>
</table>

\[ P \mapsto [P] \]
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

Syntax $\xrightarrow{[-]}$ Semantics

Recursive program $\mapsto$ Partial recursive function

Boolean circuit $\mapsto$ Boolean function

$P \mapsto [P]

Concerns:

- Abstract models (\textit{i.e.} implementation/machine independent).
  \textit{\textasciitilde}\textit{\textasciitilde} 1st 1/3 rd of course \textit{\textquoteright\textquoteright\textquoteright domain theory}

- Compositionality.
  \textit{\textasciitilde}\textit{\textasciitilde} 2nd 1/3 rd of course \textit{\textquoteright\textquoteright\textquoteright PCF}

- Relationship to computation (\textit{e.g.} operational semantics).
  \textit{\textasciitilde}\textit{\textasciitilde} last 1/3 rd of course
Basic example of denotational semantics (I)

IMP⁻ syntax

Arithmetic expressions

\[ A \in A_{\text{exp}} ::= n \mid L \mid A + A \mid \ldots \]
where \( n \) ranges over integers and \( L \) over a specified set of locations \( \mathbb{L} \)

Boolean expressions

\[ B \in B_{\text{exp}} ::= \text{true} \mid \text{false} \mid A = A \mid \ldots \]
\[ \mid \neg B \mid \ldots \]

Commands

\[ C \in \text{Comm} ::= \text{skip} \mid L := A \mid C; C \]
\[ \mid \text{if } B \text{ then } C \text{ else } C \]
Basic example of denotational semantics (II)

Semantic functions

\[ A : \mathsf{Aexp} \to (\mathsf{State} \to \mathbb{Z}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]

\[ \mathsf{State} = (\mathbb{L} \to \mathbb{Z}) \]
Basic example of denotational semantics (II)

Semantic functions

\[ A : \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]
\[ B : \text{Bexp} \rightarrow (\text{State} \rightarrow \mathbb{B}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]
\[ \mathbb{B} = \{ \text{true, false} \} \]
\[ \text{State} = (\mathbb{L} \rightarrow \mathbb{Z}) \]
Basic example of denotational semantics (II)

Semantic functions

\[ A: \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]
\[ B: \text{Bexp} \rightarrow (\text{State} \rightarrow \mathbb{B}) \]
\[ C: \text{Comm} \rightarrow (\text{State} \rightarrow \text{State}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]
\[ \mathbb{B} = \{ \text{true, false} \} \]
\[ \text{State} = (\text{L} \rightarrow \mathbb{Z}) \]
Definition. A partial function from a set $X$ to a set $Y$ is specified by any subset $f \subseteq X \times Y$ satisfying

$$(x, y) \in f \land (x, y') \in f \rightarrow y = y'$$

for all $x \in X$ and $y, y' \in Y$. 

ordered pairs $\{(x, y) \mid x \in X \land y \in Y\}$

i.e. for all $x \in X$ there is at most one $y \in Y$ with $(x, y) \in f$
Basic example of denotational semantics (III)

Semantic function $\mathcal{A}$

$\mathcal{A}[n] = \lambda s \in \text{State}. n$

$\mathcal{A}[L] = \lambda s \in \text{State}. s(L)$

$\mathcal{A}[A_1 + A_2] = \lambda s \in \text{State}. \mathcal{A}[A_1](s) + \mathcal{A}[A_2](s)$
Basic example of denotational semantics (IV)

Semantic function \( \mathcal{B} \)

\[
\begin{align*}
\mathcal{B}[\text{true}] &= \lambda s \in \text{State}. \text{true} \\
\mathcal{B}[\text{false}] &= \lambda s \in \text{State}. \text{false} \\
\mathcal{B}[A_1 = A_2] &= \lambda s \in \text{State}. \text{eq}(\mathcal{A}[A_1](s), \mathcal{A}[A_2](s)) \\
\text{where } \text{eq}(a, a') &= \begin{cases} 
\text{true} & \text{if } a = a' \\
\text{false} & \text{if } a \neq a'
\end{cases}
\end{align*}
\]
Basic example of denotational semantics (V)

