HW3 – Operational Semantics

CS 476, Fall 2023

1 Instructions

This is a written assignment. You may write your solutions by hand and scan/take a picture of the paper, or in a text editor (Notepad, Word, LaTeX, etc.) and submit a text file or PDF. If you need any help getting your solutions into a suitable format, just let the instructors know. As always, please don't hesitate to ask for help on Piazza (`https://piazza.com/class/ksknvqg6ogb2kc`).

2 Operational Semantics of Expressions and Commands

Here are the operational semantics rules for a simple imperative programming language, using the "hybrid style" of big steps for expressions and small steps for commands.

$$\text{NUM}\ \frac{(n \text{ is a number literal})}{(n, \rho) \Downarrow n} \qquad \text{BOOL}\ \frac{(b \text{ is a boolean literal})}{(b, \rho) \Downarrow b} \qquad \text{VAR}\ \frac{(\rho(x) = v)}{(x, \rho) \Downarrow v}$$

$$\text{OP}\ \frac{(e_1, \rho) \Downarrow v_1 \qquad (e_2, \rho) \Downarrow v_2 \qquad (v_1 \oplus v_2 = v)}{(e_1\ \texttt{op}\ e_2, \rho) \Downarrow v} \text{ for each operator } \texttt{op} \text{ and its meta-level equivalent } \oplus$$

$$\text{ASGN}\ \frac{(e, \rho) \Downarrow v}{(x\ \texttt{:=}\ e, \rho) \to (\texttt{skip}, \rho[x \mapsto v])} \qquad \text{SEQ-STRUCT}\ \frac{(c_1, \rho) \to (c_1', \rho')}{(c_1\ \texttt{;}\ c_2, \rho) \to (c_1'\ \texttt{;}\ c_2, \rho')}$$

$$\text{SEQ-COMP}\ \frac{}{(\texttt{skip;}\ c_2, \rho) \to (c_2, \rho)}$$

3 Problems

1. (6 points total) Consider the following program configuration:

   $$(\texttt{a := b + 2; c := 3 + 4}, \{\texttt{b} = 5\})$$

   (a) (2 points) What is the top-level operation in this configuration's program? Put another way, which rule's conclusion would match the entire configuration?

   (b) (4 points) Write a proof tree for the step taken by the configuration. The bottom of the tree should have the form $(\texttt{a := b + 2; c := 3 + 4}, \{\texttt{b} = 5\}) \to ...$, with the ... filled in according to the rules you apply. You only need to write a proof tree for a single small step.

2. (9 points total) Suppose we wanted to add a new `?` operation to our language of the form x `?` y, where x and y can be any program expression. The program x `?` y is meant to return the value of x unless that value is 0, in which case it returns the value of y instead.

   (a) (2 points) Should x `?` y be an expression or a command? Why?

   (b) (5 points) Write one or more semantic rules for the `?` operation, in the appropriate style (big-step or small-step) based on your answer to the previous question.

   (c) (2 points) Using your rules, write a proof tree showing that if the value of variable `z` is currently 0 and the value of variable `a` is currently 5, then the value of `z ? a` is 5.

   (d) (for graduate students) Now, suppose we want x `?` y to **change** the value of x (which must be a variable) to the value of y when x is 0, instead of returning the value of y. When x is not zero, x `?` y should do nothing at all. Would this change your answer to 2a? How would your rule(s) in 2b change?