Scribe: Siddharth Alexander, Kanghoe Kim Lecturer: Radu Rugina

## 1 Operational semantics of IMP (continued)

The rules described in the previous lecture are not necessarily the most efficient methods to implement their corresponding expressions. For example, consider the boolean operator, disjunction ($\vee$), that we discussed in the previous lecture. We specified disjunction as

$$\frac{\langle b_0,\,\sigma\rangle \to t_0 \quad \langle b_1,\,\sigma\rangle \to t_1}{\langle b_0 \vee b_1,\,\sigma\rangle \to t} \quad \text{where } t = b_0 \vee b_1.$$

This is not the most efficient way to implement the rule for disjunction since we need only $b_0$ or $b_1$ to evaluate to **true** for their disjunction to evaluate to **true**. With this in mind we consider "parallel evaluation" defined as

$$\frac{\langle b_0,\,\sigma\rangle \to \mathbf{true}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{true}} \quad \frac{\langle b_1,\,\sigma\rangle \to \mathbf{true}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{true}} \quad \frac{\langle b_0,\,\sigma\rangle \to \mathbf{false} \quad \langle b_1, \sigma, \to\rangle \mathbf{false}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{false}}$$

Note that this rule is non-deterministic since if both $b_0$ and $b_1$ evaluate to **true** it is not clear which rule to pick and further we need to pick a rule to evaluate first. The above rule is referred to as parallel evaluation since we start off by evaluating each one of them and pick the first one that holds true. To make the implementation deterministic, we define "left-to-right evaluation" as

$$\frac{\langle b_0,\,\sigma\rangle \to \mathbf{true}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{true}} \quad \frac{\langle b_1,\,\sigma\rangle \to \mathbf{true} \quad \langle b_0,\,\sigma\rangle \to \mathbf{false}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{true}} \quad \frac{\langle b_0,\,\sigma\rangle \to \mathbf{false} \quad \langle b_1,\,\sigma\rangle \to \mathbf{false}}{\langle b_0 \vee b_1,\,\sigma\rangle \to \mathbf{false}}$$

**Commands**

$$\frac{}{\langle \mathbf{skip},\,\sigma\rangle \to \sigma}\ (Skip)$$

$$\frac{\langle a,\,\sigma\rangle \to n}{\langle X := a,\,\sigma\rangle \to \sigma[X \mapsto n]}\ (Assignment)$$

where $\sigma[X \mapsto n](Y) = n$ if $Y = X$ and $\sigma(Y)$ otherwise. ($\sigma[X \mapsto n]$ may alternatively be written as $\sigma[n/X]$).

$$\frac{\langle c_0,\,\sigma\rangle \to \sigma' \quad \langle c_1,\,\sigma\rangle \to \sigma''}{\langle c_0; c_1,\,\sigma\rangle \to \sigma''}\ (Sequencing)$$

$$\frac{\langle b,\,\sigma\rangle \to \mathbf{true} \quad \langle c_0,\,\sigma\rangle \to \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1,\,\sigma\rangle \to \sigma'} \qquad \frac{\langle b,\,\sigma\rangle \to \mathbf{false} \quad \langle c_1,\,\sigma\rangle \to \sigma'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1,\,\sigma\rangle \to \sigma'}\ (If)$$

$$\frac{\langle b,\,\sigma\rangle \to \mathbf{false}}{\langle \mathbf{while}\, b\, \mathbf{do}\, c,\,\sigma\rangle \to \sigma} \qquad \frac{\langle b,\,\sigma\rangle \to \mathbf{true} \quad \langle c,\,\sigma\rangle \to \sigma'' \quad \langle \mathbf{while}\, b\, \mathbf{do}\, c,\,\sigma''\rangle \to \sigma'}{\langle \mathbf{while}\, b\, \mathbf{do}\, c,\,\sigma\rangle \to \sigma'}\ (While)$$

## 2 Rule instance, proof tree

A rule instance is one of the rules instantiated for an actual program statement. For example, the following rule instance is a rule for arithmetic expressions where metavariables are replaced by actual values.

$$\frac{\langle 2,\,\sigma\rangle \to 2 \quad \langle 3,\,\sigma\rangle \to 3}{\langle 2+3,\,\sigma\rangle \to 5}$$

Notice that not every rule instance can be a part of a proof tree, as the following rule instance exemplifies. $\langle 3, \sigma\rangle \to 4$ is neither a valid axiom nor a conclusion from valid premises.

$$\frac{\langle 2,\, \sigma\rangle \to 2 \quad \langle 3,\, \sigma\rangle \to 4}{\langle 2+3,\, \sigma\rangle \to 6}$$

Suppose we want to prove $\langle c, \sigma\rangle \to \sigma'$, i.e. that execution of the program $c$ on the state $\sigma$ ends up in the state $\sigma'$. To do this we build up a proof tree. It is a tree rooted at $\langle c, \sigma\rangle \to \sigma'$ where each branching is application of a rule.

