A The uML language (plus rec, sans letrec)

Syntax

$$
\begin{aligned}
n &\in \mathbb{Z} \\
s &\in \mathbf{String} \\
x &\in \mathbf{Var} \\
\oplus &::= + \mid - \mid * \mid = \mid \leq \mid \wedge \mid \ldots \\
b &::= \mathsf{true} \mid \mathsf{false} \mid n \mid s \\
e &::= b \mid x \mid e_0 \oplus e_1 \mid \mathsf{if}\ e_0\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid \#n\ e \mid (e_1, \ldots, e_n) \mid \lambda x.\, e \\
&\quad \mid e_0\ e_1 \mid \mathsf{rec}\ y.(\lambda x.\, e) \mid \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2
\end{aligned}
$$

Small-step structural Operational Semantics

Values and evaluation contexts

$$
\begin{aligned}
v &::= b \mid \lambda x.\, e \mid (v_1, \ldots v_n) \\
E[\cdot] &::= [\cdot] \oplus e \mid v \oplus [\cdot] \mid \mathsf{if}\ [\cdot]\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \mid (v_1, \ldots, v_{j-1}, [\cdot], e_{j+1}, \ldots, e_n) \mid [\cdot]\ e \mid v\ [\cdot] \mid \\
&\quad \mid \#\mathsf{n}\ [\cdot] \mid \mathsf{let}\ x = [\cdot]\ \mathsf{in}\ e
\end{aligned}
$$

Small-step transition relation

$$\frac{e \longrightarrow e'}{E[e] \longrightarrow E[e']} \qquad \frac{}{(\lambda x.\, e)\ v \longrightarrow e\{v/x\}}$$

$$\frac{}{\mathsf{if\ true}\ e_1\ e_2 \longrightarrow e_1} \qquad \frac{}{\mathsf{if\ false}\ e_1\ e_2 \longrightarrow e_2}$$

$$\frac{}{\#n\ (v_1, \ldots v_{n'}) \longrightarrow v_n}\ (\text{where } 1 \leq n \leq n') \qquad \frac{}{v_1 \oplus v_2 \longrightarrow v}\ (\text{where } v = v_1 \oplus v_2)$$

$$\frac{}{\mathsf{rec}\ y.(\lambda x.\, e) \longrightarrow e\{\mathsf{rec}\ y.(\lambda x.\, e)/y\}} \qquad \frac{}{\mathsf{let}\ x = v\ \mathsf{in}\ e \longrightarrow e\{v/x\}}$$

B IMP

Syntax

$$
\begin{array}{lcl}
x & \in & \mathbf{Var} \\
a & \in & \mathbf{Aexp} ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\
b & \in & \mathbf{Bexp} ::= \mathbf{true} \mid \mathbf{false} \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid a_1 = a_2 \mid a_1 \leq a_2 \\
c & \in & \mathbf{Com} ::= \mathbf{skip} \mid x := e \mid c_1;\ c_2 \mid \mathbf{if}\, b\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2 \mid \mathbf{while}\, b\, \mathbf{do}\, c
\end{array}
$$

Constants consist of integers $n \in \mathbb{Z}$ and truth values in $\mathbb{T} = \{\mathbf{true}, \mathbf{false}\}$.

Small-step structural operational semantics

A *configuration* is a pair $\langle c, \sigma \rangle$ where $\sigma : \mathbf{Var} \to \mathbb{Z}$ is a *store*. Final configurations are $\langle \mathbf{skip}, \sigma \rangle$.

Arithmetic Expressions

$$\frac{}{\langle n, \sigma \rangle \longrightarrow n} \qquad \frac{}{\langle x, \sigma \rangle \longrightarrow \sigma(x)}$$

$$\frac{n_3 = n_1 \oplus n_2}{\langle n_1 \oplus n_2, \sigma \rangle \longrightarrow n_3}$$

$$\frac{\langle a_1, \sigma \rangle \longrightarrow a_1'}{\langle a_1 \oplus a_2, \sigma \rangle \longrightarrow a_1' \oplus a_2} \qquad \frac{\langle a_2, \sigma \rangle \longrightarrow a_2'}{\langle n_1 \oplus a_2, \sigma \rangle \longrightarrow n_1 \oplus a_2'}$$

Boolean Expressions

Similar to arithmetic expressions.

Commands

$$\frac{}{\langle x := n, \sigma \rangle \longrightarrow \langle \mathbf{skip}, \sigma[x \to n] \rangle} \qquad \frac{\langle a, \sigma \rangle \longrightarrow a'}{\langle x := a, \sigma \rangle \longrightarrow \langle x := a', \sigma \rangle}$$

$$\frac{}{\langle \mathbf{skip}; c_2, \sigma \rangle \longrightarrow \langle c_2, \sigma \rangle} \qquad \frac{\langle c_1, \sigma \rangle \longrightarrow \langle c_1', \sigma' \rangle}{\langle c_1; c_2, \sigma \rangle \longrightarrow \langle c_1'; c_2, \sigma' \rangle}$$

$$\frac{}{\langle \mathbf{if}\, \mathbf{true}\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2, \sigma \rangle \longrightarrow \langle c_1, \sigma \rangle} \qquad \frac{}{\langle \mathbf{if}\, \mathbf{false}\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2, \sigma \rangle \longrightarrow \langle c_2, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \longrightarrow b'}{\langle \mathbf{if}\, b\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2, \sigma \rangle \longrightarrow \langle \mathbf{if}\, b'\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2, \sigma \rangle}$$

$$\frac{}{\langle \mathbf{while}\, b\, \mathbf{do}\, c, \sigma \rangle \longrightarrow \langle \mathbf{if}\, b\, \mathbf{then}\, (c; \mathbf{while}\, b\, \mathbf{do}\, c)\, \mathbf{else}\, \mathbf{skip}, \sigma \rangle}$$

