Large-step operational semantics for IMP; Properties of IMP

Lecture 4 Thursday, February 6, 2014

1 Large-step operational semantics for IMP

We define large-step evaluation relations for arithmetic expressions, boolean expressions, and commands. The relation for arithmetic expressions relates an arithmetic expression and store to the integer value that the expression evaluates to. For boolean expressions, the final value is in **Bool** $= \{\textsf{true}, \textsf{false}\}$. For commands, the final value is a store.

$$\Downarrow_{\textbf{Aexp}} \subseteq \textbf{Aexp} \times \textbf{Store} \times \textbf{Int}$$
$$\Downarrow_{\textbf{Bexp}} \subseteq \textbf{Bexp} \times \textbf{Store} \times \textbf{Bool}$$
$$\Downarrow_{\textbf{Com}} \subseteq \textbf{Com} \times \textbf{Store} \times \textbf{Store}$$

Again, we overload the symbol $\Downarrow$ and use it for any of these three relations; which relation is intended will be clear from context. We also use infix notation, for example writing $\langle c, \sigma\rangle \Downarrow \sigma'$ if $(c, \sigma, \sigma') \in \Downarrow_{\textbf{Com}}$.

Arithmetic expressions.

$$\frac{}{\langle n, \sigma\rangle \Downarrow n} \qquad \frac{}{\langle x, \sigma\rangle \Downarrow n} \text{ where } \sigma(x) = n$$

$$\frac{\langle e_1, \sigma\rangle \Downarrow n_1 \qquad \langle e_2, \sigma\rangle \Downarrow n_2}{\langle e_1 + e_2, \sigma\rangle \Downarrow n} \text{ where } n = n_1 + n_2 \qquad \frac{\langle e_1, \sigma\rangle \Downarrow n_1 \qquad \langle e_2, \sigma\rangle \Downarrow n_2}{\langle e_1 \times e_2, \sigma\rangle \Downarrow n} \text{ where } n = n_1 \times n_2$$

Boolean expressions.

$$\frac{}{\langle \textsf{true}, \sigma\rangle \Downarrow \textsf{true}} \qquad \frac{}{\langle \textsf{false}, \sigma\rangle \Downarrow \textsf{false}}$$

$$\frac{\langle a_1, \sigma\rangle \Downarrow n_1 \qquad \langle a_2, \sigma\rangle \Downarrow n_2}{\langle a_1 < a_2, \sigma\rangle \Downarrow \textsf{true}} \text{ where } n_1 < n_2 \qquad \frac{\langle a_1, \sigma\rangle \Downarrow n_1 \qquad \langle a_2, \sigma\rangle \Downarrow n_2}{\langle a_1 < a_2, \sigma\rangle \Downarrow \textsf{false}} \text{ where } n_1 \geq n_2$$

Commands.

$$\textsc{Skip}\ \frac{}{\langle \textsf{skip}, \sigma\rangle \Downarrow \sigma} \qquad \textsc{Asg}\ \frac{\langle e, \sigma\rangle \Downarrow n}{\langle x := e, \sigma\rangle \Downarrow \sigma[x \mapsto n]} \qquad \textsc{Seq}\ \frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \qquad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle c_1; c_2, \sigma\rangle \Downarrow \sigma''}$$

$$\textsc{If-T}\ \frac{\langle b, \sigma\rangle \Downarrow \textsf{true} \qquad \langle c_1, \sigma\rangle \Downarrow \sigma'}{\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle \Downarrow \sigma'} \qquad \textsc{If-F}\ \frac{\langle b, \sigma\rangle \Downarrow \textsf{false} \qquad \langle c_2, \sigma\rangle \Downarrow \sigma'}{\langle \textsf{if}\ b\ \textsf{then}\ c_1\ \textsf{else}\ c_2, \sigma\rangle \Downarrow \sigma'}$$

$$\textsc{While-F}\ \frac{\langle b, \sigma\rangle \Downarrow \textsf{false}}{\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle \Downarrow \sigma} \qquad \textsc{While-T}\ \frac{\langle b, \sigma\rangle \Downarrow \textsf{true} \qquad \langle c, \sigma\rangle \Downarrow \sigma' \qquad \langle \textsf{while}\ b\ \textsf{do}\ c, \sigma'\rangle \Downarrow \sigma''}{\langle \textsf{while}\ b\ \textsf{do}\ c, \sigma\rangle \Downarrow \sigma''}$$

