AAA616: Program Analysis

Lecture 3 — Operational Semantics

Hakjoo Oh
2024 Fall

Plan

- Notation
- Big-step operational semantics for **IMP**
- Small-step operational semantics for **IMP**

Logical Notation

For statements A and B,

- $A \,\&\, B$: A and B, the conjunction of A and B
- $A \implies B$: A implies B, if A then B
- $A \iff B$: A iff B, the logical equivalence of A and B
- $\neg A$: not A

Logical Notation

Logical quantifiers $\exists$ and $\forall$:

- $\exists x.\ P(x)$: for some x, $P(x)$
- $\forall x.\ P(x)$: for all x, $P(x)$
- Abbreviations:
  - $\exists x, y, \ldots, z.\ P(x, y, \ldots, z) \equiv \exists x \exists y \ldots \exists z.\ P(x, y, \ldots, z)$
  - $\forall x, y, \ldots, z.\ P(x, y, \ldots, z) \equiv \forall x \forall y \ldots \forall z.\ P(x, y, \ldots, z)$
  - $\forall x \in X.\ P(x) \equiv \forall x.\ x \in X \implies P(x)$
  - $\exists x \in X.\ P(x) \equiv \exists x.\ x \in X \ \&\ P(x)$
  - $\exists! x.\ P(x) \equiv (\exists x.\ P(x)) \ \&\ (\forall y, z.\ P(y) \ \&\ P(z) \implies y = z)$

Sets

- A set is a collection of objects (also called elements or members)
- $a \in X$: a is an element of the set X
- A set X is a subset of a set Y, $X \subseteq Y$, iff every element of X is an element of Y:

$$X \subseteq Y \iff \forall z \in X.\ z \in Y.$$

- Sets X and Y are equal, $X = Y$, iff $X \subseteq Y$ and $Y \subseteq X$.
- $\emptyset$: empty set
- ω: the set of natural numbers $0, 1, 2, \ldots$

Constructions on Sets

- Comprehension: If X is a set and $P(x)$ is a property, the set

$$\{x \in X \mid P(x)\}$$

  denotes the subset of X consisting of all elements x of X which satisfy $P(x)$.
- Powerset: the set of all subsets of a set:

$$\wp(X) = \{Y \mid Y \subseteq X\}.$$

- Indexed sets: Suppose I is a set and that for any $i \in I$ there is a unique object x_i. Then

$$\{x_i \mid i \in I\}$$

  is a set. The elements x_i is *indexed* by the elements $i \in I$.

Constructions on Sets

- Union and intersection:

$$\begin{array}{rcl} X \cup Y & = & \{a \mid a \in X \text{ or } a \in Y\} \\ X \cap Y & = & \{a \mid a \in X \ \& \ a \in Y\} \end{array}$$

- Big union and intersection: When X is a set of sets,

$$\begin{array}{rcl} \bigcup X & = & \{a \mid \exists x \in X.\ a \in x\} \\ \bigcap X & = & \{a \mid \forall x \in X.\ a \in x\} \end{array}$$

When $X = \{x_i \mid i \in I\}$ for some index set I,

$$\bigcup_{i \in I} x_i = \bigcup X$$

$$\bigcap_{i \in I} x_i = \bigcap X$$

Constructions on Sets

- Disjoint union:

$$X \uplus Y = (\{0\} \times X) \cup (\{1\} \times Y).$$

- Product: For sets X and Y, their product is the set

$$X \times Y = \{(a, b) \mid a \in X \ \& \ b \in Y\}.$$

In general,

$$X_1 \times X_2 \times \cdots \times X_n = \{(x_1, x_2, \ldots, x_n) \mid \forall i \in [1, n].\ x_i \in X_i\}.$$

