Semantics of Programming Languages

Denotational Semantics

Prof. Peter Müller

Software Component Technology

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Motivation

- Operational semantics is at a rather low abstraction level
  - Some arbitrariness in choice of rules (e.g., size of steps)
  - Syntax involved in description of behavior
- Semantic equivalence in natural semantics

$$\langle s_1, \sigma \rangle \rightarrow \sigma' \Leftrightarrow \langle s_2, \sigma \rangle \rightarrow \sigma'$$

- Idea
  - We can describe the behavior on an abstract level if we are only interested in equivalence
  - We specify only the partial function on states

Approach

- Denotational semantics describes the **effect** of a computation

- A semantic function is defined for each syntactic construct
  - maps syntactic construct to a mathematical object, often a function
  - the mathematical object describes the effect of executing the syntactic construct

Compositionality

- In denotational semantics, semantic functions are defined **compositionally**
- There is a semantic clause for each of the basis elements of the syntactic category
- For each method of constructing a composite element (in the syntactic category) there is a semantic clause defined in terms of the **semantic function applied to the immediate constituents** of the composite element

Examples

- The semantic functions $\mathcal{A}$: Aexp $\longrightarrow$ State $\longrightarrow$ Val and $\mathcal{B}$: Bexp $\longrightarrow$ State $\longrightarrow$ Bool are denotational definitions

$$
\begin{array}{lll}
\mathcal{A}[\![x]\!]\sigma & = \sigma(x) & \\
\mathcal{A}[\![i]\!]\sigma & = i & \text{for } i \in \mathbb{Z} \\
\mathcal{A}[\![e_1\ op\ e_2]\!]\sigma & = \mathcal{A}[\![e_1]\!]\sigma\ \overline{op}\ \mathcal{A}[\![e_2]\!]\sigma & \text{for } op \in \text{ Op}
\end{array}
$$

$$
\mathcal{B}[\![e_1\ op\ e_2]\!]\sigma = \begin{cases} tt & \text{if } \mathcal{A}[\![e_1]\!]\sigma\ \overline{op}\ \mathcal{A}[\![e_2]\!]\sigma \\ ff & \text{otherwise} \end{cases}
$$

Counterexamples

- The semantic functions $\mathcal{S}_{NS}$ and $\mathcal{S}_{SOS}$ are not denotational definitions because they are not defined compositionally

$$\mathcal{S}_{NS} : \text{Stm} \to (\text{State} \hookrightarrow \text{State})$$

$$\mathcal{S}_{NS}[\![s]\!]\sigma = \begin{cases} \sigma' & \text{if } \langle s, \sigma\rangle \to \sigma' \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\mathcal{S}_{SOS} : \text{Stm} \to (\text{State} \hookrightarrow \text{State})$$

$$\mathcal{S}_{SOS}[\![s]\!]\sigma = \begin{cases} \sigma' & \text{if } \langle s, \sigma\rangle \to_1^* \sigma' \\ \text{undefined} & \text{otherwise} \end{cases}$$

3. Denotational Semantics

3.1 Direct Style Semantics: Specification

3.1.1 Direct Style Semantics of IMP

3.1.2 Requirements on the Fixed Point

3.2 Fixed Point Theory

3.3 Direct Style Semantics: Existence

3.4 Equivalence

3.5 Extensions of IMP

Semantic Functions

- The effect of executing a statement is described by the partial function $\mathcal{S}_{DS}$

$$\mathcal{S}_{DS} : \text{Stm} \rightarrow (\text{State} \hookrightarrow \text{State})$$

- Partiality is needed to model non-termination

- The effects of evaluating expressions is defined by the functions $\mathcal{A}$ and $\mathcal{B}$

Direct Style Semantics of IMP

- `skip` does not modify the state

$$\mathcal{S}_{DS}[\![\texttt{skip}]\!] = id$$

$$id : \text{State} \to \text{State}$$

$$id(\sigma) = \sigma$$

- x`:=`e assigns the value of e to variable x

$$\mathcal{S}_{DS}[\![x\texttt{:=}e]\!]\sigma = \sigma[x \mapsto \mathcal{A}[\![e]\!]\sigma]$$

Direct Style Semantics of IMP (cont'd)

- Sequential composition $s_1 ; s_2$

$$\mathcal{S}_{DS}[\![s_1 ; s_2]\!] = \mathcal{S}_{DS}[\![s_2]\!] \circ \mathcal{S}_{DS}[\![s_1]\!]$$

- Function composition $\circ$ is defined in a **strict** way
  - If one of the functions is undefined on the given argument then the composition is undefined

$$(f \circ g)\sigma = \begin{cases} f(g(\sigma)) & \text{if } g(\sigma) \neq \text{undefined} \\ & \text{and } f(g(\sigma)) \neq \text{undefined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Direct Style Semantics of IMP (cont'd)

- Conditional statement `if` b `then` s_1 `else` s_2 `end`

$$\mathcal{S}_{DS}[\![\texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2 \texttt{ end}]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{DS}[\![s_1]\!], \mathcal{S}_{DS}[\![s_2]\!])$$

