C-Machine

Values:

$$v \ ::= \ n \ | \ b \ | \ \texttt{fun}\ x(y : \tau_1) : \tau_2\ \texttt{is}\ e$$

Note that values are a subset of expressions, and that all values are closed expressions (i.e. the fun expressions have no free variables). We will drop the types in function expressions when we give the transition rules.

Frames:

$$\begin{array}{rcll} f & ::= & \texttt{Op1}(\texttt{op}, e) & (\texttt{op}(\square, e)) \\ & | & \texttt{Op2}(\texttt{op},) & (\texttt{op}(v, \square)) \\ & | & \texttt{If}(e_1, e_2) & (\texttt{if}(\square, e_1, e_2)) \\ & | & \texttt{App1}(e) & (\texttt{apply}(\square, e)) \\ & | & \texttt{App2}(v) & (\texttt{apply}(v, \square)) \end{array}$$

Stacks:

$$k \ ::= \ \texttt{nil} \ | \ f{::}k$$

A stack is a list of frames, with the innermost frame at the head of the list.

States:

$$s \ ::= \ (k, e)$$

An initial state is of the form $(\texttt{nil}, e)$ where e is a well-typed (and therefore closed) expression. A final state is of the form $(\texttt{nil}, v)$ where v is a value expression.

Rules for $\mapsto_C$:

$$\begin{array}{ll} (CM1) & (k, \texttt{op}(e_1, e_2)) \ \mapsto_C \ (\texttt{Op1}(e_2, E){::}k, e_1) \\ (CM2) & (k, \texttt{if}(e_1, e_2, e_3)) \ \mapsto_C \ (\texttt{If}(e_2, e_3, E){::}k, e_1) \\ (CM3) & (k, \texttt{apply}(e_1, e_2)) \ \mapsto_C \ (\texttt{App1}(e_2, E){::}k, e_1) \end{array}$$

$$\begin{array}{ll} (CM4) & (\texttt{Op1}(\texttt{op}, e_2){::}k, V_1) \ \mapsto_C \ (\texttt{Op2}(\texttt{op}, V_1){::}k, e_2) \\ (CM5) & (\texttt{Op2}(\texttt{op}, V_1){::}k, V_2) \ \mapsto_C \ (k, M(\texttt{op})(V_1, V_2)) \\ (CM6) & (\texttt{If}(e_1, e_2){::}k, true) \ \mapsto_C \ (k, e_1) \\ (CM7) & (\texttt{If}(e_1, e_2){::}k, false) \ \mapsto_C \ (k, e_2) \\ (CM8) & (\texttt{App1}(e_2){::}k, V_1) \ \mapsto_C \ (\texttt{App2}(V_1){::}k, e_2) \\ (CM9) & (\texttt{App2}((\texttt{fun}\ x(y)\ \texttt{is}\ e)){::}k, v_2) \ \mapsto_C \ (k, \{v_1, v_2/f, x\}e) \end{array}$$

where $v_1 = \texttt{fun}\ x(y)\ \texttt{is}\ e$ in CM9. CM1, CM2, and CM3 are analysis rules, CM4 and CM8 are shift rules, and CM5, CM6, CM7, and CM9 are reduction rules.

Typing for C-machine Frames, Stacks, and States

We have three new typing judgments: $\vdash_F\ f :\ \tau_1 \Rightarrow \tau_2$ for frames, $\vdash_K\ k :\ \tau_1 \Rightarrow \tau_2$ for stacks, and $\vdash_S\ (k, e) :\ \tau$ for C-machine states. The notation $\tau_1 \Rightarrow \tau_2$ used in the frame and stack judgements indicates that τ_1 is the *input* type of the frame or stack (the type of the expression or value filling the hole), and τ_2 is the *output* type of the frame or stack, which is the type of the value returned by the frame or stack.

Rules for typing frames (CFT = C-machine Frame Typing):

$$\frac{\vdash e : \texttt{int} \qquad (\texttt{op} \in \{\texttt{plus}, \texttt{times}\})}{\vdash_F\ \texttt{Op1}(\texttt{op}, e) :\ \texttt{int} \Rightarrow \texttt{int}} \qquad \text{(CFT1)}$$

$$\frac{\vdash v : \texttt{int} \qquad (\texttt{op} \in \{\texttt{plus}, \texttt{times}\})}{\vdash_F\ \texttt{Op2}(\texttt{op}, v) :\ \texttt{int} \Rightarrow \texttt{int}} \qquad \text{(CFT2)}$$

$$\frac{\vdash e : \texttt{int} \qquad (\texttt{op} \in \{\texttt{equal}, \texttt{less}\})}{\vdash_F\ \texttt{Op1}(\texttt{op}, e) :\ \texttt{int} \Rightarrow \texttt{bool}} \qquad \text{(CFT3)}$$

$$\frac{\vdash v : \texttt{int} \qquad (\texttt{op} \in \{\texttt{equal}, \texttt{less}\})}{\vdash_F\ \texttt{Op2}(\texttt{op}, v) :\ \texttt{int} \Rightarrow \texttt{bool}} \qquad \text{(CFT4)}$$

$$\frac{\vdash e :\ \tau_1}{\vdash_F\ \texttt{App1}(e) :\ (\tau_1 \to \tau_2) \Rightarrow \tau_2} \qquad \text{(CFT5)}$$

$$\frac{\vdash v :\ \tau_1 \to \tau_2}{\vdash_F\ \texttt{App2}(v) :\ \tau_1 \Rightarrow \tau_2} \qquad \text{(CFT6)}$$

Rules for typing stacks (CKT = C-machine Stack Typing):

$$\frac{}{\vdash_K\ \texttt{nil} :\ \tau \Rightarrow \tau} \qquad \text{(CKT1)}$$

$$\frac{\vdash_F\ f :\ \tau_1 \Rightarrow \tau_2 \qquad \vdash_K\ k :\ \tau_2 \Rightarrow \tau_3}{\vdash_K\ f{::}k :\ \tau_1 \Rightarrow \tau_3} \qquad \text{(CKT2)}$$

Rules for typing states (CST = C-machine State Typing):

$$\frac{\vdash e :\ \tau \quad \vdash_K\ k :\ \tau \Rightarrow \tau'}{\vdash_S\ (k, e) :\ \tau'} \qquad \text{(CST)}$$

Theorem (Progress): If $\vdash_S\ (k, e) :\ \tau$, then either the state (k, e) is final (i.e. is (nil, v) where e is the value v), or there exists a state (k', e') such that $(k, e)\ \mapsto_C\ (k', e')$.

Proof sketch: The proof is by induction on the derivation of the premise $\vdash_K\ k :\ \tau' \Rightarrow \tau$, and in the case where $k = f{::}k'$ by case analysis on the derivation of the type of f.

Theorem (Preservation): If $\vdash_S\ (k, e) :\ \tau$ and $(k, e)\ \mapsto_C\ (k', e')$ then $\vdash_S\ (k', e') :\ \tau$.

Proof sketch: By case analysis on the rules for deriving $(k, e)\ \mapsto_C\ (k', e')$.