Semantic function $C$

$\boxed{\text{skip}} = \lambda s \in State. \ s$

NB: From now on the names of semantic functions are omitted!
A simple example of compositionality

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

$$[\text{if } B \text{ then } C \text{ else } C'] = \lambda s \in \text{State}. \text{if } ([B](s), [C](s), [C'])(s)$$

where

$$\text{if } (b, x, x') = \begin{cases} x & \text{if } b = \text{true} \\ x' & \text{if } b = \text{false} \end{cases}$$
Basic example of denotational semantics (VI)

Semantic function $C$

$$[L := A] = \lambda s \in \text{State}. \lambda \ell \in \mathbb{L}. \text{if} \left( \ell = L, [A](s), s(\ell) \right)$$
Denotational semantics of sequential composition

Denotation of sequential composition $C; C'$ of two commands

$$[C; C'] = [C'] \circ [C] = \lambda s \in State. [C']([C](s))$$

given by composition of the partial functions from states to states $[C], [C'] : State \to State$ which are the denotations of the commands.
Denotational semantics of sequential composition

Denotation of sequential composition $C; C'$ of two commands

$$[C; C'] = [C'] \circ [C] = \lambda s \in \text{State}. [C']( [C](s))$$

given by composition of the partial functions from states to states $[C], [C'] : \text{State} \rightarrow \text{State}$ which are the denotations of the commands.

$[C']( [C](s))$ is undefined if

- either $[C](s)$ is undefined
- or $[C](s) = s'$, say, and $[C'](s')$ is undefined.
Denotational semantics of sequential composition

Denotation of sequential composition \( C; C' \) of two commands

\[
[C; C'] = [C'] \circ [C] = \lambda s \in \text{State}. [C']([C](s))
\]

given by composition of the partial functions from states to states \([C], [C'] : \text{State} \rightarrow \text{State}\) which are the denotations of the commands.

Cf. operational semantics of sequential composition:

\[
\begin{align*}
C, s \downarrow s' & \quad C', s' \downarrow s'' \\
\hline
C; C', s \downarrow s''
\end{align*}
\]
While the language $\text{IMP}$, to a language $\text{IMP}$ by extending the grammar of commands:

$C \in \text{Comm} ::= \ldots \mid \text{while } B \text{ do } C$
Operational semantics of while-loops

\[
\langle \text{while } B \text{ do } C, s \rangle \rightarrow \langle \text{if } B \text{ then } C ; (\text{while } B \text{ do } C) \text{ else skip} , s \rangle
\]

Suggests looking for a denotation \([\text{while } B \text{ do } C]\]

Satisfying

\[
[\text{while } B \text{ do } C] =
\]

\[
[\text{if } B \text{ then } C ; (\text{while } B \text{ do } C) \text{ else skip }]
\]
Fixed point property of 
\[[\text{while } B \text{ do } C]\]

\[[\text{while } B \text{ do } C]\] = f_{[B], [C]}(\[[\text{while } B \text{ do } C]\])

where, for each \(b : \text{State} \rightarrow \{\text{true, false}\}\) and 
\(c : \text{State} \rightarrow \text{State}\), we define

\(f_{b,c} : (\text{State} \rightarrow \text{State}) \rightarrow (\text{State} \rightarrow \text{State})\)

as

\(f_{b,c} = \lambda w \in (\text{State} \rightarrow \text{State}). \lambda s \in \text{State}.
if (b(s), w(c(s)), s).\)
Fixed point property of 
\[\texttt{[while } B \texttt{ do } C]\]

\[\texttt{[while } B \texttt{ do } C] = f_{[B],[C]}(\texttt{[while } B \texttt{ do } C])\]

where, for each \(b : \text{State} \rightarrow \{\text{true, false}\}\) and \(c : \text{State} \rightarrow \text{State}\), we define

\(f_{b,c} : (\text{State} \rightarrow \text{State}) \rightarrow (\text{State} \rightarrow \text{State})\)

as

\(f_{b,c} = \lambda w \in (\text{State} \rightarrow \text{State}). \lambda s \in \text{State}. \quad \text{if } (b(s), w(c(s)), s).\)