- The function $cond$
  - takes the semantic functions for the condition and the two statements
  - when supplied with a state selects the second or third argument depending on the first

$$cond : (\text{State} \to \text{Bool}) \times (\text{State} \hookrightarrow \text{State}) \times (\text{State} \hookrightarrow \text{State}) \to (\text{State} \hookrightarrow \text{State})$$

Definition of $cond$

$$cond : \big(\text{State} \to \text{Bool}\big) \times \big(\text{State} \hookrightarrow \text{State}\big) \times \big(\text{State} \hookrightarrow \text{State}\big) \to \big(\text{State} \hookrightarrow \text{State}\big)$$

$$cond(b, f, g)\sigma = \begin{cases} f(\sigma) & \text{if } b(\sigma) = tt \\ & \text{and } f(\sigma) \neq \text{undefined} \\ g(\sigma) & \text{if } b(\sigma) = ff \\ & \text{and } g(\sigma) \neq \text{undefined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Semantics of Loop: Observations

- Defining the semantics of `while` is difficult
- The semantics of `while` b `do` s `end` must be equal to `if` b `then` s`;while` b `do` s `end else skip end`
- This requirement yields:

$$\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] \circ \mathcal{S}_{DS}[\![s]\!], id)$$

- We cannot use this equation as a definition because it is not compositional

Functionals and Fixed Points

$$\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] = cond(\mathcal{B}[\![b]\!], \mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] \circ \mathcal{S}_{DS}[\![s]\!], id)$$

- The above equation has the form $g = F(g)$
  - $g = \mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!]$
  - $F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{DS}[\![s]\!], id)$
- F is a **functional** (a function from functions to functions)
- $\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!]$ is a **fixed point** of the functional F

Fixed Points: Examples

- x is a fixed point of function f if $f(x) = x$ holds

- Consider a function $f : \mathbb{N} \to \mathbb{N}$
  - $f(x) = x + 1$ does not have a fixed point
  - $f(x) = 0$ has exactly one fixed point, 0
  - $f(x) = x^2$ has two fixed points, 0 and 1
  - $f(x) = x$ has an infinite number of fixed points

Direct Style Semantics of IMP: Loops

- Loop statement `while` b `do` s `end`

$$\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] = FIX\ F$$
$$\text{where } F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{DS}[\![s]\!], id)$$

- We write $FIX\ F$ to denote the fixed point of the functional F:

$$FIX\ :\ ((\text{State} \hookrightarrow \text{State}) \rightarrow (\text{State} \hookrightarrow \text{State})) \rightarrow (\text{State} \hookrightarrow \text{State})$$

- This defintion of $\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!]$ is compositional

Example

- Consider the statement

```
while x # 0 do skip end
```

- The functional for this loop is defined by

$$
\begin{aligned}
F'(g)\sigma &= cond(\mathcal{B}[\![\texttt{x\#0}]\!], g \circ \mathcal{S}_{DS}[\![\texttt{skip}]\!], id)\sigma \\
&= cond(\mathcal{B}[\![\texttt{x\#0}]\!], g \circ id, id)\sigma \\
&= cond(\mathcal{B}[\![\texttt{x\#0}]\!], g, id)\sigma \\
&= \begin{cases} g(\sigma) & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}
\end{aligned}
$$

Example (cont'd)

- The function

$$g_1(\sigma) = \begin{cases} \text{undefined} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

  is a fixed point of F'

- The function $g_2(\sigma) = \text{undefined}$ is not a fixed point for F'

Well-Definedness

$$\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!] = FIX\ F$$
$$\text{where } F(g) = cond(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{DS}[\![s]\!], id)$$

- The function $\mathcal{S}_{DS}[\![\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}]\!]$ is well-defined if $FIXF$ defines a **unique fixed point** for the functional F
  - There are functionals that have more than one fixed point
  - There are functionals that have no fixed point at all

Examples

- F' from the previous example has more than one fixed point

$$F'(g)\sigma = \begin{cases} g(\sigma) & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{otherwise} \end{cases}$$

  - Every function g' : State $\hookrightarrow$ State with $g'(\sigma) = \sigma$ if $\sigma(x) = 0$ is a fixed point for F'

- The functional F_1 has no fixed point if $g_1 \neq g_2$

$$F_1(g) = \begin{cases} g_1 & \text{if } g = g_2 \\ g_2 & \text{otherwise} \end{cases}$$

3. Denotational Semantics

Achieving Well-Definedness

- To make sure that there is exactly one fixed point for the functional F, we develop a framework where:

- we impose requirements on the fixed points and show that there is **at most one fixed point** fulfilling these requirements
- all functionals originating from statements in IMP **do have a fixed point** that satisfies these requirements

Requirements on the Fixed Point

- To motivate the requirements on the fixed points, we consider the three possible outcomes of a loop `while` b `do` s `end`:
  1. it terminates,
  2. it loops locally, that is, there is a construct in s that loops, or
  3. it loops globally, that is, the outer `while` construct loops

