CS 4110 – Programming Languages and Logics
Homework #2

Ameya Acharya (apa52) and Quinn Beightol (qeb2)

Exercise 1. Prove the following theorem using the large-step semantics:

Theorem. *If $\sigma(i)$ is even and $\langle \sigma, \textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ i := i + 2\rangle \Downarrow \sigma'$ then $\sigma'(i)$ is also even.*

We prove the theorem by induction, as follows:

Base case: $\sigma(i)$ is even (given). ✓

Inductive step:

Consider the following $\sigma[i \to n]$, where $n = 2x$ for some $x \in \mathbb{Z}$ (since $\sigma(i)$ is even).

Case 1: b is false.

$$\dfrac{\dfrac{false = \sigma(b)}{\langle \sigma, b\rangle \Downarrow \langle \sigma, false\rangle}\ \text{VAR}}{\langle \sigma, \textbf{while}\ b\ \textbf{do}\ i := i + 2\rangle \Downarrow \sigma}\ \text{WHILE-F}$$

In this case, since $\langle \sigma, \textbf{while}\ b\ \textbf{do}\ i := i + 2\rangle \Downarrow \sigma$, $\sigma' = \sigma$.

Therefore, $\sigma'(i) = \sigma(i) = n$, so $\sigma'(i)$ is even because $\sigma(i)$ is even.

Case 2: b is true.

$$\dfrac{\dfrac{true = \sigma(b)}{\langle \sigma, b\rangle \Downarrow \langle \sigma, true\rangle}\ \text{VAR} \quad \dfrac{\dfrac{n = \sigma(i)}{\langle \sigma, i + 2\rangle \Downarrow n + 2}\ \text{VAR}}{\langle \sigma, i := i + 2\rangle \Downarrow \sigma'}\ \text{ASSGN}}{\langle \sigma, \textbf{while}\ b\ \textbf{do}\ i := i + 2\rangle \Downarrow \sigma''}\ \text{WHILE-T}$$

TODO: is this induction? is this the right approach? Figure out.

Exercise 2. Recall that IMP commands are equivalent if they always evaluate to the same result:

$$c_1 \sim c_2 \triangleq \forall \sigma, \sigma' \in \textbf{Store}.\ \langle \sigma, c_1\rangle \Downarrow \sigma' \iff \langle \sigma, c_2\rangle \Downarrow \sigma'.$$

For each of the following pair of IMP commands, either use the large-step operational semantics to prove they are equivalent or give a concrete counter-example showing that they are not equivalent. You may assume that the language has been extended with operators such as $x! = 0$.

(a) $x := a;\ y := a$ and $y := a;\ x := a$,
where a is an arbitrary arithmetic expression

We show that these commands are not equivalent with the following counter-example.

Consider a store σ with $x = 3, y = 4$. Let our expression a be $x + y$.

Evaluating $x := a;\ y := a$:

$\langle \sigma, x := x + y; y := x + y \rangle$
$\to \langle \sigma, x := x + y; y := x + y \rangle$
$\to \langle \sigma, x := 3 + y; y := x + y \rangle$
$\to \langle \sigma, x := 3 + 4; y := x + y \rangle$
$\to \langle \sigma, x := 7; y := x + y \rangle$
$\to \langle \sigma'[x \to 7], y := x + y \rangle$
$\to \langle \sigma', y := 7 + y \rangle$
$\to \langle \sigma', y := 7 + 4 \rangle$
$\to \langle \sigma', y := 11 \rangle$
$\to \sigma''[x \to 7, y \to 11]$

Evaluating $y := a;\ x := a$:

$\langle \sigma, y := x + y; x := x + y \rangle$
$\to \langle \sigma, y := 3 + y; x := x + y \rangle$
$\to \langle \sigma, y := 3 + 4; x := x + y \rangle$
$\to \langle \sigma, y := 7; x := x + y \rangle$
$\to \langle \sigma'[y \to 7], x := x + y \rangle$
$\to \langle \sigma', x := 3 + y \rangle$
$\to \langle \sigma', x := 3 + 7 \rangle$
$\to \langle \sigma', x := 10 \rangle$
$\to \sigma''[x \to 10, y \to 7]$

We see that these two do *not* produce the same results. Therefore, the above IMP commands are not equivalent.

(b) **while** b **do** c and **if** b **then** $(\textbf{while}\ b\ \textbf{do}\ c); c$ **else skip**,
where b is an arbitrary boolean expression and c an arbitrary command.

We show that these commands are not equivalent with the following counter-example.

Consider a store σ with $i = 2$. Let b be $i = 2$ and c be $i := i + 2$.

