Operational Semantics Exercises  
CS 4610 — Spring 2017

This Review Set asks you to prepare written answers to questions on operational semantics. Each of the questions has a short answer. You may discuss this Review Set with other students and work on the problems together.

1 Definitions and Background

1. Define the following terms and give examples where appropriate.
   (a) Environment:

   (b) Store:

   (c) Call-by-value:

   (d) Call-by-reference:

2. Briefly describe the purpose of operational semantics.

3. What are the constituent parts of the context in a Cool operation semantics rule? Why is each portion of the context necessary?

4. How are side-effects modeled by operational semantics?

5. How is evaluation order enforced by the Cool operational semantics?
2 Operational Semantics

1. Consider these six operational semantics rules:

   \[
   \begin{align*}
   \text{(1)} & \quad \text{so, } E, S \vdash e_1 : \text{Bool}(false), S_1 \\
   \text{so, } E, S \vdash \text{while } e_1 \text{ loop } e_2 \text{ pool } : \text{void}, S_1 \\
   \text{so, } E, S_1 \vdash e_2 : v, S_2 \\
   \text{(2)} & \quad \text{so, } E, S \vdash e_1 : \text{Bool}(true), S_1 \\
   \text{so, } E, S_1 \vdash e_2 : v, S_2 \\
   \text{so, } E, S \vdash \text{while } e_1 \text{ loop } e_2 \text{ pool } : \text{void}, S_3 \\
   \text{so, } E, S_3 \vdash e_2 : v, S_4 \\
   \text{(3)} & \quad \text{so, } E, S \vdash \text{let id } : T \leftarrow e_1 \text{ in } e_2 : v, S_5 \\
   \text{so, } E, S_5 \vdash l_{\text{new}} = \text{newloc}(S_1) \\
   \text{so, } E[l_{\text{new}}/\text{id}], S_1[v/l_{\text{new}}] \vdash e_2 : v, S_6 \\
   \text{(4)} & \quad E(id) = l_{\text{id}} \\
   S(l_{\text{id}}) = v \\
   \text{so, } E, S \vdash id : v, S_7 \\
   \text{(5)} & \quad S_2 = S_1[v/l_{\text{id}}] \\
   \text{so, } E, S \vdash id \leftarrow e : v, S_8 \\
   \text{(6)} & \quad \text{so, } E, S \vdash e_1 : \text{Int}(n_1), S_1 \\
   \text{so, } E, S_1 \vdash e_2 : \text{Int}(n_2), S_2 \\
   \text{so, } E, S \vdash v = \begin{cases} 
   \text{Bool}(true) & \text{if } n_1 < n_2 \\
   \text{Bool}(false) & \text{if } n_1 \geq n_2
   \end{cases} \\
   \text{so, } E, S \vdash e_1 < e_2 : v, S_9
   \end{align*}
   \]

Use these rules to construct a derivation for the following piece of code:

```plaintext
1 let x : Int <- 2 in
2 while 1 < x loop
3 x <- x - 1
4 pool
```

You may assume reasonable axioms, e.g. it is always true that so, E, S \vdash 2 - 1 : Int(1), S. Start your derivation using the let rule (3) as follows:

\[
\begin{align*}
\text{so, } E, S \vdash 2 : \text{Int}(2), S \\
\text{so, } E[l_{\text{new}}/x], S[l_{\text{new}}/\text{Int}(2)] \vdash \text{while } 1 < x \text{ loop } x \leftarrow x - 1 \text{ pool } : \text{void}, S_{\text{final}} \\
\text{so, } E, S \vdash \text{let } x : \text{Int} \leftarrow 2 \text{ in } \text{while } 1 < x \text{ loop } x \leftarrow x - 1 \text{ pool } : \text{void}, S_{\text{final}} \\
\end{align*}
\]

Note that you only need to expand hypotheses that need to be proved (i.e. those containing `\vdash`).
2. Suppose we wanted to add arrays to Cool, using the following syntax:

\[
\begin{align*}
\text{let } &a: T[e_1] \text{ in } e_2 & \quad &\text{Create an array } a \text{ with size } e_1 \text{ of } T\text{'s, usable in } e_2 \\
& a[e_1] \leftarrow e_2 & \quad &\text{Assign } e_2 \text{ to element } e_1 \text{ in } a \\
& a[e] & \quad &\text{Get element } e \text{ of } a
\end{align*}
\]

Write the operational semantics for these three syntactic constructs. You may find it helpful to think of an array of type \( T[n] \) as an object with \( n \) attributes of type \( T \).
3. The operational semantics for Cool’s \texttt{while} expression show that result of evaluating such an expression is always \texttt{void}.

However, we could have used the following alternative semantics:

- If the loop body executes at least once, the result of the \texttt{while} expression is the result from the \textit{last} iteration of the loop body.
- If the loop body never executes (i.e., the condition is false the first time it is evaluated), then the result of the \texttt{while} expression is \texttt{void}.

For example, consider the following expression:

\begin{verbatim}
while (x < 10) loop x ← x+1 pool
\end{verbatim}

The result of this expression would be 10 if, initially, \(x < 10\) or \texttt{void} if \(x \geq 10\).

Write new operational rules for the \texttt{while} construct that formalize these alternative semantics.