CSE 505: Concepts of Programming Languages

Dan Grossman

Fall 2007

Lecture 3— Operational Semantics for IMP

Where we are

- Done: IMP syntax, structural induction, Caml basics
- Today: IMP operational semantics
- Tonight: You could (almost?) finish homework 1

Review

IMP's abstract syntax is defined inductively:

$$
\begin{array}{rcl}
s & ::= & \textbf{skip} \mid x := e \mid s; s \mid \textbf{if}\ e\ s\ s \mid \textbf{while}\ e\ s \\
e & ::= & c \mid x \mid e + e \mid e * e \\
(c & \in & \{\ldots, -2, -1, 0, 1, 2, \ldots\}) \\
(x & \in & \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{z}_1, \mathrm{z}_2, \ldots, \ldots\})
\end{array}
$$

We haven't said what programs mean yet! (Syntax is boring)

But we have a social understanding about variables and control flow

Expression semantics

$$H ::= \cdot \mid H, x \mapsto c$$

$$\boxed{H \; ; \; e \Downarrow c}$$

CONST

$$\frac{}{H \; ; \; c \Downarrow c}$$

VAR

$$\frac{}{H \; ; \; x \Downarrow H(x)}$$

ADD

$$\frac{H \; ; \; e_1 \Downarrow c_1 \qquad H \; ; \; e_2 \Downarrow c_2}{H \; ; \; e_1 + e_2 \Downarrow c_1 + c_2}$$

MULT

$$\frac{H \; ; \; e_1 \Downarrow c_1 \qquad H \; ; \; e_2 \Downarrow c_2}{H \; ; \; e_1 * e_2 \Downarrow c_1 * c_2}$$

"pronounce" as proofs (upward) or evaluations (downward)

Expression semantics cont'd

$$H(x) = \begin{cases} c & \text{if} \quad H = H', x \mapsto c \\ H'(x) & \text{if} \quad H = H', y \mapsto c' \\ 0 & \text{if} \quad H = \cdot \end{cases}$$

Last case avoids "errors" (makes function *total*)

We have *rule schemas* ("rules"). We *instantiate* a rule by replacing metavariables appropriately.

Instantiating rules

Example instantiation:

$$\frac{\cdot, \mathrm{y} \mapsto \mathbf{4} \; ; \; \mathbf{3} + \mathrm{y} \Downarrow \mathbf{7} \qquad \cdot, \mathrm{y} \mapsto \mathbf{4} \; ; \; \mathbf{5} \Downarrow \mathbf{5}}{\cdot, \mathrm{y} \mapsto \mathbf{4} \; ; \; (\mathbf{3} + \mathrm{y}) + \mathbf{5} \Downarrow \mathbf{12}}$$

Instantiates:

$$\frac{H \; ; \; e_1 \Downarrow c_1 \qquad H \; ; \; e_2 \Downarrow c_2}{H \; ; \; e_1 + e_2 \Downarrow c_1 + c_2}$$

with $H = \cdot, \mathrm{y} \mapsto 4$, $e_1 = (\mathbf{3} + \mathrm{y})$, $c_1 = \mathbf{7}$, $e_2 = \mathbf{5}$, $c_2 = \mathbf{5}$

Derivations

A *(complete) derivation* is a tree of instantiations with *axioms* at the leaves.

Example:

$$\dfrac{\dfrac{\dfrac{}{\cdot, \mathrm{y}\mapsto\mathbf{4} \;;\; \mathbf{3} \Downarrow \mathbf{3}} \quad \dfrac{}{\cdot, \mathrm{y}\mapsto\mathbf{4} \;;\; \mathrm{y} \Downarrow \mathbf{4}}}{\cdot, \mathrm{y}\mapsto\mathbf{4} \;;\; \mathbf{3} + \mathrm{y} \Downarrow \mathbf{7}} \quad \dfrac{}{\cdot, \mathrm{y}\mapsto\mathbf{4} \;;\; \mathbf{5} \Downarrow \mathbf{5}}}{\cdot, \mathrm{y}\mapsto\mathbf{4} \;;\; (\mathbf{3} + \mathrm{y}) + \mathbf{5} \Downarrow \mathbf{12}}$$

So $\boldsymbol{H} \;;\; \boldsymbol{e} \Downarrow \boldsymbol{c}$ if there exists a derivation with $\boldsymbol{H} \;;\; \boldsymbol{e} \Downarrow \boldsymbol{c}$ at the root.

