Due Thursday, February 25, 2010

Due to popular demand, this homework assignment focuses on providing more practice with structural induction.

1 Operational Semantics of IMP.

This homework assignment considers properties about the operational semantics of IMP. As an aid and to ensure we consider the same systems, a listing of the operational semantics of IMP is provided in this section.

Figure 1 presents the big-step operational semantics of IMP commands by defining the judgment

\[ \langle c, \sigma \rangle \Downarrow \sigma' , \]

which says that command \( c \) in state \( \sigma \) evaluates to state \( \sigma' \). This judgment relies on evaluation judgments for evaluating arithmetic expressions \( e \)

\[ \langle e, \sigma \rangle \Downarrow n \quad \text{where } n \text{ is an integer} \]

and evaluating boolean expressions \( b \)

\[ \langle b, \sigma \rangle \Downarrow t \quad \text{where } t \text{ is a boolean.} \]

In this figure, we give names to the inference rules for convenience. These rule names are prefixed with \( E \) for “evaluation.”

In Figure 2, we consider the structural small-step operational semantics of IMP commands. In particular, we define the following judgment that describes a transition relation:

\[ \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle , \]

which says command \( c \) in state \( \sigma \) steps to command \( c' \) and state \( \sigma' \). The rule names corresponding to instructions are prefixed with \( I \), while the search rules have names prefixed with \( S \). The judgment \( c \) terminal indicates the terminal programs (i.e., those that do not reduce any further).

*Updated on February 19, 2010.
\[
\begin{array}{ll}
\langle c, \sigma \rangle \Downarrow \sigma' & \text{ESkip} \\
\langle \text{skip}, \sigma \rangle \Downarrow \sigma & (e, \sigma) \Downarrow n \\
\langle c_1, \sigma \rangle \Downarrow \sigma' & \langle c_2, \sigma' \rangle \Downarrow \sigma'' & \text{ESeq} \\
\langle c_1; c_2, \sigma \rangle \Downarrow \sigma'' & \langle b, \sigma \rangle \Downarrow \text{true} & \langle c_1, \sigma \rangle \Downarrow \sigma' & \text{EIFT} \\
\langle b, \sigma \rangle \Downarrow \text{false} & \langle c_2, \sigma \rangle \Downarrow \sigma' & \langle b, \sigma \rangle \Downarrow \text{false} & \langle c_1 \text{ else } c_2, \sigma \rangle \Downarrow \sigma' & \text{EIFF} \\
\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma & (b, \sigma) \Downarrow \text{true} & \langle c, \sigma \rangle \Downarrow \sigma' & \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \text{EIFT} \\
\langle b, \sigma \rangle \Downarrow \text{false} & \langle c, \sigma \rangle \Downarrow \sigma' & \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \langle c_1 \text{ else } c_2, \sigma \rangle \Downarrow \sigma' & \text{EIFF} \\
\langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \langle b, \sigma \rangle \Downarrow \text{false} & \langle c, \sigma \rangle \Downarrow \sigma' & \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \text{EIFT} \\
\langle b, \sigma \rangle \Downarrow \text{false} & \langle c, \sigma \rangle \Downarrow \sigma' & \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \langle c_1 \text{ else } c_2, \sigma \rangle \Downarrow \sigma' & \text{EIFF} \\
\langle b, \sigma \rangle \Downarrow \text{false} & \langle c, \sigma \rangle \Downarrow \sigma' & \langle \text{while } b \text{ do } c, \sigma \rangle \Downarrow \sigma'' & \langle c_1 \text{ else } c_2, \sigma \rangle \Downarrow \sigma' & \text{EIFF} \\
\end{array}
\]

Figure 1: Big-step operational semantics of IMP.

\[
\begin{array}{ll}
\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \\
\langle x := n, \sigma \rangle \rightarrow \langle \text{skip}, \sigma[x := n] \rangle & \text{IAssign} \\
\langle \text{skip; } c, \sigma \rangle \rightarrow \langle c, \sigma \rangle & \text{ISEq} \\
\langle \text{if true then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_1, \sigma \rangle & \text{EIFT} \\
\langle \text{if false then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle c_2, \sigma \rangle & \text{EIFF} \\
\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \langle \text{if } b \text{ then } c; \text{ while } b \text{ do } c \text{ else skip}, \sigma \rangle & \text{IWhile} \\
\langle c, \sigma \rangle \rightarrow \langle e', \sigma \rangle & \text{SAssign} \\
\langle c_1; c_2, \sigma \rangle \rightarrow \langle c_1', c_2', \sigma' \rangle & \text{SSeq} \\
\langle b, \sigma \rangle \rightarrow \langle b', \sigma \rangle & \text{EIFF} \\
\langle c_1 \text{ else } c_2, \sigma \rangle \rightarrow \langle \text{if } b' \text{ then } c_1 \text{ else } c_2, \sigma \rangle & \text{EIFF} \\
\text{skip terminal} & \text{TStart} \\
\end{array}
\]