- Set difference:

$$X \setminus Y = \{x \mid x \in X \ \& \ x \notin Y\}.$$

Relations and Functions

- A binary relation R between X and Y is an element of $\wp(X \times Y)$, $R \in \wp(X \times Y)$, or $R \subseteq X \times Y$.
- When R is a binary relation $R \subseteq X \subseteq Y$, we write xRy for $(x, y) \in R$.
- A partial function f from X to Y is a relation $f \subseteq X \times Y$ such that

$$\forall x, y, y'. \ (x, y) \in f \ \& \ (x, y') \in f \implies y = y'.$$

- We use the notation $f(x) = y$ when there is y such that $(x, y) \in f$ and say $f(x)$ is *defined*, and otherwise $f(x)$ is *undefined*.
- A total function from X to Y is a partial function such that $f(x)$ is defined for all $x \in X$.
- $(X \hookrightarrow Y)$: the set of all partial functions from X to Y
- $(X \rightarrow Y)$: the set of all total functions from X to Y
- $\lambda x. \ e$: the lambda notation for functions

Relations and Functions

- Composition: When $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ are binary relations, their composition is a relation of type $X \times Z$ defined as,

  $$S \circ R = \{(x, z) \in X \times Z \mid \exists y \in Y.\ (x, y) \in R \ \&\ (y, z) \in S\}$$

- $Id_X = \{(x, x) \mid x \in X\}$

Relations and Functions

- An equivalence relation on X is a relation $R \subseteq X \times X$ which is
  - reflexive: $\forall x \in X.\ xRx$,
  - symmetric: $\forall x, y \in X.\ xRy \implies yRx$, and
  - transitive: $\forall x, y, z \in X.\ xRy \ \&\ yRz \implies xRz$.
- Example: $=$ on numbers, the relation "has the same age" on people
- We sometime write $x \equiv y \pmod{R}$ for $(x, y) \in R$.
- The equivalence class of x under R, denoted $\{x\}_R$ or $[x]_R$:

$$[x]_R = \{y \in X \mid xRy\}.$$

- Quotient set: the set of all equivalence classes of X by R:

$$X/R = \{[x]_R \mid x \in X\}.$$

- For any equivalence relation R, X/R is a partition of X.

Relations and Functions

- Let R be a relation on a set X. Define $R^0 = Id_X$, and $R^1 = R$, and
$$R^{n+1} = R \circ R^n.$$
- The transitive closure of R:
$$R^+ = \bigcup_{n \in \omega} R^{n+1}$$
- The reflexive transitive closure of R:
$$R^* = \bigcup_{n \in \omega} R^n = Id_X \cup R^+.$$

Sequences

- Given a set S, S^+ denotes the set of all finite nonempty sequences of elements of S
- When σ is a finite sequence, σ_k denotes the $(k+1)$th element of the sequence, σ_0 the first element, and $\sigma_\dashv$.
- Given a sequence $\sigma \in S^+$ and an element $s \in S$, $\sigma \cdot s$ denotes a sequence obtained by appending s to σ.

Syntax vs. Semantics

A programming language is defined with syntax and semantics.

- The syntax is concerned with the grammatical structure of programs.
  - Context-free grammar
- The semantics is concerned with the meaning of grammatically correct programs.
  - Operational semantics: The meaning is specified by the computation steps executed on a machine. It is of intrest how it is obtained.
  - Denotational semantics: The meaning is modeled by mathematical objects that represent the effect of executing the program. It is of interest the effect, not how it is obtained.
  - Axiomatic semantics: The meaning is given as proof rules within a program logic. It is of interest how to prove program correctness.

IMP: Abstract Syntax

n, m will range over numerals, **N**
t will range over truth values, **T**$= \{$ **true**, **false** $\}$
X, Y will range over locations, **Loc**
a will range over arithmetic expressions, **Aexp**
b will range over boolean expressions, **Bexp**
c will range over statements, **Com**

$$
\begin{array}{lcl}
a & ::= & n \mid X \mid a_0 + a_1 \mid a_0 \star a_1 \mid a_0 - a_1 \\
b & ::= & \texttt{true} \mid \texttt{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1 \\
c & ::= & X := a \mid \texttt{skip} \mid c_0; c_1 \mid \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1 \mid \texttt{while } b \texttt{ do } c
\end{array}
$$

