Denotational Semantics

10 lectures for Part II CST 2017/18

Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/1718/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

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

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

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

the set of functions from State to $\mathbb{Z}$.

$A \in \underline{Aexp}$ $\quad \mathcal{A}[\![A]\!] \in \underline{State} \to \mathbb{Z}$ s.t $\mathcal{A}[\![A]\!](s)$

$\}$ the value of A in state $s \in \underline{State}$.

where

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

$$State \quad = \quad \underbrace{(\mathbb{L} \to \mathbb{Z})}_{\text{functions from Locations to } \mathbb{Z}}$$

Idea

$s \in \underline{State}$

$L \in \mathbb{L}$

$\Rightarrow s(L)$

$\}$ value of L in s

Basic example of denotational semantics (II)

Semantic functions

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

where

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

Basic example of denotational semantics (II)

Semantic functions

$C \in \underline{Comm}$ $\mathcal{C}[\![C]\!] \sim$ state transformer.

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

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

$$\mathcal{C} : \quad \mathbf{Comm} \to \underbrace{(State \rightharpoonup State)}_{\text{partial function from } \underline{State} \text{ to } \underline{State}}$$

where

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

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

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

$\underline{\text{Idea}}$: $\mathcal{C}[\![C]\!](s)$ is undefined means C loops.

Basic example of denotational semantics (III)

Token standing for $n \in \mathbb{N}$

Semantic function $\mathcal{A}$

$\mathcal{A}[\![\underline{n}]\!](s) = n \in \mathbb{N}$

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

$\underline{State} = \mathbb{L} \to \mathbb{Z}$

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

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

syntax

semantics.

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

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

$\in (State \to State)$

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

identity function

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

COMPOSITIONAL

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

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 $\mathcal{C}$

$$\llbracket L := A \rrbracket \quad = \quad \lambda s \in \mathit{State}.\, \underbrace{\lambda \ell \in \mathbb{L}.\, \mathit{if}\big(\ell = L, \llbracket A \rrbracket(s), s(\ell)\big)}_{\text{a } \mathit{State} = (\mathbb{L} \to \mathbb{Z})}$$

by Compositionality given in terms of $\llbracket A \rrbracket$.

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:

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

One <u>can</u> show:

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

$[\![\textbf{while } B \textbf{ do } C]\!] \in (\text{State} \to \text{State})$

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

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

$= f\big([\![B]\!](s),\ [\![\text{while } B \text{ do } C]\!]([\![C]\!]s)),\ s\big)$

* a definition in terms of itself ✗

* not compositional

Idea:

while B do C

$\equiv$ if B then (C; while B do C) else skip