Some theorems

- Progress: For all H and e, there exists a c such that $H\ ;\ e \Downarrow c$.
- Determinacy: For all H and e, there is at most one c such that $H\ ;\ e \Downarrow c$.

We rigged it that way...

what would division, undefined-variables, or gettime() do?

Note: Our semantics is *syntax-directed*.

Some theory comments

Inference rules are PL notation for some standard math...

- "H and e evaluating to c" is a *relation* on triples of the form (H, e, c) (i.e., $H\ ;\ e \Downarrow c$)
- Relation defined inductively on the derivation *height*
- Can define syntax the same way:

$$\frac{}{c \in E} \qquad \frac{}{x \in E}$$

$$\frac{e_1 \in E \qquad e_2 \in E}{e_1 + e_2 \in E} \qquad \frac{e_1 \in E \qquad e_2 \in E}{e_1 * e_2 \in E}$$

Less metanotation for you, but not what "we" do

Statement semantics

$$\boxed{H_1 \; ; \; s_1 \rightarrow H_2 \; ; \; s_2}$$

ASSIGN

$$\frac{H \; ; \; e \Downarrow c}{H \; ; \; x := e \rightarrow H, x \mapsto c \; ; \; \mathsf{skip}}$$

SEQ1

$$\frac{}{H \; ; \; \mathsf{skip}; s \rightarrow H \; ; \; s}$$

SEQ2

$$\frac{H \; ; \; s_1 \rightarrow H' \; ; \; s_1'}{H \; ; \; s_1; s_2 \rightarrow H' \; ; \; s_1'; s_2}$$

IF1

$$\frac{H \; ; \; e \Downarrow c \qquad c > 0}{H \; ; \; \mathsf{if}\; e \; s_1 \; s_2 \rightarrow H \; ; \; s_1}$$

IF2

$$\frac{H \; ; \; e \Downarrow c \qquad c \leq 0}{H \; ; \; \mathsf{if}\; e \; s_1 \; s_2 \rightarrow H \; ; \; s_2}$$

Statement semantics cont'd

What about **while** e s (do s and loop if $e > 0$)?

$$\frac{\text{WHILE}}{H\ ;\ \textbf{while}\ e\ s \rightarrow H\ ;\ \textbf{if}\ e\ (s; \textbf{while}\ e\ s)\ \textbf{skip}}$$

Many other equivalent definitions possible

Program semantics

We defined $H \; ; \; s \rightarrow H' \; ; \; s'$, but what does "$s$" mean/do?

Our machine iterates: $H_1;s_1 \rightarrow H_2;s_2 \rightarrow H_3;s_3 \ldots$

Let $H_1 \; ; \; s_1 \rightarrow^* H_2 \; ; \; s_2$ mean "becomes after some number of steps" and pick a special "answer" variable ans

The program s produces c if $\cdot \; ; \; s \rightarrow^* H \; ; \;$ **skip** and $H(ans) = c$

Does every s produce a c?

