Denotational Semantics

10 lectures for Part II CST 2013/14

Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/1314/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

$$\text{Syntax} \xrightarrow{[\![-]\!]} \text{Semantics}$$

$$\text{Recursive program} \mapsto \text{Partial recursive function}$$

$$\text{Boolean circuit} \mapsto \text{Boolean function}$$

$$P \mapsto [\![P]\!]$$

Reg. Exp. $\longmapsto$ (Reg) Languages.

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

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

Commands

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

Basic example of denotational semantics (II)

Semantic functions

$\mathcal{A}[\![A]\!](s) \in \mathbb{Z}$ ~ the integer value of the arithmetic expression A in state s.

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

where

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

$\mathcal{C}[\![C]\!]$ is a state transformer

Def Given sets A and B we let $(A \to B)$ be the set of all functions from A to B.

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

Idea here is that for a state $s \in \text{State}$ and a location $L \in \mathbb{L}$,

$s(L)$ is the integer stored in L at state s.

Def For sets A and B, the set $(A \rightarrow B)$ is that of partial functions from A to B.

Example $(State \rightarrow State)$

Semantic function $\mathcal{A}$.

syntax

semantics

$$\mathcal{A}[\![\underline{n}]\!]s = n \in \mathbb{Z}$$

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

$$\mathcal{A}[\![A_1 + A_2]\!] = \mathcal{A}[\![A_1]\!] + \mathcal{A}[\![A_2]\!]$$

syntactic addition

mathematical addition

Basic example of denotational semantics (III)

Semantic function $\mathcal{A}$

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

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

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

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

$$
\begin{aligned}
\mathcal{B}[\![\mathbf{true}]\!] &= \lambda s \in State.\, true \\
\mathcal{B}[\![\mathbf{false}]\!] &= \lambda s \in State.\, false \\
\mathcal{B}[\![A_1 = A_2]\!] &= \lambda s \in State.\, eq\big(\mathcal{A}[\![A_1]\!](s), \mathcal{A}[\![A_2]\!](s)\big)
\end{aligned}
$$

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

$\mathcal{C} : Comm \to (State \to State)$

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

$$[\![\mathbf{skip}]\!] \quad = \quad \underbrace{\lambda s \in State.\, s}_{\text{The identity function}}$$

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

$$[\![\text{if } B \text{ then } C_1 \text{ else } C_2]\!](s)$$

$$= \begin{cases} [\![C_1]\!](s) & \text{if } [\![B]\!](s) = \text{true} \\ [\![C_2]\!](s) & \text{if } [\![B]\!](s) = \text{false} \end{cases}$$

A simple example of compositionality

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

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

where

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

$[\![L := A]\!](s) \in State$

$\rightarrow$ the state as s, but modified so that L is mapped to $[\![A]\!](s)$.

Basic example of denotational semantics (VI)

Semantic function $\mathcal{C}$

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

Recall $(f \circ g) = \lambda x . f(g x)$

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.

sequencing $\mapsto$ composition

Cf. operational semantics of sequential composition:

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

Prop: $C, s \Downarrow s'$ iff $[\![C]\!](s) = s'$

$$[\![\text{while } B \text{ do } C]\!]$$

$$= \sim [\![B]\!] \sim [\![C]\!] \sim$$