Operational Semantics Exercises
CS 364 — Spring 2022

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 snail operational 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 snail operational semantics?

2 Operational Semantics

1. Consider these seven operational semantics rules:

$$(1)\ \frac{so, E, S \vdash e_1 : Bool(false), E_1, S_1}{so, E, S \vdash \text{while } (e_1)\ e_{body} : void, E_1, S_1}$$

$$(4)\ \frac{\begin{array}{c} E(id) = l_{id} \\ S(l_{id}) = v \end{array}}{so, E, S \vdash id : v, E, S}$$

$$(2)\ \frac{\begin{array}{c} so, E, S \vdash e_1 : Bool(true), E_1, S_1 \\ so, E_1, S_1 \vdash e_{body} : v, E_2, S_2 \\ so, E_2, S_2 \vdash \text{while } (e_1)\ e_{body} \text{ pool } : void, E_3, S_3 \end{array}}{so, E, S \vdash \text{while } (e_1)\ e_{body} \text{ pool } : void, E_3, S_3}$$

$$(5)\ \frac{\begin{array}{c} so, E, S \vdash e : v, E_1, S_1 \\ E_1(id) = l_{id} \\ S_2 = S_1[v/l_{id}] \end{array}}{so, E, S \vdash id = e : v, E_1, S_2}$$

$$(3)\ \frac{\begin{array}{c} so, E, S \vdash e_1 : v_1; E_1, S_1 \\ l_{Id} = newloc(S_1) \\ S_2 = S_1\,[v_1/l_{Id}] \\ E_2 = E_1\,[l_{Id}/Id] \end{array}}{so, E, S \vdash let\ Id = e_1 : v_1, E_2, S_2}$$

$$(6)\ \frac{\begin{array}{c} so, E, S \vdash e_1 : v_1, E_1, S_1 \\ so, E_1, S_1 \vdash e_2 : v_2, E_2, S_2 \\ \vdots \\ so, E_{n-1}, S_{n-1} \vdash e_n : v_n, E_n, S_n \end{array}}{so, E, S \vdash \{e_1; e_2; \cdots; e_n;\} : v_n, E, S_n}$$

$$(7)\ \frac{\begin{array}{c} so, E, S \vdash e_1 : Int(n_1), E_1, S_1 \\ so, E_2, S_1 \vdash e_2 : Int(n_2), E_2, S_2 \\ v = \begin{cases} Bool(true) & \text{if } n_1 < n_2 \\ Bool(false) & \text{if } n_1 \geq n_2 \end{cases} \end{array}}{so, E, S \vdash e_1 < e_2 : v, E_2, S_2}$$

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

```
{
  let x = 2;
  while (1 < x) {
    x = x - 1;
  };
}
```

You may assume reasonable axioms, e.g. it is always true that $so, E, S \vdash 2-1 : Int(1), E, S$. Start your derivation using the `let` rule (6) as follows:

$$\frac{\dfrac{\cdots}{so, E, S \vdash (let)x = 2 : Int(2), E_{let}, S_{let}}(3) \qquad \dfrac{\cdots}{so, E_{let}, S_{let} \vdash \text{ while } (1 < x)\,\{x = x - 1;\} : void, E_{final}, S_{final}}(2)}{so, E, S \vdash \{\text{let } x = 2;\ \text{while } (1 < x)\,\{x = x - 1;\};\} : void, E_{final}, S_{final}}(6)$$

Note that you only need to expand hypotheses that need to be proved (i.e. those containing $\vdash$).

2. The operational semantics for snail's `while` expression show that result of evaluating such an expression is always `void`.

   However, we could have used the following alternative semantics:

   - If the loop body executes at least once, the result of the `while` expression is the result from the *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 `while` expression is `void`.

   For example, consider the following expression:

```
while (x < 10) { x = x + 1; }
```

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

   Write new operational rules for the `while` construct that formalize these alternative semantics.