It's interesting to see that the rule for while loops does not rely on using an if command (as we needed in the case of small-step semantics). Why does this rule work?

1.1 Command equivalence

The small-step operational semantics suggest that the loop **while** b **do** c should be equivalent to the command **if** b **then** $(c;$ **while** b **do** $c)$ **else skip**. Can we show that this indeed the case that the language is defined using the above large-step evaluation?

First, we need to to be more precise about what "equivalent commands" mean. Our formal model allows us to define this concept using large-step evaluations as follows. (One can write a similar definition using $\longrightarrow^*$ in small-step semantics.)

Definition (Equivalence of commands)**.** Two commands c and c' are equivalent (written $c \sim c'$) if, for any stores σ and σ', we have

$$\langle c, \sigma\rangle \Downarrow \sigma' \iff \langle c', \sigma\rangle \Downarrow \sigma'.$$

We can now state and prove the claim that **while** b **do** c and **if** b **then** $(c;$ **while** b **do** $c)$ **else skip** are equivalent.

Theorem. *For all $b \in$ **Bexp** and $c \in$ **Com** we have*

$$\textbf{while } b \textbf{ do } c \sim \textbf{if } b \textbf{ then } (c; \textbf{while } b \textbf{ do } c) \textbf{ else skip}.$$

Proof. Let W be an abbreviation for **while** b **do** c. We want to show that for all stores σ, σ', we have:

$$\langle W, \sigma\rangle \Downarrow \sigma' \text{ if and only if } \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip} \Downarrow \sigma'$$

For this, we must show that both directions ($\Longrightarrow$ and $\Longleftarrow$) hold. We'll show only direction $\Longrightarrow$; the other is similar.

Assume that σ and σ' are stores such that $\langle W, \sigma\rangle \Downarrow \sigma'$. It means that there is some derivation that proves for this fact. Inspecting the evaluation rules, we see that there are two possible rules whose conclusions match this fact: WHILE-F and WHILE-T. We analyze each of them in turn.

- WHILE-F. The derivation must look like the following.

$$\textsc{While-F}\ \dfrac{\dfrac{\vdots^1}{\langle b, \sigma\rangle \Downarrow \textbf{false}}}{\langle W, \sigma\rangle \Downarrow \sigma}$$

  Here, we use $\vdots^1$ to refer to the derivation of $\langle b, \sigma\rangle \Downarrow$ **false**. Note that in this case, $\sigma' = \sigma$.

  We can use $\vdots^1$ to derive a proof tree showing that the evaluation of **if** b **then** $(c; W)$ **else skip** yields the same final state σ:

$$\textsc{If-F}\ \dfrac{\dfrac{\vdots^1}{\langle b, \sigma\rangle \Downarrow \textbf{false}} \qquad \textsc{Skip}\ \dfrac{}{\langle \textbf{skip}, \sigma\rangle \Downarrow \sigma}}{\langle \textbf{if } b \textbf{ then } (c; W) \textbf{ else skip}, \sigma\rangle \Downarrow \sigma}$$

- WHILE-T. In this case, the derivation has the following form.

$$\textsc{While-T}\ \dfrac{\dfrac{\vdots^2}{\langle b, \sigma\rangle \Downarrow \textbf{true}} \qquad \dfrac{\vdots^3}{\langle c, \sigma\rangle \Downarrow \sigma''} \qquad \dfrac{\vdots^4}{\langle W, \sigma''\rangle \Downarrow \sigma'}}{\langle W, \sigma\rangle \Downarrow \sigma'}$$

