Homework 3
COSE212, Fall 2017

Hakjoo Oh

Due: 11/14, 24:00

Problem 1 Consider the following programming language, called minML, that features (recursive) procedures and explicit references.

Syntax The syntax of minML is defined as follows:

$$
\begin{array}{rcl}
P & \to & E \\
E & \to & n \\
& \mid & x \\
& \mid & E + E \mid E - E \mid E * E \mid E / E \\
& \mid & \texttt{iszero}\ E \\
& \mid & \texttt{read} \\
& \mid & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
& \mid & \texttt{let}\ x = E\ \texttt{in}\ E \\
& \mid & \texttt{letrec}\ f(x) = E\ \texttt{in}\ E \\
& \mid & \texttt{proc}\ x\ E \\
& \mid & E\ E \\
& \mid & \texttt{ref}\ E \\
& \mid & !\ E \\
& \mid & E := E \\
& \mid & E; E \\
& \mid & \texttt{begin}\ E\ \texttt{end}
\end{array}
$$

Semantics The semantics is defined with the domain:

$$
\begin{array}{rcl}
Val & = & \mathbb{Z} + Bool + Procedure + RecProcedure + Loc \\
Procedure & = & Var \times E \times Env \\
RecProcedure & = & Var \times Var \times E \times Env \\
\rho \in Env & = & Var \to Val \\
\sigma \in Mem & = & Loc \to Val
\end{array}
$$

and the evaluation rules:

$$\frac{}{\rho, \sigma \vdash n \Rightarrow n, \sigma} \qquad \frac{}{\rho, \sigma \vdash x \Rightarrow \rho(x), \sigma}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow n_1, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow n_2, \sigma_2}{\rho, \sigma_0 \vdash E_1 \oplus E_2 \Rightarrow n_1 \oplus n_2, \sigma_2} \ \oplus \in \{+, *, -, /\}$$

$$\frac{\rho, \sigma_0 \vdash E \Rightarrow 0, \sigma_1}{\rho, \sigma_0 \vdash \texttt{iszero}\ E \Rightarrow \mathit{true}, \sigma_1} \qquad \frac{\rho, \sigma_0 \vdash E \Rightarrow n, \sigma_1}{\rho, \sigma_0 \vdash \texttt{iszero}\ E \Rightarrow \mathit{false}, \sigma_1} \ n \neq 0$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow \mathit{true}, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v, \sigma_2}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow \mathit{false}, \sigma_1 \qquad \rho, \sigma_1 \vdash E_3 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v, \sigma_2}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \qquad [x \mapsto v_1]\rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \texttt{let}\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v, \sigma_2}$$

$$\frac{[f \mapsto (f, x, E_1, \rho)]\rho, \sigma_0 \vdash E_2 \Rightarrow v, \sigma_1}{\rho, \sigma_0 \vdash \texttt{letrec}\ f(x) = E_1\ \texttt{in}\ E_2 \Rightarrow v, \sigma_1}$$

$$\frac{}{\rho, \sigma \vdash \texttt{proc}\ x\ E \Rightarrow (x, E, \rho), \sigma}$$

$$\frac{\begin{array}{c}\rho, \sigma_0 \vdash E_1 \Rightarrow (x, E, \rho'), \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2 \\ [x \mapsto v]\rho', \sigma_2 \vdash E \Rightarrow v', \sigma_3\end{array}}{\rho, \sigma_0 \vdash E_1\ E_2 \Rightarrow v', \sigma_3}$$

$$\frac{\begin{array}{c}\rho, \sigma_0 \vdash E_1 \Rightarrow (f, x, E, \rho'), \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2 \\ [x \mapsto v, f \mapsto (f, x, E, \rho')]\rho', \sigma_2 \vdash E \Rightarrow v', \sigma_3\end{array}}{\rho, \sigma_0 \vdash E_1\ E_2 \Rightarrow v', \sigma_3}$$

