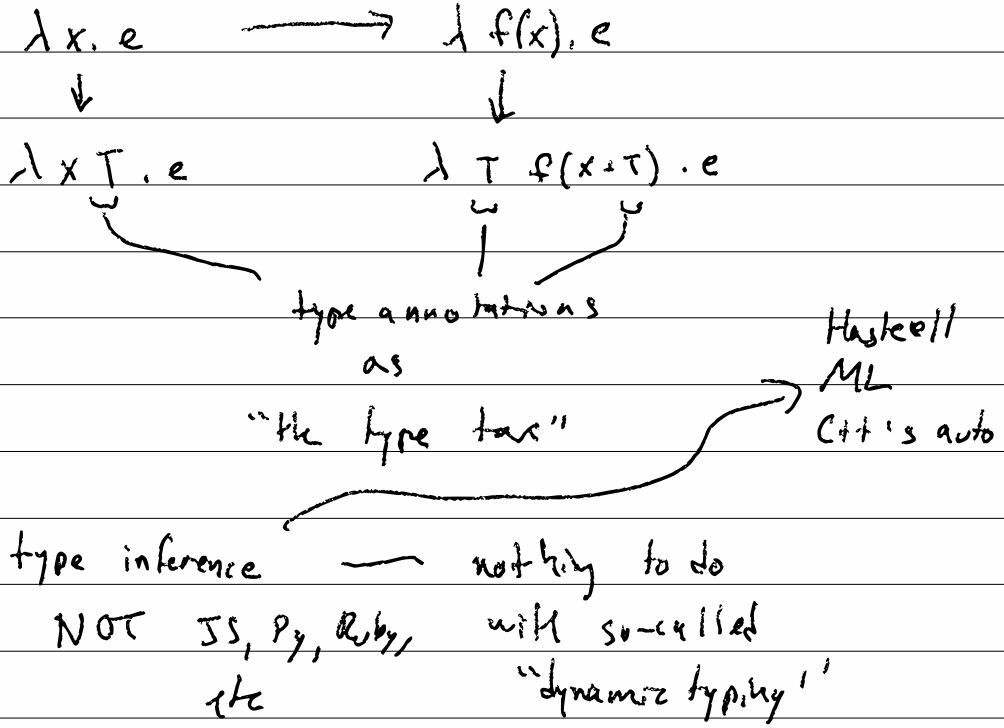
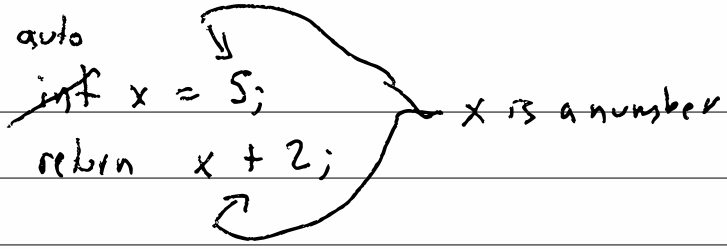
26-1


λx. e
λ f(x). e
λ x T . e
λ T f(x:T) . e
type annotations
as
"the type tax"
Haskell
ML
C++'s auto
type inference
nothing to do
with so-called
"dynamic typing"
NOT JS, Py, Ruby, etc

28-2 /



auto x = f(2); int f (int y);

use the programs to figure out what types variables should have ... do

263) similar to solving a system of equations.

$$\begin{aligned} x + y &= 10 \\ 4y &= 20 \\ 2y &= 15 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x + y &= 10 \\ y &= 5 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x + 5 &= 10 \\ y &= 5 \end{aligned}$$

$$\downarrow$$

$$x = y = 5$$

26-4/ auto x = 5; ← x is a num

printf(x, 2); ← x is a string

over-constrained
inconsistent
or
untyped

z + x + y = 10
y = 20
⇒
x + z = 5
y = 5

under-constrained
or
"polymorphic"

auto x = 5;
auto y;
return x + 2;

26-5 / type judgement that returns two things:
- first, a type
- second, a set of constraints
- third, a ~~list~~ set of variables used

next, we'll write an algorithm for solving such systems.

$\vdash e : T, \{\text{constraints}\}, \{\text{vars}\}$

$\vdash \sqcup : X, \{Num \to X = Num \to Num, Y = Z, Z = Num\}, \{X, Y, Z\}$

26-6/ $e = v \mid x \mid e\ e \mid \text{if } e\ e\ e$

$v = \lambda x.e \mid b$

$\Delta : b \Rightarrow T$

$T = B \mid T \Rightarrow T \mid X$

$\mathcal{X} = \mid \mathcal{X}, T = T$

$\Gamma = \mid \Gamma, x : T$

$\Gamma \vdash e : T, \mathcal{X}, \vec{\mathcal{X}}$

26-7/

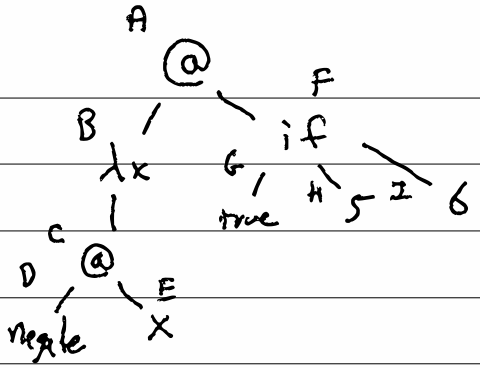
$$\frac{}{\Gamma \vdash b : \Delta(b), \emptyset, \emptyset} \qquad \frac{}{\Gamma \vdash x : \Gamma(x), \emptyset, \emptyset}$$

