Denotational Semantics

Lectures for Part II CST 2021/22

Prof Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/2122/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).

  $\leadsto$ Lectures 2, 3 and 4.

- Compositionality.

  $\leadsto$ Lectures 5 and 6.

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

  $\leadsto$ 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

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

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

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

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

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

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

domain of State transformers.

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 (III)

Semantic function $\mathcal{A}$

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

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

syntax

semantics

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

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

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

$\mathcal{C}[\![c]\!] : \underline{State} \to \underline{State}$

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

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

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

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

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 $\mathcal{C}$

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

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.

Cf. operational semantics of sequential composition:

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

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

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

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

$\}$ is the totally undefined partial function

I.e. the partial function with empty graph

(sometimes denoted $\emptyset$ or $\bot$)

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

└ The identity function

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

$= \lambda s \in \underline{\text{State}} \ldots \quad [\![B]\!] \ldots \quad [\![C]\!] \ldots$

Operationally:

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

Expect:

$$[\![\texttt{while } B \texttt{ do } C]\!] = [\![\texttt{if } B \texttt{ then } (C, \texttt{while } B \texttt{ do } C) \texttt{ else skip}]\!]$$

$$[\![\texttt{while } B \texttt{ do } C]\!](s) \stackrel{\text{def?}\ \times}{=} \text{if}\left([\![B]\!](s),\ [\![\texttt{while } B \texttt{ do } C]\!]([\![C]\!]s),\ s\right)$$

$[\![\underline{\text{while}}\ B\ \underline{\text{do}}\ C]\!]$ is a fixed point of a function

$$f_{[\![B]\!],[\![C]\!]}$$

That is,

$$[\![\text{while } B \text{ do } C]\!] = f_{[\![B]\!],[\![C]\!]}\left([\![\text{while } B \text{ do } C]\!]\right)$$

for

$$f_{[\![B]\!],[\![C]\!]} = \lambda w.\ \lambda s.\ \text{if}\left([\![B]\!]s,\ w([\![C]\!]s),\ s\right)$$

$$: (\text{State} \to \text{State}) \to (\text{State} \to \text{State})$$

$$[\![\text{while } B \text{ do } C]\!] \overset{\text{def}}{=} \underline{fix}\left(f_{[\![B]\!],[\![C]\!]} \right)$$

need a notion of fixed point that is computationally meaningful.

calculate by approximation.

$[\![\text{while } B \text{ do } C]\!] : \underline{State} \rightharpoonup \underline{State}.$

Idea:

$$w_0 \sqsubseteq w_1 \sqsubseteq \cdots \sqsubseteq w_n \sqsubseteq \cdots \quad (n \in \mathbb{N}) \qquad \bigsqcup_{n \in \mathbb{N}} w_n$$

$$\parallel$$

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

$$w_0 = \bot$$

$$w_{n+1} = f_{[\![B]\!],[\![C]\!]}(w_n)$$