Homework 4 (Term Project)
COSE212, Fall 2015

Hakjoo Oh

Due: 11/14, 24:00

In this project, we design and implement an imperative language, called B. The language is a small subset of C (this is why it is named so). The syntax and semantics are defined in Sections 1 and 2, respectively. Your task is to understand the language and implement an interpreter for it.

1 Syntax

Expression e	$\rightarrow$	$\texttt{unit}$	unit
	$\mid$	$x \;\texttt{:=}\; e$	assignment
	$\mid$	$e \;\texttt{;}\; e$	sequence
	$\mid$	$\texttt{if}\; e\; \texttt{then}\; e\; \texttt{else}\; e$	branch
	$\mid$	$\texttt{while}\; e\; \texttt{do}\; e$	while loop
	$\mid$	$\texttt{read}\; x$	input
	$\mid$	$\texttt{write}\; e$	output
	$\mid$	$\texttt{let}\; x \;\texttt{:=}\; e\; \texttt{in}\; e$	variable binding
	$\mid$	$\texttt{let proc}\; f\texttt{(}x_1\texttt{,}x_2\texttt{,}\cdots\texttt{,}x_n\texttt{)} \;\texttt{=}\; e\; \texttt{in}\; e$	procedure binding
	$\mid$	$f\texttt{(}e_1\texttt{,}e_2\texttt{,}\cdots\texttt{,}e_n\texttt{)}$	call by value
	$\mid$	$f\texttt{<}x_1\texttt{,}x_2\texttt{,}\cdots\texttt{,}x_n\texttt{>}$	call by reference
	$\mid$	n	integer
	$\mid$	$\texttt{true} \mid \texttt{false}$	boolean
	$\mid$	$\texttt{\{\}} \mid \texttt{\{}x_1\texttt{:=}e_1\texttt{,}x_2\texttt{:=}e_2\texttt{,}\cdots\texttt{,}x_n\texttt{:=}e_n\texttt{\}}$	record (i.e., struct)
	$\mid$	$e\texttt{.}x$	record lookup
	$\mid$	$e\texttt{.}x \;\texttt{:=}\; e$	record assignment
	$\mid$	x	identifier
	$\mid$	$e \;\texttt{+}\; e \mid e \;\texttt{-}\; e \mid e \;\texttt{*}\; e \mid e \;\texttt{/}\; e$	arithmetic operation
	$\mid$	$e \;\texttt{<}\; e \mid e \;\texttt{=}\; e \mid \texttt{not}\; e$	boolean operation

A program is an expression. Expressions include unit, assignments, sequences, conditional expressions (branch), while loops, read, write, let expressions, let expressions for procedure binding, procedure calls (by either call-by-value or call-by-reference), integers, boolean constants, records (i.e., structs), record lookup, record assignment, identifier, arithmetic expressions, and boolean expressions. Note that procedures may have multiple arguments.

2 Semantics

Domain

$$
\begin{array}{rcrcl}
x, y & \in & \mathit{Id} & & \text{identifier (variable)} \\
l & \in & \mathit{Addr} & & \text{address (memory location)} \\
n & \in & \mathbb{Z} & & \text{integer} \\
b & \in & \mathbb{B} & = & \{\mathit{true}, \mathit{false}\} \\
r & \in & \mathit{Record} & = & \mathit{Id} \to \mathit{Addr} \\
v & \in & \mathit{Val} & = & \mathbb{Z} + \mathbb{B} + \{\cdot\} + \mathit{Record} \\
\sigma & \in & \mathit{Env} & = & \mathit{Id} \to \mathit{Addr} + \mathit{Procedure} \\
M & \in & \mathit{Mem} & = & \mathit{Addr} \to \mathit{Val} \\
 & & \mathit{Procedure} & = & (\mathit{Id} \times \mathit{Id} \times \cdots) \times \mathit{Expression} \times \mathit{Env}
\end{array}
$$

A record (i.e., struct) is defined as a (finite) function from identifiers to memory addresses. A value is either an integer, boolean value, unit value $(\cdot)$, or a record. An environment maps identifiers to memory addresses or procedure values. A memory is a finite function from addresses to values. Note that we design B in a way that procedures are not stored in memory, which means that procedures are not first-class objects in B.