We can use subderivations $\vdots^{2}$, $\vdots^{3}$, and $\vdots^{4}$ to show that the evaluation of **if** b **then** $(c; W)$ **else skip** yields the same final state σ.

$$\textsc{If-T}\ \dfrac{\dfrac{\vdots^{2}}{\langle b,\sigma\rangle \Downarrow \textbf{true}} \qquad \textsc{Seq}\ \dfrac{\dfrac{\vdots^{3}}{\langle c,\sigma\rangle \Downarrow \sigma''} \qquad \dfrac{\vdots^{4}}{\langle W,\sigma''\rangle \Downarrow \sigma'}}{\langle c; W,\sigma\rangle \Downarrow \sigma'}}{\langle \textbf{if } b \textbf{ then } (c;W) \textbf{ else skip},\sigma\rangle \Downarrow \sigma'}$$

Hence, we showed that in each of the two possible cases, the command **if** b **then** $(c; W)$ **else skip** evaluates to the same final state as the command W. □

2 Some properties of IMP

2.1 Equivalence of semantics

The small-step and large-step semantics are equivalent. We state this formally in the following theorem.

Theorem (Equivalence of IMP semantics)**.** *For all commands $c \in$ **Com** and stores $\sigma, \sigma' \in$ **Store** we have*

$$\langle c, \sigma\rangle \longrightarrow^{*} \langle \textbf{skip}, \sigma'\rangle \iff \langle c, \sigma\rangle \Downarrow \sigma'.$$

2.2 Non-termination

For a command c and initial state σ, the execution of the command may *terminate* with some final store σ', or it may *diverge* and never yield a final state. For example, the command **while true do** $\mathsf{foo} := \mathsf{foo} + 1$ always diverges; the command **while** $0 < \mathsf{i}$ **do** $\mathsf{i} := \mathsf{i} + 1$ will diverge if and only if the value of variable i in the initial state is positive.

If $\langle c, \sigma\rangle$ is a configuration that diverges, then there is no state σ' such that $\langle c, \sigma\rangle \Downarrow \sigma'$ or $\langle c, \sigma\rangle \longrightarrow^{*} \langle \textbf{skip}, \sigma'\rangle$. However, in small-step semantics, a diverging computation has an infinite sequence of configurations: $\langle c, \sigma\rangle \longrightarrow \langle c_1, \sigma_1\rangle \longrightarrow \langle c_2, \sigma_2\rangle \longrightarrow \ldots$. Small-step semantics can allow us to state, and prove, properties about programs that may diverge. Later in the course, we will specify and prove properties that are of interest in potentially diverging computations.

2.3 Determinism of commands

The semantics of IMP (both small-step and large-step) are *deterministic*. That is, each IMP command c and each initial store σ evaluates to at most one final store. We state this formally for the large-step semantics below.

Theorem. *For all commands $c \in$ **Com** and stores $\sigma, \sigma_1, \sigma_2 \in$ **Store**, if $\langle c, \sigma\rangle \Downarrow \sigma_1$ and $\langle c, \sigma\rangle \Downarrow \sigma_2$ then $\sigma_1 = \sigma_2$.*

We need an inductive proof to prove this theorem. However, induction on the structure of command c does not work. (Why? Which of the cases does it fail for?) Instead, we need to perform induction on the derivation of $\langle c, \sigma\rangle \Downarrow \sigma_1$.

Before we commence the proof of the theorem, we will need two lemmas, related to the determinism of the arithmetic and boolean semantics, $\Downarrow_{\mathbf{Aexp}}$ and $\Downarrow_{\mathbf{Bexp}}$.

