HW 9 – Structural Operational and Transition Semantics

CS 421 – Fall 2012
Revision 1.1

Assigned Saturday, November 17, 2012
Due Thursday, November 29, 2012, 11:59 PM
Extension 48 hours (20% penalty)

1 Change Log

1.1 Changed "operational semantics" to "structural operational semantics".

1.0 Initial Release.

2 Turn-In Procedure

Answer the problems below, save your work as a PDF (either scanned if handwritten or converted from a program), and hand in the PDF. Your file should be named `hw9.pdf`.

3 Objectives and Background

The purpose of this HW is to test your understanding of

- The difference between structural operational semantics and transition semantics.
- How to create rules for structural operational semantics.
- How to write rules for transition semantics.

All the homework will use the syntax as:

$I \in Identifiers$
$N \in Numerals$
$B ::= \texttt{true} \mid \texttt{false} \mid B \ \& \ B \mid B \ \texttt{or} \ B \mid \texttt{not} \ B \mid E < E \mid E = E$
$E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid -E$
$C ::= \texttt{skip} \mid C; C \mid I \texttt{::=} E \mid \texttt{if} \ B \ \texttt{then} \ C \ \texttt{else} \ C \ \texttt{fi} \mid \texttt{while} \ B \ \texttt{do} \ C \ \texttt{od}$

4 Problems

1. (20 points) Add a new increment operator `++I` to the syntax of expression `E`.

   a. (10 points) Add the structural operational semantics (*a.k.a.* natural semantics) for this operator. The semantics of the operator is to add one with the current value I, then store the new value into I. Also, modify the set of structural operational semantics rules for other operators in order to fit the new changes caused by adding the increment operator if any modification is needed.

b. (10 points) Add the transition semantics for this operator. It has the same meaning as part a. Also, modify the set of transition semantics rules for other operators in order to fit the new changes caused by adding the increment operator if any modification is needed.

Solution:
a. Natural Semantics:
Expressions:

$$\frac{}{(N,m) \Downarrow (N,m)} \qquad \frac{}{(I,m) \Downarrow (m(I),m)} \qquad \frac{v = m(I)+1}{(++I,m) \Downarrow (v, m[I \leftarrow v])} \qquad \frac{(E,m) \Downarrow (u,m') \quad v = -u}{-E \Downarrow (v,m')}$$

$$\frac{(E_1,m) \Downarrow (u,m') \quad (E_2,m') \Downarrow (v,m'') \quad w = u \oplus v \quad \oplus \in \{+,-,*\}}{(E_1 \oplus E_2, m) \Downarrow (w,m'')}$$

Booleans:

$$\frac{}{(\texttt{true}\,,m) \Downarrow (\texttt{true}\,,m)} \qquad \frac{}{(\texttt{false}\,,m) \Downarrow (\texttt{false}\,,m)} \qquad \frac{(B,m) \Downarrow (b,m') \quad b' = \neg b}{(\texttt{not}\ B, m) \Downarrow (b',m')}$$

$$\frac{(B_1,m) \Downarrow (\texttt{false}\,,m')}{(B_1\ \&\ B_2, m) \Downarrow (\texttt{false}\,,m')} \qquad \frac{(B_1,m) \Downarrow (\texttt{true}\,,m') \quad (B_2,m') \Downarrow (b,m'')}{(B_1\ \&\ B_2, m) \Downarrow (b,m'')}$$

$$\frac{(B_1,m) \Downarrow (\texttt{true}\,,m') \qquad (B_1,m) \Downarrow (\texttt{false}\,,m') \quad (B_2,m') \Downarrow (b,m'')}{(B_1\ \texttt{or}\ B_2, m) \Downarrow (\texttt{true}\,,m') \qquad\qquad (B_1\ \texttt{or}\ B_2, m) \Downarrow (b,m'')}$$

$$\frac{(E_1,m) \Downarrow (u,m') \quad (E_2,m') \Downarrow (v,m'') \quad b = u \sim v \quad \sim \in \{=,<,<=,\ldots\}}{(E_1 \sim E_2, m) \Downarrow (w,m'')}$$

Commands:
The rules for `skip` and sequencing remain unchanged.

$$\frac{(E,m) \Downarrow (v,m')}{(I ::= E, m) \Downarrow m'[I \rightarrow v]} \qquad \frac{(B,m) \Downarrow (\texttt{true}\,,m') \quad (C,m') \Downarrow m''}{\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C'\texttt{fi} \Downarrow m''} \qquad \frac{(B,m) \Downarrow (\texttt{false}\,,m') \quad (C',m') \Downarrow m''}{\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C'\texttt{fi} \Downarrow m''}$$

$$\frac{(B,m) \Downarrow (\texttt{false}\,,m')}{(\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}\,,m) \Downarrow m'} \qquad \frac{(B,m) \Downarrow (\texttt{true}\,,m') \quad (C,m') \Downarrow m'' \quad ((\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}\,,m'') \Downarrow m'''}{(\texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}\,,m) \Downarrow m'}$$

b. Transition Semantics:
Expressions:

$$\frac{}{(I,m) \longrightarrow (m(I),m)} \qquad \frac{v = m(I)+1}{(++I,m) \longrightarrow (v, m[I \leftarrow v])} \qquad \frac{(E,m) \longrightarrow (E',m')}{-E \longrightarrow (-E',m')} \qquad \frac{v = -u}{-u \longrightarrow (v,m')}$$

$$\frac{(E_1,m) \longrightarrow (E_1',m')}{(E_1 \oplus E_2, m) \longrightarrow (E_1' \oplus E_2, m')} \qquad \frac{(E_2,m) \longrightarrow (E_2',m')}{(v \oplus E, m) \longrightarrow (v \oplus E_2, m')} \qquad \frac{w = u \oplus v}{(u \oplus v, m) \longrightarrow (w,m)}$$

