21-3/

m = ... | (fix m)

E = ... | (fix E)

T = num | bool | T→T

E[(fix (λX:T. m))]
⟶ E[m[X ⟵ (fix (λX:T. m))]]

fixed-point of a fun F is a value x_0 s.t. $F\ x_0 = x_0$

$$\frac{\Gamma \vdash m \vdash ((T_1 \to T_2) \to (T_1 \to T_2))}{\Gamma \vdash (\text{fix } m) : (T_1 \to T_2)}$$

```
int f(int n){
  f(n);
}
```

((fix (λf: (num→(num→num)). λn:num. (f n))) 0)

E = ([] 0) [(fix m)]
↦ [λn:num. ((fix m) n)]
E = [] [(λn:num. (fix m) n) 0]
↦ [(fix m) 0]

∀T. ∅ ⊢ (fix (λX:T. X)) : T

2-4/ Data — Pairs

$M = \ldots \mid (\text{pair } M\ M) \mid (\text{fst } M) \mid (\text{snd } M)$

$V = (\text{pair } V\ V) \mid \ldots$

$E = \text{pair } E\ M \mid \text{pair } V\ E \mid \text{fst } E \mid \text{snd } E$

$T = \ldots \mid (T \times T)$

$$E[(\text{fst } (\text{pair } V_1\ V_2))] \longrightarrow E[V_1]$$
$$E[(\text{snd } (\text{pair } V_1\ V_2))] \longrightarrow E[V_2]$$

$$\frac{\Gamma \vdash e_1 : (T_1 \times T_2)}{\Gamma \vdash (\text{fst } e_1) : T_1} \qquad \frac{\Gamma \vdash e_1 : (T_1 \times T_2)}{\Gamma \vdash (\text{snd } e_1) : T_2} \qquad \frac{\Gamma \vdash e_i : T_i}{\Gamma \vdash (\text{pair } e_1\ e_2) : T_1 \times T_2}$$

Data — unions (dog or cat)

$M = \ldots \mid (\text{inL } M) \mid (\text{inR } M) \mid (\text{match } M\ (\lambda X_1.N_1)\ (\lambda X_2.N_2))$

$V = (\text{inL } V) \mid (\text{inR } V)$

$E = (\text{inL } E) \mid (\text{inR } E) \mid (\text{match } E\ M\ M)$

$T = \ldots \mid (T + T)$

$$\frac{\Gamma \vdash e_1 : T_1}{\Gamma \vdash (\text{inL } e_1) : T_1 + T_2} \qquad \frac{\Gamma \vdash e_2 : T_2}{\Gamma \vdash (\text{inR } e_2) : T_1 + T_2}$$

$$\frac{\Gamma \vdash M : T_1 + T_2 \quad \Gamma \vdash N_1 : T_1 \to T \quad \Gamma \vdash N_2 : T_2 \to T}{\Gamma \vdash (\text{match } M\ N_1\ N_2) : T}$$

$$E[(\text{match } (\text{inL } V)\ M\ N)] \mapsto E[M\ V]$$
$$E[(\text{match } (\text{inR } V)\ M\ N)] \mapsto E[N\ V]$$

$M = \ldots \mid (\text{useL } M\ M)$

$$\frac{\Gamma \vdash e_1 : T + T' \quad \Gamma \vdash e_2 : T \to T''}{\Gamma \vdash (\text{useL } e_1\ e_2) : T''}$$

$E[(\text{useL } (\text{inL } V)\ M)] \mapsto E[M\ V]$

$(\text{useL } (\text{inR } V)\ M)$ is stuck

22-1

Identity fun in ISWIM

$\lambda x.\ x$

In Typed-ISWIM:

$\lambda x:num.\ x$

$\lambda x:bool.\ x$

$\lambda x:(num \to bool).\ x$

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

value application

Poly-ISWIM (polymorphic – many shaped)

$\Lambda\alpha.\ \lambda x:\alpha.\ x$

normal fun – value abstraction

type abstraction $(\forall A.\ A \to A)$

$M = x \mid \ldots \mid \lambda x:T.M \mid (M\ M) \mid \Lambda\alpha.M \mid M[T]$

$T = \ldots \mid \alpha$

C++ / Java

```
class List<X> {
    X elem;
    List<X> rest;
    ...
    int f(int x) { return x + X; }
    main { List<int> l = ... }
}
```

