Lecture 19

Thursday, October 28, 2021 11:12 AM

Understanding types

$3 + 5 \Rightarrow^* 8$

$(\text{if } 3 \leq 5 \text{ then } 3 \text{ else } 5) + 8 \Rightarrow^* 11$

$3 + \text{true} \Rightarrow^* ?$

$\text{if } 3 \text{ then } 4 \text{ else } 5 \Rightarrow^* ?$

Language L_1 (Untyped expressions)

$$e ::= \underbrace{0 \mid 1 \mid 2 \mid \ldots}_{c \,\in\, \text{Int}} \mid \underbrace{\text{true} \mid \text{false}}_{\text{Booleans}}$$

$$\mid e_1 + e_2 \mid e_1 - e_2$$

$$\mid e_1 \text{ and } e_2 \mid e_1 \text{ or } e_2 \mid \text{not } e_1$$

$$\mid e_1 \leq e_2$$

| if e_1 then e_2 else e_3

Language L_2 (Separating Ints & Bools)

Arithmetic expressions	Boolean expressions
$a ::= \underbrace{0 \mid 1 \mid 2 \mid \ldots}_{c \,\in\, Int}$	$b ::= \underbrace{\text{true} \mid \text{false}}_{\text{Booleans}}$
$\mid a_1 \pm a_2$	$\mid b_1$ and b_2 $\mid b_1$ or b_2
$\mid$ if b then a_1 else a_2	$\mid$ not b_1
	$\mid$ if b_1 then b_2 else b_3
	$\equiv$ $\boxed{(b_1 \text{ and } b_2) \text{ or } (\text{not } b_1 \text{ and } b_3)}$

Grammar in L_2 already encodes our typing rules

Guarantee: No runtime type errors.

Problem: What if the language has infinitely many "types"?

Language L_3 (Ints, Bools, Lists)

$$e ::= c \in \text{Int} \mid c \in \text{Bool} \mid [e_1 ; e_2 ; \cdots ; e_k]$$
$$\mid e_1 \pm e_2 \mid e_1 \ {}^{\text{and}}_{\text{or}} \ e_2 \mid \text{not } e_1$$
$$\mid e_1 [e_2]$$
$$\mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3$$

Int, Bool, List

$$[3;4;5][1] + 2 \Rightarrow^* 6$$

$$[\text{true}; \text{false}; \text{false}][1] + 2 \Rightarrow^* \times$$

– Need to distinguish

List [Int] from List [Bool]

from List [List [Int]]

List [List [Bool]]

⋮

---

a ::= $c \in \text{Int}$ | $\cancel{c \in \text{Bool}}$ | $\cancel{[e_1; e_2; \ldots; e_k]}$

| $a_1 \pm a_2$ | $\cancel{e_1 \ \text{and/or} \ e_2}$ | $\cancel{\text{not } e_1}$

| $\cancel{e_1[e_2]}$ $\qquad l_1[a_2]$

| if b then a_2 else a_3

l ::= $\cancel{c \in \text{Int}}$ | $c \in \text{Bool}$ | $[a_1; a_2; \ldots; a_k]$

| $\cancel{e_1 + e_2}$ | $\cancel{e_1 \ \text{and/or} \ e_2}$ | $\cancel{\text{not } e_1}$

| $l_1[a_2]$

| if b_1 then l_2 else l_3

Program source code → $\boxed{\text{Parser}}$ → AST → $\boxed{\text{Type checker}}$ → Yes, program is sensible → …

$\boxed{\text{Type checker}}$ → No, the program …

No, the program is possibly nonsensical

$$(\text{if true then } 3 \text{ else false}) + 3$$

How to make sense of expressions in L_3

if c is an integer literal,

then c is of type Int

$$\frac{c \in \text{Int}}{c : \text{Int}}$$

If everything above the line is true

Then everything below the line is true

Typing judgment

$$\frac{c \in \text{Bool}}{c : \text{Bool}} \qquad \frac{e_1 : T \quad e_2 : T \quad \cdots \quad e_k : T}{[e_1; e_2; \cdots; e_k] : \text{List}[T]}$$

$$\frac{e_1 : \text{Int} \qquad e_2 : \text{Int}}{\begin{array}{c} e_1 + e_2 : \text{Int} \\ e_1 - e_2 : \text{Int} \end{array}}$$

If e_1 is of type Int & e_2 is of type Int

then (we define) $e_1 + e_2$ to have type Int.

