Abstract Machine

Configurations: $(c,\ e,\ s)$ where

- c is the code to be executed
  $c\ \in$ Code = Inst*

- e is the evaluation stack
  $e\ \in$ Stack = $(\mathsf{Z}\ \cup\ \mathsf{T})^\star$

- s is the storage
  $s\ \in$ State

Instructions:

$$
\begin{array}{rcl}
inst & ::= & \textsc{push}\text{-}n \mid \textsc{add} \mid \textsc{mult} \mid \textsc{sub} \\
 & \mid & \textsc{true} \mid \textsc{false} \\
 & \mid & \textsc{eq} \mid \textsc{le} \mid \textsc{and} \mid \textsc{neg} \\
 & \mid & \textsc{fetch}\text{-}x \mid \textsc{store}\text{-}x \\
 & \mid & \textsc{noop} \mid \textsc{branch}(c,\ c) \\
 & \mid & \textsc{loop}(c,\ c)
\end{array}
$$

$$\boxed{(c,\ e,\ s)\ \rhd\ (c',\ e',\ s')}$$

$(\text{PUSH-}n{:}c,\ e,\ s)\ \rhd\ (c,\ \mathcal{N}[n]{:}e,\ s)$

$(\text{ADD}{:}c,\ z_1{:}z_2{:}e,\ s)\ \rhd\ (c,\ (z_1 + z_2){:}e,\ s)$

if $z_1,\ z_2\ \in \mathsf{Z}$

$(\text{MULT}{:}c,\ z_1{:}z_2{:}e,\ s)\ \rhd\ (c,\ (z_1 \star z_2){:}e,\ s)$

if $z_1,\ z_2\ \in \mathsf{Z}$

$(\text{SUB}{:}c,\ z_1{:}z_2{:}e,\ s)\ \rhd\ (c,\ (z_1 - z_2){:}e,\ s)$

if $z_1,\ z_2\ \in \mathsf{Z}$

$(\text{TRUE}{:}c,\ e,\ s)\ \rhd\ (c,\ \mathsf{tt}{:}e,\ s)$

$(\text{FALSE}{:}c,\ e,\ s)\ \rhd\ (c,\ \mathsf{ff}{:}e,\ s)$

$(\text{EQ}{:}c,\ z_1{:}z_2{:}e,\ s)\ \rhd\ (c,\ (z_1{=}z_2){:}e,\ s)$

if $z_1,\ z_2\ \in \mathsf{Z}$

$(\text{LE}{:}c,\ z_1{:}z_2{:}e,\ s)\ \rhd\ (c,\ (z_1 \leq z_2){:}e,\ s)$

if $z_1,\ z_2\ \in \mathsf{Z}$

$(\text{AND}{:}c,\ t_1{:}t_2{:}e,\ s)\ \rhd\ (c,\ (t_1\ \wedge\ t_2){:}e,\ s)$

if $t_1,\ t_2\ \in \mathsf{T}$

$(\text{NEG}{:}c,\ t{:}e,\ s)\ \rhd\ (c,\ (\neg t){:}e,\ s)$

if $t\ \in \mathsf{T}$

$$\boxed{(c,\ e,\ s)\ \rhd\ (c',\ e',\ s')}$$

$$\big(\textsc{fetch}\text{-}x{:}c,\ e,\ s\big)\ \rhd\ (c,\ (s\ x){:}e,\ s)$$

$$\big(\textsc{store}\text{-}x{:}c,\ z{:}e,\ s\big)\ \rhd\ (c,\ e,\ s[x \mapsto z]) \quad \text{if } z \in \mathsf{Z}$$

$$\big(\textsc{noop}{:}c,\ e,\ s\big)\ \rhd\ (c,\ e,\ s)$$

$$\big(\textsc{branch}\big(c_1,\ c_2\big){:}c,\ t{:}e,\ s\big)\ \rhd \begin{cases} (c_1 : c, e, s) & \text{if } t{=}\mathsf{tt} \\ (c_2 : c, e, s) & \text{if } t{=}\mathsf{ff} \end{cases}$$

$$\big(\textsc{loop}\big(c_1,\ c_2\big){:}c,\ e,\ s\big)\ \rhd\ \big(c_1{:}\textsc{branch}\big(c_2{:}\textsc{loop}\big(c_1,\ c_2\big),\ \textsc{noop}\big){:}c,\ e,\ s\big)$$

