Programming Language Theory

A compiler for **While**

2011-04-03

Grammar for instructions

$$
\begin{array}{lll}
[::=]inst & \rightarrow & PUSH - n \mid ADD \mid MULT \mid SUB \mid TRUE \mid FALSE \\
inst & \rightarrow & EQ \mid LE \mid AND \mid NEG \mid FETCH - x \mid STORE - x \\
inst & \rightarrow & NOOP \mid BRANCH(c, c) \mid LOOP(c, c) \\
c & \rightarrow & \epsilon \mid inst : c
\end{array}
$$

Abstract machine configuration

$$
\begin{aligned}
\langle\, c, e, \sigma \,\rangle &\in Code \times Stack \times State \\
Code &= Inst^* \\
Stack &= (\mathbb{Z} \cup \mathbb{B})^* \\
State &= Var \rightarrow \mathbb{Z}
\end{aligned}
$$

Abstract machine transition

$$\langle\, c, e, \sigma \,\rangle \rhd \langle\, c,' e', \sigma' \,\rangle$$

Operational semantics

$$
\begin{array}{ll}
[\triangleright]\langle\, PUSH - n : c, e, \sigma \,\rangle & \rightarrow \langle\, c, \mathcal{N}[\![n]\!] : e, \sigma \,\rangle \\
\langle\, ADD : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 + z_2) : e, \sigma \,\rangle \\
\langle\, MULT : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 * z_2) : e, \sigma \,\rangle \\
\langle\, SUB : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 - z_2) : e, \sigma \,\rangle \\
\langle\, TRUE : c, e, \sigma \,\rangle & \rightarrow \langle\, c, \mathbf{tt} : e, \sigma \,\rangle \\
\langle\, FALSE : c, e, \sigma \,\rangle & \rightarrow \langle\, c, \mathbf{ff} : e, \sigma \,\rangle \\
\langle\, EQ : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 = z_2) : e, \sigma \,\rangle \\
\langle\, LE : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 \leq z_2) : e, \sigma \,\rangle
\end{array}
$$

Operational semantics

$$
\begin{array}{ll}
[\triangleright]\langle\, AND : c, b_1 : b_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (b_1 \wedge b_2) : e, \sigma \,\rangle \\
\langle\, NEG : c, b : e, \sigma \,\rangle & \rightarrow \langle\, c, (\neg b) : e, \sigma \,\rangle \\
\langle\, FETCH - x : c, e, \sigma \,\rangle & \rightarrow \langle\, c, (\sigma\ x) : e, \sigma \,\rangle \\
\langle\, STORE - x : c, z : e, \sigma \,\rangle & \rightarrow \langle\, c, e, \sigma[x \mapsto z] \,\rangle \\
\langle\, SUB : c, z_1 : z_2 : e, \sigma \,\rangle & \rightarrow \langle\, c, (z_1 - z_2) : e, \sigma \,\rangle \\
\langle\, NOOP : c, e, \sigma \,\rangle & \rightarrow \langle\, c, e, \sigma \,\rangle \\
\langle\, BRANCH(c_1, c_2) : c, b : e, \sigma \,\rangle & \rightarrow \langle\, c_1 : c, e, \sigma \,\rangle \ \texttt{if}\ \ b = \mathbf{tt} \\
\langle\, BRANCH(c_1, c_2) : c, b : e, \sigma \,\rangle & \rightarrow \langle\, c_2 : c, e, \sigma \,\rangle \ \texttt{if}\ \ b = \mathbf{ff} \\
\langle\, LOOP(c_1, c_2) : c, b : e, \sigma \,\rangle & \rightarrow \\
\multicolumn{2}{r}{\langle\, c_1 : BRANCH(c_2 : LOOP(c_1, c_2), NOOP) : c, e, \sigma \,\rangle}
\end{array}
$$

Execution function

$$
\mathcal{M}[\![c]\!]\sigma = \begin{cases} \sigma' & \text{if } \langle\, c, \epsilon, \sigma \,\rangle \triangleright^* \langle\, \epsilon, e, \sigma' \,\rangle \\ \text{undefined} & \text{otherwise} \end{cases}
$$

