Operational Semantics for My While Language

Swapnil Mohan Patil
San José State University

1 Introduction

In this Project, I have implemented the semantics for a small imperative language, named WHILE using Haskell Parsec libraries.

The language for WHILE is given in Figure 2. 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 Small-step semantics

The small-step semantics for WHILE are given in Figure 2. For the sake of brevity, these rules use *evaluation contexts* (C), which specify which *redex* will be evaluated next. The evaluation rules then apply to the "hole" ($\bullet$) in this context.

Note the final value of this expression once the while loop completes. It will *always* be `false` when it completes. We could have created a special value, such as `null`, or we could have made the while loop a statement that returns no value. Both choices, however, would complicate our language needlessly.

$e ::=$		*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
$v ::=$		*Values*
	i	integer values
	b	boolean values
$op ::=$	$+ \mid - \mid * \mid / \mid > \mid >= \mid < \mid <=$	*Binary operators*

Figure 1: The WHILE language

Runtime Syntax:

$$C \in \textit{Context} ::= C; e \mid C\ op\ e \mid v\ op\ C \mid x := C \mid \texttt{if}\ C\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \mid \bullet$$
$$\sigma \in \textit{Store} = \textit{variable} \rightarrow v$$

Evaluation Rules: $\boxed{e, \sigma \rightarrow e', \sigma'}$

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

[SS-ASSIGN]
$$\frac{}{C[x := v], \sigma \rightarrow C[v], \sigma[x := v]}$$

[SS-OP]
$$\frac{v = v_1\ op\ v_2}{C[v_1\ op\ v_2], \sigma \rightarrow C[v], \sigma}$$

[SS-SEQ]
$$\frac{}{C[v; e], \sigma \rightarrow C[e], \sigma}$$

[SS-IFTRUE]
$$\frac{}{C[\texttt{if true then}\ e_1\ \texttt{else}\ e_2], \sigma \rightarrow C[e_1], \sigma}$$

[SS-IFFALSE]
$$\frac{}{C[\texttt{if false then}\ e_1\ \texttt{else}\ e_2], \sigma \rightarrow C[e_2], \sigma}$$

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

Figure 2: Small-step semantics for WHILE