Assignment #2:
Language Design and Implementation

CSCI 5535 / ECEN 5533: Fundamentals of Programming Languages

Spring 2018: Due Friday, February 23, 2018

The tasks in this homework ask you to formalize and prove meta-theoretical properties of an imperative core language IMP based on your experience with E. This homework also asks you to implement an extension of E in OCaml to gain experience translating formalization to implementation.

1 Language Design: IMP

In this section, we will formalize a variant of IMP from Chapter 2 of FSPL based on our experience from Homework Assignment 1. Consider the following syntax chart for IMP:

Typ	τ	::=	`num`	`num`	numbers
			`bool`	`bool`	booleans
Exp	e	::=	$\texttt{addr}[a]$	a	addresses (or “assignables”)
			$\texttt{num}[n]$	n	numeral
			$\texttt{bool}[b]$	b	boolean
			$\texttt{plus}(e_1; e_2)$	$e_1 + e_2$	addition
			$\texttt{times}(e_1; e_2)$	$e_1 * e_2$	multiplication
			$\texttt{eq}(e_1; e_2)$	$e_1 == e_2$	equal
			$\texttt{le}(e_1; e_2)$	$e_1 <= e_2$	less-than-or-equal
			$\texttt{not}(e_1)$	$!e_1$	negation
			$\texttt{and}(e_1; e_2)$	$e_1 \mathbin{\&\&} e_2$	conjunction
			$\texttt{or}(e_1; e_2)$	$e_1 \mathbin{\|\|} e_2$	disjunction
Cmd	c	::=	$\texttt{set}[a](e)$	$a := e$	assignment
			`skip`	`skip`	skip
			$\texttt{seq}(c_1; c_2)$	$c_1; c_2$	sequencing
			$\texttt{if}(e; c_1; c_2)$	$\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	conditional
			$\texttt{while}(e; c_1)$	$\texttt{while}\ e\ \texttt{do}\ c_1$	looping
Addr	a				

Addresses a represent static memory store locations and are drawn from some unbounded set Addr. For simplicity, we fix all memory locations to only store numbers (as in FSPL). A store σ is thus a mapping from addresses to numbers, written as follows:

$$\sigma \quad ::= \quad \cdot \mid \sigma, a \hookrightarrow n$$

Extra Credit. Complete this section where instead memory locations can store any values (i.e., numbers or booleans).

1.1. Formalize the statics for IMP with two judgment forms $e : \tau$ and $c\ \mathsf{ok}$.

1.2. Formalize the dynamics for IMP by the following:

(a) Define values and final states $e\ \mathsf{val}$ and $\langle c, \sigma \rangle\ \mathsf{final}$

(b) Define a big-step operational semantics with the judgment forms $\langle e, \sigma \rangle \Downarrow e'$ and $\langle c, \sigma \rangle \Downarrow \sigma'$.

(c) Define a small-step operational semantics with the judgment forms $\langle e, \sigma \rangle \longmapsto \langle e', \sigma' \rangle$ and $\langle c, \sigma \rangle \longmapsto \langle c', \sigma' \rangle$.

(d) State canonical forms. Then, state and prove progress and preservation.

2 Language Implementation: T with Products and Sums

3 Final Project Preparation