Booleans:

$$\frac{}{(\texttt{true}\ \&\ B, m) \longrightarrow (B,m)} \qquad \frac{}{(\texttt{false}\ \&\ B, m) \longrightarrow (\texttt{false}\,,m)} \qquad \frac{(B_1,m) \longrightarrow (B_1',m')}{(B_1\ \&\ B_2, m) \longrightarrow (B_1'\ \&\ B_2, m')}$$

$$\frac{}{(\texttt{true or}\ B, m) \longrightarrow (\texttt{true}\ , m)} \qquad \frac{}{(\texttt{false or}\ B, m) \longrightarrow (B, m)} \qquad \frac{(B_1, m) \longrightarrow (B_1', m')}{(B_1\ \texttt{or}\ B_2, m) \longrightarrow (B_1'\ \texttt{or}\ B_2, m')}$$

$$\frac{}{(\texttt{not true}\ , m) \longrightarrow (\texttt{false}\ , m)} \qquad \frac{}{(\texttt{not false}\ , m) \longrightarrow (\texttt{true}\ , m)} \qquad \frac{(B, m) \longrightarrow (B', m')}{(\texttt{not}\ B, m) \longrightarrow (\texttt{not}\ B', m')}$$

Commands:
The rules for `skip`, assignment of a value, sequencing, `if true`, `if false`, and `while` remain unchanged. That leaves:

$$\frac{(E, m) \longrightarrow (E', m')}{(I ::= E, m) \longrightarrow (I ::= E', m')} \qquad \frac{(B, m) \longrightarrow (B, m')}{(\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C'\texttt{fi}, m) \longrightarrow (\texttt{if}\ B'\ \texttt{then}\ C\ \texttt{else}\ C'\texttt{fi}, m')}$$

2. Extra Credit (20 points)

   Add a new for-loop operator $\texttt{for (}C\texttt{; }B\texttt{; }C\texttt{) do }C\texttt{ od}$ to the syntax of commands `C`.

   a. (10 points) Add the structural operational semantics for this operator. Also, modify the set of structural operational semantics rules for other operators in order to fit the new changes caused by adding the for-loop operator if any modification is needed. The for-loop behaves as follows:
      (a) At the start of the execution of the for-loop $\texttt{for (}C_1\texttt{; }B\texttt{; }C_2\texttt{) do }C_3\texttt{ od}$ the command C_1 is executed. This may be used to initialize an indexing variable, for example. This command is executed only once per executing of the whole for-loop. We shall refer to the exeuction point after this initialization as the "start of the loop".
      (b) At the start of the loop, the boolean expression B is evaluated. If its value is `false`, the whole for-loop terminates.
      (c) If the value for B is `true`, then the main body of the loop C_3 is executed.
      (d) At the completion of the body of the loop C_4, the update command C_3 is executed (to update an indexing variable, for example) and then execution returns to the start of the loop (*i.e.* we go back to evaluating B in step (b)).

      Solution: Only Commands require any changes, and there we only require the addition of two new rules for the new `for` construct:

$$\frac{(C_1, m) \Downarrow m' \quad (B, m) \Downarrow \texttt{false}}{(\texttt{for (}C_1\texttt{; }B\texttt{; }C_2\texttt{) do }C_3\texttt{ od}, m) \Downarrow m'}$$

$$\frac{(C_1, m) \Downarrow m_1 \quad (B, m_1) \Downarrow \texttt{true} \quad (C_3, m_1) \Downarrow m_2 \quad (C_2, m_2 \Downarrow m_2) \qquad (\texttt{for (skip; }B\texttt{; }C_2\texttt{) do }C_3\texttt{ od}, m_2) \Downarrow m'}{(\texttt{for (}C_1\texttt{; }B\texttt{; }C_2\texttt{) do }C_3\texttt{ od}, m) \Downarrow m'}$$

   b. (10 points) Add the transition semantics for this operator. Also, modify the set of transition semantics rules for other operators in order to fit the new changes caused by adding the for-loop operator if any modification is needed.

Solution: b. Transition Semantics:
Only Commands require any changes, and there we only require the addition of one new rule for the new `for` construct:

$$\frac{}{(\texttt{for (}C_1\texttt{; }B\texttt{; }C_2\texttt{) do }C_3\texttt{ od}, m) \longrightarrow (C_1\texttt{; while }B\texttt{ do }C_3\texttt{; }C_2\texttt{ od}\ , m)}$$