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

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

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

Basic idea of denotational semantics

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

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

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

Basic idea of denotational semantics

Syntax	$\xrightarrow{[\![-]\!]}$	Semantics
Recursive program	$\mapsto$	Partial recursive function
Boolean circuit	$\mapsto$	Boolean function
P	$\mapsto$	$[\![P]\!]$

Basic idea of denotational semantics

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

Concerns:

- Abstract models (*i.e.* implementation/machine independent).
  $\rightsquigarrow$ Lectures 2, 3 and 4.

Basic idea of denotational semantics

Syntax	$\xrightarrow{[\![-]\!]}$	Semantics
Recursive program	$\mapsto$	Partial recursive function
Boolean circuit	$\mapsto$	Boolean function
P	$\mapsto$	$[\![P]\!]$

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

$$B \in \mathbf{Bexp} \quad ::= \quad \mathbf{true} \mid \mathbf{false} \mid A = A \mid \ldots$$
$$\mid \quad \neg B \mid \ldots$$

Commands

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

Basic example of denotational semantics (II)

Semantic functions

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

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

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

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 $\mathcal{A}$

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

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

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

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

$$\mathcal{B}[\![\mathbf{true}]\!] = \lambda s \in State.\, true$$

$$\mathcal{B}[\![\mathbf{false}]\!] = \lambda s \in State.\, false$$

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

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 $\mathcal{C}$

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

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

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

where

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

Basic example of denotational semantics (VI)

Semantic function $\mathcal{C}$

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

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:

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

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

Fixed point property of $[\![\textbf{while } B \textbf{ do } C]\!]$

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

where, for each $b : State \to \{true, false\}$ and $c : State \rightharpoonup State$, we define

$$f_{b,c} : (State \rightharpoonup State) \to (State \rightharpoonup State)$$

as

$$f_{b,c} = \lambda w \in (State \rightharpoonup State).\, \lambda s \in State.\, if\, \big(b(s), w(c(s)), s\big).$$

Fixed point property of $[\![\textbf{while } B \textbf{ do } C]\!]$

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

where, for each $b : State \to \{true, false\}$ and $c : State \rightharpoonup State$, we define

$$f_{b,c} : (State \rightharpoonup State) \to (State \rightharpoonup State)$$

as

$$f_{b,c} = \lambda w \in (State \rightharpoonup State).\, \lambda s \in State.\, if\,\big(b(s), w(c(s)), s\big).$$

- Why does $w = f_{[\![B]\!],[\![C]\!]}(w)$ have a solution?
- What if it has several solutions—which one do we take to be $[\![\textbf{while } B \textbf{ do } C]\!]$?

Approximating $[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!]$

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

$$f_{[\![B]\!],[\![C]\!]}{}^{n}(\bot)$$

$$= \lambda s \in State.$$

$$\begin{cases} [\![C]\!]^k(s) & \text{if } \exists\, 0 \leq k < n.\, [\![B]\!]([\![C]\!]^k(s)) = false \\ & \text{and } \forall\, 0 \leq i < k.\, [\![B]\!]([\![C]\!]^i(s)) = true \\ \\ \uparrow & \text{if } \forall\, 0 \leq i < n.\, [\![B]\!]([\![C]\!]^i(s)) = true \end{cases}$$

$$D \stackrel{\text{def}}{=} (State \rightharpoonup State)$$

- **Partial order $\sqsubseteq$ on D:**

  $w \sqsubseteq w'$ iff for all $s \in State$, if w is defined at s then so is w' and moreover $w(s) = w'(s)$.

  iff the graph of w is included in the graph of w'.

- **Least element $\bot \in D$ w.r.t. $\sqsubseteq$:**

  $\bot$ = totally undefined partial function

  = partial function with empty graph

  (satisfies $\bot \sqsubseteq w$, for all $w \in D$).