Evaluating **while** $i = 2$ **do** $i := i + 2$:

$\langle \sigma, \textbf{while}\ i = 2\ \textbf{do}\ i := i + 2 \rangle$
$\to \langle \sigma, \textbf{while}\ 2 = 2\ \textbf{do}\ i := i + 2 \rangle$

$\to \langle \sigma, i := i + 2; \textbf{while } i = 2 \textbf{ do } i := i + 2 \rangle$
$\to \langle \sigma, i := 2 + 2; \textbf{while } i = 2 \textbf{ do } i := i + 2 \rangle$
$\to \langle \sigma, i := 4; \textbf{while } i = 2 \textbf{ do } i := i + 2 \rangle$
$\to \langle \sigma'[i \to 4], \textbf{while } i = 2 \textbf{ do } i := i + 2 \rangle$
$\to \langle \sigma', \textbf{while } 4 = 2 \textbf{ do } i := i + 2 \rangle$
$\to \sigma'[i \to 4]$

Evaluating **if** $i = 2$ **then** (**while** $i = 2$ **do** $i := i + 2$); $i := i + 2$ **else skip**:

$\langle \sigma, \textbf{if } i = 2 \textbf{ then } (\textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \textbf{ else skip} \rangle$
$\to \langle \sigma, \textbf{if } 2 = 2 \textbf{ then } (\textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \textbf{ else skip} \rangle$
$\to \langle \sigma, (\textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma, (i := i + 2; \textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma, (i := 2 + 2; \textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma, (i := 4; \textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma'[i \to 4], (\textbf{while } i = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma'[i \to 4], (\textbf{while } 4 = 2 \textbf{ do } i := i + 2); i := i + 2 \rangle$
$\to \langle \sigma'[i \to 4], i := i + 2 \rangle$
$\to \langle \sigma'[i \to 4], i := 4 + 2 \rangle$
$\to \langle \sigma'[i \to 4], i := 6 \rangle$
$\to \sigma''[i \to 6]$

We see that these two do *not* produce the same results. Therefore, the above IMP commands are not equivalent.

(c) **while** $x \mathrel{!}= 0$ **do** $x := 0$ and $x := 0 * x$

TODO: do this exercise

Exercise 3. Let $\langle \sigma, c \rangle \to \langle \sigma', c' \rangle$ be the small-step operational semantics relation for IMP. Consider the following definition of the multi-step relation:

$$\frac{}{\langle \sigma, c \rangle \to^* \langle \sigma, c \rangle} \text{ R1} \qquad \frac{\langle \sigma, c \rangle \to \langle \sigma', c' \rangle \qquad \langle \sigma', c' \rangle \to^* \langle \sigma'', c'' \rangle}{\langle \sigma, c \rangle \to^* \langle \sigma'', c'' \rangle} \text{ R2}$$

Prove the following theorem, which states that $\to^*$ is transitive.

Theorem. *If* $\langle \sigma, c \rangle \to^* \langle \sigma', c' \rangle$ *and* $\langle \sigma', c' \rangle \to^* \langle \sigma'', c'' \rangle$ *then* $\langle \sigma, c \rangle \to^* \langle \sigma'', c'' \rangle$.

TODO: type this up.

Exercise 4. In this exercise, you will extend the IMP language with exceptions. These exceptions are intended to behave like the analogous constructs found in languages such as Java. We will proceed in several steps.

First, we fix a set of exceptions, which will ranged over by metavariables e, and we extend the syntax of the language with new commands for throwing and handling exceptions:

$$
\begin{array}{rl}
c ::= & \textbf{skip} \\
\mid & x := a \\
\mid & c_1; c_2 \\
\mid & \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\
\mid & \textbf{while } b \textbf{ do } c \\
\mid & \textbf{throw } e \\
\mid & \textbf{try } c_1 \textbf{ with } e \textbf{ do } c_2
\end{array}
$$

Intuitively, evaluating a command either yields a modified store or a pair comprising a modified store and an (uncaught) exception. We let metavariables r range over such results:

$$r ::= \sigma \mid (\sigma, e)$$

Second, we change the type of the large-step evaluation relation so it yields a result instead of a store: $\langle \sigma, c \rangle \Downarrow r$.

Third, we will extend the large-step semantics rules so they handle **throw** and **try** commands. This is your task in this exercise.

Informally, **throw** e should return exception e, and **try** c_1 **with** e **do** c_2 should execute c_1 and return the result it produces, unless the result contains an exception e, in which case it should discard e executes the handler c_2. You will also need to modify many other rules so they have the right type and also propagate exceptions.

TODO: type this up.

Debriefing

(a) How many hours did you spend on this assignment?
(b) Would you rate it as easy, moderate, or difficult?
(c) How deeply do you feel you understand the material it covers (0%100%)?
(d) If you have any other comments, we would like to hear them! Please write them here or send email to `jnfoster@cs.cornell.edu`.