Typing and Semantics of Imperative Language

A4M36TPJ, 2014/2015

Today's Lecture

- We specify a syntax for a turing complete language I++.
- We design type rules for I++.
- We specify operational semantics.

I++

- I++ is a simple imperative language.
- We start with a basic version of the language, which contains expressions, commands and variables.
- After that we will add new features into the language as: iteration and functions.

I++ Syntax (Expr)

$$
\begin{aligned}
Expr ::= \; & Num \mid \\
& Bool \mid \\
& \triangle Expr \mid \\
& Expr \odot Expr \mid \\
& Expr \leq Expr \mid \\
& Expr \; \texttt{nand} \; Expr \mid \\
& \texttt{if} \; Expr \; \texttt{then} \; Expr \; \texttt{else} \; Expr \mid \\
& VarName
\end{aligned}
$$

I++ Syntax

$$
\begin{aligned}
Type &::= \texttt{Number} \mid \texttt{Boolean} \\
Cmd &::= Type\ VarName \mid \\
&\quad\ VarName\ \texttt{=}\ Expr \\
Program &::= Cmd\texttt{;}\ Program \mid \\
&\quad\ Cmd\texttt{,}
\end{aligned}
$$

BOS Semantics (1)

$$n \in Num, b \in Bool, s \in (VarName \to (Num \cup Bool))$$

$$\frac{}{(s, n) \Rightarrow n}$$

$$\frac{}{(s, b) \Rightarrow b}$$

BOS Semantics (2)

$$e, e' \in Expr, n, n' \in Num, b \in Bool, s \in (VarName \to (Num \cup Bool))$$

$$\frac{(s, e) \Rightarrow n}{(s, \triangle e) \Rightarrow -n}$$

$$\frac{(s, e) \Rightarrow n \quad (s, e') \Rightarrow n'}{(s, e \odot e') \Rightarrow n + n'}$$

BOS Semantics (3)

$$e, e' \in Expr, n, n' \in Num, b, b' \in Bool, s \in (VarName \rightarrow (Num \cup Bool))$$

$$\frac{(s, e) \Rightarrow n \quad (s, e') \Rightarrow n'}{(s, e \leq e') \Rightarrow n \leq n'}$$

$$\frac{(s, e) \Rightarrow b \quad (s, e') \Rightarrow b'}{(s, e \ \textbf{nand} \ e') \Rightarrow not(b \ \ \&\& \ \ b')}$$

BOS Semantics (4)

$$t \in (Num \cup Bool), e_1, e_2 \in Expr, b \in Bool, s \in (VarName \rightarrow (Num \cup Bool))$$

$$\frac{(s, b) \Rightarrow false \quad (s, e_2) \Rightarrow t}{(s, if \;\; b \;\; then \;\; e_1 \;\; else \;\; e_2) \Rightarrow t}$$

$$\frac{(s, b) \Rightarrow true \quad (s, e_1) \Rightarrow t}{(s, if \;\; b \;\; then \;\; e_1 \;\; else \;\; e_2) \Rightarrow t}$$

BOS Semantics (5)

Reading a variable value

$$x \in VarName, s \in (VarName \rightarrow (Num \cup Bool))$$

$$\frac{}{(s, x) \Rightarrow s(x)}$$

BOS Semantics Open Issues

$$\Rightarrow \in (VarName \rightarrow (Num \cup Bool)) \times Expr \times (Num \cup Bool)$$

- How can we realize assignment in BOS for I++?

BOS Semantics (6)

$$x \in VarName, s \in (VarName \to (Num \cup Bool)), t \in \{Number, Boolean\}$$

$$\Longrightarrow \in (VarName \to (Num \cup Bool)) \times Expr \times (VarName \to (Num \cup Bool))$$

Declaration is an axiom! Why?

$$\frac{}{(s, t\ x) \Longrightarrow s}$$

BOS Semantics (7)

$$x \in VarName, s \in (VarName \to (Num \cup Bool)), e \in Expr, v \in Num \cup Bool$$

$$\frac{(s, e) \Rightarrow v}{(s, x \texttt{ = } e) \Longrightarrow s[x \mapsto v]}$$

BOS Semantics (8)

$$s, s', s'' \in (VarName \to (Num \cup Bool)), c, p \in Command$$

$$\frac{(s, c) \Longrightarrow s' \quad (s', p) \Longrightarrow s''}{(s, c\,;\ p) \Longrightarrow s''}$$

Type Rules (1)

$$\frac{}{\Gamma \vdash n : Number}$$

$$\frac{}{\Gamma \vdash b : Boolean}$$

Type Rules (2)

$$\frac{\Gamma \vdash e : Number}{\Gamma \vdash \triangle e : Number}$$

$$\frac{\Gamma \vdash e : Number \quad \Gamma \vdash e' : Number}{\Gamma \vdash e \odot e' : Number}$$

Type Rules (3)

$$\frac{\Gamma \vdash e : Number \quad \Gamma \vdash e' : Number}{e \leq e' : Boolean}$$

$$\frac{\Gamma \vdash e : Boolean \quad \Gamma \vdash e' : Boolean}{e \ \texttt{nand} \ e' : Boolean}$$

$$\frac{\Gamma \vdash e : Boolean \quad \Gamma \vdash e' : t \quad \Gamma \vdash e'' : t}{\Gamma \vdash \texttt{if} \ e \ \texttt{then} \ e' \ \texttt{else} \ e'' : t}$$

Type Rules (4)

$$\frac{}{\Gamma \vdash v : \Gamma(v)}$$

$$\frac{\Gamma \vdash v : t \quad \Gamma \vdash e : t}{\Gamma \vdash v \texttt{ = } e : \diamond}$$

$$\frac{v \notin dom(\Gamma)}{\Gamma \vdash t\ v : \diamond}$$

Type Rules (5)

$$\frac{\Gamma \vdash v \texttt{ = } e : \diamond \quad \Gamma \vdash p : \diamond}{\Gamma \vdash v \texttt{ = } e \texttt{;}\ p : \diamond}$$

$$\frac{\Gamma \vdash t\ v : \diamond \quad \Gamma \cup \{(v, t)\} \vdash p : \diamond}{\Gamma \vdash t\ v \texttt{;}\ p : \diamond}$$

Iteration

- We add a new statement *while* with the following syntax:

$$Cmd ::= \ while\ Expr\ do\ Program\ end$$

BOS While Semantics

$$\frac{(s,e) \Rightarrow true \quad (s,c) \Longrightarrow s' \quad (s', while\ e\ do\ c\ end) \Longrightarrow s''}{(s, while\ e\ do\ c\ end) \Longrightarrow s''}$$

$$\frac{(s,e) \Rightarrow false}{(s, while\ e\ do\ c\ end) \Longrightarrow s}$$

Type Rule - While

$$\frac{\Gamma \vdash e : Boolean \quad \Gamma \vdash c : \diamond}{\Gamma \vdash while\ e\ do\ c\ end : \diamond}$$

Adding command ++

- We want to have a ++ command in I++ language.
- This command can be used with variables of type Number.

BOS ++ Semantics

$$\frac{}{(s, var++) \Longrightarrow s[var \mapsto (s(var) + 1)]}$$

Type Rule - ++

$$\frac{\Gamma \vdash var : Number}{\Gamma \vdash var ++ : \diamond}$$

Functions

```
int f(int x) {
  return x+50;
};
f(20)
```

Functions

- We add a new syntax construct, to define a function and a function call:

$$
\begin{aligned}
Cmd &::= \ Type\ FunName(Type\ VarName)\{return\ Expr\} \\
Expr &::= \ FunName(Expr)
\end{aligned}
$$