$$\frac{\Gamma \vdash f : T_f, \mathcal{X}_f, V_f \qquad \Gamma \vdash a : T_a, \mathcal{X}_a, V_a}{\Gamma \vdash f\ a : T_r,\ \mathcal{X}_f \cup \mathcal{X}_a \cup \{T_f = T_a \Rightarrow T_r\},\ V_f \cup V_a \cup \{T_r\}}$$

$$\frac{\Gamma \vdash e_c : T_c, \mathcal{X}_c, V_c \qquad \Gamma \vdash e_t : T_t, \mathcal{X}_t, V_t \qquad \Gamma \vdash e_f : T_f, \mathcal{X}_f, V_f}{\Gamma \vdash \text{if } e_c\ e_t\ e_f : T_t,\ \mathcal{X}_c \cup \mathcal{X}_t \cup \mathcal{X}_f \cup \{T_c = \text{Bool}, T_t = T_f\},\ V_c \cup V_t \cup V_f}$$

$$\frac{\Gamma[x \mapsto \not{X}_d] \vdash e : T_r, \mathcal{X}_r, V_r}{\Gamma \vdash \lambda x.\, e : \not{X}_d \Rightarrow T_r, \mathcal{X}_r, V_r \cup \{\not{X}_d\}}$$

26-8/ ((λx. (negate x))
(if true 5 6))



A/ B = F → A

B/ ~~F~~ = X → C

C/ D = E → C

D/ D = Num → Num

E/ E = X

F/ G = Bool
H = I
F = H

G/ G = Bool

H/ H = Num

I/ I = Num

$B = F \to A \Rightarrow B = X \to A \Rightarrow B = X \to C \Rightarrow B = X \to Num \Rightarrow B = N \to N$

26.9 $B = X \to C \Rightarrow F \to A = X \to C \Rightarrow F = X$ ($F = Num$) $\quad A = C \Rightarrow A = Num$

$D = E \to C \Rightarrow D = Num \to C \Rightarrow D = Num \to Num$

$D = Num \to Num \Rightarrow E \to C = N \to N \Rightarrow$ ($E = Num$, $C = Num$)

$E = X \Rightarrow Num = X \Rightarrow X = Num$

$G = Bool$

$H = I \Rightarrow H = Num$

$F = H \Rightarrow X = H \Rightarrow Num = H \Rightarrow Num = I \Rightarrow I = Num$

$G = Bool \Rightarrow Bool \to Bool \Rightarrow$

$H = Num \Rightarrow I = Num \Rightarrow Num = Num \Rightarrow$

$I = Num \Rightarrow Num = Num \Rightarrow$

$B = Num \to Num$	$A = Num$	$E = N$	$X = N$	$H = N$
$F = Num$	$D = N \to N$	$C = N$	$G = B$	$I = N$

26-10/ Solve : Constraints × Constraints → Constraints

solve subst ∅ = subst

solve subst (T=T) : cs = solve subst cs

solve subst (X=T) : cs = solve subst' cs'

where subst' = (X=T) : subst[X←T]

cs' = cs[X←T]

where X ∉ T

solve subst (T=X) : cs = solve subst (X=T) : cs

solve subst (A⇒B = P⇒Q) : cs =

solve subst (A=P) : (B=Q) : cs

∅ ⊢ e : T, 𝒳, V ⇒

solve ∅ 𝒳 ⇒ subst[T]

26-11/ what about programs w/ no types?

negate false ⇒ B = C → A ⇒

B = Num → Num C → A = Num → Num

C = Bool C = Num

A = Num

Num = Bool

↙

unsolvable equations

26+2) what about polymorphism?

$(\lambda x.x)\ 5 \Rightarrow B = D \Rightarrow A \Rightarrow B = Num \to Num$

$B = X \to C \quad C = Num$

$C = X \quad D = Num$

$D = Num \quad A = Num$

if $((\lambda x.x)\ true)$

$((\lambda x.x)\ 5)$

6

$(\lambda x.x) \Rightarrow A = X \to B \Rightarrow A = X \to X$

$B = X \quad B = X$

26-13/ let id = $\lambda x.x$ in

if (id true) $\Rightarrow$ id = Bool $\Rightarrow$ Bool

(id 5) $\Rightarrow$ id = Num $\Rightarrow$ Num

6

type error

let-polymorphism $\rightarrow$ "value restriction" of ML

$$\frac{\Gamma \vdash e_b[x \leftarrow e_x] : T, \mathcal{K}, V}{\Gamma \vdash \text{let } x = e_x \text{ in } e_b : T, \mathcal{K}, V}$$

26-14 / performance

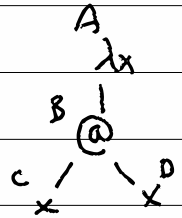
type-checking =

constraint generation $O(n)$

+ constraint solving $O(n^3)$

$= O(n^3)$

$(\lambda x.\ x\ x)$



"occurs check"

$A = X \Rightarrow B$

$C = D \Rightarrow B$

$C = X$

$D = X$

$\Rightarrow$ $A = X \Rightarrow B$

$X = X \Rightarrow B$

$C = X$

$D = X$

violates an assumption