Notes on Foundations of Programming Languages
Operational Semantics

Sandra Alves

September 25, 2019

Abstract

Semantics of programming languages allows us to specify meaning, behavior and properties, with the purpose of detecting errors/ambiguities, serve as a basis for implementation, program analysis and verification, etc. Semantics are usually divide into three main categories: operational, denotational and axiomatic.

Operational semantics: The meaning of a programming constructor is specified by the computation it induces when executed on an abstract machine. This type of semantics aims at answering the question *"How is a program executed?"*

Denotational semantics: The meaning of a programming constructor is modeled by a mathematical object that represents the effect of executing the constructor. This type of semantics aims at answering the question *"What is the effect of the program's execution?"*

Axiomatic semantics: Aims at specifying properties of the effect of executing the constructors as sets of pre and post conditions, whilst ignoring several aspects of execution itself. This type of semantics aims at answering the question *"What are the valid assertions after execution?"*

In this course we will focus on the first two categories. For full details on the notions here presented see [Winskel, 1993, Nielson and Nielson, 1992].

1 The IMP Programming Language

We start by describing a simple imperative language, named IMP[1]. We start by defining the list of syntactic sets:

- Num, consisting of the set integers;
- $\mathtt{T} = \{\mathtt{true}, \mathtt{false}\}$, consisting of the set of truth values;
- Var, consisting of an infinite enumerable set of program variables;
- AExp, consisting of the set of arithmetic expressions;
- BExp, consisting of the set of boolean expressions;
- Com, consisting of the set of commands.

We will use $n, m, \ldots$ to range over Num, $x, y, \ldots$ to range over Var, $a_0, a_1, \ldots$ to range over AExp, $b_0, b_1, \ldots$ to range over BExp and $c_0, c_1, \ldots$ to range over Com.

The sets AExp, BExp, and Com are given by the following grammars:

$$\begin{array}{rcl} a_1, a_2 \in \mathtt{AExp} & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\ b_1, b_2 \in \mathtt{BExp} & ::= & \mathtt{true} \mid \mathtt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\ c_1, c_2 \in \mathtt{Com} & ::= & \mathtt{skip} \mid x := a \mid c_1; c_2 \mid \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2 \mid \mathtt{while}\ b\ \mathtt{do}\ c \end{array}$$

2 Operational Semantics of IMP

Before presenting the rules for the natural operational semantics for IMP, we define the set of states Σ.

Definition 2.1 *The set of states Σ, consists of the functions $\sigma : \mathtt{Var} \to \mathtt{Num}$. The value of variable x is state σ is denoted by $\sigma(x)$.*

We now define the natural operational semantics for arithmetic expressions.

Definition 2.2 *We represent an evaluation relation between pairs $\langle a, \sigma \rangle$ and numbers n, such that $\langle a, \sigma \rangle \to n$, if the arithmetic expression a in state σ evaluates to n. Pairs of the form $\langle a, \sigma \rangle$, are called configurations. The evaluation relation is given by the following rules:*

$$\langle n, \sigma \rangle \to n$$

$$\langle x, \sigma \rangle \to \sigma(x)$$

$$\frac{\langle a_1, \sigma \rangle \to n_1 \qquad \langle a_2, \sigma \rangle \to n_2}{\langle a_1 + a_2, \sigma \rangle \to n} \quad \textit{where } n \textit{ is the sum of } n_1, n_2$$

$$\frac{\langle a_1, \sigma \rangle \to n_1 \qquad \langle a_2, \sigma \rangle \to n_2}{\langle a_1 - a_2, \sigma \rangle \to n} \quad \textit{where } n \textit{ is the difference between of } n_1, n_2$$

$$\frac{\langle a_1, \sigma \rangle \to n_1 \qquad \langle a_2, \sigma \rangle \to n_2}{\langle a_1 * a_2, \sigma \rangle \to n} \quad \textit{where } n \textit{ is the product of } n_1, n_2$$

[1] In some textbooks, a similar language is called WHILE.

We define the natural operational semantics for boolean expressions in a similar way.

