Programming Languages — Homework 3
Operational semantics

Due: Thursday, 14 March 2013, 08:30

Several problems in this assignment use the small-step operational semantics for IMP, given on the next page.

1. [3 pts] **Evaluation of while.** Given the statement

$$i = 1;\ k = 0;\ \texttt{while}\ i < 3\ \textbf{do}\ (i := i + 1; k := k + i)$$

   show through a sequence of derivations of individual evaluation steps how this expression is evaluated to a configuration $\sigma, \texttt{pass}$ starting with a store σ_0. Show the resulting store σ.

2. [3 pts] **Auto-increment.** Suppose we add an auto-increment expression to IMP. Given a variable x containing an integer value, the expression x++ increments x by 1, updating the store, and evaluates to the previous value of x.

   (a) Since expressions can now modify the store, the form of the evaluation relation for expressions must change to include the updated store. The evaluation judgments now look like:

$$\sigma, e \longrightarrow \sigma', e'$$

   Show how the inference rules SC-ASN and EC-NOT change.

   (b) Write inference rules for the small-step operational semantics for the new expression x++. Note: you should not specify the behavior of ++ on boolean expressions.

3. [4 pts] **Expression evaluation is deterministic.**

   (a) Using the operational semantics of IMP on the next page—not your modified semantics above, prove by structural induction that expression evaluation is deterministic. That is, prove that for all expressions e (both arithmetic and boolean) and all stores σ if $\sigma \vdash e \longrightarrow e'$ and $\sigma \vdash e \longrightarrow e''$, then $e' = e''$. You need not prove that evaluation of statements s is deterministic.

   (b) Do the changes you made to the semantics in question 2 above change whether or not the language is deterministic? Argue why or why not. You need not do a full proof.

Submission

1. Complete the survey linked from the course web page after completing this assignment.
2. Submit a PDF on Moodle by the beginning of class on 14 March 2013. Include your name in the file. **OR** submit your solutions on paper in class on 14 March.

IMP

$$s ::= \texttt{pass} \mid x := a \mid s_1;\ s_2 \mid \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2 \mid \texttt{while}\ b\ \texttt{do}\ s \qquad \text{statements}$$
$$a ::= x \mid n \mid a_1 + a_2 \qquad \text{arithmetic expressions}$$
$$b ::= \texttt{true} \mid \texttt{false} \mid a_1 < a_2 \mid b_1\ \texttt{and}\ b_2 \mid \texttt{not}\ b \qquad \text{boolean expressions}$$

Statement evaluation is defined by judgments of the form $\sigma, s \longrightarrow \sigma', s'$ ("s in store σ reduces to s' in σ'"). Evaluation halts in the configuration $\sigma, \texttt{pass}$. Note that the language is restricted syntactically to ensure that variables contain only integers. A store σ is a function from variables x to values n. $\sigma[x \mapsto n]$ is the store that maps x to n and y ($\neq x$) to $\sigma(y)$.

$$\sigma, \texttt{pass};\ s \longrightarrow \sigma, s \qquad \text{(S-SEQ)}$$

$$\frac{\sigma, s_1 \longrightarrow \sigma', s_1'}{\sigma, s_1;\ s_2 \longrightarrow \sigma', s_1';\ s_2} \qquad \text{(SC-SEQ)}$$

$$\sigma, x := n \longrightarrow \sigma[x \mapsto n], \texttt{pass} \qquad \text{(S-ASN)}$$

$$\frac{\sigma \vdash a \longrightarrow a'}{\sigma, x := a \longrightarrow \sigma, x := a'} \qquad \text{(SC-ASN)}$$

$$\sigma, \texttt{if true then}\ s_1\ \texttt{else}\ s_2 \longrightarrow \sigma, s_1 \qquad \text{(S-IFTRUE)}$$

$$\sigma, \texttt{if false then}\ s_1\ \texttt{else}\ s_2 \longrightarrow \sigma, s_2 \qquad \text{(S-IFFALSE)}$$

$$\frac{\sigma \vdash b \longrightarrow b'}{\sigma, \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2 \longrightarrow \sigma, \texttt{if}\ b'\ \texttt{then}\ s_1\ \texttt{else}\ s_2} \qquad \text{(SC-IF)}$$

$$\sigma, \texttt{while}\ b\ \texttt{do}\ s \longrightarrow \sigma, \texttt{if}\ b\ \texttt{then}\ (s;\ \texttt{while}\ b\ \texttt{do}\ s)\ \texttt{else pass} \qquad \text{(S-WHILE)}$$

Expression evaluation is defined by judgments of the form $\sigma \vdash e \longrightarrow e'$ ("e reduces to e' with store σ").

To simplify the rules, we add the following syntax:

$$e ::= a \mid b \qquad \text{expressions}$$
$$v ::= n \mid \texttt{true} \mid \texttt{false} \qquad \text{values}$$
$$o ::= + \mid < \mid \texttt{and} \qquad \text{binary operations}$$

$$\sigma \vdash x \longrightarrow \sigma(x) \qquad \text{(E-VAR)}$$

$$\sigma \vdash n_1 < n_2 \longrightarrow \texttt{true} \qquad (\text{where } n_1 < n_2) \qquad \text{(E-LT)}$$

$$\sigma \vdash n_1 < n_2 \longrightarrow \texttt{false} \qquad (\text{where } n_1 \geq n_2) \qquad \text{(E-GE)}$$

$$\sigma \vdash n_1 + n_2 \longrightarrow n \qquad (\text{where } n = n_1 + n_2) \qquad \text{(E-ADD)}$$

$$\sigma \vdash \texttt{false and}\ b \longrightarrow \texttt{false} \qquad \text{(E-ANDFALSE)}$$

$$\sigma \vdash \texttt{true and}\ b \longrightarrow b \qquad \text{(E-ANDTRUE)}$$

$$\frac{\sigma \vdash e_1 \longrightarrow e_1'}{\sigma \vdash e_1\ o\ e_2 \longrightarrow e_1'\ o\ e_2} \qquad \text{(EC-L)}$$

$$\frac{\sigma \vdash e \longrightarrow e'}{\sigma \vdash v\ o\ e \longrightarrow v\ o\ e'} \qquad \text{(EC-R)}$$

$$\sigma \vdash \texttt{not true} \longrightarrow \texttt{false} \qquad \text{(E-NOTTRUE)}$$

$$\sigma \vdash \texttt{not false} \longrightarrow \texttt{true} \qquad \text{(E-NOTFALSE)}$$

$$\frac{\sigma \vdash e \longrightarrow e'}{\sigma \vdash \texttt{not}\ e \longrightarrow \texttt{not}\ e'} \qquad \text{(EC-NOT)}$$