CS 440: Programming Languages

Assignment: Big-step semantics

IMP Rules

The following are the big-step semantic rules of the simple imperative language (IMP) as described in class.

$$\text{LITERAL}\ \frac{}{< i, \sigma >\Downarrow_e i}\ \text{WHEN}\ i \in \mathbb{Z} \qquad \text{VAR}\ \frac{}{< u, \sigma >\Downarrow_e v}\ \text{IF}\ u{:=}v \in \sigma$$

$$\text{ARITH}\ \frac{< e_1, \sigma >\Downarrow_e v_1 \qquad < e_2, \sigma >\Downarrow_e v_2}{< e_1 \oplus e_2, \sigma >\Downarrow_e v_1 \oplus v_2}$$

Figure 1: Arithmetic expressions

$$\text{LITERAL}\ \frac{}{< b, \sigma >\Downarrow_b b}\ \text{IF}\ b \in \{\mathbf{true}, \mathbf{false}\} \qquad \text{VAR}\ \frac{}{< u, \sigma >\Downarrow_b v}\ \text{IF}\ u{:=}v \in \sigma$$

$$\text{REL}\ \frac{< e_1, \sigma >\Downarrow_e v_1 \qquad < e_2, \sigma >\Downarrow_e v_2}{< e_1 \backsim e_2, \sigma >\Downarrow_b v_1 \backsim v_2}$$

Figure 2: Boolean expressions

$$\text{SKIP}\ \frac{}{< \mathbf{skip}, \sigma >\Downarrow \sigma} \qquad \text{ASSIGN}\ \frac{< e, \sigma >\Downarrow_e v}{< x := e, \sigma >\Downarrow \sigma[x := v]}$$

$$\text{SEQ}\ \frac{< S_1, \sigma >\Downarrow \sigma' \qquad < S_2, \sigma' >\Downarrow \sigma''}{< S_1; S_2, \sigma >\Downarrow \sigma''} \qquad \text{IF-T}\ \frac{< b, \sigma >\Downarrow_b \mathbf{true} \qquad < S_1, \sigma >\Downarrow \sigma'}{< \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, \sigma >\Downarrow \sigma'}$$

$$\text{IF-F}\ \frac{< b, \sigma >\Downarrow_b \mathbf{false} \qquad < S_2, \sigma >\Downarrow \sigma'}{< \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, \sigma >\Downarrow \sigma'} \qquad \text{WHILE-F}\ \frac{< b, \sigma >\Downarrow_b \mathbf{false}}{< \mathbf{while}\ b\ \mathbf{do}\ S, \sigma >\Downarrow \sigma}$$

$$\text{WHILE-T}\ \frac{< b, \sigma >\Downarrow_b \mathbf{true} \qquad < S, \sigma >\Downarrow \sigma' \qquad < \mathbf{while}\ b\ \mathbf{do}\ S, \sigma' >\Downarrow \sigma''}{< \mathbf{while}\ b\ \mathbf{do}\ S, \sigma >\Downarrow \sigma''}$$

Figure 3: Statements

Logistics and Submission

Please submit your solutions as a PDF (typed or *neatly* handwritten!) on Blackboard by the due date.

1 Extending the language

1. (5 points) We wish to add Boolean negation to IMP, via the `!` operator. Write down inference rules to describe the big-step semantics of this operator.

2. (10 points) We wish to add **for** loops to IMP, which will have the form "**for** v **in** a_0 **to** a_1 **do** S", which will run statement S with the variable v taking on values $a_0, a_0 + 1, ..., a_1$. E.g., "**for x in** 1 **to** 5 **do** S" will run S with **x** taking on values 1, 2, 3, 4, 5, in that order.

   Write down inference rules to describe the big-step semantics of the **for** statement. Note that the loop variable is allowed to clash with pre-existing variables, and it may remain in the environment after the loop completes.

2 Proofs

3. (5 points) Draw a proof tree for the following assertion:
   $< t := a + b; a := b; b := t, \{a := 5, b := 10\} >\Downarrow \{t := 15, a := 10, b := 15\}$

4. (5 points) Draw a proof tree for the following assertion:
   $< \textbf{if } x > y \textbf{ then } m := x * 10 \textbf{ else } m := y * 10, \{x := 10, y := 20\} >\Downarrow \{x := 10, y := 20, m := 200\}$

5. Consider the program P = "**for x in** m **to** n **do sum** := **sum** + **x**" (which makes use of the **for** statement defined in the previous section). Note that m and n are $n$ot variables, but represent arbitrary integer constants.

   (a) (5 points) Describe the environment σ' such that $< P, \sigma_0 >\Downarrow \sigma'$, where σ_0 is an environment which maps all variables to 0. Hint: you may make use of the summation operator (Σ).

   (b) (5 points) Write down a program Q using the **while** statement, which is semantically equivalent to P.

   (c) (10 points) Prove that P and Q are equivalent when run in starting state σ_0.