Definition 2.3 *We represent an evaluation relation between pairs* $\langle b, \sigma\rangle$ *and truth values* t*, such that* $\langle b, \sigma\rangle \to t$*, if the boolean expression* b *in state* σ *evaluates to* b*. The evaluation relation is given by the following rules:*

$$\langle \texttt{true}, \sigma\rangle \to \texttt{true}$$

$$\langle \texttt{false}, \sigma\rangle \to \texttt{false}$$

$$\frac{\langle a_1, \sigma\rangle \to n_1 \qquad \langle a_2, \sigma\rangle \to n_2}{\langle a_1 = a_2, \sigma\rangle \to \texttt{true}} \quad \textit{if } n_1, n_2 \textit{ are equal}$$

$$\frac{\langle a_1, \sigma\rangle \to n_1 \qquad \langle a_2, \sigma\rangle \to n_2}{\langle a_1 = a_2, \sigma\rangle \to \texttt{false}} \quad \textit{if } n_1, n_2 \textit{ are different}$$

$$\frac{\langle a_1, \sigma\rangle \to n_1 \qquad \langle a_2, \sigma\rangle \to n_2}{\langle a_1 \leq a_2, \sigma\rangle \to \texttt{true}} \quad \textit{if } n_1 \textit{ is less or equal than } n_2$$

$$\frac{\langle a_1, \sigma\rangle \to n_1 \qquad \langle a_2, \sigma\rangle \to n_2}{\langle a_1 \leq a_2, \sigma\rangle \to \texttt{false}} \quad \textit{if } n_1 \textit{ is greater that } n_2$$

$$\frac{\langle b, \sigma\rangle \to \texttt{true}}{\langle \neg b, \sigma\rangle \to \texttt{false}} \qquad \frac{\langle b, \sigma\rangle \to \texttt{false}}{\langle \neg b, \sigma\rangle \to \texttt{true}}$$

$$\frac{\langle b_1, \sigma\rangle \to t_1 \qquad \langle b_2, \sigma\rangle \to t_2}{\langle b_1 \wedge b_2, \sigma\rangle \to \texttt{true}} \quad \textit{if } t_1 \textit{ and } t_2 \textit{ are both } \texttt{true}$$

$$\frac{\langle b_1, \sigma\rangle \to t_1 \qquad \langle b_2, \sigma\rangle \to t_2}{\langle b_1 \wedge b_2, \sigma\rangle \to \texttt{false}} \quad \textit{if } t_1 \textit{ or } t_2 \textit{ is } \texttt{false}$$

More efficiently,

$$\frac{\langle b_1, \sigma\rangle \to \texttt{false}}{\langle b_1 \wedge b_2, \sigma\rangle \to \texttt{false}} \qquad \frac{\langle b_1, \sigma\rangle \to \texttt{true} \qquad \langle b_2, \sigma\rangle \to t}{\langle b_1 \wedge b_2, \sigma\rangle \to t}$$

We now define the evaluation relation for commands in $\texttt{Com}$.

Definition 2.4 *We consider the following definitions on states.*

1. *The initial state, denoted* σ_0 *is a total function such that, for every* $x \in \mathrm{Var}$*,* $\sigma_0(x) = 0$*.*
2. *Let* $\sigma \in \Sigma$*,* $n \in \texttt{Num}$ *and* $x \in \texttt{Var}$*. We write* $\sigma[n/x]$ *to denote the state defined in the following way:*

$$\sigma[n/x](y) = \begin{cases} n & \textit{if } x = y \\ \sigma(y) & \textit{otherwise} \end{cases}$$

Definition 2.5 *We represent an evaluation relation between pairs* $\langle c, \sigma\rangle$ *and states* σ'*, such that* $\langle c, \sigma\rangle \to \sigma'$*, if the command* c *in state* σ *evaluates to state* σ'*. The evaluation relation is given* *by the following rules:*

$$\langle \texttt{skip}, \sigma\rangle \to \sigma \qquad\qquad \frac{\langle a, \sigma\rangle \to n}{\langle x := a, \sigma\rangle \to \sigma[n/x]}$$

$$\frac{\langle c_1, \sigma\rangle \to \sigma'' \qquad \langle c_2, \sigma''\rangle \to \sigma'}{\langle c_1; c_2, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \texttt{true} \qquad \langle c_1, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'} \qquad \frac{\langle b, \sigma\rangle \to \texttt{false} \qquad \langle c_2, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \texttt{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma} \qquad \frac{\langle b, \sigma\rangle \to \texttt{false} \quad \langle c, \sigma\rangle \to \sigma'' \quad \langle \texttt{while } b \texttt{ do } c, \sigma''\rangle \to \sigma'}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma'}$$