Semantics Rules

$$\text{TRUE}\ \frac{}{\sigma, M \vdash \texttt{true} \Rightarrow \mathit{true}, M} \qquad \text{FALSE}\ \frac{}{\sigma, M \vdash \texttt{false} \Rightarrow \mathit{false}, M}$$

$$\text{NUM}\ \frac{}{\sigma, M \vdash \texttt{n} \Rightarrow n, M} \qquad \text{UNIT}\ \frac{}{\sigma, M \vdash \texttt{unit} \Rightarrow \cdot, M}$$

$$\text{VAR}\ \frac{}{\sigma, M \vdash x \Rightarrow M(\sigma(x)), M} \qquad \text{RECF}\ \frac{}{\sigma, M \vdash \texttt{\{\}} \Rightarrow \cdot, M}$$

$$\text{RECT}\ \frac{\begin{array}{c}\sigma, M \vdash e_1 \Rightarrow v_1, M_1 \\ \sigma, M_1 \vdash e_2 \Rightarrow v_2, M_2 \\ \vdots \\ \sigma, M_{n-1} \vdash e_n \Rightarrow v_n, M_n\end{array}}{\begin{array}{c}\sigma, M \vdash \texttt{\{}x_1 \texttt{ := } e_1\texttt{,} \cdots \texttt{,} x_n \texttt{ := } e_n\texttt{\}} \Rightarrow \\ \{x_1 \mapsto l_1, \cdots, x_n \mapsto l_n\}, M_n\{l_1 \mapsto v_1, \cdots, l_n \mapsto v_n\}\end{array}}\ l_i \notin \mathit{Dom}(M_n)$$

$$\text{ADD}\ \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 \texttt{ + } e_2 \Rightarrow n_1 + n_2, M''}$$

$$\text{SUB}\ \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 \texttt{ - } e_2 \Rightarrow n_1 - n_2, M''}$$

$$\text{MUL}\ \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 \texttt{ * } e_2 \Rightarrow n_1 * n_2, M''}$$

$$\text{DIV}\ \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1 \texttt{ / } e_2 \Rightarrow n_1 / n_2, M''}$$

$$\text{EQUALT}\ \frac{\sigma, M \vdash e_1 \Rightarrow v_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow v_2, M''}{\sigma, M \vdash e_1\ \texttt{=}\ e_2 \Rightarrow \texttt{true}, M''}\ \begin{array}{l} v_1 = v_2 = n \\ \lor\ v_1 = v_2 = b \\ \lor\ v_1 = v_2 = \cdot \end{array}$$

$$\text{EQUALF}\ \frac{\sigma, M \vdash e_1 \Rightarrow v_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow v_2, M''}{\sigma, M \vdash e_1\ \texttt{=}\ e_2 \Rightarrow \texttt{false}, M''}\ \text{otherwise}$$

$$\text{LESS}\ \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow n_2, M''}{\sigma, M \vdash e_1\ \texttt{<}\ e_2 \Rightarrow n_1 < n_2, M''}$$

$$\text{NOT}\ \frac{\sigma, M \vdash e \Rightarrow b, M'}{\sigma, M \vdash \texttt{not}\ e \Rightarrow \mathit{not}\ b, M'}$$

$$\text{ASSIGN}\ \frac{\sigma, M \vdash e \Rightarrow v, M'}{\sigma, M \vdash x\ \texttt{:=}\ e \Rightarrow v, M'\{\sigma(x) \mapsto v\}}$$

$$\text{RECASSIGN}\ \frac{\sigma, M \vdash e_1 \Rightarrow r, M_1 \qquad \sigma, M_1 \vdash e_2 \Rightarrow v, M_2}{\sigma, M \vdash e_1\,\texttt{.}x\ \texttt{:=}\ e_2 \Rightarrow v, M_2\{r(x) \mapsto v\}}$$

$$\text{RECLOOKUP}\ \frac{\sigma, M \vdash e \Rightarrow r, M'}{\sigma, M \vdash e\,\texttt{.}x \Rightarrow M'(r(x)), M'}$$

$$\text{SEQ}\ \frac{\sigma, M \vdash e_1 \Rightarrow v_1, M' \qquad \sigma, M' \vdash e_2 \Rightarrow v_2, M''}{\sigma, M \vdash e_1\ \texttt{;}\ e_2 \Rightarrow v_2, M''}$$

$$\text{IFT}\ \frac{\sigma, M \vdash e \Rightarrow \mathit{true}, M' \qquad \sigma, M' \vdash e_1 \Rightarrow v, M''}{\sigma, M \vdash \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Rightarrow v, M''}$$