Example

The factorial program:

```
Y:=1; while ¬(X=1) do (Y:=Y⋆X; X:=X-1)
```

The abstract syntax tree:

States

- The meaning of a program depends on the values bound to the locations that occur in the program, e.g., $\boldsymbol{X + 3}$.
- A state is a function from locations to values:

$$\boldsymbol{\sigma, s \in \Sigma = \mathbf{Loc} \to \mathbf{N}}$$

- Let $\boldsymbol{\sigma}$ be a state. Let $\boldsymbol{m \in \mathbf{N}}$. Let $\boldsymbol{X \in \mathbf{Loc}}$. We write $\boldsymbol{\sigma[m/X]}$ (or $\boldsymbol{\sigma[X \mapsto m]}$) for the state obtained from $\boldsymbol{\sigma}$ by replacing its contents in $\boldsymbol{X}$ by $\boldsymbol{m}$, i.e.,

$$\boldsymbol{\sigma[m/X](Y) = \sigma[X \mapsto m] = \begin{cases} m & \text{if } Y = X \\ \sigma(Y) & \text{if } Y \neq X \end{cases}}$$

- $\boldsymbol{\Sigma_\perp = \Sigma \cup \{\perp\}}$

Operational Semantics

Operational semantics is concerned about how to execute programs and not merely what the execution results are.

- *Big-step operational semantics* describes how the overall results of executions are obtained.
- *Small-step operational semantics* describes how the individual steps of the computations take place.

In both kinds, the semantics is specified by a transition system $(\mathbb{S}, \to)$ where $\mathbb{S}$ is the set of configurations with two types (for $\mathbf{Aexp}$):

- $\langle \boldsymbol{a}, \boldsymbol{\sigma} \rangle$: a nonterminal configuration, i.e. the expression $\boldsymbol{a}$ is to be evaluated in the state $\boldsymbol{\sigma} \in \boldsymbol{\Sigma} = \mathbf{Loc} \to \mathbf{N}$
- $\boldsymbol{n}$: a terminal configuration

The transition relation $(\to) \subseteq \mathbb{S} \times \mathbb{S}$ describes how the execution takes place. The difference between the two approaches are in the definitions of transition relation.

Evaluation of Arithmetic Expressions

$$\frac{}{\langle n, \sigma \rangle \to n}$$

$$\frac{}{\langle X, \sigma \rangle \to \sigma(X)}$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 + a_1, \sigma \rangle \to n_0 + n_1}$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 - a_1, \sigma \rangle \to n_0 - n_1}$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \qquad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \star a_1, \sigma \rangle \to n_0 \star n_1}$$

Example

When $\sigma(X) = 0$,

$$\langle (X + 5) + (7 + 9), \sigma \rangle \to 21$$

Semantic Equivalence of Arithmetic Expressions

$$a_0 \sim a_1 \text{ iff } (\forall n \in \mathbf{N} \forall \sigma \in \Sigma.\ \langle a_0, \sigma \rangle \to n \iff \langle a_1, \sigma \rangle \to n)$$

Evaluation of Boolean Expressions

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

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 = a_1, \sigma \rangle \to \mathbf{true}} \; n_0 = n_1 \qquad \frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 = a_1, \sigma \rangle \to \mathbf{false}} \; n_0 \neq n_1$$

$$\frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \leq a_1, \sigma \rangle \to \mathbf{true}} \; n_0 \leq n_1 \qquad \frac{\langle a_0, \sigma \rangle \to n_0 \quad \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \leq a_1, \sigma \rangle \to \mathbf{false}} \; n_0 > n_1$$

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

$$\frac{\langle b_0, \sigma \rangle \to \mathbf{true} \quad \langle b_1, \sigma \rangle \to \mathbf{true}}{\langle b_0 \wedge b_1, \sigma \rangle \to \mathbf{true}} \qquad \frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \wedge b_1, \sigma \rangle \to \mathbf{false}} \; \mathbf{false} \in \{t_0, t_1\}$$

$$\frac{\langle b_0, \sigma \rangle \to \mathbf{false} \quad \langle b_1, \sigma \rangle \to \mathbf{false}}{\langle b_0 \vee b_1, \sigma \rangle \to \mathbf{false}} \qquad \frac{\langle b_0, \sigma \rangle \to t_0 \quad \langle b_1, \sigma \rangle \to t_1}{\langle b_0 \vee b_1, \sigma \rangle \to \mathbf{true}} \; \mathbf{true} \in \{t_0, t_1\}$$