A structural operational semantics for IMP

Definition 2.6 *The structural (one-step) operational semantics for IMP is defined by the following rules:*

$$\langle \texttt{skip}, \sigma\rangle \Rightarrow \sigma \qquad \langle x := a, \sigma\rangle \Rightarrow \sigma[a/x] \quad \textit{if } \langle a, \sigma\rangle \to n$$

$$\frac{\langle c_1, \sigma\rangle \to \sigma'}{\langle c_1; c_2, \sigma\rangle \Rightarrow \langle c_2, \sigma'\rangle} \qquad \frac{\langle c_1, \sigma\rangle \to \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \Rightarrow \langle c_1'; c_2, \sigma'\rangle}$$

$$\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \;\Rightarrow\; \langle c_1, \sigma\rangle \quad \textit{if } \langle b, \sigma\rangle \to \texttt{true}$$

$$\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \;\Rightarrow\; \langle c_2, \sigma\rangle \quad \textit{if } \langle b, \sigma\rangle \to \texttt{false}$$

$$\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \;\Rightarrow\; \langle \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}, \sigma\rangle$$

2.1 Properties of the operational semantics for IMP

We start by defining the notions of semantic equivalence.

Definition 2.7 *We define equivalence between arithmetic/boolean expressions and commands in the, respectively denoted* $a_1 \sim a_2$*,* $b_1 \sim b_2$ *and* $c_1 \sim c_2$*, in the following way:*

$$a_1 \sim a_2 \quad \textit{iff} \quad \forall n \in \texttt{Num},\ \sigma \in \Sigma.\ \ \langle a_1, \sigma\rangle \to n \quad \Leftrightarrow \quad \langle a_2, \sigma\rangle \to n$$

$$b_1 \sim b_2 \quad \textit{iff} \quad \forall t \in \mathrm{T},\ \sigma \in \Sigma.\ \ \langle b_1, \sigma\rangle \to t \quad \Leftrightarrow \quad \langle b_2, \sigma\rangle \to t$$

$$c_1 \sim c_2 \quad \textit{iff} \quad \forall \sigma, \sigma' \in \Sigma.\ \ \langle c_1, \sigma\rangle \to \sigma' \quad \Leftrightarrow \quad \langle c_2, \sigma\rangle \to \sigma'$$

Proposition 2.8 *Let* $w \equiv \texttt{while } b \texttt{ do } c$*, then:*

$$w \sim \texttt{if } b \texttt{ then } c; w \texttt{ else skip}$$

Proof: We want to show that

$$\langle w, \sigma\rangle \to \sigma' \ \text{ iff } \ \langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip}, \sigma\rangle \to \sigma', \quad \forall \sigma, \sigma' \in \Sigma.$$

$(\Rightarrow)$: Suppose that $\langle w, \sigma\rangle \to \sigma'$. Then there exist two possible derivations for $\langle w, \sigma\rangle \to \sigma'$:

1)

$$\dfrac{\begin{array}{c}\vdots\\ \langle b,\sigma\rangle \to \texttt{false}\end{array}}{\langle w,\sigma\rangle \to \sigma}$$

from which we can build the following derivation:

$$\dfrac{\begin{array}{c}\vdots\\ \langle b,\sigma\rangle \to \texttt{false}\end{array} \qquad \langle \texttt{skip},\sigma\rangle \to \sigma}{\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip},\sigma\rangle \to \sigma}$$

2)

$$\dfrac{\begin{array}{c}\vdots\\ \langle b,\sigma\rangle \to \texttt{true}\end{array} \qquad \begin{array}{c}\vdots\\ \langle c,\sigma\rangle \to \sigma''\end{array} \qquad \begin{array}{c}\vdots\\ \langle w,\sigma''\rangle \to \sigma'\end{array}}{\langle w,\sigma\rangle \to \sigma'}$$

from which we can build the following derivation:

$$\dfrac{\begin{array}{c}\vdots\\ \langle b,\sigma\rangle \to \texttt{true}\end{array} \qquad \dfrac{\begin{array}{c}\vdots\\ \langle c,\sigma\rangle \to \sigma''\end{array} \qquad \begin{array}{c}\vdots\\ \langle w,\sigma''\rangle \to \sigma'\end{array}}{\langle c; w,\sigma\rangle \to \sigma'}}{\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip},\sigma\rangle \to \sigma'}$$

$\therefore\ \langle w,\sigma\rangle \to \sigma' \Rightarrow \langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip},\sigma\rangle \to \sigma'$. ($\Leftarrow$) : The reverse implication is proved in a similar way. $\square$

The natural operational semantics for arithmetic/boolean expressions and commands is deterministic.

