25-1

$(\forall A, T)$

A is unknown

Set of Numbers

- ↳ Array/List of members
- ↳ Map of (member → bool)
- ↳ Tree of members
- ↳ n-bit bitmask

Interface
Virtual Abstract Class

type of set
mem : set × num → bool
add : set × num → set
rem : set × num → set
mt : set

```
class U {
  void f ( SetOfNumbers s ) {
      s.mem (s.add (s.mt, 1), 1) == #true } ;
```

Interface (Representation^{R})
$$= (R \times (R \times Num \to Bool) \times (R \times Num \to R))$$

BitClass : Interface (R = Num)
$$= (\underset{\emptyset}{Num} \times \underbrace{(Num \times Num \to Bool)}_{(1 \ll n^{k}\ \&)} \times \underset{n \mid (1 \ll^{k})}{(Num \times Num \to Num)})$$

MapClass : Interface (R = Map)
$$= (Map \times (Map \times Num \to Bool) \times (Map \times Num \to Map))$$

$$f : \forall R.\ Interface(R) \to Bool$$
$$: \forall R.\ (R \times (R \times Num \to Bool) \times (R \times Num \to R)) \to Bool$$

((fst MapClass) map-count) ⇒ Num

25-2/

R = representation

Library = Interface (concrete R)
Client = ∀R. (Interface (R) → Answer)
└ trusts clients

Client(A) = ∀R. (Interface (R) → Answer)   doesn't trust clients
Library' = ∀A. (Client (A) → A)

MapClass client :=
    (client [R = Map] (empty-map × map-member? × map-add))
Program :=
    (MapClass [A = Bool] (λ si.
        ((snd si) ((third si) (fst si) 1) 1)))
    ;; (set-member? (set-add set-mt 1) 1)

existential type	universal type
∃A. T	∀A. T

MapClass : ∃R. Interface (R)
ListClass : ∃R. Interface (R)
BitClass : ∃R. Interface (R)   → unpack [R] SetInstance from MapClass in code that calls set funcs

T = ... | ∃A. T
m = ... | pack [A = T] m as T'

name it this 3
hide this 2
run this and get a value 1
new exposed type 4

→ pack [A = Map] MapClass as (A × (A×Num → Bool) × (A×Num → A))

| unpack [A] X from M in M'

expose the hidden type as A 2
name it this 3
run this
run this with x bound to expose value 4

$V = \ldots \mid \text{pack}\,[A=T]\ V \text{ as } T'$

$E = \ldots \mid \text{pack}\,[A=T]\ E \text{ as } T'$

$\mid \text{unpack}\,[A]\ X \text{ from } E \text{ in } M$

$$E\left[(\text{unpack}\,[A]\ X \text{ from } (\text{pack}\,[A'=T']\ V \text{ as } T) \text{ in } M\right]$$
$$\longmapsto E\left[M\,[X \leftarrow V]\,[A \leftarrow T']\right]$$

$$\frac{\Gamma \vdash M : T[A \leftarrow T'] \qquad \Gamma \vdash T' \qquad \Gamma, A \vdash T}{\Gamma \vdash \text{pack}\,[A=T']\ M \text{ as } T : \exists A.T}$$

$$\frac{\Gamma \vdash M_1 : \exists A'.T' \qquad \Gamma, X : T'[A' \leftarrow A] \vdash M_2 : T \qquad A \notin FV(T)}{\Gamma \vdash \text{unpack}\,[A]\ X \text{ from } M_1 \text{ in } M_2 : T}$$

unpack $[A]$ X from
  $(\text{pack}\,[A' = Num]\ 17 \text{ as } A') \qquad : \exists A'.A'$
in
  $X \qquad : A$

$\forall \beta \ldots \quad \lambda x : T_\beta \quad \ldots \quad \exists \alpha \ldots \quad T_{\alpha,\beta}$

β is the imported ~~abstracte~~ abstract types

T_β is the modules imported that use β

α are yourexported abstract types

$T_{\alpha,\beta}$ are the exported values (mention α or β)

25-4

$$
\begin{aligned}
T = \;& \mathrm{Num} \mid \mathrm{Bool} \mid \text{base-types} \\
\mid\;& T \to T \\
\mid\;& T \times T \qquad (T \wedge T) \\
\mid\;& T + T \qquad (T \vee T) \\
\mid\;& \forall A.\, T \\
\mid\;& A \\
\mid\;& \exists A.\, T \\
\mid\;& \mu A.\, T
\end{aligned}
$$

$$
\begin{tikzcd}
\text{Types} \arrow[r, leftrightarrow] \arrow[d, shift right, "n"'] & \text{Propositions} \arrow[d, shift right, "n"'] \\
\text{Program} \arrow[r, leftrightarrow] \arrow[u, shift right] & \text{Proofs} \arrow[u, shift right]
\end{tikzcd}
$$

Curry-Howard Isomorphism

$\boxed{\text{COMP 3010 - 202 McCarthy}}$