$$\frac{e_1 : \text{Int} \qquad e_2 : \text{Int}}{e_1 \le e_2 : \text{Bool}}$$

$$\frac{e_1 : \text{Bool} \qquad e_2 : \text{Bool}}{e_1 \ \substack{\text{and} \\ \text{or}} \ e_2 : \text{Bool}}$$

$$\text{not } e_1 : \text{Bool}$$

$$\frac{e_1 : \underline{\text{List}[T]} \qquad e_2 : \underline{\text{Int}}}{e_1[e_2] : \underline{\quad T \quad}}$$

$$3 : \text{int}$$

$$x : \text{int}$$

$$x + 3 : \text{int}$$

$$x : \{\text{int where } \_ < 3\}$$

$\text{arr}[x]$

$$\text{arr} : \{\text{List}[\text{Int}] \text{ with len} > x\}$$

$$x : \{\text{Int} > 0\}$$

$$\frac{e_1 : \text{Bool} \qquad e_2 : T \qquad e_3 : T}{\quad}$$

if e_1 then e_2 else e_3 : T

Claim : Well-typed programs do not get stuck.

Ill-typed programs *may* get stuck.

	Static	Dynamic
Strong	Agda, Coq, Idris Haskell, OCaml, Rust Java C++-98 C	Python Javascript
Weak		Perl Bash

Origin of types : Late 1800s / early 1900s

Whitehead & Russell: Principia Mathematica

$A = \{ x \notin x \}$: All elements which don't belong to themselves

$A \in A \Rightarrow A \notin A$

$A \notin A \Rightarrow A \in A$

$\boxed{\text{Russell's Paradox}}$

$\dots A \notin A \quad \Rightarrow A \in A$ $\boxed{\text{Paradox}}$

Language L_4 : (Integers & functions)

Expressions $e ::= c \in \mathrm{Int}$
$\quad\quad\quad\quad | \ x \in \mathrm{Var}$
$\quad\quad\quad\quad | \ e_1 \pm e_2$
$\quad\quad\quad\quad | \ e_1 \ e_2$
$\quad\quad\quad\quad | \ \mathrm{fun}\ (x : T) \Rightarrow e$

Var

Types, $T ::= \mathrm{Int} \ | \ \underbrace{T_1 \rightarrow T_2}$

Functions which take values of type T_1 as input & produce values of type T_2.

Infinitely many types

Int $\quad$ $\text{Int} \to \text{Int}$ $\quad$ $\text{Int} \to (\text{Int} \to \text{Int})$

3 $\quad$ $(\text{fun } x \to x+2)$ $\quad$ $\text{fun } x \to \text{fun } y \to x+y$

$(\text{Int} \to \text{Int}) \to \text{Int}$ $\quad$ ...

$\text{fun } f \to f\ 0$

<u>Typing rules for L_4</u>

$$\frac{c \in \text{Int}}{Ctx \vdash c : \text{Int}} \qquad \frac{(x:T) \in Ctx}{Ctx \vdash x : T}$$

$\vdash$: Turnstile

$Ctx,\ \Gamma$

$$\frac{Ctx \vdash e_1, e_2 : \text{Int}}{Ctx \vdash e_1 \pm e_2 : \text{Int}} \qquad \frac{Ctx \vdash e_1 : T' \to T \quad Ctx \vdash e_2 : T'}{Ctx \vdash e_1\ e_2 : T}$$

$$\frac{Ctx, x : T \vdash e : T'}{Ctx \vdash \text{fun } (x:T) \to e : T \to T'}$$

$\text{let } x = 3 \text{ in } \underline{x} : \text{Int}$

$\text{let } x = \text{false in } x : \text{Bool}$

let $x = 3$ in $\underline{x + x}$

let $x = \text{false}$ in $\underline{x + x}$

<u>Example</u> : $\boxed{\left(\text{fun } (x : \text{Int} \to \text{Int}) \to x\ 5\right)}$ $(\text{Int} \to \text{Int}) \to \text{Int}$

$(\text{fun } (y : \text{Int}) \to y + 1)$: Int

---

① $\text{Empty Ctx} \vdash \left(\text{fun } (y : \text{Int}) \to \underbrace{y + 1}\right) : \text{Int} \to \text{Int}$

$(y : \text{Int}) \vdash (y + 1) : \text{Int}$

② $\left(\text{fun } (\underline{x : \text{Int} \to \text{Int}}) \to \underbrace{x\ 5}\right) : (\text{Int} \to \text{Int}) \to \text{Int}$

$(x : \text{Int} \to \text{Int}) \vdash (\underline{x}\ 5) : \underline{\text{Int}}$

$\text{Ctx} \vdash 5 : \text{Int}$

$x : \text{Int} \to \text{Int}$