Theorem 2.9 *Let* $c \in \texttt{Com}$ *and* $\sigma, \sigma', \sigma'' \in \Sigma$, *if* $\langle c,\sigma\rangle \to \sigma'$ *and* $\langle c,\sigma\rangle \to \sigma''$, *then* $\sigma' = \sigma''$.

Proof: Let us assume that $\langle c,\sigma\rangle \to \sigma'$. We need to prove that, if $\langle c,\sigma\rangle \to \sigma''$ then $\sigma' = \sigma''$.

- **[skip]:** There is only one axiom producing $\langle \texttt{skip},\sigma\rangle \to \sigma'$ and $\langle \texttt{skip},\sigma\rangle \to \sigma''$, therefore $\sigma' = \sigma'' = \sigma$.
- **[atrb]:** Similar to the previous case, taking into account determinism for $\langle a,\sigma\rangle$. In both cases $\sigma' = \sigma'' = [n/x]$.
- **[seqn]:** If $\langle c_1; c_2, \to\rangle\sigma'$, which follows from $\langle c_1,\sigma\rangle \to \sigma_0$ and $\langle c_2,\sigma_0\rangle \to \sigma'$. If $\langle c_1; c_2,\sigma\rangle \to \sigma''$, then this follows from $\langle c_1,\sigma\rangle \to \sigma_1$ and $\langle c_2,\sigma_1\rangle \to \sigma''$. By induction, from $\langle c_1,\sigma\rangle \to \sigma_0$ and $\langle c_1,\sigma\rangle \to \sigma_1$ it follows that $\sigma_0 = \sigma_1$. Similarly, from $\langle c_2,\sigma_0\rangle \to \sigma'$ and $\langle c_2,\sigma_0\rangle \to \sigma''$ it follows that $\sigma' = \sigma''$.
- **[if$_{\texttt{true}}$]:** If $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \to \sigma'$, which follows from $\langle b,\sigma\rangle \to \texttt{true}$ and $\langle c_1,\sigma\rangle \to \sigma'$. Since the operational semantics of boolean expressions is deterministic, then $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2,\sigma\rangle \to \sigma''$ follows from $\langle c_1,\sigma\rangle \to \sigma''$. By induction hypothesis, $\langle c_1,\sigma\rangle \to \sigma'$ and $\langle c_1,\sigma\rangle \to \sigma''$, imply that $\sigma' = \sigma''$.
- **[if$_{\texttt{false}}$]:** Analogous to the previous case.
- **[while$_{\texttt{true}}$]:** If $\langle \texttt{while } b \texttt{ do } c,\sigma\rangle \to \sigma'$, which follows from $\langle b,\sigma\rangle \to \texttt{true}$, $\langle c,\sigma\rangle \to \sigma_0$ and $\langle \texttt{while } b \texttt{ do } c,\sigma_0\rangle \to \sigma'$. Since the operational semantics of boolean expressions is deterministic, $\langle \texttt{while } b \texttt{ do } c,\sigma\rangle \to \sigma''$, which follows from $\langle b,\sigma\rangle \to \texttt{true}$, $\langle c,\sigma\rangle \to \sigma_1$ and $\langle \texttt{while } b \texttt{ do } c,\sigma_1\rangle \to \sigma''$. By induction hypothesis $\langle c,\sigma\rangle \to \sigma_0$ and $\langle c,\sigma\rangle \to \sigma_1$ imply that $\sigma_0 = \sigma_1$. And $\langle \texttt{while } b \texttt{ do } c,\sigma_0\rangle \to \sigma'$ and $\langle \texttt{while } b \texttt{ do } c,\sigma_0\rangle \to \sigma''$ imply that $\sigma' = \sigma''$.

- **[while$_{\texttt{false}}$]:** Analogous to the previous case.

