Denotational Semantics

10 lectures for Part II CST 2015/16

Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/1516/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?

Why do we care?

- Rigour.

  ... specification of programming languages

  ... justification of program transformations

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

- Feedback into language design.

  . . . continuations

  . . . monads

- Reasoning principles.

  . . . Scott induction

  . . . Logical relations

  . . . Co-induction

Styles of formal semantics

Operational.

Axiomatic.

Denotational.

Styles of formal semantics

Operational.

Meanings for program phrases defined in terms of the *steps of computation* they can take during program execution.

Axiomatic.

Denotational.

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.

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

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

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

Basic idea of denotational semantics

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

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

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

Basic idea of denotational semantics

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

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.

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.

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

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

Notation: $X \to Y$ (for X and Y sets) is the set of functions from X to Y

Basic example of denotational semantics (II)

Semantic functions

$E \in \mathbf{Aexp}$

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

$\mathcal{A}(E) \in (State \to \mathbb{Z})$

$\mathcal{A}(E)(s) \in \mathbb{Z}$

$s \in State$

the value of E in state s

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

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

where

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

Notation: $(X \rightharpoonup Y)$ the set of partial functions from X to Y.

Basic example of denotational semantics (II)

$K \in Comm$

Semantic functions

$\mathcal{C}(K) \in (State \rightharpoonup State)$ — state transformers.

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

where

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

Notation $\mathcal{A}(A) \approx \mathcal{A}[\![A]\!] \approx [\![A]\!] \in (State \to \mathbb{Z})$

Basic example of denotational semantics (III)

λ-notation

Semantic function $\mathcal{A}$

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

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

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

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

Idea $\lambda x. \ldots\ldots x \sim \approx$ the function that given x outputs $\sim x \sim$

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

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

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

$[\![C]\!] \in (State \to State)$

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

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

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

where

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

Idea: the state after executing the assignment from state s.

Basic example of denotational semantics (VI)

Semantic function $\mathcal{C}$

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

①

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

Denotational semantics of sequential composition

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

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

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

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

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

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

$[\![\text{while } B \text{ do } C]\!] \in (\text{State} \rightharpoonup \text{State})$ —— ?

$\quad\; \|$

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

Operational heuristic:

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

$\leadsto$

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

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

Def A fixed point of a function f is a value x such that $f(x) = x$.

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

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

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

$\leadsto$ The interpretation of $[\![\text{while } B \text{ do } C]\!]$ is given by a fixed point