IMP Rules

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

LITERAL \( <i, \sigma> \Downarrow_e i \) WHEN \( i \in \mathbb{Z} \)

VAR \( <u, \sigma> \Downarrow_e v \) IF \( u := v \in \sigma \)

ARITH \( <e_1, \sigma> \Downarrow_e v_1 < e_2, \sigma> \Downarrow_e v_2 \)
\( <e_1 + e_2, \sigma> \Downarrow_e v_1 + v_2 \)

Figure 1: Arithmetic expressions

LITERAL \( <b, \sigma> \Downarrow_b b \) IF \( b \in \{\text{true, false}\} \)

VAR \( <u, \sigma> \Downarrow_b v \) IF \( u := v \in \sigma \)

REL \( <e_1, \sigma> \Downarrow_e v_1 < e_2, \sigma> \Downarrow_e v_2 \)
\( <e_1 \sim e_2, \sigma> \Downarrow_b v_1 \sim v_2 \)

Figure 2: Boolean expressions

SKIP \( <\text{skip}, \sigma> \Downarrow \sigma \)

ASSIGN \( <e, \sigma> \Downarrow_e v < x := e, \sigma > \Downarrow e \sigma[x := v] \)

SEQ \( <S_1, \sigma> \Downarrow \sigma' \) \( <S_2, \sigma> \Downarrow \sigma'' \)
\( <S_1; S_2, \sigma> \Downarrow \sigma''' \)

IF-T \( <b, \sigma> \Downarrow_b \text{true} \) \( <S_1, \sigma> \Downarrow \sigma' \)
\( <\text{if} \; b \; \text{then} \; S_1 \; \text{else} \; S_2, \sigma > \Downarrow \sigma'' \)

IF-F \( <b, \sigma> \Downarrow_b \text{false} \)
\( <\text{if} \; b \; \text{then} \; S_1 \; \text{else} \; S_2, \sigma > \Downarrow \sigma'' \)

WHILE-F \( <b, \sigma> \Downarrow_b \text{false} \)
\( <\text{while} \; b \; \text{do} \; S, \sigma > \Downarrow \sigma'' \)

WHILE-T \( <b, \sigma> \Downarrow_b \text{true} \) \( <S, \sigma > \Downarrow \sigma' \)
\( <\text{while} \; b \; \text{do} \; S, \sigma' > \Downarrow \sigma'' \)
\( <\text{while} \; b \; \text{do} \; S, \sigma > \Downarrow \sigma'' \)

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:

$\langle x > y \text{ then } m := x * 10 \text{ else } m := y * 10, \{x := 10, y := 20\} \downarrow \{x := 10, y := 20, m := 200\}$

5. Consider the program $P = \text{"for } x \text{ in } m \text{ to } n \text{ do sum := sum + x"}$ (which makes use of the for statement defined in the previous section). Note that $m$ and $n$ are not variables, but represent arbitrary integer constants.

(a) (5 points) Describe the environment $\sigma'$ such that $\langle P, \sigma_0 \rangle \downarrow \sigma'$, where $\sigma_0$ is an environment which maps all variables to 0. Hint: you may make use of the summation operator ($\Sigma$).

(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 $\sigma_0$. 