$\square$

Definition 2.10 *Considering the relations defined above, we define the following semantic functions:*

$$\mathcal{S}_{sn}[\![c]\!]\sigma \;=\; \begin{cases} \sigma' & \textit{if } \langle c,\sigma\rangle \to \sigma' \\ \textit{not def} & \textit{otherwise} \end{cases} \qquad \mathcal{S}_{ss}[\![c]\!]\sigma \;=\; \begin{cases} \sigma' & \textit{if } \langle c,\sigma\rangle \Rightarrow^{*} \sigma' \\ \textit{not def} & \textit{otherwise} \end{cases}$$

In the structural operational semantics, a derivation sequence of a command c, with initial state σ_0 is:

1. a finite sequence of configurations $\gamma_0, \gamma_1, \ldots, \gamma_k$, written as:

$$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_k$$

   such $\gamma_0 = \langle c, \sigma_0\rangle$, $\gamma_i \Rightarrow \gamma_{i+1}$, for $0 \le i < k$, and where γ_k is a terminal or blocked[2] configuration.

2. an infinite sequence of configurations $\gamma_0, \gamma_1, \ldots$ written as:

$$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots$$

   such $\gamma_0 = \langle c, \sigma_0\rangle$, $\gamma_i \Rightarrow \gamma_{i+1}$, for $i \ge 0$.

We write $\gamma_0 \Rightarrow^{i} \gamma_i$ to indicate that there exist i reduction steps in the execution from γ_0 to γ_i.

Lemma 2.11 *If $\langle c_1; c_2, \sigma\rangle \Rightarrow^{k} \sigma''$ then there exists a state σ' and natural numbers k_1, k_2 such that $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma'$, $\langle c_2, \sigma'\rangle \Rightarrow^{k_2} \sigma''$ and $k = k_1 + k_2$.*

Proof: By induction on k. For $k = 0$ the property holds trivially.

Suppose that the result holds for $k \le k_0$, let us show that it also holds for k_0+1. Let $\langle c_1; c_2, \sigma\rangle \Rightarrow^{k_0+1} \sigma''$, which means that, for some configuration γ, we have:

$$\langle c_1; c_2, \sigma\rangle \Rightarrow \gamma \Rightarrow^{k_0} \sigma''$$

We then have two cases:

1. $\langle c_1; c_2, \sigma\rangle \Rightarrow \langle c_1'; c_2, \sigma'\rangle$, which follows from $\langle c_1, \sigma\rangle \Rightarrow \langle c_1', \sigma'\rangle$. We then have $\langle c_1'; c_2, \sigma'\rangle \Rightarrow^{k_0} \sigma''$. By induction hypothesis, $\langle c_1', \sigma'\rangle \Rightarrow^{k_1} \sigma_0$ and $\langle c_2, \sigma_0\rangle \Rightarrow^{k_2} \sigma''$ and $k_0 = k_1 + k_2$. From $\langle c_1, \sigma\rangle \Rightarrow \langle c_1', \sigma'\rangle \Rightarrow^{k_1} \sigma_0$ if follows that $\langle c_1, \sigma\rangle \Rightarrow^{k_0+1} \sigma_0$. And we have $\langle c_2, \sigma_0\rangle \Rightarrow^{k_2} \sigma''$. The result follows with $k_0 + 1 = (k_1 + 1) + k_2$.

2. $\langle c_1; c_2, \sigma\rangle \Rightarrow \sigma'$, therefore $\gamma = \langle c_2, \sigma'\rangle \Rightarrow^{k_0} \sigma''$. The result follows taking $k_1 = 1$ and $k_2 = k_0$.

$\square$

Lemma 2.12 *If $\langle c_1, \sigma\rangle \Rightarrow^{k} \sigma'$, then $\langle c_1; c_2, \sigma\rangle \Rightarrow^{k} \langle c_2, \sigma'\rangle$.*

Proof: Proposed as an exercise. $\square$

The structural operational semantics is also deterministic.

Theorem 2.13 *If $\langle c, \sigma\rangle \Rightarrow \gamma$ and $\langle c, \sigma\rangle \Rightarrow \gamma'$, then $\gamma = \gamma'$.*

We now prove the equivalence between the two notions of operational semantics.

[2] A configuration γ_k is blocked, if there is no γ such that $\gamma_k \Rightarrow \gamma$

Theorem 2.14 *For every* $c \in \texttt{Com}$, $\mathcal{S}_{sn}[\![c]\!] = \mathcal{S}_{ss}[\![c]\!]$.

