### Homework 2: Operational Semantics for WHILE

CS 252: Advanced Programming Languages Prof. Thomas H. Austin San José State University

#### 1 Introduction

For this assignment, you will implement the semantics for a small imperative language, named WHILE.  $(e_1; e_2)$ 

### 2 Small-step semantics

Most of these rules are fairly straightforward, but there are a couple of points to note with the [SS-WHILE] rule.

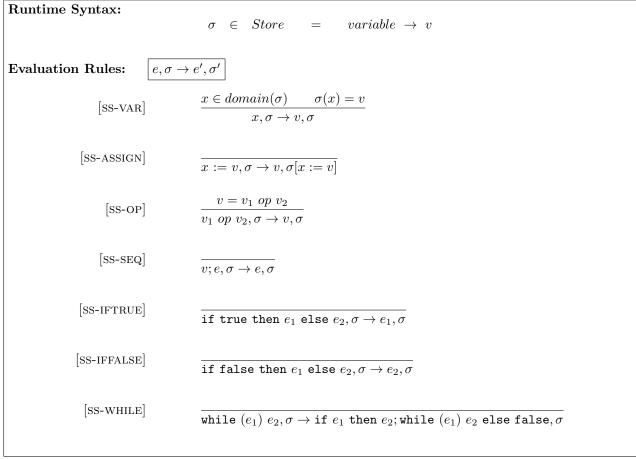


Figure 2: Small-step semantics for WHILE

```
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
                                                                  not expressions
            \mathtt{not}\ e
            and (e) e
                                                                  and expressions
            or (e) e
                                                                    or expressions
v ::=
                                                                            Values
                                                                    integer values
            i
                                                                   boolean values
            + | - | * | / | > | >= | < | <=
                                                                Binary operators
op ::=
```

Figure 1: The WHILE language

# Variable Evaluation Rules: $[\text{SS-VAR}] \qquad \qquad \frac{x \in domain(\sigma) \qquad \sigma(x) = v}{x, \sigma \to v, \sigma}$

Figure 3: Variable Small-Step Semantics Evaluation Order Rules

Set/Assignment Evaluation Rules: 
$$\frac{e,\sigma\to e',\sigma'}{x:=e,\sigma\to x:=e',\sigma'}$$
 [SS-ASSIGNREDUCTION] 
$$\frac{x:=v,\sigma\to v,\sigma[x:=v]}{x:=v,\sigma\to v,\sigma[x:=v]}$$

Figure 4: Set/Assignment Small-Step Semantics Evaluation Order Rules

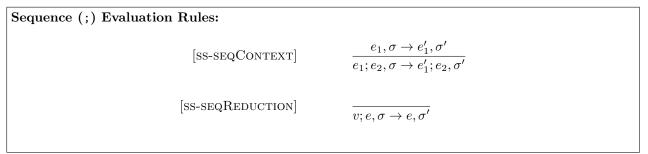


Figure 5: Sequence (;) Evaluation Order Rules

## 

Figure 6: Conditional (if) Small-Step Semantics Evaluation Order Rules

```
while Evaluation Rules:  \frac{}{\text{[SS-WHILE]}} \qquad \frac{}{\text{while } (e_1) \ e_2, \sigma \to \text{if } e_1 \text{ then } e_2; \text{while } (e_1) \ e_2 \text{ else false}, \sigma}
```

Figure 7: while Small-Step Semantics Evaluation Order Rules

### 3 Boolean Expressions Small-Step Semantics

In this section, I add three new expression types to the WHILE language namely: not, and, and or. The evaluation order rules for each are below.

Do I need the parentheses is in the "and" and "or" statements? Is an infix style more typically used?

$$[\text{SS-NOTCONTEXT}] \qquad \qquad \frac{e,\sigma \to e',\sigma'}{\text{not } e,\sigma \to \text{not } e',\sigma'}$$

Not sure if I need this. If I do, then why? Why is this not like the "op" case:

[SS-NOTREDUCTION] 
$$\frac{}{\mathsf{not}\ v, \sigma \to \mathsf{if}\ v \mathsf{\ then\ false\ else\ true}, \sigma}$$

I believe the above rule makes these unnecessary. Would most define as above or like below?

$$[\text{SS-NOTTRUE}] \hspace{1cm} \overline{\text{not true}, \sigma \rightarrow \text{false}, \sigma}$$

$$[\text{SS-NOTFALSE}] \qquad \qquad \overline{\texttt{not false}, \sigma \to \texttt{true}, \sigma}$$

Figure 8: not Small-Step Semantics Evaluation Order Rules

### and Evaluation Rules:

$$[\text{SS-ANDCONTEXT}] \qquad \qquad \frac{e_1, \sigma \to e_1', \sigma'}{\text{and } (e_1) \ e_2, \sigma \to \text{and } (e_1') \ e_2, \sigma'}$$

$$\boxed{ \boxed{ \text{and } (v) \ e, \sigma \rightarrow \text{if} \ v \ \text{then} \ e \ \text{else} \ \text{false}, \sigma} }$$

Using the above, I think I do not need these:

$$[\text{SS-ANDCONTEXT2}] \qquad \qquad \frac{e,\sigma \to e',\sigma'}{\text{and }(v)\ e,\sigma \to \text{and }(v)\ e',\sigma'}$$

[SS-ANDALLTRUE] 
$$\frac{}{\text{and (true) true}, \sigma \rightarrow \text{true}, \sigma}$$

[SS-ANDFALSE1] 
$$\frac{}{\text{and (false) } v, \sigma \rightarrow \text{false}, \sigma}$$

[SS-ANDFALSE2] 
$$\frac{}{\text{and }(v) \text{ false}, \sigma \rightarrow \text{false}, \sigma}$$

Figure 9: and Small-Step Semantics Evaluation Order Rules

or Evaluation Rules: (Is defining "temporary variables" as I did allowed in small step semantics?)

$$[\text{SS-ORREDUCTION}] \qquad \qquad \frac{e_1' = \text{not } e_1 \qquad e_2' = \text{not } e_2 \qquad e_3 = \text{and } (e_1') \ e_2'}{\text{or } (e_1) \ e_2, \sigma \to \text{not } e_3, \sigma}$$

Figure 10: or Evaluation Order Rule