- We investigate the functional F and its fixpoints for these three cases

Terminating Loops

- Execution of `while` b `do` s `end` from state σ_0 terminates
- There are states $\sigma_0, \ldots, \sigma_n$ such that

$$\mathcal{B}[\![b]\!]\sigma_i = \begin{cases} tt & \text{if } i < n \\ ff & \text{if } i = n \end{cases}$$

  and $\mathcal{S}_{DS}[\![s]\!]\sigma_i = \sigma_{i+1}$ for $i < n$

- **Every fixed point** g_0 of F satisfies $g_0(\sigma_0) = \sigma_n$
- This case does not give us any help for choosing "the right" fixed point

Terminating Loops (cont'd)

- Let g_0 be any fixed point of F
- For $i < n$ we get

$$
\begin{aligned}
& g_0(\sigma_i) \\
&= F(g_0)\sigma_i \\
&= cond(\mathcal{B}[\![b]\!], g_0 \circ \mathcal{S}_{DS}[\![s]\!], id)\sigma_i \\
&= g_0(\mathcal{S}_{DS}[\![s]\!](\sigma_i)) \\
&= g_0(\sigma_{i+1})
\end{aligned}
$$

- For $i = n$ we get

$$
\begin{aligned}
& g_0(\sigma_n) \\
&= F(g_0)\sigma_n \\
&= cond(\mathcal{B}[\![b]\!], g_0 \circ \mathcal{S}_{DS}[\![s]\!], id)\sigma_n \\
&= id(\sigma_n) \\
&= \sigma_n
\end{aligned}
$$

- Thus, any fixed point g_0 satisfies $g_0(\sigma_0) = \sigma_n$

Local Looping

- Execution of `while` b `do` s `end` from state σ_0 loops locally
- There are states $\sigma_0, \ldots, \sigma_n$ such that $\mathcal{B}[\![b]\!]\sigma_i = tt$ for $i \leq n$ and

$$\mathcal{S}_{DS}[\![s]\!]\sigma_i = \begin{cases} \sigma_{i+1} & \text{if } i < n \\ \text{undefined} & \text{if } i = n \end{cases}$$

- **Every fixed point** g_0 of F satisfies $g_0(\sigma_0) =$ undefined
- This case does not give us any help for choosing "the right" fixed point

Local Looping (cont'd)

- Let g_0 be any fixed point of F
- For $i < n$ we get

$$
\begin{aligned}
& g_0(\sigma_i) \\
&= F(g_0)\sigma_i \\
&= cond(\mathcal{B}[\![b]\!], g_0 \circ \mathcal{S}_{DS}[\![s]\!], id)\sigma_i \\
&= g_0(\mathcal{S}_{DS}[\![s]\!](\sigma_i)) \\
&= g_0(\sigma_{i+1})
\end{aligned}
$$

- For $i = n$ we get

$$
\begin{aligned}
& g_0(\sigma_n) \\
&= F(g_0)\sigma_n \\
&= cond(\mathcal{B}[\![b]\!], g_0 \circ \mathcal{S}_{DS}[\![s]\!], id)\sigma_n \\
&= (g_0 \circ \mathcal{S}_{DS}[\![s]\!])(\sigma_n) \\
&= \text{undefined}
\end{aligned}
$$

- Thus, any fixed point g_0 satisfies $g_0(\sigma_0) = \text{undefined}$

Global Looping

- Execution of `while` b `do` s `end` from state σ_0 loops globally
- There are states $\sigma_0, \sigma_1, \ldots$ such that $\mathcal{B}[\![b]\!]\sigma_i = tt$ and $\mathcal{S}_{DS}[\![s]\!]\sigma_i = \sigma_{i+1}$ for all i
- Let g_0 be any fixed point of F
- Like in the other cases, we get $g_0(\sigma_i) = g_0(\sigma_{i+1})$ for all i
- Therefore, we get $g_0(\sigma_0) = g_0(\sigma_i)$ for all i
- We cannot determine the value of $g_0(\sigma_0)$ in this way

Global Looping: Example

- We revisit the example `while x#0 do skip end` with its functional F'

$$F'(g)\sigma = \begin{cases} g(\sigma) & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{otherwise} \end{cases}$$

- Every function g' : State $\hookrightarrow$ State with $g'(\sigma) = \sigma$ if $\sigma(x) = 0$ is a fixed point for F'
- However, we want to record the looping. Therefore, our **preferred fixed point** is g_1:

$$g_1(\sigma) = \begin{cases} \text{undefined} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

The Desired Fixed Point

$$g_1(\sigma) = \begin{cases} \text{undefined} & \text{if } \sigma(x) \neq 0 \\ \sigma & \text{if } \sigma(x) = 0 \end{cases}$$

- The property that distinguishes g_1 from all other fixed points g' of F' is that $g_1(\sigma) = \sigma' \Rightarrow g'(\sigma) = \sigma'$, but not vice versa

- **Requirement:** The desired fixed point $FIX\ F$ should be some partial function $g_0 : \text{State} \hookrightarrow \text{State}$ such that
  - g_0 is a fixed point of F, that is, $F(g_0) = g_0$
  - if g' is another fixed point of F, then $g_0(\sigma) = \sigma' \Rightarrow g'(\sigma) = \sigma'$ for all σ, σ'