Proof: We need to show the following two implications

$$\begin{array}{lcl} \langle c, \sigma\rangle \to \sigma' & \Rightarrow & \langle c, \sigma\rangle \Rightarrow^* \sigma' \\ \langle c, \sigma\rangle \Rightarrow^k \sigma' & \Rightarrow & \langle c, \sigma\rangle \to \sigma' \end{array}$$

$(\Rightarrow)$: By induction on the derivation tree of $\langle c, \sigma\rangle \to \sigma'$.

- **[skip]:** We have $\langle \texttt{skip}, \sigma\rangle \to \sigma$ and $\langle \texttt{skip}, \sigma\rangle \Rightarrow \sigma$.
- **[assn]:** We have $\langle x := a, \sigma\rangle \to \sigma[n/x]$, that follows from $\langle a, \sigma\rangle \to n$, from which we get $\langle x := a, \sigma\rangle \Rightarrow \sigma[n/x]$
- **[seqn]:** Let $\langle c_1; c_2, \sigma\rangle \to \sigma'$, which follows from $\langle c_1, \sigma\rangle \to \sigma''$ and $\langle c_2, \sigma''\rangle \to \sigma'$.By induction hypothesis we have $\langle c_1, \sigma\rangle \Rightarrow^* \sigma''$ and $\langle c_2, \sigma''\rangle \Rightarrow^* \sigma'$. By a previous lemma, $\langle c_1, \sigma\rangle \Rightarrow^* \sigma''$ implies that $\langle c_1; c_2, \sigma\rangle \Rightarrow^* \langle c_2, \sigma''\rangle$. Therefore

$$\langle c_1; c_2, \sigma\rangle \Rightarrow^* \langle c_2, \sigma''\rangle \Rightarrow^* \sigma'.$$

- **[if$_{\texttt{true}}$]:** Let $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'$, which follows from $\langle b, \sigma\rangle \to \texttt{true}$ and $\langle c_1, \sigma\rangle \to \sigma'$. By induction hypothesis, $\langle c_1, \sigma\rangle \Rightarrow^* \sigma'$. Since $\langle b, \sigma\rangle \to \texttt{true}$ we then have:

$$\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \Rightarrow \langle c_1, \sigma\rangle \Rightarrow^* \sigma'.$$

- **[if$_{\texttt{false}}$]:** Analogous to the previous case.
- **[while$_{\texttt{true}}$]:** Let $\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma'$, which follows from $\langle b, \sigma\rangle \to \texttt{true}$, $\langle c, \sigma\rangle \to \sigma''$ and $\langle \texttt{while } b \texttt{ do } c, \sigma''\rangle \to \sigma'$. By induction hypothesis, $\langle c, \sigma\rangle \Rightarrow^* \sigma''$ and $\langle \texttt{while } b \texttt{ do } c, \sigma''\rangle \Rightarrow^* \sigma'$. Since $\langle b, \sigma\rangle \to \texttt{true}$ we then have:

$$\begin{array}{lcl} \langle \texttt{while } b \texttt{ do } c, \sigma\rangle & \Rightarrow & \langle \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}, \sigma\rangle \\ & \Rightarrow & \langle c; \texttt{while } b \texttt{ do } c, \sigma\rangle \\ & \Rightarrow^* & \langle \texttt{while } b \texttt{ do } c, \sigma''\rangle \\ & \Rightarrow^* & \sigma' \end{array}$$

- **[while$_{\texttt{false}}$]:** Let $\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma$, which follows from $\langle b, \sigma\rangle \to \texttt{false}$. we then have:

$$\begin{array}{lcl} \langle \texttt{while } b \texttt{ do } c, \sigma\rangle & \Rightarrow & \langle \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}, \sigma\rangle \\ & \Rightarrow & \langle \texttt{skip}, \sigma\rangle \\ & \Rightarrow & \sigma \end{array}$$

$(\Leftarrow)$: By induction on k, where k is the number of steps in the sequence $\langle c, \sigma\rangle \Rightarrow^k \sigma'$. If $k = 0$, the result follows trivially. Supposing that the property is true for $k \leq k_0$, we will show that it also is true for $k_0 + 1$. We proceed by cases over the first step of $\langle c, \sigma\rangle \Rightarrow^{k_0+1} \sigma'$.