Example program execution

$\mathtt{x} := \mathbf{3}; (\mathtt{y} := \mathbf{1}; \textsf{while}\ \mathtt{x}\ (\mathtt{y} := \mathtt{y} * \mathtt{x}; \mathtt{x} := \mathtt{x} - \mathbf{1}))$

(Let's write some of the state sequence. You can justify each step with a full derivation. Let $s = (\mathtt{y} := \mathtt{y} * \mathtt{x}; \mathtt{x} := \mathtt{x} - \mathbf{1})$.)

$$
\begin{array}{ll}
 & \cdot; \mathtt{x} := \mathbf{3}; \mathtt{y} := \mathbf{1}; \textsf{while}\ \mathtt{x}\ s \\
\rightarrow & \cdot, \mathtt{x} \mapsto \mathbf{3}; \textsf{skip}; \mathtt{y} := \mathbf{1}; \textsf{while}\ \mathtt{x}\ s \\
\rightarrow & \cdot, \mathtt{x} \mapsto \mathbf{3}; \mathtt{y} := \mathbf{1}; \textsf{while}\ \mathtt{x}\ s \\
\rightarrow^{2} & \cdot, \mathtt{x} \mapsto \mathbf{3}, \mathtt{y} \mapsto \mathbf{1}; \textsf{while}\ \mathtt{x}\ s \\
\rightarrow & \cdot, \mathtt{x} \mapsto \mathbf{3}, \mathtt{y} \mapsto \mathbf{1}; \textsf{if}\ \mathtt{x}\ (s; \textsf{while}\ \mathtt{x}\ s)\ \textsf{skip} \\
\rightarrow & \cdot, \mathtt{x} \mapsto \mathbf{3}, \mathtt{y} \mapsto \mathbf{1}; \mathtt{y} := \mathtt{y} * \mathtt{x}; \mathtt{x} := \mathtt{x} - \mathbf{1}; \textsf{while}\ \mathtt{x}\ s
\end{array}
$$

Continued...

$$\longrightarrow^{\mathbf{2}} \quad \cdot, \mathtt{x} \mapsto \mathbf{3}, \mathtt{y} \mapsto \mathbf{1}, \mathtt{y} \mapsto \mathbf{3}; \mathtt{x} := \mathtt{x} - \mathbf{1}; \textbf{while}\ \mathtt{x}\ s$$

$$\longrightarrow^{\mathbf{2}} \quad \cdot, \mathtt{x} \mapsto \mathbf{3}, \mathtt{y} \mapsto \mathbf{1}, \mathtt{y} \mapsto \mathbf{3}, \mathtt{x} \mapsto \mathbf{2}; \textbf{while}\ \mathtt{x}\ s$$

$$\longrightarrow \quad \ldots, \mathtt{y} \mapsto \mathbf{3}, \mathtt{x} \mapsto \mathbf{2}; \textbf{if}\ \mathtt{x}\ (s; \textbf{while}\ \mathtt{x}\ s)\ \textbf{skip}$$

$$\ldots$$

$$\longrightarrow \quad \ldots, \mathtt{y} \mapsto \mathbf{6}, \mathtt{x} \mapsto \mathbf{0}; \textbf{skip}$$

Where we are

We have defined $H\ ;\ e \Downarrow c$ and $H\ ;\ s \rightarrow H'\ ;\ s'$ and extended the latter to give s a meaning.

The way we did expressions is "large-step" or "natural".

The way we did statements is "small-step".

So now you have seen both.

Large-step does not distinguish errors and divergence.

Establishing Properties

We can prove a property of a terminating program by "running" it.

Example: Our last program terminates with $\mathtt{x}$ holding $\mathbf{0}$.

We can prove a program diverges, i.e., for all H and n,
$\cdot \; ; \; s \;\rightarrow^{n}\; H \; ; \;$ **skip** cannot be derived.

Example: **while** 1 **skip**

By induction on n with stronger induction hypothesis: If we can derive
$\cdot \; ; \; s \;\rightarrow^{n}\; H \; ; \; s'$ then s' is **while** 1 **skip** or
if 1 (**skip; while** 1 **skip**) **skip** or **skip; while** 1 **skip**.

More General Proofs

We can prove properties of executing all programs (satisfying another property)

Example: If H and s have no negative constants and $H\ ;\ s \rightarrow^* H'\ ;\ s'$, then H' and s' have no negative constants.

Example: If for all H, we know s_1 and s_2 terminate, then for all H, we know $H;(s_1; s_2)$ terminates.