CS 4110

Programming Languages & Logics

Lecture 5

The IMP Language

Simple imperative language

We’ll now consider a more realistic programming language...

Simple imperative language

We'll now consider a more realistic programming language...

arithmetic expressions $a \in \textbf{Aexp} \quad a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2$

Simple imperative language

We'll now consider a more realistic programming language...

$$\begin{array}{rll} \text{arithmetic expressions} & a \in \mathbf{Aexp} & a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2 \\ \text{boolean expressions} & b \in \mathbf{Bexp} & b ::= \mathbf{true} \mid \mathbf{false} \mid a_1 < a_2 \end{array}$$

Simple imperative language

We'll now consider a more realistic programming language...

$$
\begin{array}{rll}
\text{arithmetic expressions} & a \in \textbf{Aexp} & a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2 \\
\text{boolean expressions} & b \in \textbf{Bexp} & b ::= \textbf{true} \mid \textbf{false} \mid a_1 < a_2 \\
\text{commands} & c \in \textbf{Com} & c ::= \textbf{skip} \\
 & & \quad \mid \; x := a \\
 & & \quad \mid \; c_1 ; c_2 \\
 & & \quad \mid \; \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\
 & & \quad \mid \; \textbf{while } b \textbf{ do } c
\end{array}
$$

Large-Step Semantics

Three relations, one for each syntactic category:

$$\Downarrow_{\mathbf{Aexp}} \subseteq (\mathbf{Store} \times \mathbf{Aexp}) \times \mathbf{Store}$$

$$\Downarrow_{\mathbf{Bexp}} \subseteq (\mathbf{Store} \times \mathbf{Bexp}) \times \mathbf{Store}$$

$$\Downarrow_{\mathbf{Com}} \subseteq (\mathbf{Store} \times \mathbf{Com}) \times \mathbf{Store}$$

We'll typically just use $\Downarrow$ where the specific relation we mean is clear from context.

Large-Step Semantics

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

$$\frac{\langle \sigma, e_1 \rangle \Downarrow n_1 \qquad \langle \sigma, e_2 \rangle \Downarrow n_2 \qquad n = n_1 + n_2}{\langle \sigma, e_1 + e_2 \rangle \Downarrow n}$$

$$\frac{\langle \sigma, e_1 \rangle \Downarrow n_1 \qquad \langle \sigma, e_2 \rangle \Downarrow n_2 \qquad n = n_1 \times n_2}{\langle \sigma, e_1 \times e_2 \rangle \Downarrow n}$$

Large-Step Semantics

$$\frac{}{\langle \sigma, \textbf{true} \rangle \Downarrow \textbf{true}} \qquad \frac{}{\langle \sigma, \textbf{false} \rangle \Downarrow \textbf{false}}$$

$$\frac{\langle \sigma, a_1 \rangle \Downarrow n_1 \qquad \langle \sigma, a_2 \rangle \Downarrow n_2 \qquad n_1 < n_2}{\langle \sigma, a_1 < a_2 \rangle \Downarrow \textbf{true}}$$

$$\frac{\langle \sigma, a_1 \rangle \Downarrow n_1 \qquad \langle \sigma, a_2 \rangle \Downarrow n_2 \qquad n_1 \geq n_2}{\langle \sigma, a_1 < a_2 \rangle \Downarrow \textbf{false}}$$

Large-Step Semantics

$$\frac{\textsc{Skip}}{\langle \sigma, \textbf{skip} \rangle \Downarrow \sigma}$$

Large-Step Semantics

$$\textsc{Assgn}\ \frac{\langle \sigma, e \rangle \Downarrow n}{\langle \sigma, x := e \rangle \Downarrow \sigma[x \mapsto n]}$$

Large-Step Semantics

Seq

$$\frac{\langle \sigma, c_1 \rangle \Downarrow \sigma' \qquad \langle \sigma', c_2 \rangle \Downarrow \sigma''}{\langle \sigma, c_1; c_2 \rangle \Downarrow \sigma''}$$

Large-Step Semantics

$$\textsc{If-T}\quad \frac{\langle \sigma, b\rangle \Downarrow \textbf{true} \qquad \langle \sigma, c_1\rangle \Downarrow \sigma'}{\langle \sigma, \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2\rangle \Downarrow \sigma'}$$

Large-Step Semantics

IF-F

$$\frac{\langle \sigma, b \rangle \Downarrow \textbf{false} \qquad \langle \sigma, c_2 \rangle \Downarrow \sigma'}{\langle \sigma, \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \rangle \Downarrow \sigma'}$$

Large-Step Semantics

WHILE-F

$$\frac{\langle \sigma, b \rangle \Downarrow \textbf{false}}{\langle \sigma, \textbf{while } b \textbf{ do } c \rangle \Downarrow \sigma}$$

WHILE-T

$$\frac{\langle \sigma, b \rangle \Downarrow \textbf{true} \qquad \langle \sigma, c \rangle \Downarrow \sigma' \qquad \langle \sigma', \textbf{while } b \textbf{ do } c \rangle \Downarrow \sigma''}{\langle \sigma, \textbf{while } b \textbf{ do } c \rangle \Downarrow \sigma''}$$

Command Equivalence

Intuitively, two commands are equivalent if they produce the same result under any store...

Definition (Equivalence of commands)

Two commands c and c' are equivalent (written $c \sim c'$) if, for any stores σ and σ', we have

$$\langle \sigma, c \rangle \Downarrow \sigma' \iff \langle \sigma, c' \rangle \Downarrow \sigma'.$$

Command Equivalence

For example, we can prove that every **while** command is equivalent to its unfolding:

Theorem

For all $b \in \mathbf{Bexp}$ *and* $c \in \mathbf{Com}$ *we have*

$$\textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ c \sim \textbf{\textit{if}}\ b\ \textbf{\textit{then}}\ (c; \textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ c)\ \textbf{\textit{else}}\ \textbf{\textit{skip}}.$$

Proof.

We show each implication separately... $\square$