Denotational Semantics

10 lectures for Part II CST 2016/17

Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/1617/DenotSem/

Topic 1

Introduction

What is this course about?

- General area.

  Formal methods: Mathematical techniques for the specification, development, and verification of software and hardware systems.

- Specific area.

  Formal semantics: Mathematical theories for ascribing meanings to computer languages.

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

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.

Basic idea of denotational semantics

$$\begin{array}{rcl} \text{Syntax} & \xrightarrow{[\![-]\!]} & \text{Semantics} \\ \text{Recursive program} & \mapsto & \text{Partial recursive function} \\ \text{Boolean circuit} & \mapsto & \text{Boolean function} \\ P & \mapsto & [\![P]\!] \end{array}$$

Concerns:

- Abstract models (*i.e.* implementation/machine independent).

  $\rightsquigarrow$ Lectures 2, 3 and 4.

- Compositionality.

  $\rightsquigarrow$ Lectures 5 and 6.

- Relationship to computation (*e.g.* operational semantics).

  $\rightsquigarrow$ Lectures 7 and 8.

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 example of denotational semantics (I)

IMP$^-$ syntax

Arithmetic expressions

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

where n ranges over *integers* and
L over a specified set of *locations* $\mathbb{L}$

Boolean expressions

$$\begin{aligned} B \in \mathbf{Bexp} \quad ::= & \quad \mathbf{true} \mid \mathbf{false} \mid A = A \mid \ldots \\ \mid & \quad \neg B \mid \ldots \end{aligned}$$

Commands

$$\begin{aligned} C \in \mathbf{Comm} \quad ::= & \quad \mathbf{skip} \mid L := A \mid C; C \\ \mid & \quad \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C \end{aligned}$$

Basic example of denotational semantics (II)

Semantic functions

$$\mathcal{A} : \quad \mathbf{Aexp} \rightarrow (State \rightarrow \mathbb{Z})$$

where

idea:

$$\mathbb{Z} \quad = \quad \{\ldots, -1, 0, 1, \ldots\}$$

$$s \in State \quad = \quad (\mathbb{L} \rightarrow \mathbb{Z})$$

in L

the set of functions from the set of locations to the set of integers.

s(L) = value of L in state s

Basic example of denotational semantics (II)

Semantic functions

$$
\begin{aligned}
\mathcal{A}: &\quad \mathbf{Aexp} \rightarrow (State \rightarrow \mathbb{Z}) \\
\mathcal{B}: &\quad \mathbf{Bexp} \rightarrow (State \rightarrow \mathbb{B})
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbb{Z} &= \{\ldots, -1, 0, 1, \ldots\} \\
\mathbb{B} &= \{\, true, false \,\} \\
State &= (\mathbb{L} \rightarrow \mathbb{Z})
\end{aligned}
$$

Idea: $\mathcal{C}[\![C]\!] : State \to State$ s.t. $\mathcal{C}[\![C]\!](s)$ undefined if C loops from s or gives the new state obtained after executing C.

Basic example of denotational semantics (II)

Semantic functions

$$
\begin{aligned}
\mathcal{A} : \quad & \mathbf{Aexp} \to (\mathit{State} \to \mathbb{Z}) \\
\mathcal{B} : \quad & \mathbf{Bexp} \to (\mathit{State} \to \mathbb{B}) \\
\mathcal{C} : \quad & \mathbf{Comm} \to \underbrace{(\mathit{State} \rightharpoonup \mathit{State})}_{\text{The set of state transformers}}
\end{aligned}
$$

The set of state transformers

i.e. partial functions from states to states.

where

$$
\begin{aligned}
\mathbb{Z} &= \{\ldots, -1, 0, 1, \ldots\} \\
\mathbb{B} &= \{\, \mathit{true}, \mathit{false} \,\} \\
\mathit{State} &= (\mathbb{L} \to \mathbb{Z})
\end{aligned}
$$

$\mathcal{A}[\![A]\!](s) \in \mathbb{Z}$

Basic example of denotational semantics (III)

Semantic function $\mathcal{A}$

lambda notation / anonymous function notation

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

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

$$\mathcal{A}[\![A_1 + A_2]\!] = \lambda s \in State.\, \mathcal{A}[\![A_1]\!](s) + \mathcal{A}[\![A_2]\!](s)$$

addition notation

mathematical addition on $\mathbb{Z}$

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

$\mathcal{B}[\![B]\!](s) \in \mathbb{B}$

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

$$\mathcal{B}[\![\mathbf{true}]\!] = \lambda s \in State.\, true$$

$$\mathcal{B}[\![\mathbf{false}]\!] = \lambda s \in State.\, false$$

$$\mathcal{B}[\![A_1 = A_2]\!] = \lambda s \in State.\, eq\big(\mathcal{A}[\![A_1]\!](s), \mathcal{A}[\![A_2]\!](s)\big)$$

where $eq(a, a') = \begin{cases} true & \text{if } a = a' \\ false & \text{if } a \neq a' \end{cases}$

$\mathcal{C}[\![c]\!] \in (State \to State)$

state transformers.

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

the identity function

$$[\![\mathbf{skip}]\!] \quad = \quad \lambda s \in State.\, s$$

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

A simple example of compositionality

Given partial functions $[\![C]\!], [\![C']\!] : \mathit{State} \rightharpoonup \mathit{State}$ and a function $[\![B]\!] : \mathit{State} \to \{\mathit{true}, \mathit{false}\}$, we can define

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

where

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

Basic example of denotational semantics (VI)

Semantic function $\mathcal{C}$

$$[\![L := A]\!] \quad = \quad \lambda s \in State.\, \lambda \ell \in \mathbb{L}.\, \mathit{if}\big(\ell = L, [\![A]\!](s), s(\ell)\big)$$

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

Denotational semantics of sequential composition

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

$$[\![C; C']\!] = [\![C']\!] \circ [\![C]\!] = \lambda s \in \mathit{State}.\, [\![C']\!]\big([\![C]\!](s)\big)$$

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

idea

$$\text{State} \xrightarrow{[\![C]\!]} \text{State} \xrightarrow{[\![C']\!]} \text{State}$$

Denotational semantics of sequential composition

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

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

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

Cf. operational semantics of sequential composition:

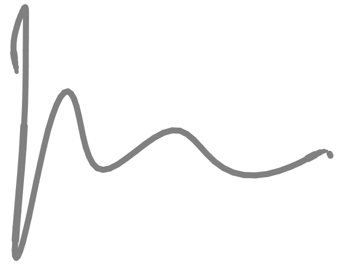
$$\frac{C, s \Downarrow s' \quad C', s' \Downarrow s''}{C; C', s \Downarrow s''} .$$

Result
$\forall C, s, s'$: $\quad C, s \Downarrow s' \iff [\![C]\!](s) = s'$

$[\![\textbf{while } B \textbf{ do } C]\!]$: State $\to$ State

$[\![B]\!] : \text{State} \to \mathbb{B}$

$[\![C]\!] : \text{State} \to \text{State}$



$$[\![\underline{\text{while}}\ B\ \underline{\text{do}}\ C]\!](s)$$

$$= \cdots [\![B]\!](s) \cdots [\![C]\!](s) \cdots$$

$$= \text{if}\Big([\![B]\!]s,\ \underline{[\![C]\!]\big(\cdots [\![\text{while } B \text{ do } C]\!] \cdots\big)},\ s\Big)$$