$$\frac{\rho, \sigma_0 \vdash E \Rightarrow v, \sigma_1}{\rho, \sigma_0 \vdash \texttt{ref}\ E \Rightarrow l, [l \mapsto v]\sigma_1} \ l \notin \mathrm{Dom}(\sigma_1)$$

$$\frac{\rho, \sigma_0 \vdash E \Rightarrow l, \sigma_1}{\rho, \sigma_0 \vdash !\ E \Rightarrow \sigma_1(l), \sigma_1}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow l, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash E_1 := E_2 \Rightarrow v, [l \mapsto v]\sigma_2}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v_2, \sigma_2}{\rho, \sigma_0 \vdash E_1; E_2 \Rightarrow v_2, \sigma_2}$$

$$\frac{\rho, \sigma_0 \vdash E \Rightarrow v, \sigma_1}{\rho, \sigma_0 \vdash \texttt{begin}\ E\ \texttt{end} \Rightarrow v, \sigma_1}$$

Implement an interpreter of minML. Raise an exception `UndefinedSemantics` whenever the semantics is undefined. Skeleton code will be provided.

Problem 2 In class, we have focused on designing an expression-oriented, functional language. Designing a statement-oriented language like C follows a similar path. In this problem, let us design and implement a small imperative language, called minC.

Syntax The syntax of minC is defined as follows:

$$
\begin{array}{rcl}
A & \to & n \mid x \mid A_1 + A_2 \mid A_1 \star A_2 \mid A_1 - A_2 \\
B & \to & \texttt{true} \mid \texttt{false} \mid A_1 = A_2 \mid A_1 \leq A_2 \mid \neg B \mid B_1 \wedge B_2 \\
S & \to & x := A \\
 & \mid & \texttt{skip} \\
 & \mid & S_1; S_2 \\
 & \mid & \texttt{if}\ B\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \\
 & \mid & \texttt{while}\ B\ \texttt{do}\ S \\
 & \mid & \texttt{begin var}\ x := A;\ S\ \texttt{end} \\
 & \mid & \texttt{read}\ x \\
 & \mid & \texttt{print}\ A
\end{array}
$$

The language has three syntactic categories: A (arithmetic expressions), B (boolean expressions), and S (statements). Arithmetic expressions include integers (n), variables (x), addition ($A_1 + A_2$), multiplication ($A_1 \star A_2$), and subtraction ($A_1 - A_2$). Boolean expressions include boolean constants (true, false), comparison ($A_1 = A_2, A_1 \leq A_2$), negation ($\neg B$), and conjunction ($B_1 \wedge B_2$). Statements consist of assignment ($x := A$), skip (skip), sequence ($S_1; S_2$), conditional (if B then S_1 else S_2), loop (while B do S), block (begin var $x := A$; S end), read (read x), and print (print A) statements. Note that the language supports local blocks and every variable must be declared before its use: e.g.,

```
begin var y:=1;
  begin var x:=1;
    begin var x:=2;
      y:=x+1;          // x is 2, y is 3
    end;
    y:=y+x;            // x is 1, (old) y is 3
    print y            // 4 is printed
  end
end
```

Semantics The semantics of minC is similar to that of C, where variables refer to references (i.e. implicit references). Thus, we define the environment and memory as follows:

$$
\begin{array}{rcl}
\rho \in \mathit{Env} & = & \mathit{Var} \to \mathit{Loc} \\
s \in \mathit{Mem} & = & \mathit{Loc} \to \mathit{Value} \\
n \in \mathit{Value} & = & \mathbb{Z}
\end{array}
$$

Given environment (ρ) and memory state (s), arithmetic (A) and boolean (B) expressions compute integers and booleans, represented by $\mathcal{A}[\![A]\!](\rho)(s)$ and