Lemma 1. *For all arithmetic expressions $a \in$ **Aexp**, stores $\sigma \in$ **Store**, and integers $n_1, n_2 \in$ **Int**, if $\langle a, \sigma\rangle \Downarrow n_1$ and $\langle a, \sigma\rangle \Downarrow n_2$ then $n_1 = n_2$.*

Lemma 2. *For all boolean expressions $b \in$ **Aexp**, stores $\sigma \in$ **Store**, and integers $b_1, b_2 \in$ **Bool**, if $\langle b, \sigma\rangle \Downarrow b_1$ and $\langle a, \sigma\rangle \Downarrow b_2$ then $b_1 = b_2$.*

These lemmas are straightforward to prove, and can be proved using strucutral induction on arithmetic and boolean expressions respectively.

Proof. We proceed by induction on the derivation of $\langle c, \sigma\rangle \Downarrow \sigma_1$. The inductive hypothesis P is

$$P(\langle c, \sigma\rangle \Downarrow \sigma_1) = \forall \sigma_2 \in \textbf{Store}, \text{if } \langle c, \sigma\rangle \Downarrow \sigma_2 \text{ then } \sigma_1 = \sigma_2.$$

Suppose we have a derivation for $\langle c, \sigma\rangle \Downarrow \sigma_1$, for some c, σ, and σ_1. Assume that the inductive hypothesis holds for any subderivation $\langle c', \sigma'\rangle \Downarrow \sigma''$ used in the derivation of $\langle c, \sigma\rangle \Downarrow \sigma_1$.

Assume that for some σ_2 we have $\langle c, \sigma\rangle \Downarrow \sigma_2$. We need to show that $\sigma_1 = \sigma_2$.

We consider the possible cases for the last rule used in derivation of

$$\langle c, \sigma\rangle \Downarrow \sigma_1$$

.

- Skip. In this case, the derivation looks like

  $$\textsc{Skip}\ \frac{\vdots}{\langle \textbf{skip}, \sigma\rangle \Downarrow \sigma}\ ,$$

  and we have $c \equiv \textbf{skip}$ and $\sigma_1 = \sigma$. Since by assumption we have $\langle c, \sigma\rangle \Downarrow \sigma_2$, there must be a derivation of $\langle c, \sigma\rangle \Downarrow \sigma_2$. Moreover, the last rule used in this derivation must be Skip, as it is the only rule that has the command **skip** in its conclusion. So we have $\sigma_2 = \sigma$ and the result holds.

- Asg

  In this case, the derivation looks like

  $$\textsc{Asg}\ \frac{\dfrac{\vdots}{\langle a, \sigma\rangle \Downarrow n}}{\langle x := a, \sigma\rangle \Downarrow \sigma_1}\ ,$$

  and we have $c \equiv x := a$ and $\sigma_1 = \sigma[x \mapsto n]$. The last rule used in the derivation of $\langle c, \sigma\rangle \Downarrow \sigma_2$ must also be Asg, and so we have $\sigma_2 = \sigma[x \mapsto m]$, where $\langle a, \sigma\rangle \Downarrow m$. By the determinism of arithmetic expressions, $m = n$ and so $\sigma_2 = \sigma_1$ and the result holds.

- Seq

  In this case, the derivation looks like

  $$\textsc{Seq}\ \frac{\dfrac{\vdots}{\langle c_1, \sigma\rangle \Downarrow \sigma'} \qquad \dfrac{\vdots}{\langle c_2, \sigma'\rangle \Downarrow \sigma_1}}{\langle c_1; c_2, \sigma\rangle \Downarrow \sigma_1}\ ,$$

  and we have $c \equiv c_1; c_2$. The last rule used in the derivation of $\langle c, \sigma\rangle \Downarrow \sigma_2$ must also be Seq, and so we have

  $$\textsc{Seq}\ \frac{\dfrac{\vdots}{\langle c_1, \sigma\rangle \Downarrow \sigma''} \qquad \dfrac{\vdots}{\langle c_2, \sigma''\rangle \Downarrow \sigma_2}}{\langle c_1; c_2, \sigma\rangle \Downarrow \sigma_2}\ .$$

  By the inductive hypothesis applied to the derivation $\dfrac{\vdots}{\langle c_1, \sigma\rangle \Downarrow \sigma'}$, we have $\sigma' = \sigma''$. By another application of the inductive hypothesis, to the derivation $\dfrac{\vdots}{\langle c_2, \sigma'\rangle \Downarrow \sigma_1}$, we have $\sigma_1 = \sigma_2$ and the result holds.