Denotational semantics

Arithmetic denotations

The semantic function $\mathcal{A} : \mathbf{Aexp} \to (\Sigma \to \mathbb{N})$ is defined by induction on the structure of arithmetic expressions:

$$\begin{array}{rcl}
\mathcal{A}[\![n]\!] & = & \lambda\sigma \in \Sigma \,.\, n \\
\mathcal{A}[\![x]\!] & = & \lambda\sigma \in \Sigma \,.\, \sigma x \\
\mathcal{A}[\![a_0 + a_1]\!] & = & \lambda\sigma \in \Sigma \,.\, \mathcal{A}[\![a_0]\!]\sigma + \mathcal{A}[\![a_1]\!]\sigma \\
\mathcal{A}[\![a_0 - a_1]\!] & = & \lambda\sigma \in \Sigma \,.\, \mathcal{A}[\![a_0]\!]\sigma - \mathcal{A}[\![a_1]\!]\sigma \\
\mathcal{A}[\![a_0 \times a_1]\!] & = & \lambda\sigma \in \Sigma \,.\, \mathcal{A}[\![a_0]\!]\sigma \times \mathcal{A}[\![a_1]\!]\sigma
\end{array}$$

Boolean denotations

The semantic function $\mathcal{B} : \mathbf{Bexp} \to (\Sigma \to T)$ is defined by induction on the structure of boolean expressions:

$$\begin{array}{rcl}
\mathcal{B}[\![\mathbf{true}]\!]\sigma & = & \lambda\sigma \in \Sigma \,.\, \mathit{true} \\
\mathcal{B}[\![\mathbf{false}]\!]\sigma & = & \lambda\sigma \in \Sigma \,.\, \mathit{false} \\
\mathcal{B}[\![a_0 = a_1]\!]\sigma & = & \mathit{if}\ \mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma\ \mathit{then\ true\ else\ false} \\
\mathcal{B}[\![a_0 \leq a_1]\!]\sigma & = & \mathit{if}\ \mathcal{A}[\![a_0]\!]\sigma \leq \mathcal{A}[\![a_1]\!]\sigma\ \mathit{then\ true\ else\ false} \\
\mathcal{B}[\![b_0 \wedge b_1]\!]\sigma & = & \mathit{if}\ \mathcal{B}[\![b_0]\!]\sigma\ \wedge\ \mathcal{B}[\![b_1]\!]\sigma\ \mathit{then\ true\ else\ false} \\
\mathcal{B}[\![b_0 \vee b_1]\!]\sigma & = & \mathit{if}\ \mathcal{B}[\![b_0]\!]\sigma\ \vee\ \mathcal{B}[\![b_1]\!]\sigma\ \mathit{then\ true\ else\ false}
\end{array}$$

Command denotations

The semantic function $\mathcal{C} : \mathbf{Com} \to \Sigma \to \Sigma_\perp$ is also defined using induction on the structure of commands:

$$\begin{array}{rcl}
\mathcal{C}[\![\mathbf{skip}]\!]\sigma & = & \lfloor\sigma\rfloor \\
\mathcal{C}[\![x := a]\!]\sigma & = & \lfloor\sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]\rfloor \\
\mathcal{C}[\![c_0;\, c_1]\!]\sigma & = & \mathcal{C}[\![c_1]\!]^*(\mathcal{C}[\![c_0]\!]\sigma) = \left\{ \begin{array}{ll} \mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) & (\text{if } \mathcal{C}[\![c_0]\!]\sigma) \neq \perp) \\ \perp & (\text{otherwise}) \end{array} \right. \\
\mathcal{C}[\![\mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1]\!]\sigma & = & \mathit{if}\ \mathcal{B}[\![b]\!]\sigma\ \mathit{then}\ \mathcal{C}[\![c_0]\!]\sigma\ \mathit{else}\ \mathcal{C}[\![c_1]\!]\sigma \\
\mathcal{C}[\![\mathbf{while}\, b\, \mathbf{do}\, c]\!] & = & \mathit{fix}(\lambda d \in \Sigma \to \Sigma_\perp \,.\, \lambda\sigma \in \Sigma.\ \mathit{if}\ \neg\mathcal{B}[\![b]\!]\sigma\ \mathit{then}\ \lfloor\sigma\rfloor\ \mathit{else}\ d^*(\mathcal{C}[\![c]\!]\sigma))
\end{array}$$

Note: Given a function $f : D \to E_\perp$, $f^* = \lambda d \in D_\perp \,.\, \mathit{if}\ d = \perp\ \mathit{then}\ \perp\ \mathit{else}\ f(d)$. That is,

$$\cdot^* = \lambda f \in D \to E_\perp \,.\, \lambda d \in D_\perp \,.\, \mathit{if}\ d = \perp\ \mathit{then}\ \perp\ \mathit{else}\ f(d)$$