CSE 114A lecture 22

Agenda:

- type system for mini-Nano, continued
- tiny, terrible type inferencer

$$e ::= n^{\checkmark} \mid b^{\checkmark} \mid x^{\checkmark} \mid e_1 + e_2^{\checkmark} \mid \backslash x \rightarrow e \mid e_1\ e_2 \mid \text{let } x = e_1 \text{ in } e_2$$

$$T ::= \text{Int} \mid \text{Bool} \mid T_1 \rightarrow T_2$$

True :: Bool

3 + 4 :: Int

$\backslash x \rightarrow x + x$ :: Int → Int

typing relation

Greek letter gamma: $\Gamma \vdash e :: T$

"In type environment Γ, e has type T"

Γ = [(x, Int), (f, Int → Bool), (y, Bool) ...]

$$\frac{}{\Gamma \vdash n :: \text{Int}}$$

$$\frac{}{\Gamma \vdash b :: \text{Bool}}$$

$$\frac{\Gamma \vdash e_1 :: \text{Int} \qquad \Gamma \vdash e_2 :: \text{Int}}{\Gamma \vdash e_1 + e_2 :: \text{Int}}$$

$$\frac{(x, T) \text{ in } \Gamma}{\Gamma \vdash x :: T}$$

What should this expression's type be?

```
let x = 3 in
  let y = 4 in
    x + y
```

:: Int

Types are a way to classify program expressions according to what they compute.

$$\frac{\Gamma \vdash e_1 :: T_1 \qquad (x, T_1) : \Gamma \vdash e_2 :: T_2}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 :: T_2}$$

The type of a let-expression is the type of its body in a type environment that's been extended with the type of the variable that the let-expression binds.

Lecture quiz access code:

Brent