Homework 2: Operational Semantics for WHILE

CS 252: Advanced Programming Languages Student Ji An Lee San José State University

1 MY ASSIGNMENT

Part 1: Rewrite the operational semantic rules for WHILE in LATEX to use big-step operational semantics instead. Submit both your LATEX source and the generated PDF file.

Extend your semantics with features to handle boolean values. **Do not treat these a binary operators.** Specifically, add support for:

- \bullet and
- or
- not

The exact behavior of these new features is up to you, but should seem reasonable to most programmers.

Part 2: Once you have your semantics defined, download WhileInterp.hs and implement the evaluate function, as well as any additional functions you need. Your implementation must be consistent with your operational semantics, *including your extensions for* and, or, *and* not. Also, you may not change any type signatures provided in the file.

Finally, implement the interpreter to match your semantics.

Zip all files together into hw2.zip and submit to Canvas.

```
Expressions
e ::=
                                                             variables/addresses
            x
                                                                            values
            v
                                                                      assignment
            x := e
                                                          sequential expressions
            e; e
                                                               binary operations
            e op e
            \mathtt{if}\ e\ \mathtt{then}\ e\ \mathtt{else}\ e
                                                         conditional expressions
            while (e) e
                                                                while expressions
                                                                           Values
v ::=
            i
                                                                   integer values
                                                                  boolean values
            + | - | * | / | > | >= | < | <=
                                                                Binary operators
op ::=
```

Figure 1: The WHILE language

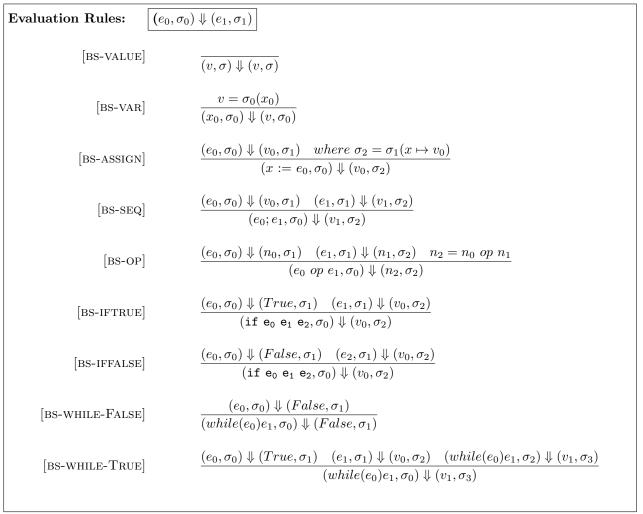


Figure 2: Big-step semantics for WHILE

$$[\text{BS-AND-TRUE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (True,\sigma_1) \quad (e_1,\sigma_1) \Downarrow (b,\sigma_2) \quad | \ b \in bool}{(AND\ e_0\ e_1,\sigma_0) \Downarrow (b,\sigma_2)}$$

$$[\text{BS-AND-FALSE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (False,\sigma_1)}{(AND\ e_0\ e_1,\sigma_0) \Downarrow (False,\sigma_1)}$$

$$[\text{BS-OR-TRUE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (True,\sigma_1)}{(OR\ e_0\ e_1,\sigma_0) \Downarrow (True,\sigma_1)}$$

$$[\text{BS-OR-FALSE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (False,\sigma_1) \quad (e_1,\sigma_1) \Downarrow (b,\sigma_2) \quad | \ b \in bool}{(OR\ e_0\ e_1,\sigma_0) \Downarrow (b,\sigma_2)}$$

$$[\text{BS-NOT-TRUE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (True,\sigma_1)}{(NOT\ e_0,\sigma_0) \Downarrow (False,\sigma_1)}$$

$$[\text{BS-NOT-FALSE}] \qquad \frac{(e_0,\sigma_0) \Downarrow (True,\sigma_1)}{(NOT\ e_0,\sigma_0) \Downarrow (True,\sigma_1)}$$

Figure 3: Big-step semantics for WHILE