Computation Sequences

A computation sequence for c and s is either

- a finite sequence

  $$\gamma_0,\ \gamma_1,\ \gamma_2,\ \cdots,\ \gamma_k$$

  of configurations satisfying
  - $\gamma_0 = (c,\ \varepsilon,\ s)$
  - $\gamma_i \ \triangleright\ \gamma_{i+1}$ for $0 \leq i < k$, $k \ \geq\ 0$
  - there is no γ such that $\gamma_k \ \triangleright\ \gamma$

- an infinite sequence

  $$\gamma_0,\ \gamma_1,\ \gamma_2,\ \cdots$$

  of configurations satisfying
  - $\gamma_0 = (c,\ \varepsilon,\ s)$
  - $\gamma_i \ \triangleright\ \gamma_{i+1}$ for $0 \leq i$

Code generation for expressions

$$
\begin{aligned}
\mathcal{CA}[n] &= \text{PUSH-}n\\
\mathcal{CA}[x] &= \text{FETCH-}x\\
\mathcal{CA}[a_1{+}a_2] &= \mathcal{CA}[a_2]{:}\mathcal{CA}[a_1]{:}\text{ADD}\\
\mathcal{CA}[a_1 \star a_2] &= \mathcal{CA}[a_2]{:}\mathcal{CA}[a_1]{:}\text{MULT}\\
\mathcal{CA}[a_1{-}a_2] &= \mathcal{CA}[a_2]{:}\mathcal{CA}[a_1]{:}\text{SUB}\\
\mathcal{CB}[\mathtt{true}] &= \text{TRUE}\\
\mathcal{CB}[\mathtt{false}] &= \text{FALSE}\\
\mathcal{CB}[a_1 = a_2] &= \mathcal{CA}[a_2]{:}\mathcal{CA}[a_1]{:}\text{EQ}\\
\mathcal{CB}[a_1 \leq a_2] &= \mathcal{CA}[a_2]{:}\mathcal{CA}[a_1]{:}\text{LE}\\
\mathcal{CB}[\neg\, b] &= \mathcal{CB}[b]{:}\text{NEG}\\
\mathcal{CB}[b_1 \wedge b_2] &= \mathcal{CB}[b_2]{:}\mathcal{CB}[b_1]{:}\text{AND}
\end{aligned}
$$

Code generation for statements

$$\mathcal{CS}[x := a]$$
$$= \mathcal{CA}[a]\text{:}\textsc{store}\text{-}x$$

$$\mathcal{CS}[\texttt{skip}]$$
$$= \textsc{noop}$$

$$\mathcal{CS}[s_1;s_2]$$
$$= \mathcal{CS}[s_1]\text{:}\mathcal{CS}[s_2]$$

$$\mathcal{CS}[\texttt{if }b\texttt{ then }s_1\texttt{ else }s_2]$$
$$= \mathcal{CB}[b]\text{:}\textsc{branch}(\mathcal{CS}[s_1],\mathcal{CS}[s_2])$$

$$\mathcal{CS}[\texttt{while }b\texttt{ do }s]$$
$$= \textsc{loop}(\mathcal{CB}[b],\mathcal{CS}[s])$$

Factorial example

$$\mathcal{CS}[\texttt{y := 1;}$$
$$\quad \texttt{while } \neg(\texttt{x = 1})$$
$$\quad \texttt{do } (\texttt{y := y} \star \texttt{x; x := x} - \texttt{1})]$$

$$= \textsc{push-}1{:}\textsc{store-}\mathtt{y}{:}$$
$$\quad \textsc{loop}(\textsc{push-}1{:}\textsc{fetch-}\mathtt{x}{:}$$
$$\qquad\qquad \textsc{eq}{:}\textsc{neg},$$
$$\qquad \textsc{fetch-}\mathtt{x}{:}\textsc{fetch-}\mathtt{y}{:}$$
$$\qquad\qquad \textsc{mult}{:}\textsc{store-}\mathtt{y}{:}$$
$$\qquad \textsc{push-}1{:}\textsc{fetch-}\mathtt{x}{:}$$
$$\qquad\qquad \textsc{sub}{:}\textsc{store-}\mathtt{x})$$

Semantic functions

For the abstract machine:

$\mathcal{M}$: Code $\rightarrow$ (State $\hookrightarrow$ State)

$$\mathcal{M}[c]\ s = \begin{cases} s' & \text{if } (c,\ \varepsilon,\ s) \ \rhd^* \ (\varepsilon,\ e,\ s') \\ \text{undef} & \text{otherwise} \end{cases}$$

For the statements:

$\mathcal{S}_{am}$: Stm $\rightarrow$ (State $\hookrightarrow$ State)

$$\mathcal{S}_{am}[s] = (\mathcal{M} \circ \mathcal{CS})[s]$$