Homework 2: Operational Semantics for the WHILE Language

Zayd Hammoudeh
(zayd.hammoudeh@sjsu.edu)

1 Introduction to the WHILE Language

The "WHILE" language is a basic language that was defined in class. Figure 1 defines the expressions, values, and operators in this language. This notation for expressions (e.g. e), values (v), variables/addresses (x), and store (σ) applies to all sections of this document.

$e ::=$		Expressions
	x	variables/addresses
	v	values
	$x := e$	assignment
	$e; e$	sequential expressions
	$e\ op\ e$	binary operations
	if e then e else e	conditional expressions
	while $(e)\ e$	while expressions
	not e	not expressions
	and $(e)\ e$	and expressions
	or $(e)\ e$	or expressions
$v ::=$		Values
	i	integer values
	b	boolean values
$op ::=$	$+\ \mid\ -\ \mid\ *\ \mid\ /\ \mid\ >\ \mid\ >=\ \mid\ <\ \mid\ <=$	Binary operators
σ		Store

Figure 1: The WHILE language

2 Base WHILE Language

The following figures enumerate the execution order, small-step semantics rules for the WHILE language as defined in class.
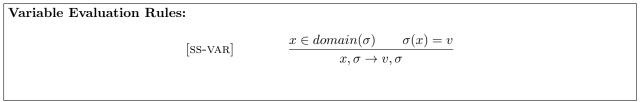
Variable Evaluation Rules:

$$[\text{SS-VAR}] \qquad \frac{x \in domain(\sigma) \qquad \sigma(x) = v}{x, \sigma \rightarrow v, \sigma}$$

Figure 2: Variable Small-Step Semantics Evaluation Order Rules

Set/Assignment Evaluation Rules:

$$[\text{SS-ASSIGNCONTEXT}] \qquad \frac{e, \sigma \to e', \sigma'}{x := e, \sigma \to x := e', \sigma'}$$

$$[\text{SS-ASSIGNREDUCTION}] \qquad \frac{}{x := v, \sigma \to v, \sigma[x := v]}$$

Figure 3: Set/Assignment Small-Step Semantics Evaluation Order Rules

Binary Operator (op) Evaluation Rules:

$$[\text{SS-OPCONTEXT1}] \qquad \frac{e_1, \sigma \to e_1', \sigma'}{e_1 \ op \ e_2, \sigma \to e_1' \ op \ e_2, \sigma'}$$

$$[\text{SS-OPCONTEXT2}] \qquad \frac{e, \sigma \to e', \sigma'}{v \ op \ e, \sigma \to v \ op \ e', \sigma'}$$

Is there a reason you used the infix op notation here instead of the notation from class "$v_3 = \text{apply}(op, v_1, v_2)$"

$$[\text{SS-OPREDUCTION}] \qquad \frac{v_3 = v_1 \ op \ v_2}{v_1 \ op \ v_2, \sigma \to v_3, \sigma}$$

Figure 4: Binary Operator (op) Evaluation Order Rules

Sequence (;) Evaluation Rules:

$$[\text{SS-SEQCONTEXT}] \qquad \frac{e_1, \sigma \to e_1', \sigma'}{e_1; e_2, \sigma \to e_1'; e_2, \sigma'}$$

$$[\text{SS-SEQREDUCTION}] \qquad \frac{}{v; e, \sigma \to e, \sigma'}$$

Figure 5: Sequence (;) Evaluation Order Rules

Conditional Statement (`if`) Evaluation Rules:

$$[\text{SS-IF}\textsc{Context}] \qquad \frac{e_1, \sigma \rightarrow e_1', \sigma'}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3, \sigma \rightarrow \texttt{if } e_1' \texttt{ then } e_2 \texttt{ else } e_3, \sigma'}$$

$$[\text{SS-IF}\textsc{TrueReduction}] \qquad \frac{}{\texttt{if true then } e_1 \texttt{ else } e_2, \sigma \rightarrow e_1, \sigma}$$

$$[\text{SS-IF}\textsc{FalseReduction}] \qquad \frac{}{\texttt{if false then } e_1 \texttt{ else } e_2, \sigma \rightarrow e_2, \sigma}$$

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

---

`while` Evaluation Rules:

$$[\text{SS-WHILE}] \qquad \frac{}{\texttt{while } (e_1)\ e_2, \sigma \rightarrow \texttt{if } e_1 \texttt{ then } e_2 \texttt{;while } (e_1)\ e_2 \texttt{ else false}, \sigma}$$

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?

not Evaluation Rules:

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

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

$$[\text{SS-NOTREDUCTION}] \qquad \frac{}{\texttt{not } v,\sigma \rightarrow \texttt{if } v \texttt{ then false else true},\sigma}$$

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

$$[\text{SS-NOTTRUE}] \qquad \frac{}{\texttt{not true},\sigma \rightarrow \texttt{false},\sigma}$$

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

Figure 8: not Small-Step Semantics Evaluation Order Rules

---

and Evaluation Rules:

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

$$[\text{SS-ANDREDUCTION}] \qquad \frac{}{\texttt{and } (v) \ e,\sigma \rightarrow \texttt{if } v \texttt{ then } e \texttt{ else false},\sigma}$$

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

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

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

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

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

Figure 9: and Small-Step Semantics Evaluation Order Rules

or **Evaluation Rules:** (Is defining "temporary variables" as I did allowed in small step semantics? I assumed it was because of how you handled the "op" case. I also assume that defining these temp variables is required since they enforce the evaluation order (correct me if I am wrong))

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

Figure 10: or Evaluation Order Rule