Introduction to Types

February 25, 2002

CS 131: Programming Languages

A Simple Static Semantics

- We start with just two types

  type ::= int | bool

- We will define a typing relation

  exp : *type*

- A type system is frequently called the "static semantics" of the language.

Static Semantic Rules

$$\frac{}{\texttt{n : int}} \qquad \frac{}{\texttt{tt : bool}} \qquad \frac{}{\texttt{ff : bool}}$$

$$\frac{}{exp_1 \texttt{ + } exp_2 \texttt{ :}}$$

$$\frac{}{exp_1 \texttt{ == } exp_2 \texttt{ :}}$$

$$\frac{}{\texttt{(}exp_1 \texttt{ ? } exp_2 \texttt{ : } exp_3\texttt{)} \texttt{ :}}$$

Fill in the Proof Tree

$$\frac{}{\texttt{((3>4) ? 5+6 : 7+8) :}}$$

Pairs

type ::= **int** | **bool** | *type* ***** *type*

$$\frac{}{\langle exp_1, exp_2 \rangle \ :}$$

$$\frac{}{\textbf{fst}\ exp \ :}$$

$$\frac{}{\textbf{snd}\ exp \ :}$$

But...

- Some programs wouldn't get stuck but still don't typecheck

```
(ff ? tt : 4) + 1
```

- For any interesting language, a type system preventing all bad programs also rejects programs that would run without problems.
- Research topic: type systems that catch as many errors as possible, but don't reject useful programs

Languages with Variables

- Well-typedness is *context-sensitive*
  - What is the type of x ?
  - Is x+3 well-typed?

```
let x be 4 in x+3
```

```
let x be tt in x+3
```

- To determine if an expression is well-typed we need to know the types of the *free* variables

Conditional Judgments

- The typing judgments are now *conditional*

  "If x : int then x+3 : int"

  "If x : bool then (x ? 3 : 4) : int"

- Given assumptions about the types of variables, we can conclude code is well-typed

```
x:int ⊢ x+3 : int
```

```
x:bool ⊢ (x ? 3 : 4) : int
```

Typing Environments

- An environment is a lookup table for variables
  - A *type environment* associates variables with types
- Notation
  - We use *tenv* to denote an arbitrary type environment.
  - Type environments can be written as a list
    - e.g., `x:int, y:bool, z:int`
  - The notation *tenv*`(x)` gives the type of `x`
  - The notation *tenv*`,x:int` is the extension of *tenv* that maps `x` to `int`

Updated Static Semantics

$$\frac{}{tenv \vdash n : \textbf{int}} \qquad \frac{}{tenv \vdash \texttt{tt} : \textbf{bool}} \qquad \frac{}{tenv \vdash \texttt{ff} : \textbf{bool}}$$

$$\frac{tenv \vdash exp_1 : \textbf{int} \quad tenv \vdash exp_2 : \textbf{int}}{\texttt{tenv} \vdash exp_1 \textbf{ + } exp_2 : \textbf{int}}$$

$$\frac{tenv \vdash exp_1 : \textbf{int} \quad tenv \vdash exp_2 : \textbf{int}}{\texttt{tenv} \vdash exp_1 \textbf{ == } exp_2 : \texttt{bool}}$$

$$\frac{tenv \vdash exp_1 : \textbf{bool} \quad tenv \vdash exp_2 : type \quad tenv \vdash exp_3 : type}{tenv \vdash (exp_1 \textbf{ ? } exp_2 \textbf{ : } exp_3) : \texttt{type}}$$

More Interesting Rules

$$\frac{}{\texttt{tenv} \vdash var : }$$

$$\frac{}{tenv \vdash \textbf{let } var \textbf{ be } exp_1 \textbf{ in } exp_2 : }$$

$$\frac{}{\vdash \textbf{let x be 3 in (let y be 4 in : ((x+y)<6))}}$$

Functions

type ::= **int** | **bool** |
| *type* ***** *type* | *type* **->** *type*

$$\frac{}{\Gamma \vdash exp_1\ exp_2 :}$$

$$\frac{}{\Gamma \vdash ((var)\ \mathbf{=>}\ exp) :}$$

$$\frac{}{\Gamma \vdash (\mathbf{rec}\ var_1(var_2)\ \mathbf{=>}\ exp) :}$$

$$\frac{}{\vdash \boxed{\begin{array}{l}\mathbf{let\ f\ be}\\ \quad \mathbf{(rec\ g(x)=>\ x==0\ ?\ 0\ :\ x+g(x-1))}\\ \mathbf{in}\\ \quad \mathbf{f(f(4))}\end{array}} :}$$

Are These Rules Right?