

Homework 2: Operational Semantics for WHILE

CS 252: Advanced Programming Languages
Yuri Tatishchev
San José State University

1 Introduction

For this assignment, you will implement the semantics for a small imperative language, named WHILE.

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

Figure 1: The WHILE language

The language for WHILE is given in Figure 1. Unlike the Bool* language we discussed previously, WHILE supports *mutable references*. The state of these references is maintained in a *store*, a mapping of references to values. (“Store” can be thought of as a synonym for heap.) Once we have mutable references, other language constructs become more useful, such as sequencing operations $(e_1; e_2)$.

2 Semantics

Runtime Syntax:

$$\sigma \in Store = variable \rightarrow v$$

Evaluation Rules:

$$e, \sigma \Downarrow e', \sigma'$$

[B-VALUE]

$$\frac{}{v, \sigma \Downarrow v, \sigma}$$

[B-VAR]

$$\frac{x \in domain(\sigma) \quad \sigma(x) = v}{x, \sigma \Downarrow v, \sigma}$$

[B-ASSIGN]

$$\frac{e, \sigma \Downarrow v, \sigma'}{x := e, \sigma \Downarrow v, \sigma' [x := v]}$$

[B-SEQ]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow v_1, \sigma' \\ e_2, \sigma' \Downarrow v_2, \sigma'' \end{array}}{e_1; e_2, \sigma \Downarrow v_2, \sigma''}$$

[B-OP]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow v_1, \sigma' \\ e_2, \sigma' \Downarrow v_2, \sigma'' \\ v = apply(op, v_1, v_2) \end{array}}{e_1 \ op \ e_2, \sigma \Downarrow v, \sigma''}$$

[B-IFTRUE]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow \mathbf{true}, \sigma' \\ e_2, \sigma' \Downarrow v, \sigma'' \end{array}}{\mathbf{if} \ e_1 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3, \sigma \Downarrow v, \sigma''}$$

[B-IFFALSE]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow \mathbf{false}, \sigma' \\ e_3, \sigma' \Downarrow v, \sigma'' \end{array}}{\mathbf{if} \ e_1 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3, \sigma \Downarrow v, \sigma''}$$

[B-WHILETRUE]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow \mathbf{true}, \sigma' \\ e_2, \sigma' \Downarrow v_1, \sigma'' \\ \mathbf{while} \ (e_1) \ e_2, \sigma'' \Downarrow v, \sigma''' \end{array}}{\mathbf{while} \ (e_1) \ e_2, \sigma \Downarrow v, \sigma'''}$$

[B-WHILEFALSE]

$$\frac{e_1, \sigma \Downarrow \mathbf{false}, \sigma'}{\mathbf{while} \ (e_1) \ e_2, \sigma \Downarrow \mathbf{false}, \sigma'}$$

[B-NOT]

$$\frac{e, \sigma \Downarrow b, \sigma'}{\mathbf{not} \ e, \sigma \Downarrow \neg b, \sigma'}$$

[B-BOOLOP]

$$\frac{\begin{array}{c} e_1, \sigma \Downarrow b_1, \sigma' \\ e_2, \sigma' \Downarrow b_2, \sigma'' \\ b = apply(boolop, b_1, b_2) \end{array}}{e_1 \ boolop \ e_2, \sigma \Downarrow b, \sigma''}$$

Figure 2: Big-step semantics for WHILE