- **[skip,assn]:** Both cases are immediate with $k_0 = 0$.
- **[seqn]:** Let $\langle c_1; c_2, \sigma\rangle \Rightarrow^{k_0+1} \sigma'$. It follows from a previous lemma that $\langle c_1, \sigma\rangle \Rightarrow^{k_1} \sigma''$ and $\langle c_2, \sigma''\rangle \Rightarrow^{k_2} \sigma'$, with $k_0 + 1 = k_1 + k_2$. By induction hypothesis we then have $\langle c_1, \sigma\rangle \to \sigma''$ and $\langle c_2, \sigma''\rangle \to \sigma'$, from which follows that $\langle c_1; c_2, \sigma\rangle \to \sigma'$.

- **[if$_{\texttt{true}}$]:** If $\langle b, \sigma\rangle \to \texttt{true}$, then we have:

$$\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \Rightarrow \langle c_1, \sigma\rangle \Rightarrow^{k_0} \sigma'.$$

  By induction hypothesis $\langle c_1, \sigma\rangle \to \sigma'$. From $\langle b, \sigma\rangle \to \texttt{true}$ and $\langle c_1, \sigma\rangle \to \sigma'$, it follows that $\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'$.

- **[if$_{\texttt{false}}$]:** Analogous to the previous case.

- **[while]:** We have

$$\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \Rightarrow \langle \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}, \sigma\rangle \Rightarrow^{k_0} \sigma'.$$

  By induction hypothesis $\langle \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}, \sigma\rangle \to \sigma'$, which implies, by the equivalence $\texttt{while } b \texttt{ do } c \sim \texttt{if } b \texttt{ then } c; \texttt{while } b \texttt{ do } c \texttt{ else skip}$, $\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma'$.

□

Exercises

1 Considering $\sigma \equiv \sigma_0[3/x, 5/y]$ calculate:

(a) $\langle x + 1, \sigma\rangle$

(b) $\langle \neg(x = 1), \sigma\rangle$

(c) $\langle \neg(x \leq 1) \wedge (y = 5), \sigma\rangle$

2 Considering $\sigma_0[3/x]$, determine the state after the execution of:

$$y := 1; \texttt{while } \neg(x = 1) \texttt{ do } (y := y \times x; x := x - 1)$$

3 Considering $\sigma_0[10/x, 5/y]$, determine the state after the execution of:

$$z := 0; \texttt{while } y \leq x \texttt{ do } (z := z + 1; x := x - y)$$

4 Consider the structural operational semantics for IMP. Write all the evaluation steps for the following commands:

(a) $z := x; (x := y; y := z)$ with initial state $\sigma_0 \equiv [3/x; 5/y]$;

(b) $\texttt{while } x < 3 \texttt{ do } x := x * 2$ with initial state $\sigma_0 \equiv [0/x]$.

5 Show the equivalence between the commands $(c_1; c_2); c_3$ and $c_1; (c_2; c_3)$.

6 Recall the natural operational semantics for arithmetic and boolean expressions.

(a) Define structural operational semantics for arithmetic and boolean expressions.

(b) Show the equivalence between the natural and structural operational semantics for arithmetic and boolean expressions.

7 Consider the command $\texttt{repeat } c \texttt{ until } b$.

(a) Define an operational natural semantics for this command (without using the semantics of $\texttt{while}$).

(b) Show that $\texttt{repeat}\ c\ \texttt{until}\ b$ is semantically equivalent to

$$c; \texttt{if}\ b\ \texttt{then skip else}\ (\texttt{repeat}\ c\ \texttt{until}\ b)$$

(c) Show that $\texttt{repeat}\ c\ \texttt{until}\ b$ is semantically equivalent to

$$c; \texttt{while}\ \neg b\ \texttt{do}\ c$$

8 Prove that, if $\langle c_1, \sigma\rangle \Rightarrow^k \sigma'$, then $\langle c_1; c_2, \sigma\rangle \Rightarrow^k \langle c_2, \sigma'\rangle$.

9 Prove that, if $\langle c, \sigma\rangle \Rightarrow \gamma$ and $\langle c, \sigma\rangle \Rightarrow \gamma'$, then $\gamma = \gamma'$.

References

[Nielson and Nielson, 1992] Nielson, H. R. and Nielson, F. (1992). *Semantics with Applications: A Formal Introduction*. John Wiley & Sons, Inc., New York, NY, USA.

[Winskel, 1993] Winskel, G. (1993). *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge, MA, USA.