p5q11
PMS

Semantics of Programming Languages 2003

Q1 Solution and marking guide.

This requires understanding of the semantics of the simple imperative language (lectures 1 and 2) and the semantics of functions (lectures 6,7), with the various possible design options.

(a) Here e ranges over expressions – abstract syntax trees[1] of the grammar, up to alpha conversion[1], – and s ranges over stores, which are finite partial maps from a set of locations to the (ML-representable subrange of the[1]) integers[1].

$$\begin{array}{ll}
\text{(deref')} & \langle !\ell, s\rangle \longrightarrow \langle n, s\rangle \quad \text{if } \ell \in \text{dom}(s) \text{ and } s(\ell) = n \\
\text{(deref'')} & \langle !\ell, s\rangle \longrightarrow \langle 0, s + \{\ell \mapsto 0\}\rangle \quad \text{if } \ell \notin \text{dom}(s) \\
\text{(assign1')} & \langle \ell := n, s\rangle \longrightarrow \langle \textbf{skip}, s + \{\ell \mapsto n\}\rangle \quad \text{if } \ell \in \text{dom}(s) \\
\text{(assign1'')} & \langle \ell := n, s\rangle \longrightarrow \langle \textbf{skip}, s + \{\ell \mapsto n\}\rangle \quad \text{if } \ell \notin \text{dom}(s)
\end{array}$$

$$\text{(assign2)}\ \frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \ell := e, s\rangle \longrightarrow \langle \ell := e', s'\rangle}$$

$$\text{(CBN-app)}\ \frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1\, e_2, s\rangle \longrightarrow \langle e_1'\, e_2, s'\rangle}$$

$$\text{(CBN-fn)}\quad \langle (\textbf{fn}\ x{:}T \Rightarrow e) e_2, s\rangle \longrightarrow \langle \{e_2/x\}e, s\rangle$$

[7]

(b)

1. In L2 $!l$ is stuck if l is not in the store, whereas in HMM it reduces to 0 and $l \mapsto 0$ is added to the store. The standard rule is

$$\text{(deref)}\quad \langle !\ell, s\rangle \longrightarrow \langle n, s\rangle \quad \text{if } \ell \in \text{dom}(s) \text{ and } s(\ell) = n$$

   If ℓ is not in the store, there is likely to be some programmer error – which it would be useful to discover sooner rather than later. [2]

2. In L2 $l := n$ is stuck if l is not in the store, whereas in HMM it reduces to **skip** and $l \mapsto n$ is added to the store. The standard rules are

$$\text{(assign1)}\quad \langle \ell := n, s\rangle \longrightarrow \langle \textbf{skip}, s + \{\ell \mapsto n\}\rangle \quad \text{if } \ell \in \text{dom}(s)$$

$$\text{(assign2)}\ \frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \ell := e, s\rangle \longrightarrow \langle \ell := e', s'\rangle}$$

   The rationale for this is as above, albeit weaker. [2]

3. L2 has call-by-value function application, whereas HMM is call-by-name[1]. For example, $\langle (\textbf{fn}\ \text{x:unit} \Rightarrow ())(l := 1),\ \{l = 0\}\rangle$ reduces to a state with $l = 1$ in L2 but $l = 0$ in HMM[1]. The standard rules are:

$$\text{(app1)}\ \frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1\ e_2, s\rangle \longrightarrow \langle e_1'\ e_2, s'\rangle}$$

$$\text{(app2)}\ \frac{\langle e_2, s\rangle \longrightarrow \langle e_2', s'\rangle}{\langle v\ e_2, s\rangle \longrightarrow \langle v\ e_2', s'\rangle}$$

$$\text{(fn)}\quad \langle (\textbf{fn}\ x{:}T \Rightarrow e)\ v, s\rangle \longrightarrow \langle \{v/x\}e, s\rangle$$

   where values are $v ::= n \mid \textbf{skip} \mid \textbf{fn}\ x{:}T \Rightarrow e$. The combination of CBN reduction and store is counter-intuitive – it's hard to see how many times the argument (which may be an assignment) of a function will be executed. [3].