$\Lambda\alpha.\ 5+6$

$(\Lambda\alpha.\ \lambda x:\beta.\ x)$

$\Lambda\alpha.\ (vector\ (\lambda x:\alpha.\ x)\ (\lambda y:nt.\ \lambda x:\alpha.\ y))$

$M = \ldots \mid \Lambda A.m \mid m[T]$

$V = \ldots \mid \Lambda A.m$

$T = \ldots \mid A \mid \forall A.T$

$E = \ldots$

$E[(\Lambda A.m)[T]] \mapsto E[m[A \leftarrow T]]$

$\Gamma = \varepsilon \mid \Gamma, x:T \mid \Gamma, A$ — records valid type variables

$$\frac{\Gamma, A \vdash M : T}{\Gamma \vdash (\Lambda A.m) : \forall A.T}$$

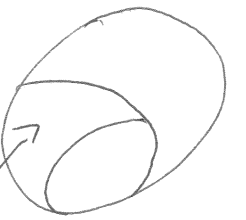
$$\frac{\Gamma \vdash M : \forall A.T' \qquad \Gamma \vdash T}{\Gamma \vdash M[T] : T'[A \leftarrow T]}$$

$$\frac{\Gamma, x:T \vdash M : T' \qquad \Gamma \vdash T}{\Gamma \vdash (\lambda x:T.M) : T \to T'}$$

$\Gamma \vdash A$ iff $\Gamma(A)$

12-2

$$\frac{\dfrac{\dfrac{\dfrac{(\Lambda\alpha.\ \lambda x{:}\alpha.\ x+5) : \forall\alpha.\ \alpha \to num}{\emptyset\alpha \vdash \lambda x{:}\alpha.\ x+5 : \alpha \to num}}{\emptyset, \alpha, x{:}\alpha \vdash x+5 : num} \qquad \dfrac{\emptyset, \alpha, x{:}\alpha \vdash x : num}{\alpha = num}\ \times \qquad \emptyset, \alpha, x{:}\alpha \vdash 5 : num\ \checkmark}$$



$((\Lambda\alpha.\ \lambda x{:}\alpha.\ x+5)\ [num]\ 7) \Rightarrow^{*} 12$

$((\Lambda\alpha.\ \lambda x{:}\alpha.\ x+5)\ [bool]\ true) \Rightarrow^{*} true + 5$ stuck

parametricity — a promise to not look at polymorphic types (enforced by a lang that can't)

T	#s
bool	∞ (infinity)
num	∞
$\forall\alpha.\ \alpha \to \alpha$	1*

* modulo eta $\lambda x.\ ((\lambda y.\ y)\ x) =_{eta} \lambda x.\ x$

$\forall\alpha,\beta.$ l f

map : (List α) × ($\alpha \to \beta$) → (List β)

f can only be called with αs from l

```
(define (map l f)
  (cond [(empty? l) empty]
        [(number? (first l)) (list (f 7))]
        [else (cons (f (first l))
                    (map (rest l) f))]))
```

Violates parametricity

α

Racket, Clojure
Scheme, JS, Python

22-3/

$E[(\Lambda A.m)[T]] \mapsto E[m[A \leftarrow T]]$

```
( let id = ΛA. λx:A. x in
    (+  (id[num] 5)                          List<int>
        (id[num] 7)))
```

$\mapsto$

```
(+  ((ΛA. λx:A. x)[num] 5)
    ((ΛA. λx:A. x)[num] 7))
```

$\mapsto$

```
(+  ((λx:num. x) 5)        "     )
```

C++ — Templates —

Java — Generics —

Standard ML — exactly this —

4bs 80bs 4k

List<int>, List<Dog>, List<Image>

monomorphization → C++

3 implementations

Specialized to each shape

− many binaries, slow compiles

− more i-cache needed

+ compact mem, no deref

List<int*>

List<void*>

copies code

parametric: No, because of ptrs

List<int> {

:

}

→ Java

1 implementation

compatible w/ all shapes

+ one binary, faster compiles

+ less i-cache

− all data is beyond a pointer

8bs of mem, mem deref

one impl

that is generic

List<X>

add(X ex) {

:

tell what ex is?

} : instanceof

X