$$\text{IFF}\ \frac{\sigma, M \vdash e \Rightarrow \mathit{false}, M' \qquad \sigma, M' \vdash e_2 \Rightarrow v, M''}{\sigma, M \vdash \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Rightarrow v, M''}$$

$$\text{WHILEF}\ \frac{\sigma, M \vdash e_1 \Rightarrow \mathit{false}, M'}{\sigma, M \vdash \texttt{while}\ e_1\ \texttt{do}\ e_2 \Rightarrow \cdot, M'}$$

$$\text{WHILET}\ \frac{\begin{array}{c}\sigma, M \vdash e_1 \Rightarrow \mathit{true}, M' \\ \sigma, M' \vdash e_2 \Rightarrow v_1, M_1 \qquad \sigma, M_1 \vdash \texttt{while}\ e_1\ \texttt{do}\ e_2 \Rightarrow v_2, M_2\end{array}}{\sigma, M \vdash \texttt{while}\ e_1\ \texttt{do}\ e_2 \Rightarrow v_2, M_2}$$

$$\text{LETV}\ \frac{\sigma, M \vdash e_1 \Rightarrow v, M' \qquad \sigma\{x \mapsto l\}, M'\{l \mapsto v\} \vdash e_2 \Rightarrow v', M''}{\sigma, M \vdash \texttt{let}\ x\ \texttt{:=}\ e_1\ \texttt{in}\ e_2 \Rightarrow v', M''}\ l \notin Dom\, M'$$

$$\text{LETF}\ \frac{\sigma\{f \mapsto \langle (x_1, \cdots, x_n), e_1, \sigma \rangle\}, M \vdash e_2 \Rightarrow v, M'}{\sigma, M \vdash \texttt{let proc}\ f\texttt{(}x_1\texttt{,}\cdots\texttt{,}x_n\texttt{)}\ \texttt{=}\ e_1\ \texttt{in}\ e_2 \Rightarrow v, M'}$$

$$\text{CALLV}\ \frac{\begin{array}{c}\sigma, M \vdash e_1 \Rightarrow v_1, M_1 \qquad \sigma, M_1 \vdash e_2 \Rightarrow v_2, M_2 \\ \vdots \\ \sigma, M_{n-1} \vdash e_n \Rightarrow v_n, M_n \\ \sigma'\{x_1 \mapsto l_1\} \cdots \{x_n \mapsto l_n\}\{f \mapsto \langle (x_1, \cdots, x_n), e', \sigma' \rangle\}, \\ M_n\{l_1 \mapsto v_1\} \cdots \{l_n \mapsto v_n\} \vdash e' \Rightarrow v', M'\end{array}}{\sigma, M \vdash f\texttt{(}e_1\texttt{,}\cdots\texttt{,}e_n\texttt{)} \Rightarrow v', M'}\ \begin{array}{l}\sigma(f) = \langle (x_1, \cdots, x_n), e', \sigma' \rangle \\ l_i \notin Dom\, M_n\end{array}$$

$$\text{CALLR}\ \frac{\begin{array}{c}\sigma'\{x_1 \mapsto \sigma(y_1)\} \cdots \{x_n \mapsto \sigma(y_n)\}\{f \mapsto \langle (x_1, \cdots, x_n), e, \sigma' \rangle\}, \\ M \vdash e \Rightarrow v, M'\end{array}}{\sigma, M \vdash f\texttt{<}y_1\texttt{,}\cdots\texttt{,}y_n\texttt{>} \Rightarrow v, M'}\ \sigma(f) = \langle (x_1, \cdots, x_n), e, \sigma' \rangle$$

$$\text{READ}\ \frac{}{\sigma, M \vdash \texttt{read}\ x \Rightarrow n, M\{\sigma(x) \mapsto n\}}$$

$$\text{WRITE}\ \frac{\sigma, M \vdash e \Rightarrow n, M'}{\sigma, M \vdash \texttt{write}\ e \Rightarrow n, M'}$$

3 Instruction

1. Clone the project framework from Github in your Ubuntu system:
   `git clone https://github.com/hakjoooh/ProgrammingLanguagesProject2015.git`
   The framework contains a skeleton implementation, including a parser.

2. Your job is to complete the interpreter implementation in `b.ml`. Specifically, implement the function

   ```
   eval:  memory -> env -> exp -> (value * memory)
   ```

   in module `B` (in `b.ml`). Before implementing it, carefully read `readme.txt`.

3. Submit the single file `b.ml` via Blackboard.

Enjoy!