Compiler types

$$
\begin{array}{l}
\mathcal{CA} \in Aexp \rightarrow Code \\
\mathcal{CB} \in Bexp \rightarrow Code \\
\mathcal{CS} \in Stm \rightarrow Code
\end{array}
$$

$\mathcal{CA}$

$$
\begin{array}{ll}
[=]\mathcal{CA}[\![n]\!] & \rightarrow PUSH - n \\
\mathcal{CA}[\![x]\!] & \rightarrow FETCH - x \\
\mathcal{CA}[\![a_1 + a_2]\!] & \rightarrow \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : ADD \\
\mathcal{CA}[\![a_1 * a_2]\!] & \rightarrow \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : MULT \\
\mathcal{CA}[\![a_1 - a_2]\!] & \rightarrow \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : SUB
\end{array}
$$

$\mathcal{CB}$

$$
\begin{array}{ll}
[=]\mathcal{CB}[\![true]\!] & \rightarrow TRUE \\
\mathcal{CB}[\![false]\!] & \rightarrow FALSE \\
\mathcal{CB}[\![a_1 = a_2]\!] & \rightarrow \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : EQ \\
\mathcal{CB}[\![a_1 \leq a_2]\!] & \rightarrow \mathcal{CA}[\![a_2]\!] : \mathcal{CA}[\![a_1]\!] : LE \\
\mathcal{CB}[\![b_1 \wedge b_2]\!] & \rightarrow \mathcal{CB}[\![b_2]\!] : \mathcal{CB}[\![b_1]\!] : AND \\
\mathcal{CB}[\![\neg b]\!] & \rightarrow \mathcal{CB}[\![b]\!] : NEG
\end{array}
$$

$\mathcal{CS}$

$$
\begin{array}{ll}
[=]\mathcal{CS}[\![skip]\!] & \rightarrow NOOP \\
\mathcal{CS}[\![x := a]\!] & \rightarrow \mathcal{CA}[\![a]\!] : STORE - x \\
\mathcal{CS}[\![S_1; S_2]\!] & \rightarrow \mathcal{CS}[\![S_1]\!] : \mathcal{CS}[\![S_2]\!] \\
\mathcal{CS}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] & \rightarrow \mathcal{CB}[\![b]\!] : BRANCH(\mathcal{CS}[\![S_1]\!], \mathcal{CS}[\![S_2]\!]) \\
\mathcal{CS}[\![\texttt{while } b \texttt{ do } S]\!] & \rightarrow LOOP(\mathcal{CB}[\![b]\!], \mathcal{CS}[\![S]\!])
\end{array}
$$

Semantic function

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

$$\mathcal{S}_{am}[\![S]\!]\sigma = \mathcal{M}(\mathcal{CS}[\![S]\!])\sigma$$

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

$$\mathcal{S}_{am} = \mathcal{M} \circ \mathcal{CS}$$

Correctness

Theorem

$$\mathcal{S}_{ns} = \mathcal{S}_{am}$$

Proof

By

Lemma
$\langle \mathcal{CA}[\![a]\!], \epsilon, \sigma \rangle \rhd^* \langle \epsilon, \mathcal{A}[\![a]\!]\sigma, \sigma \rangle$

Lemma
$\langle \mathcal{CB}[\![b]\!], \epsilon, \sigma \rangle \rhd^* \langle \epsilon, \mathcal{B}[\![b]\!]\sigma, \sigma \rangle$

Lemma
If $\langle S, \sigma \rangle \rightarrow \sigma'$ *then* $\langle \mathcal{CS}[\![S]\!], \epsilon, \sigma \rangle \rhd^* \langle \epsilon, \epsilon, \sigma' \rangle$

Lemma
If $\langle \mathcal{CS}[\![S]\!], \epsilon, \sigma \rangle \rhd^k \langle \epsilon, e, \sigma' \rangle$ *then* $\langle S, \sigma \rangle \rightarrow \sigma'$ *and* $e = \epsilon$