Figure 2: Small-step operational semantics of IMP.
To express evaluation using small-step operational semantics, we define a judgment that captures some number of steps of the one-step relation:

\[ \langle c, \sigma \rangle \rightarrow^* \langle c', \sigma' \rangle \]

The above judgment can be read as command \( c \) in state \( \sigma \) evaluates to \( c' \) and \( \sigma' \) in some number of steps. This relation is the reflexive-transitive closure of the transition relation \( \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \) and is defined in Figure 3. The rule names are prefixed with MS for “multi-step.”

2 Exercises

Exercise 1: Bookkeeping. Indicate in a sentence or two how much time you spent on this homework, how difficult you found it subjectively, and what you found to be the hardest part. Any non-empty answer will receive full credit.

If there is something else you would like to share about yourself, please do so. Also, if your opinions have changed since the last homework, indicate one thing you like about the class so far and one thing you would change about it.

Exercise 2: Determinism of Small-Step Evaluation. Consider our structural small-step operational semantics of IMP (without any extensions). We can show that with these semantics, either we have reached a value or there is a unique next step of evaluation. In other words, if have not reached a value, we can always make progress and evaluation is deterministic.

Lemma 1 (Deterministic Progress of Arithmetic Expression Evaluation). For all arithmetic expressions \( e \) and states \( \sigma \), either \( e \text{ value} \) or there exists a unique \( e' \) such that \( e \rightarrow^*_\sigma e' \).
Proof. By induction on the structure of \( e \). \( \square \)

**Lemma 2** (Deterministic Progress of Boolean Expression Evaluation). For all boolean expressions \( b \) and states \( \sigma \), either \( b \) value or there exists a unique \( b' \) such that \( b \rightarrow_\sigma b' \).

Proof. By induction on the structure of \( b \). \( \square \)

**Theorem 3** (Deterministic Progress of Command Evaluation). For all commands \( c \) and states \( \sigma \), either \( c \) terminal or there exists a unique \( c' \) and \( \sigma' \) such that \( \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \).

- For this exercise, prove Theorem 3. You may assume we have proofs for Lemma 1 and Lemma 2.

**Exercise 3:** Equivalence of Big-Step and Small-Step Evaluation.
Consider our big-step operational semantics and our structural small-step operational semantics of IMP. As a sanity check, we would like to show that the two forms of operational semantics correspond, that is,

\[
\langle c, \sigma \rangle \Downarrow \sigma' \quad \text{ iff } \quad \langle c, \sigma \rangle \rightarrow^* \langle \text{skip}, \sigma' \rangle.
\]

In this exercise, we will build up a proof of this correspondence.

1. We first consider the direction from small-step to big-step semantics (right-to-left in the above). This statement is difficult to prove directly. Instead, we first show that if we have a one-step evaluation in the small-step semantics and from that configuration there is an evaluation in the big-step semantics, then original configuration evaluates to that same result.

**Lemma 4.** If \( e \rightarrow_\sigma e' \) and \( \langle e', \sigma \rangle \Downarrow n \), then \( \langle e, \sigma \rangle \Downarrow n \).

**Lemma 5.** If \( b \rightarrow_\sigma b' \) and \( \langle b', \sigma \rangle \Downarrow t \), then \( \langle b, \sigma \rangle \Downarrow t \).

**Lemma 6.** If \( S :: \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \) and \( E :: \langle c', \sigma' \rangle \Downarrow \sigma'' \), then \( E' :: \langle c, \sigma \rangle \Downarrow \sigma'' \).

- For this part, prove Lemma 6. You may assume we have proofs for Lemma 4 and Lemma 5.

2. Now, we can show that if we reach \textbf{skip} (the terminal program) using the small-step operational semantics, then there is an evaluation using the big-step operational semantics.
Theorem 7. If $\mathcal{M} :: \langle c, \sigma \rangle \xrightarrow{\ast} \langle \text{skip}, \sigma' \rangle$, then $\mathcal{E} :: \langle c, \sigma \rangle \Downarrow \sigma'$.

- For this part, prove Theorem 7.

3. (Extra Credit). To complete this correspondence, we can also prove the other direction.

Theorem 8. If $\langle c, \sigma \rangle \Downarrow \sigma'$, then $\langle c, \sigma \rangle \xrightarrow{\ast} \langle \text{skip}, \sigma' \rangle$.

Note that this proof is rather long and requires the introduction of some lemmas. Please attempt this part only after completing the rest of the homework.