$\mathcal{B}[\![B]\!](\rho)(s)$, respectively. Evaluation functions $\mathcal{A}[\![A]\!]$ and $\mathcal{B}[\![B]\!]$ are inductively defined:

$$
\begin{array}{rcl}
\mathcal{A}[\![A]\!] & : & Env \to Mem \to \mathbb{Z} \\
\mathcal{A}[\![n]\!](\rho)(s) & = & n \\
\mathcal{A}[\![x]\!](\rho)(s) & = & s(\rho(x)) \\
\mathcal{A}[\![A_1 + A_2]\!](\rho)(s) & = & \mathcal{A}[\![A_1]\!](\rho)(s) + \mathcal{A}[\![A_2]\!](\rho)(s) \\
\mathcal{A}[\![A_1 \star A_2]\!](\rho)(s) & = & \mathcal{A}[\![A_1]\!](\rho)(s) \times \mathcal{A}[\![A_2]\!](\rho)(s) \\
\mathcal{A}[\![A_1 - A_2]\!](\rho)(s) & = & \mathcal{A}[\![A_1]\!](\rho)(s) - \mathcal{A}[\![A_2]\!](\rho)(s)
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{B}[\![B]\!] & : & Env \to Mem \to Bool \\
\mathcal{B}[\![\texttt{true}]\!](\rho)(s) & = & true \\
\mathcal{B}[\![\texttt{false}]\!](\rho)(s) & = & false \\
\mathcal{B}[\![A_1 = A_2]\!](\rho)(s) & = & \mathcal{A}[\![A_1]\!](\rho)(s) = \mathcal{A}[\![A_2]\!](\rho)(s) \\
\mathcal{B}[\![A_1 \leq A_2]\!](\rho)(s) & = & \mathcal{A}[\![A_1]\!](\rho)(s) \leq \mathcal{A}[\![A_2]\!](\rho)(s) \\
\mathcal{B}[\![\neg B]\!](\rho)(s) & = & \mathcal{B}[\![B]\!](\rho)(s) = false \\
\mathcal{B}[\![B_1 \wedge B_2]\!](\rho)(s) & = & \mathcal{B}[\![B_1]\!](\rho)(s) \wedge \mathcal{B}[\![B_2]\!](\rho)(s)
\end{array}
$$

With $\mathcal{A}[\![A]\!]$ and $\mathcal{B}[\![B]\!]$, the semantics rules for statements are defined as follows:

$$\frac{}{\rho, s \vdash x := A \Rightarrow \boxed{\quad ? \quad}}$$

$$\frac{}{\rho, s \vdash \texttt{skip} \Rightarrow s}$$

$$\frac{\rho, s \vdash S_1 \Rightarrow s' \quad \rho, s' \vdash S_2 \Rightarrow s''}{\rho, s \vdash S_1; S_2 \Rightarrow s''}$$

$$\frac{\rho, s \vdash S_1 \Rightarrow s'}{\rho, s \vdash \texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \Rightarrow s'} \ \mathcal{B}[\![B]\!](\rho)(s) = true$$

$$\frac{\rho, s \vdash S_2 \Rightarrow s'}{\rho, s \vdash \texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \Rightarrow s'} \ \mathcal{B}[\![B]\!](\rho)(s) = false$$

$$\frac{}{\rho, s \vdash \texttt{while } B \texttt{ do } S \Rightarrow s} \ \mathcal{B}[\![B]\!](\rho)(s) = false$$

$$\frac{\boxed{\quad ? \quad}}{\rho, s \vdash \texttt{while } B \texttt{ do } S \Rightarrow s''} \ \mathcal{B}[\![B]\!](\rho)(s) = true$$

$$\frac{\boxed{\quad ? \quad}}{\rho, s \vdash \texttt{begin var } x := A;\ S \texttt{ end} \Rightarrow s'} \ l \notin \mathrm{Dom}(s)$$

$$\frac{}{\rho, s \vdash \texttt{read } x \Rightarrow [\rho(x) \mapsto n]s} \qquad \frac{}{\rho, s \vdash \texttt{print } A \Rightarrow s}$$

Complete the definition and implement an interpreter. Skeleton code will be provided.