Denotational Semantics

Lectures for Part II CST 2022/23

Prof Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/2223/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

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

$A \in \underline{\underline{Aexp}}$ $\qquad \mathcal{A}[\![A]\!] : State \to \mathbb{Z}$

where

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

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

Basic example of denotational semantics (II)

Semantic functions

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

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

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

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

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

The set of partial functions from State to State.

where

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

Basic example of denotational semantics (III)

syntax in Aexp

Semantic function $\mathcal{A}$ $[\![A]\!] : State \to \mathbb{Z}$

semantics in $\mathbb{Z}$

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

$\mathcal{A}[\![\underline{n}]\!](s) = n \;\; \forall s$

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

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

syntax

semantics (addition)

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

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

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

State transformers.

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}[\![c]\!] : (State \to State)$

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

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 $\mathcal{C}$

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

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.

$$(c_1; c_2); c_3 \approx c_1; (c_2; c_3)$$

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:

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

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

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

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

$$[\![\underline{\text{while}}\ \underline{\text{true}}\ \text{do}\ \underline{\text{skip}}]\!](s) = \uparrow \rightsquigarrow \text{undefined}$$

$\forall s \in \underline{\text{State}}$.

$[\![\underline{\text{while}}\ \underline{\text{true}}\ \underline{\text{do}}\ \underline{\text{skip}}]\!] : \text{State} \to \text{State}$

is the undefined partial function; i.e, with empty graph denoted $\emptyset$ or $\perp$

$$[\![\underline{\text{while}}\ \underline{\text{false}}\ \underline{\text{do}}\ \underline{\text{skip}}]\!](s) = s$$

$$[\![\underline{\text{while}}\ \underline{\text{false}}\ \underline{\text{do}}\ \underline{\text{skip}}]\!] = \mathrm{id}_{\underline{\text{state}}}$$

$\underline{\underline{\text{NB}}}$: $\underline{\text{while}}\ \underline{\text{false}}\ \underline{\text{do}}\ \underline{\text{skip}} \simeq \underline{\text{skip}}$

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

$$\underline{\text{while}}\ B\ \underline{\text{do}}\ C \simeq \underline{\text{if}}\ B\ \underline{\text{then}}\ (C;\ \underline{\text{while}}\ B\ \underline{\text{do}}\ C)\ \underline{\text{else}}\ \underline{\text{skip}}$$

~~operationally~~

$$\llbracket \text{while } B \text{ do } C \rrbracket(s)$$

$$= \llbracket \text{if } B \text{ then } (C;\ \text{while } B \text{ do } C) \text{ else skip} \rrbracket(s)$$

$$= \text{if}(\llbracket B \rrbracket(s),\ \llbracket C;\ \text{while } B \text{ do } C \rrbracket(s),\ s)$$

$$= \text{if}(\llbracket B \rrbracket(s),\ \llbracket \text{while } B \text{ do } C \rrbracket(\llbracket C \rrbracket s),\ s)$$

NB: $\llbracket \text{while } B \text{ do } C \rrbracket$ is a state transformer

$w : \text{State} \to \text{State}$ satisfying

$$w = \lambda s.\ \text{if}(\llbracket B \rrbracket(s),\ w(\llbracket C \rrbracket s),\ s)$$

$$\text{fix}(\lambda w.\lambda s.\ \text{if}(\llbracket B \rrbracket(s),\ w(\llbracket C \rrbracket s),\ s))$$