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

\[ \text{State} \overset{\text{def}}{=} (\mathbb{L} \to \mathbb{Z}) \]
integer assignments to locations

\[ D \overset{\text{def}}{=} (\text{State} \to \text{State}) \]
partial functions on states

For \[ \text{while } X > 0 \text{ do } Y := X \times Y ; X := X - 1 \] \[ \in D \] we seek a minimal solution to \[ w = f(w) \], where \[ f : D \to D \] is defined by:

\[
 f(w)([X \mapsto x, Y \mapsto y]) = \begin{cases} 
 [X \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\
 w([X \mapsto x - 1, Y \mapsto x \times y]) & \text{if } x > 0.
\end{cases}
\]
\[ D \overset{\text{def}}{=} (\text{State} \to \text{State}) \]

- **Partial order \( \sqsubseteq \) on \( D \):**
  
  \[ w \sqsubseteq w' \quad \text{iff} \quad \text{for all } s \in \text{State}, \text{ if } w \text{ is defined at } s \text{ then so is } w' \text{ and moreover } w(s) = w'(s). \]
  
  \[ \text{iff} \quad \text{the graph of } w \text{ is included in the graph of } w'. \]

- **Least element \( \bot \in D \) w.r.t. \( \sqsubseteq \):**
  
  \[ \bot = \text{totally undefined partial function} \]
  
  \[ = \text{partial function with empty graph} \]
  
  (satisfies \( \bot \sqsubseteq w \), for all \( w \in D \)).
\[ f : \mathcal{D} \to \mathcal{D} \text{ is given by} \]
\[ f(w) \left[ X \mapsto x, Y \mapsto y \right] = \begin{cases} 
[X \mapsto x, Y \mapsto y] & \text{if } x \leq 0 \\
W \left[ X \mapsto x^{-1}, Y \mapsto x \cdot y \right] & \text{if } x > 0
\end{cases} \]

Want to find \( w \in \mathcal{D} \) s.t. \( w = f(w) \)

Define \( w_0 = 1 \), \( w_1 = f(1) \), \( w_2 = f(f(1)) \), etc.

\[ w_0 \left[ X \mapsto x, Y \mapsto y \right] = \text{undefined} \]
$f : D \to D$ is given by

\[ f(w) \ [x \mapsto x, \ y \mapsto y] = \begin{cases} 
[x \mapsto x, \ y \mapsto y] & \text{if } x \leq 0 \\
 w \ [x \mapsto x-1, \ y \mapsto x \cdot y] & \text{if } x > 0
\end{cases} \]

Want to find $w \in D$ s.t. $w = f(w)$

Define $w_0 = 1$, $w_1 = f(1)$, $w_2 = f(f(1))$, etc.

\[ w_1 \ [x \mapsto x, \ y \mapsto y] = \begin{cases} 
[x \mapsto x, \ y \mapsto y] & \text{if } x \leq 0 \\
 \text{undefined} & \text{if } x \geq 1
\end{cases} \]
$f : D \to D$ is given by

$$f(w) [x \mapsto x, y \mapsto y] = \begin{cases} 
[x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
 w [x \mapsto x-1, y \mapsto xy] & \text{if } x > 0
\end{cases}$$

Want to find $w \in D$ s.t. $w = f(w)$

Define $w_0 = 1$, $w_1 = f(1)$, $w_2 = f(f(1))$, etc.

$$w_2 [x \mapsto x, y \mapsto y] = \begin{cases} 
[x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
 [x \mapsto 0, y \mapsto y] & \text{if } x = 1 \\
 \text{undefined} & \text{if } x > 2
\end{cases}$$
\( f : D \rightarrow D \) is given by

\[
f(w) [x \mapsto x, \, \mathsf{y} \mapsto y] = \begin{cases} 
[x \mapsto x, \, \mathsf{y} \mapsto y] & \text{if } x \leq 0 \\
 w [x \mapsto x-1, \, \mathsf{y} \mapsto x \cdot y] & \text{if } x > 0
\end{cases}
\]