- IF-T

  In this case, the derivation looks like

$$\text{IF-T}\ \dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \Downarrow \textbf{true}} \qquad \dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma_1}}{\langle \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma \rangle \Downarrow \sigma_1},$$

  and we have $c \equiv \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2$. The last rule used in the derivation of $\langle c, \sigma \rangle \Downarrow \sigma_2$ must be either IF-T or IF-F (since these are the only rules that can be used to derive a conclusion of the form $\langle \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma \rangle \Downarrow \sigma_2$). But by the determinism of boolean expressions, we must have $\langle b, \sigma \rangle \Downarrow \textbf{true}$, and so the derivation of $\langle c, \sigma \rangle \Downarrow \sigma_2$ must have the following form.

$$\text{IF-T}\ \dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \Downarrow \textbf{true}} \qquad \dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma_2}}{\langle \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \sigma \rangle \Downarrow \sigma_2}$$

  The result holds by the inductive hypothesis applied to the derivation $\dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma_1}$.

- IF-F

  Similar to the case for IF-T.

- WHILE-F

  Straightforward, similar to the case for SKIP.

- WHILE-T

  Here we have

$$\text{WHILE-T}\ \dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \Downarrow \textbf{true}} \qquad \dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma'} \qquad \dfrac{\vdots}{\langle c, \sigma' \rangle \Downarrow \sigma_1}}{\langle \textbf{while}\ b\ \textbf{do}\ c_1, \sigma \rangle \Downarrow \sigma_1},$$

  and we have $c \equiv \textbf{while}\ b\ \textbf{do}\ c_1$. The last rule used in the derivation of $\langle c, \sigma \rangle \Downarrow \sigma_2$ must also be WHILE-T (by the determinism of boolean expressions), and so we have

$$\text{WHILE-T}\ \dfrac{\dfrac{\vdots}{\langle b, \sigma \rangle \Downarrow \textbf{true}} \qquad \dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma''} \qquad \dfrac{\vdots}{\langle c, \sigma'' \rangle \Downarrow \sigma_2}}{\langle \textbf{while}\ b\ \textbf{do}\ c_1, \sigma \rangle \Downarrow \sigma_2}.$$

  By the inductive hypothesis applied to the derivation $\dfrac{\vdots}{\langle c_1, \sigma \rangle \Downarrow \sigma'}$, we have $\sigma' = \sigma''$. By another application of the inductive hypothesis, to the derivation $\dfrac{\vdots}{\langle c, \sigma' \rangle \Downarrow \sigma_1}$, we have $\sigma_1 = \sigma_2$ and the result holds.

  Note: Even though the command $c \equiv \textbf{while}\ b\ \textbf{do}\ c_1$ appears in the derivation of $\langle \textbf{while}\ b\ \textbf{do}\ c_1, \sigma \rangle \Downarrow \sigma_1$, we do not run in to problems, as the induction is over the *derivation*, not over the structure of the command.

So we have shown that $P(\langle c, \sigma\rangle \Downarrow \sigma_1)$ for any c, σ, and σ_1 such that $\langle c, \sigma\rangle \Downarrow \sigma_1$. This is equivalent to

$$\forall c \in \textbf{Com}.\ \forall \sigma, \sigma_1, \sigma_2 \in \textbf{Store}, \text{if } \langle c, \sigma\rangle \Downarrow \sigma_1 \text{ and } \langle c, \sigma\rangle \Downarrow \sigma_2 \text{ then } \sigma_1 = \sigma_2$$

which proves the theorem. $\square$