CS 565 Spring 2022 Homework 3 (Big-Step Semantics)

Your name: Shuang Wu (wu1716@purdue.edu)

February 13, 2022

Using the big-step operational semantics of Imp (shown in Figure 1), either (i) fill in the final states produced by each of the following Imp programs and give a derivation justifying your answer **or** (ii) state that the program does not terminate. **Note**: You don't need to supply the derivations for any arithmetic or boolean expressions, just for the Imp statements.

Problem 1 (1 point).

$$\dfrac{\textsc{E-Ass}\ \dfrac{\textsc{E-Const}\ \dfrac{}{\emptyset, 0 \Downarrow_A 0}}{\emptyset\ , \mathrm{X} := 0 \Downarrow [\mathrm{X}\mapsto 0]} \qquad \dfrac{\dfrac{}{[\mathrm{X}\mapsto 0], 1 \Downarrow_A 1}\ \textsc{E-Const}}{[\mathrm{X}\mapsto 0],\ \mathrm{Y} := 1 \Downarrow [\mathrm{X}\mapsto 0][\mathrm{Y}\mapsto 1]}\ \textsc{E-Ass}}{\emptyset\ , \mathrm{X} := 0;\ \mathrm{Y} := 1 \Downarrow [\mathrm{X}\mapsto 0][\mathrm{Y}\mapsto 1]}\ \textsc{E-Seq}$$

Problem 2 (1 point).

$$\dfrac{\textsc{E-False}\ \dfrac{}{[X \mapsto 2], X \leq 1 \Downarrow_B \text{false}} \qquad \dfrac{\dfrac{}{[X \mapsto 2], 4 \Downarrow_A 4}\ \textsc{E-Count}}{[X \mapsto 2], Z := 4 \Downarrow [X \mapsto 2][Z \mapsto 4]}\ \textsc{E-Ass}}{[X \mapsto 2], \textbf{if } (X \leq 1) \textbf{ then } Y := X + 3 \textbf{ else } Z := 4 \textbf{ end} \Downarrow [X \mapsto 2][Z \mapsto 4]}\ \textsc{E-IfFalse}$$

Problem 3 (1 point).

$$[X \mapsto 0], \textbf{while } (X \leq 1) \textbf{ do } Y := Y + 1 \textbf{ end } \Downarrow$$

The program does not terminate.

Problem 4 (1 point).

$$
\begin{prooftree}
\AxiomC{}
\LeftLabel{E-True}
\UnaryInfC{$[Y \mapsto 0], Y \leq 1 \Downarrow_B \text{true}$}
\AxiomC{}
\LeftLabel{E-Var}
\UnaryInfC{$[Y \mapsto 0], Y \Downarrow_A 0$}
\AxiomC{}
\RightLabel{E-Const}
\UnaryInfC{$[Y \mapsto 0], 1 \Downarrow_A 1$}
\RightLabel{E-Add}
\BinaryInfC{$[Y \mapsto 0], Y + 1 \Downarrow_A 1$}
\RightLabel{E-Ass}
\UnaryInfC{$[Y \mapsto 0], Y := Y + 1 \Downarrow [Y \mapsto 1]$}
\AxiomC{}
\LeftLabel{E-True}
\UnaryInfC{$[Y \mapsto 1], (Y \leq 1) \Downarrow_B \text{true}$}
\AxiomC{}
\LeftLabel{E-Var}
\UnaryInfC{$[Y \mapsto 1], Y \Downarrow_A 1$}
\AxiomC{}
\RightLabel{E-Const}
\UnaryInfC{$[Y \mapsto 1], 1 \Downarrow_A 1$}
\RightLabel{E-Add}
\BinaryInfC{$[Y \mapsto 1], Y := Y + 1 \Downarrow_A 2$}
\RightLabel{E-Ass}
\UnaryInfC{$[Y \mapsto 1], Y := Y + 1 \Downarrow [Y \mapsto 2]$}
\AxiomC{}
\LeftLabel{E-False}
\UnaryInfC{$[Y \mapsto 2], Y \leq 1 \Downarrow_B \text{false}$}
\RightLabel{E-WhileFalse}
\UnaryInfC{$[Y \mapsto 2], \textbf{while } (Y \leq 1) \textbf{ do } Y := Y + 1 \textbf{ end} \Downarrow [Y \mapsto 2]$}
\RightLabel{E-WhileTrue}
\TrinaryInfC{$[Y \mapsto 1], \textbf{while } (Y \leq 1) \textbf{ do } Y := Y + 1 \textbf{ end} \Downarrow [Y \mapsto 2]$}
\RightLabel{E-WhileTrue}
\TrinaryInfC{$[Y \mapsto 0], \textbf{while } (Y \leq 1) \textbf{ do } Y := Y + 1 \textbf{ end} \Downarrow [Y \mapsto 2]$}
\end{prooftree}
$$

Problem 5 (2 points). Several cutting edge languages, including Perl, Visual Basic, and Pascal include a repeat c until b loop construct. These loops differ from the standard **while** loops in two ways:

1. the loop guard is checked /after/ the execution of the body, so the loop always executes at least once.
2. the loop continues executing as long as the condition is false.

