

# 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 ( $e$ ), values ( $v$ ), variables/addresses ( $x$ ), and store ( $\sigma$ ) applies to all sections of this document.

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

Figure 1: The WHILE language

## 2 Base WHILE Language Small-Step Semantics Rules

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

**Variable Evaluation Rule:**

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

**Figure 2:** Variable Small-Step Semantics Evaluation Order Rule**Set/Assignment Evaluation Rules:**

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

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

**Figure 3:** Set/Assignment Small-Step Semantics Evaluation Order Rules**Binary Operator (op) Evaluation Rules:**

$$[\text{SS-OPCONTEXT1}] \quad \frac{e_1, \sigma \rightarrow e'_1, \sigma'}{e_1 \text{ op } e_2, \sigma \rightarrow e'_1 \text{ op } e_2, \sigma'}$$

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

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

**Figure 4:** Binary Operator (op) Evaluation Order Rules**Sequence (;) Evaluation Rules:**

$$[\text{SS-SEQCONTEXT}] \quad \frac{e_1, \sigma \rightarrow e'_1, \sigma'}{e_1; e_2, \sigma \rightarrow e'_1; e_2, \sigma'}$$

$$[\text{SS-SEQREDUCTION}] \quad \frac{}{v; e, \sigma \rightarrow e, \sigma}$$

**Figure 5:** Sequence (;) Evaluation Order Rules

**Conditional Statement (if) Evaluation Rules:**

[SS-IFCONTEXT]	$\frac{e_1, \sigma \rightarrow e'_1, \sigma'}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3, \sigma \rightarrow \text{if } e'_1 \text{ then } e_2 \text{ else } e_3, \sigma'}$
[SS-IFTRUEREDUCTION]	$\frac{}{\text{if true then } e_1 \text{ else } e_2, \sigma \rightarrow e_1, \sigma}$
[SS-IFFALSEREDUCTION]	$\frac{}{\text{if false then } e_1 \text{ else } e_2, \sigma \rightarrow e_2, \sigma}$

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

**while Evaluation Rule:**

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

**Figure 7:** while Small-Step Semantics Evaluation Order Rule

### 3 Boolean Expressions Small-Step Semantics Rules

In this section, I describe three additional expression types in the updated the WHILE language namely: **not**, **and**, and **or**. The evaluation order rules for each are below.

**not Evaluation Rule:**

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

**Figure 8:** not Small-Step Semantics Evaluation Order Rule

and Evaluation Rules: (In the case of rule “SS-ANDREDUCTION”, “ $e$ ” could return an integer. I do not enforce any typing here while I do in the lower set of rules. Is that ok? Rather than just making it “ $e$ ”, I could make it “AND  $e$  True”)

$$\begin{array}{c}
 \text{[SS-ANDCONTEXT]} \quad \frac{e_1, \sigma \rightarrow e'_1, \sigma'}{\text{and } (e_1) (e_2), \sigma \rightarrow \text{and } (e'_1) (e_2), \sigma'} \\
 \\
 \text{[SS-ANDREDUCTION]} \quad \frac{}{\text{and } (v) (e), \sigma \rightarrow \text{if } v \text{ then } e \text{ else false}, \sigma}
 \end{array}$$

Using the above, I think I do not need these. However, I believe the implementation from an execution perspective of these is slightly different since the above case is short circuit compare (which could affect the store) while the lower case is not. Correct me if I am wrong.

$$\begin{array}{c}
 \text{[SS-ANDCONTEXT2]} \quad \frac{e, \sigma \rightarrow e', \sigma'}{\text{and } (v) (e), \sigma \rightarrow \text{and } (v) (e'), \sigma'} \\
 \\
 \text{[SS-ANDALLTRUE]} \quad \frac{}{\text{and } (\text{true}) (\text{true}), \sigma \rightarrow \text{true}, \sigma} \\
 \\
 \text{[SS-ANDFALSE1]} \quad \frac{}{\text{and } (\text{false}) (v), \sigma \rightarrow \text{false}, \sigma} \\
 \\
 \text{[SS-ANDFALSE2]} \quad \frac{}{\text{and } (v) (\text{false}), \sigma \rightarrow \text{false}, \sigma}
 \end{array}$$

**Figure 9:** and Small-Step Semantics Evaluation Order Rules

or Evaluation Rule:

$$\text{[SS-ORREDUCTION]} \quad \frac{e'_1 = \text{not } e_1 \quad e'_2 = \text{not } e_2 \quad e_3 = \text{and } (e'_1) (e'_2)}{\text{or } (e_1) (e_2), \sigma \rightarrow \text{not } e_3, \sigma}$$

**Figure 10:** or Small-Step Semantics Evaluation Order Rule