F11

A compiler for **While**

Lennart Andersson

Revision 2009-04-22

2009

Grammar for instructions

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

Abstract machine configuration

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

Abstract machine transition

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

Operational semantics

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

Operational semantics

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

Execution function

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

Compiler types

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

$\mathcal{CA}$

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

$\mathcal{CB}$

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

$\mathcal{CS}$

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

Semantic function

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

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

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

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

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* $e = \epsilon$ ■