**Example:** Let $p \equiv \mathbf{if}\, x < y\, \mathbf{then}\, x := x + 1\, \mathbf{else\, skip}$. We construct a proof tree for this program to prove $\langle p, \sigma\rangle \to \sigma[x \mapsto 1]$, provided $\sigma(x) = 0$ and $\sigma(y) = 1$.

$$\dfrac{\dfrac{\dfrac{}{\langle x,\sigma\rangle \to 0} \quad \dfrac{}{\langle y,\sigma\rangle \to 1}}{\langle x < y, \sigma\rangle \to \mathbf{true}} \quad \dfrac{\dfrac{\dfrac{}{\langle x,\sigma\rangle \to 0} \quad \dfrac{}{\langle 1,\sigma\rangle \to 1}}{\langle x+1, \sigma\rangle \to 1\rangle}}{\langle x := x+1, \sigma\rangle \to \sigma[x \mapsto 1]}}{\langle p, \sigma\rangle \to \sigma[x \mapsto 1]}$$

We see that all leaves in the tree are instances of axioms. And note that depth first search on the proof tree exactly corresponds to the execution of the program.

## 3 Expressing program properties

We can use the formalism of operational semantics to express the following program properties:

- Termination

  $\forall a \in \mathbf{Aexp}, \sigma \in \Sigma \;.\; \exists n \in \mathbb{Z}$ s.t. $\langle a, \sigma\rangle \to n$

  $\forall b \in \mathbf{Bexp}, \sigma \in \Sigma \;.\; \exists t \in \mathbb{T}$ s.t. $\langle b, \sigma\rangle \to t$

  $\forall c \in \mathbf{Com}, \sigma \in \Sigma \;.\; \exists \sigma' \in \Sigma$ s.t. $\langle c, \sigma\rangle \to \sigma'$

  Every arithmetic and boolean expression always result in a value in **IMP**. However, not every command terminates.

- Equivalence of syntactic constructs (including programs)

  $a_0, a_1 \in \mathbf{Aexp}$ are equivalent: $a_0 \sim a_1$ iff $\forall \sigma, n \;.\; \langle a_0, \sigma\rangle \to n \iff \langle a_1, \sigma\rangle \to n$

  $b_0, b_1 \in \mathbf{Bexp}$ are equivalent: $b_0 \sim b_1$ iff $\forall \sigma, t \;.\; \langle b_0, \sigma\rangle \to t \iff \langle b_1, \sigma\rangle \to t$

  $c_0, c_1 \in \mathbf{Com}$ are equivalent: $c_0 \sim c_1$ iff $\forall \sigma, \sigma' \;.\; \langle c_0, \sigma\rangle \to \sigma' \iff \langle c_1, \sigma\rangle \to \sigma'$

- Determinism (of semantics)

  $\forall c, \sigma, \sigma', \sigma'' \;.\; \langle c, \sigma\rangle \to \sigma' \wedge \langle c, \sigma\rangle \to \sigma'' \Rightarrow \sigma' = \sigma''$

**Example.** (simple proof of equivalence) Let $\mathrm{if}_0 \equiv \mathbf{if}\, \mathrm{b}\, \mathbf{then}\, \mathrm{c}_0\, \mathbf{else}\, \mathrm{c}_1$ and $\mathrm{if}_1 \equiv \mathbf{if}\, \neg\mathrm{b}\, \mathbf{then}\, \mathrm{c}_1\, \mathbf{else}\, \mathrm{c}_0$
We want to show $\mathrm{if}_0$ is equivalent to $\mathrm{if}_1$ ($\mathrm{if}_0 \sim \mathrm{if}_1$) using proof trees.

Proof. $(\Rightarrow) : \forall \sigma, \sigma', \langle \mathrm{if}_0, \sigma\rangle \to \sigma' \Rightarrow \langle \mathrm{if}_1, \sigma\rangle \to \sigma'$

The $b$ evaluates either to **true** or to **false**. Consider first that $b$ evaluates to **true**. Suppose the left hand side holds. Then there exists its proof tree like below.

$$\dfrac{\dfrac{\vdots(1)}{\langle b, \sigma\rangle \to \mathbf{true}} \quad \dfrac{\vdots(2)}{\langle c_0, \sigma\rangle \to \sigma'}}{\langle \mathrm{if}_0, \sigma\rangle \to \sigma'}$$

Now we construct the proof tree for $\text{if}_1$.

$$\dfrac{\dfrac{\dfrac{\vdots(1)}{\langle b, \sigma \rangle \to \mathbf{true}}}{\langle \neg b, \sigma \rangle \to \mathbf{false}} \quad \dfrac{\vdots(2)}{\langle c_0, \sigma \rangle \to \sigma'}}{\langle \text{if}_1, \sigma \rangle \to \sigma'}$$

As shown above, we can substitute subtrees (1) and (2) from the first proof tree into the second to show $\langle \text{if}_1, \sigma \rangle \to \sigma'$.

Similar for the case where $b$ evaluates to **false**, and for the converse($\Leftarrow$).