Want to find \( w \in D \) s.t. \( w = f(w) \)

Define \( w_0 = 1 \), \( w_1 = f(1) \), \( w_2 = f(f(1)) \), etc.

\[
w_2 [x \mapsto x, \, \mathsf{y} \mapsto y] = \begin{cases} 
[x \mapsto x, \, \mathsf{y} \mapsto y] & \text{if } x \leq 0 \\
[x \mapsto 0, \, \mathsf{y} \mapsto y] & \text{if } x = 1 \\
[x \mapsto 0, \, \mathsf{y} \mapsto 2y] & \text{if } x = 2 \\
\text{undefined} & \text{if } x \geq 3
\end{cases}
\]
$f : D \to D$ is given by

$$f(w) \ [x \mapsto x, y \mapsto y] = \begin{cases} [x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\ w \ [x \mapsto x-1, y \mapsto xy] & \text{if } x > 0 \end{cases}$$

Want to find $w \in D$ s.t. $w = f(w)$

Define $w_0 = 1$, $w_1 = f(1)$, $w_2 = f(f(1))$, etc.

$$w_4 \ [x \mapsto x, y \mapsto y] = \begin{cases} [x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\ [x \mapsto 0, y \mapsto y] & \text{if } x = 1 \\ [x \mapsto 0, y \mapsto 2y] & \text{if } x = 2 \\ [x \mapsto 0, y \mapsto 6y] & \text{if } x = 3 \\ \text{undefined} & \text{if } x > 4 \end{cases}$$
Define \( w_0 = 1 \), \( w_1 = f(1) \), \( w_2 = f(f(1)) \), etc.
Union \( w_\infty = w_0 \cup w_1 \cup w_2 \cup \ldots \) is the function

\[
\begin{align*}
f(x) = \begin{cases} 
[x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
[w \mapsto x-1, y \mapsto x \cdot y] & \text{if } x > 0
\end{cases}
\end{align*}
\]
\[ f : \mathcal{D} \to \mathcal{D} \text{ is given by} \]
\[ f(w) [x \mapsto x, y \mapsto y] = \]
\[ \begin{cases} 
[x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
 w [x \mapsto x-1, y \mapsto x*y] & \text{if } x > 0
\end{cases} \]

Want to find \( w \in \mathcal{D} \) s.t. \( w = f(w) \)

Define \( w_0 = 1 \), \( w_1 = f(1) \), \( w_2 = f(f(1)) \), etc.

Union \( w_\infty = w_0 \cup w_1 \cup w_2 \cup \ldots \) is the function

\[ w_\infty [x \mapsto x, y \mapsto y] = \]
\[ \begin{cases} 
[x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
 [x \mapsto 0, y \mapsto ] & \text{if } x > 0
\end{cases} \]

It satisfies \( w_\infty = f(w_\infty ) \) — fixed point we seek for definition of \( \{ \text{while } x > 0 \ do \ (y := y*x; x := x-1) \} \)
Given \( f : D \rightarrow D \) is given by
\[
    f(w) \left[ x \mapsto x, y \mapsto y \right] = \begin{cases} 
        [x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
        w \left[ x \mapsto x-1, y \mapsto x \cdot y \right] & \text{if } x > 0
    \end{cases}
\]

Want to find \( w \in D \) s.t. \( w = f(w) \).

Define \( w_0 = 1 \), \( w_1 = f(1) \), \( w_2 = f(f(1)) \), etc.

Union \( w_\infty = w_0 \cup w_1 \cup w_2 \cup \ldots \) is the function
\[
w_\infty \left[ x \mapsto x, y \mapsto y \right] = \begin{cases} 
        [x \mapsto x, y \mapsto y] & \text{if } x \leq 0 \\
        [x \mapsto 0, y \mapsto !x \cdot y] & \text{if } x > 0
    \end{cases}
\]

It satisfies \( w_\infty = f(w_\infty) \) and
\[
    (\forall w) \ w = f(w) \Rightarrow w_\infty \subseteq w
\]

- \( w_\infty \) is a least fixed point for \( f \).