Semantic Equivalence of Boolean Expressions

$$b_0 \sim b_1 \text{ iff } (\forall t \in \mathbf{T} \forall \sigma \in \Sigma.\ \langle b_0, \sigma \rangle \to t \iff \langle b_1, \sigma \rangle \to t)$$

Short-Circuit Evaluation

A more efficient evaluation strategy for $b_0 \wedge b_1$ is to first evaluate b_0 and then only in the case where its evaluation yields $\mathbf{true}$ to proceed with the evaluation of b_1.

$$\frac{\langle b_0, \sigma \rangle \to \mathbf{false}}{\langle b_0 \wedge b_1, \sigma \rangle \to \mathbf{false}}$$

$$\frac{\langle b_0, \sigma \rangle \to \mathbf{true} \qquad \langle b_1, \sigma \rangle \to \mathbf{false}}{\langle b_0 \wedge b_1, \sigma \rangle \to \mathbf{false}}$$

$$\frac{\langle b_0, \sigma \rangle \to \mathbf{true} \qquad \langle b_1, \sigma \rangle \to \mathbf{true}}{\langle b_0 \wedge b_1, \sigma \rangle \to \mathbf{true}}$$

Exercise) Define short-circuit evaluation for $b_0 \vee b_1$.

Execution of Commands

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

$$\frac{\langle c_0, \sigma\rangle \to \sigma'' \qquad \langle c_1, \sigma''\rangle \to \sigma'}{\langle c_0; c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{true} \qquad \langle c_0, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{false} \qquad \langle c_1, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_0 \texttt{ else } c_1, \sigma\rangle \to \sigma'}$$

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

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

cf) Non-Terminating Program

For any state σ, there is no state σ' such that

$$\langle \texttt{while true do skip}, \sigma \rangle \to \sigma'$$

Semantic Equivalence of Commands

$$c_0 \sim c_1 \text{ iff } (\forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma \rangle \to \sigma' \iff \langle c_1, \sigma \rangle \to \sigma')$$

Example

Let $w \equiv \texttt{while}\ b\ \texttt{do}\ c$ with $b \in \mathbf{Bexp}, c \in \mathbf{Com}$. Prove that

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

Proof) To show:

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

$\Rightarrow$: Suppose $\langle w, \sigma \rangle \to \sigma'$ for states σ, σ'. Then there must be a derivation of $\langle w, \sigma \rangle \to \sigma'$, where the final rule is either

$$\frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle w, \sigma \rangle \to \sigma} \quad (1)$$

or

$$\frac{\langle b, \sigma \rangle \to \mathbf{true} \qquad \langle c, \sigma \rangle \to \sigma'' \qquad \langle w, \sigma'' \rangle \to \sigma'}{\langle w, \sigma \rangle \to \sigma'} \quad (2)$$

In case (1), the derivation must have the form

$$\dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \to \mathbf{false}}}{\langle w, \sigma \rangle \to \sigma}$$

which includes a derivation of $\langle b, \sigma \rangle \to \mathbf{false}$. Using this derivation, we can build the following derivation:

$$\dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \to \mathbf{false}} \qquad \overline{\langle \mathtt{skip}, \sigma \rangle \to \sigma}}{\langle \mathtt{if}\ b\ \mathtt{then}\ c; w\ \mathtt{else}\ \mathtt{skip}, \sigma \rangle \to \sigma}$$

In case (2), the derivation must have the form

$$\dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \to \mathbf{true}} \qquad \dfrac{\vdots}{\langle c, \sigma \rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w, \sigma'' \rangle \to \sigma'}}{\langle w, \sigma \rangle \to \sigma'}$$