Write down the big-step reduction rules for repeat loops in IMP.

$$\frac{\sigma, \mathrm{c} \Downarrow \sigma_1 \qquad \sigma_1, \mathrm{b} \Downarrow_B \mathrm{true}}{\sigma, \textbf{repeat}\ \mathrm{c}\ \textbf{until}\ \mathrm{b} \Downarrow \sigma_1}\ \textsc{E-RepeatTrue}$$

$$\frac{\sigma, \mathrm{c} \Downarrow \sigma_1 \qquad \sigma_1, \mathrm{b} \Downarrow_B \mathrm{false} \qquad \sigma_1, \textbf{repeat}\ \mathrm{c}\ \textbf{until}\ \mathrm{b} \Downarrow \sigma_2}{\sigma, \textbf{repeat}\ \mathrm{c}\ \textbf{until}\ \mathrm{b} \Downarrow \sigma_2}\ \textsc{E-RepeatFalse}$$

$$\frac{}{\sigma, \mathsf{x} \Downarrow_A \sigma(x)}\ (\text{E-Var}) \qquad \frac{}{\sigma, \mathsf{n} \Downarrow_A \mathsf{n}}\ (\text{E-Const}) \qquad \frac{\sigma, \mathsf{a}_1 \Downarrow_A \mathsf{m} \qquad \mathsf{a}_2 \Downarrow_A \mathsf{n}}{\sigma, \mathsf{a}_1 + \mathsf{a}_2 \Downarrow_A \mathsf{m} + \mathsf{n}}\ (\text{E-Add})$$

$$\frac{}{\sigma, \mathsf{true} \Downarrow_B \mathsf{true}}\ (\text{E-True}) \qquad \frac{}{\sigma, \mathsf{false} \Downarrow_B \mathsf{false}}\ (\text{E-False}) \qquad \frac{\sigma, \mathsf{b}_1 \Downarrow_B \mathsf{t} \qquad \mathsf{b}_2 \Downarrow_B \mathsf{v}}{\sigma, \mathsf{b}_1 \wedge \mathsf{b}_2 \Downarrow_B \mathsf{t} \ \wedge \mathsf{v}}\ (\text{E-And})$$

$$\frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{true}}{\sigma, \neg\mathsf{b} \Downarrow_B \mathsf{false}}\ (\text{E-NotT}) \qquad \frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{false}}{\sigma, \neg\mathsf{b} \Downarrow_B \mathsf{true}}\ (\text{E-NotF})$$

$$\frac{\sigma, \mathsf{a}_1 \Downarrow_A \mathsf{n} \qquad \sigma, \mathsf{a}_2 \Downarrow_A \mathsf{n}}{\sigma, \mathsf{a}_1 = \mathsf{a}_2 \Downarrow_B \mathsf{true}}\ (\text{E-CmpT}) \qquad \frac{\sigma, \mathsf{a}_1 \Downarrow_A \mathsf{m} \qquad \sigma, \mathsf{a}_2 \Downarrow_A \mathsf{n} \qquad m \neq n}{\sigma, \mathsf{a}_1 = \mathsf{a}_2 \Downarrow_B \mathsf{true}}\ (\text{E-CmpF})$$

$$\frac{}{\sigma, \textbf{skip} \Downarrow \sigma}\ (\text{E-Skip}) \qquad \frac{\sigma, \mathsf{a} \Downarrow_A \mathsf{n}}{\sigma, \mathsf{x} := \mathsf{a} \Downarrow \sigma[\mathsf{x} \mapsto \mathsf{n}]}\ (\text{E-Ass})$$

$$\frac{\sigma, \mathsf{c}_1 \Downarrow \sigma_1 \qquad \sigma_1, \mathsf{c}_2 \Downarrow \sigma_2}{\sigma, \mathsf{c}_1; \mathsf{c}_2 \Downarrow \sigma_2}\ (\text{E-Seq}) \qquad \frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{true} \qquad \sigma, \mathsf{c}_1 \Downarrow \sigma_1}{\sigma, \textbf{if}\ \mathsf{b}\ \textbf{then}\ \mathsf{c}_1\ \text{else}\ \mathsf{c}_2 \Downarrow \sigma_1}\ (\text{E-IfTrue})$$

$$\frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{false} \qquad \sigma, \mathsf{c}_2 \Downarrow \sigma_2}{\sigma, \textbf{if}\ \mathsf{b}\ \textbf{then}\ \mathsf{c}_1\ \text{else}\ \mathsf{c}_2 \Downarrow \sigma_2}\ (\text{E-IfFalse}) \qquad \frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{false}}{\sigma, \textbf{while}\ \mathsf{b}\ \textbf{do}\ \mathsf{c} \Downarrow \sigma}\ (\text{E-WhileFalse})$$

$$\frac{\sigma, \mathsf{b} \Downarrow_B \mathsf{true} \qquad \sigma, \mathsf{c} \Downarrow \sigma_1 \qquad \sigma_1, \textbf{while}\ \mathsf{b}\ \textbf{do}\ \mathsf{c} \Downarrow \sigma_2}{\sigma, \textbf{while}\ \mathsf{b}\ \textbf{do}\ \mathsf{c} \Downarrow \sigma_2}\ (\text{E-WhileTrue})$$

Figure 1: Big-step semantics of arithemetic expressions, boolean expressions, and regular IMP.