Pairs/Data/Box

24-1/ $e = \ldots \mid \text{pair } e\ e \mid \text{fst } e \mid \text{snd } e$
$\mid \text{inl } e\ T \mid \text{inr } T\ e \mid \text{case } e \text{ of inl } x \Rightarrow e$
$\text{or inr } x \Rightarrow e$
$\mid \text{box } e \mid \text{setbox } e\ e \mid \text{unbox } e$

$T = \ldots \mid T \times T \mid T + T \mid \text{Box}(T)$

$$\frac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2}{\Gamma \vdash \text{pair } e_1\ e_2 : T_1 \times T_2} \qquad \frac{\Gamma \vdash e : T_1 \times T_2}{\Gamma \vdash \text{fst } e : T_1} \qquad \frac{\Gamma \vdash e : T_1 \times T_2}{\Gamma \vdash \text{snd } e : T_2}$$

$$\frac{\Gamma \vdash e : T_1}{\Gamma \vdash \text{inl } e\ T_2 : T_1 + T_2} \qquad \frac{\Gamma \vdash e : T_2}{\Gamma \vdash \text{inr } T_1\ e : T_1 + T_2}$$

$$\frac{\Gamma \vdash e_1 : T_1 + T_2 \quad \Gamma[x \mapsto T_1] \vdash e_2 : T_3 \quad \Gamma[y \mapsto T_2] \vdash e_3 : T_3}{\Gamma \vdash \text{case } e_1 \text{ of inl } x \Rightarrow e_2 \text{ or inr } y \Rightarrow e_3}$$

24-2/

$$\frac{\Gamma \vdash e : T}{\Gamma \vdash \text{box } e : \text{Box } T} \qquad \frac{\Gamma \vdash e : \text{Box } T}{\Gamma \vdash \text{unbox } e : T}$$

$$\frac{\Gamma \vdash e_1 : \text{Box } T \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash \text{setbox } e_1\ e_2 : \text{Unit}}$$

24-3/ untyped : $\lambda x.x$

$(((\lambda x.x)+)\ ((\lambda x.x)\ 5))\ 5 \Rightarrow 10$

$\lambda x:(N \to N \to N).x$ $\quad$ $\lambda x:N.x$

Java and C++

Generics $\quad$ Templates

Id < Image >
Id < Cat >
Id < Num >

class Id <X> {
public
X thing;
Id (X t) { this.thing = x; } }

24-1/ Polymorphism

$$e := \ldots \mid \Lambda X.\, e \mid e\langle T \rangle$$
$$T := \ldots \mid \forall x.\, e \mid X$$

untyped	typed
$\lambda x.x$	$\Lambda \alpha.\, \lambda\, \alpha\, f\, (x : \alpha).\, x$
	$: \forall \alpha.\, \alpha \Rightarrow \alpha$

OLD $\Gamma = \omega \mid \Gamma, X : T$

NEW $\Gamma = \ldots \mid X$

24-5/

$$\frac{\Gamma, X \vdash e : T}{\Gamma \vdash \Lambda X.\, e : \forall X.\, T} \qquad \frac{\Gamma \vdash e : \forall X.\, T_2}{\Gamma \vdash e\ \langle T_1 \rangle : T_2[X \leftarrow T_1]}$$

$$\frac{\vdash id\ \langle num \rangle\ 5 : Num}{\vdash 5 : Num \qquad \dfrac{\dfrac{\dfrac{\vdash id\ \langle num \rangle : \alpha \to \alpha[\alpha \leftarrow Num] = Num \to Num}{\vdash id : \forall \alpha.\, \alpha \to \alpha}}{\alpha \vdash \lambda \alpha\, f(x:\alpha), x : \alpha \to \alpha}}{\alpha, x:\alpha \vdash x : \alpha}}$$

24-6)

$v = \ldots \mid \Lambda X.\, v$

$E = \ldots \mid \Lambda X.\, E \mid E\langle T \rangle$

$E[(\Lambda X.\, v)\langle T \rangle] \rightarrow E[v[X \leftarrow T]]$

$id\langle Num \rangle\ 5 = (\Lambda\alpha.\, \lambda x{:}\alpha.\, x)\langle Num \rangle\ 5$

$\rightarrow (\lambda x{:}Num.\, x)\ 5$

$\rightarrow 5$

let $id = \Lambda\alpha.\, \lambda x{:}\alpha.\, x$ in

if $(id\langle Bool \rangle\ true)\ (id\langle Num \rangle\ 3)\ 4$

24-7/ strategy 1:

Polymorphism assumes arguments are pointers

... thus we only compile once

Java generics, C#

Strategy 2:

Make unique copies for every type ... C++

```
let id_Bool = λx:Bool. x in
    id_Num = λx:Num. x in
if (id_Bool true) (id_Num 3) 4
```

24-8/ Parametric Polymorphism

f : $\forall \alpha.\ \alpha \to \alpha$

you call (f <Num> 42)

... what number comes back?

... must 42

g : Num → Num

g 42 ... no idea

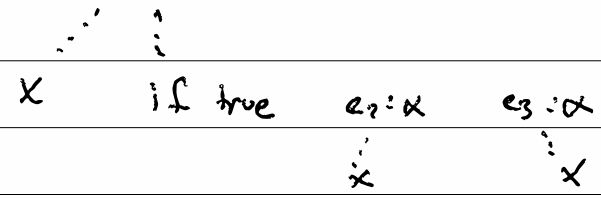
g x = 0 g x = x

g x = x+1 g x = x*4

24-9/

$$\frac{\vdash \Lambda\alpha.\ e_1 \ : \forall\alpha.\ \alpha \to \alpha}{\alpha \vdash \lambda x{:}\alpha.\ e_2 \ : \ \alpha \to \alpha}$$

$$\alpha, x{:}\alpha \vdash e_2 \ : \ \alpha$$



polymorphism provides a security channel

$$map : \forall\alpha.\forall\beta.\ (\alpha \to \beta) \to List\langle\alpha\rangle \to List\langle\beta\rangle$$

24-10/ Java / C++ / Typed Racket
don't provide this

```
class Id<X> {
  X t;   Id(X t) { this.t = t; }
  X get() {
    if (t instanceof Cat) {
      return Garfield; }
    else { return t; } } }
```

But Haskell / ML do provide this