Using this, we can build the following derivation:

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

In either case, (1) and (2), we obtain a derivation of

$$\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip}, \sigma\rangle \to \sigma'$$

Thus,

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

$\Leftarrow$: Suppose $\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip}, \sigma\rangle \to \sigma'$ for states σ, σ'. Then, there is a derivation with one of two possible forms:

$$\frac{\dfrac{\vdots}{\langle b, \sigma\rangle \to \mathbf{false}} \quad \dfrac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma}}{\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip}, \sigma\rangle \to \sigma} \tag{3}$$

$$\frac{\dfrac{\vdots}{\langle b, \sigma\rangle \to \mathbf{true}} \quad \dfrac{\vdots}{\langle c; w, \sigma\rangle \to \sigma'}}{\langle \texttt{if } b \texttt{ then } c; w \texttt{ else skip}, \sigma\rangle \to \sigma'} \tag{4}$$

From either derivation, we can construct a derivation of $\langle w, \sigma\rangle \to \sigma'$. Consider the second case, (4), which has a derivation of $\langle c; w, \sigma\rangle \to \sigma'$ of the form

$$\frac{\dfrac{\vdots}{\langle c, \sigma\rangle \to \sigma''} \quad \dfrac{\vdots}{\langle w, \sigma''\rangle \to \sigma'}}{\langle c; w, \sigma\rangle \to \sigma'}$$

for some state σ''.

Using the derivations of $\langle c, \sigma \rangle \to \sigma''$, $\langle w, \sigma'' \rangle \to \sigma'$, and $\langle b, \sigma \rangle \to \mathbf{true}$, we can produce the derivation

$$\frac{\dfrac{\vdots}{\langle b, \sigma \rangle \to \mathbf{true}} \qquad \dfrac{\vdots}{\langle c, \sigma \rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w, \sigma'' \rangle \to \sigma'}}{\langle w, \sigma \rangle \to \sigma'}$$

Similarly, we can construct a derivation of $\langle w, \sigma \rangle \to \sigma'$ from (3). Thus,

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

We can now conclude that

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

and hence

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

Small-step Operational Semantics

$$\frac{\langle a_0, \sigma\rangle \to_1 \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \to_1 \langle a_0' + a_1, \sigma\rangle}$$

$$\frac{\langle a_1, \sigma\rangle \to_1 \langle a_1', \sigma\rangle}{\langle n + a_1, \sigma\rangle \to_1 \langle n + a_1', \sigma\rangle}$$

$$\frac{}{\langle n + m, \sigma\rangle \to_1 \langle p, \sigma\rangle}\ p \text{ is the sum of } n \text{ and } m$$

Exercise) Complete the rules for $\mathbf{Aexp}$ and $\mathbf{Bexp}$.

Small-step Operational Semantics

$$\frac{}{\langle \mathtt{skip}, \sigma\rangle \to_1 \sigma}$$

$$\frac{}{\langle X := n, \sigma\rangle \to_1 s[n/X]} \qquad \frac{\langle a, \sigma\rangle \to_1 \langle a', \sigma\rangle}{\langle X := a, \sigma\rangle \to_1 \langle X := a', \sigma\rangle}$$

$$\frac{\langle c_0, \sigma\rangle \to_1 \langle c_0', \sigma'\rangle}{\langle c_0; c_1, \sigma\rangle \to_1 \langle c_0'; c_1, \sigma'\rangle} \qquad \frac{\langle c_0, \sigma\rangle \to_1 \sigma'}{\langle c_0; c_1, \sigma\rangle \to_1 \langle c_1, \sigma'\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \langle \mathbf{true}, \sigma\rangle}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle \to_1 \langle c_0, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \langle \mathbf{false}, \sigma\rangle}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle \to_1 \langle c_1, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \to \langle b', \sigma\rangle}{\langle \mathtt{if}\ b\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle \to_1 \langle \mathtt{if}\ b'\ \mathtt{then}\ c_0\ \mathtt{else}\ c_1, \sigma\rangle}$$

$$\frac{}{\langle \mathtt{while}\ b\ \mathtt{do}\ c, \sigma\rangle \to_1 \langle \mathtt{if}\ b\ \mathtt{then}\ c; \mathtt{while}\ b\ \mathtt{do}\ c\ \mathtt{else}\ \mathtt{skip}, \sigma\rangle}$$

