CS4450: Simple While Language

Bill Harrison

November 7, 2018

Announcements

- Today: While language static semantics + transition semantics

State Machines

Transition Semantics:

- Define the meaning of a language with transition rules. Execute input program p in state m_0:
- $(p, m_0) \rightarrow (p_1, m_1) \rightarrow \ldots \rightarrow (p_n, m_n) \rightarrow \ldots$

The Simple Imperative Language with Loops

(Abstract Syntax of While)

$$
\begin{array}{rcl}
I & \in & \mathit{Identifier} \\
N & \in & \mathit{Numeral} \\
B & ::= & \mathbf{true} \mid \mathbf{false} \mid B\ \mathbf{and}\ B \mid B\ \mathbf{or}\ B \mid \mathbf{not}\ B \mid E < E \mid E = E \\
E & ::= & N \mid I \mid E + E \mid E * E \mid E - E \mid -E \\
C & ::= & \mathbf{skip} \mid C\,;C \mid I{:=}E \mid \\
 & & \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C\ \mathbf{fi} \mid \mathbf{while}\ B\ \mathbf{do}\ C\ \mathbf{od}
\end{array}
$$

While Abstract Syntax in Haskell

```
data E = N Number
       | I Ident
       | Add E E | Mult E E | Subt E E | Inv E

data B = T | F | Conj B B | Disj B B
       | Negt B | LTC E E | EQL E E

data C = Nop
       | Seq C C
       | IfElse B C C | While B C
```

The Memory

(Memory maps I to Z)

$$\begin{aligned} \mathit{lookup}\ m\ i &= \langle \text{current value of i} \rangle \\ m[i \mapsto v] &= \langle \text{new memory s.t. i is bound to v} \rangle \end{aligned}$$

$$\mathit{lookup}\ m[i \mapsto v]\ j = \begin{cases} v & i \text{ is } j \\ \mathit{lookup}\ m\ j & \text{otherwise} \end{cases}$$

While Semantics in Haskell

```
type Ident  = String
type Number = Int
type Memory = [(Ident, Number)]

--
-- memory look-up
--
lkup :: Memory -> Ident -> Number
lkup ((x, n):ms) x' = if x == x' then n else lkup ms x'
lkup [ ] _          = error (x ++ " is unbound!")
```

E and B semantics

$$
\begin{aligned}
ev_E\ n\ m &= n \\
ev_E\ i\ m &= lookup\ i\ m \\
ev_E\ {-e}\ m &= -\ (ev_E\ e\ m) \\
&\ldots \\
ev_B\ \mathbf{true}\ m &= true \\
ev_B\ (e_1 = e_2)\ m &= (ev_E\ e_1\ m\ ==\ ev_E\ e_2\ m) \\
&\ldots
\end{aligned}
$$

E and B semantics

$$
\begin{array}{lll}
ev_E\ n\ m & = & n \\
ev_E\ i\ m & = & lookup\ i\ m \\
ev_E\ {-e}\ m & = & -\ (ev_E\ e\ m) \\
\quad \ldots & & \\
ev_B\ \textbf{true}\ m & = & true \\
ev_B\ (e_1 = e_2)\ m & = & (ev_E\ e_1\ m\ ==\ ev_E\ e_2\ m) \\
\quad \ldots & &
\end{array}
$$

Question: what are the types of ev_E and ev_B?

Transition Semantics of While

$$(i := e, m) \rightarrow m[i \mapsto ev_E\ e\ m] \qquad (\textbf{skip}, m) \rightarrow m$$

$$\frac{(c_1, m) \rightarrow (c_1', m')}{(c_1; c_2, m) \rightarrow (c_1'; c_2, m')} \qquad \frac{(c_1, m) \rightarrow m'}{(c_1; c_2, m) \rightarrow (c_2, m')}$$

$$\frac{ev_B\ b\ m = true}{(\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \textbf{fi}, m) \rightarrow (c_1, m)}$$

$$\frac{ev_B\ b\ m = false}{(\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \textbf{fi}, m) \rightarrow (c_2, m)}$$

Transition Semantics of While (cont'd)

$$\frac{ev_B\ b\ m = true}{(\textbf{while}\ b\ \textbf{do}\ c\ \textbf{od}, m) \rightarrow (c; \textbf{while}\ b\ \textbf{do}\ c\ \textbf{od}, m)}$$

$$\frac{ev_B\ b\ m = false}{(\textbf{while}\ b\ \textbf{do}\ c\ \textbf{od}, m) \rightarrow m}$$

Exercise

Formulate the transition semantics for While in Haskell.
Hint:

1. Define each of E, B, and C as **data** declarations;
2. Define ev_E and ev_B as Haskell functions;
3. Define the Memory data type next;
4. Finally, define the transitions for C as a function of type: $(C, Memory) \rightarrow Memory$.