Example

Consider the statement:

```
(z:=x; x:=y); y:=z
```

Let σ_0 be the state that maps all variables except x and y and has $\sigma_0(\mathrm{x}) = 5$ and $\sigma_0(\mathrm{y}) = 7$. We then have the derivation sequence:

$$
\begin{aligned}
&\langle (\mathrm{z} := \mathrm{x}; \mathrm{x} := \mathrm{y}); \mathrm{y} := \mathrm{z}, \sigma_0 \rangle \\
&\to_1 \langle \mathrm{x} := \mathrm{y}; \mathrm{y} := \mathrm{z}, \sigma_0[z \mapsto 5] \rangle \\
&\to_1 \langle \mathrm{y} := \mathrm{z}, \sigma_0[z \mapsto 5, x \mapsto 7] \rangle \\
&\to_1 \sigma_0[z \mapsto 5, x \mapsto 7, y \mapsto 5]
\end{aligned}
$$

Each step has a derivation tree explaining why it takes place, e.g.,

$$
\dfrac{\dfrac{\langle \mathrm{z} := \mathrm{x}, \sigma_0 \rangle \to_1 \sigma_0[z \mapsto 5]}{\langle \mathrm{z} := \mathrm{x}; \mathrm{x} := \mathrm{y}, \sigma_0 \rangle \to_1 \langle \mathrm{x} := \mathrm{y}, \sigma_0[z \mapsto 5] \rangle}}{\langle (\mathrm{z} := \mathrm{x}; \mathrm{x} := \mathrm{y}); \mathrm{y} := \mathrm{z}, \sigma_0 \rangle \to_1 \langle \mathrm{x} := \mathrm{y}; \mathrm{y} := \mathrm{z}, \sigma_0[z \mapsto 5] \rangle}
$$

Example: Factorial

Assume that $\sigma(x) = 3$.

$\langle$`y:=1; while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma\rangle$
$\rightarrow_1 \langle$`while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 1]\rangle$
$\rightarrow_1 \langle$`if ¬(x=1) then ((y:=y⋆x; x:=x-1);while ¬(x=1) do (y:=y⋆x; x:=x-1))`
`else skip`$, \sigma[y \mapsto 1]\rangle$
$\rightarrow_1 \langle$`(y:=y⋆x; x:=x-1);while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 1]\rangle$
$\rightarrow_1 \langle$`x:=x-1;while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 3]\rangle$
$\rightarrow_1 \langle$`while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 3][x \mapsto 2]\rangle$
$\rightarrow_1 \langle$`if ¬(x=1) then ((y:=y⋆x; x:=x-1);while ¬(x=1) do (y:=y⋆x; x:=x-1))`
`else skip`$, \sigma[y \mapsto 3][x \mapsto 2]\rangle$
$\rightarrow_1 \langle$`(y:=y⋆x; x:=x-1);while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 3][x \mapsto 2]\rangle$
$\rightarrow_1 \langle$`x:=x-1;while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 6][x \mapsto 2]\rangle$
$\rightarrow_1 \langle$`while ¬(x=1) do (y:=y⋆x; x:=x-1)`$, \sigma[y \mapsto 6][x \mapsto 1]\rangle$
$\rightarrow_1 s[y \mapsto 6][x \mapsto 1]$

Summary

We have defined the operational semantics of **IMP**.

- *Big-step operational semantics* describes how the overall results of executions are obtained.
- *Small-step operational semantics* describes how the individual steps of the computations take place.

The big-step and small-step operational semantics are equivalent:

Theorem

$\forall c \in \mathbf{Com} \forall \sigma, \sigma' \in \Sigma.\ \langle c, \sigma \rangle \to \sigma' \iff \langle c, \sigma \rangle \to_1^* \sigma'.$