Subtyping

~~class Animal~~

```
typedef struct { int x;  int y; }  2dpt;

int bigger ( 2dpt  p )  { return max(p.x, p.y); }

typedef struct { int x; int y; int z; }  3dpt;
   ...     3dpt p = ... ;
           bigger(p);

int biggera ( int *p )  { return max (p[0], p[1]); }
```

---

f : D → R

(f a)

→ must be EXACTLY D (original rule)

→ a must be COMPATIBLE (subtyping)

(bigger)

suppose $D = (\boxed{int, int}, str)$

$A = (int, int, int)$

$M = \ldots \mid \langle L = M, \ldots, L = M \rangle \mid M.L$

$L \in$ some set of labels (records)

$\langle x = 3, y = (+\ 3\ 4) \rangle . y$

$V = \ldots \mid \langle L = V, \ldots, L = V \rangle$

$E = \ldots \mid E.L \mid \langle L = V, \ldots, L = E, L = M, \ldots \rangle$

$E[\langle L_0 = V_0, \ldots, L_i = V_i, \ldots, L_n = V_n \rangle . L_i]$

$\longmapsto E[V_i]$

26-2/ $T = \ldots \mid \langle L : T, \ldots, L : T \rangle$

$\langle x : Num,\ y : Num,\ get : (Bool \to String) \rangle$ ∈ T

$\mu Class.\ \langle field : T,\ method : Class \times Args \to Rng \rangle$

$$\frac{\Gamma \vdash M_0 : T_0 \quad \ldots \quad \Gamma \vdash M_n : T_n}{\Gamma \vdash \langle L_0 : M_0, \ldots, L_n : M_n \rangle : \langle L_0 : T_0, \ldots, L_n : T_n \rangle}$$

$$\frac{\Gamma \vdash M : \langle L_0 : M_0, \ldots, L_i : T_i, \ldots, L_n : T_n \rangle}{\Gamma \vdash M.L_i : T_i}$$

2dpt : $\langle x : Num, y : Num \rangle$ = $\langle y : Num, x : Num \rangle$

3dpt : $\langle x : Num, y : Num, z : Num \rangle$

3dpt $\underbrace{\text{is compatible with}}_{\text{a partial order}}$ 2dpt

Bool <: Bool

Num <: Num

$\boxed{<:}$ $\preccurlyeq:$ $=<:$ $<\!\!=:$

T <: T

3dpt <: 2dpt

X ⊆ Y

$\langle L_0 : T_0, \ldots, L_n : T_n \rangle$
$<: \langle L'_0 : T'_0, \ldots, L'_m : T'_m \rangle$
iff. $\{L'_0 : T'_0, \ldots, L'_m : T'_m\}$
$\subseteq \{L_0 : T_0, \ldots, L_n : T_n\}$

(addendum:
$T'_i <: T_i$
for all matching
i)
pg 162

OLD

$$\frac{\Gamma \vdash f : D \to R \qquad \Gamma \vdash a : D}{\Gamma \vdash (f\ a) : R}$$

NEW

$$\frac{\Gamma \vdash f : D \to R \qquad \Gamma \vdash a : D'}{\Gamma \vdash (f\ a) : R}$$

1. D <: D'

$\boxed{2.\ D' <: D}$

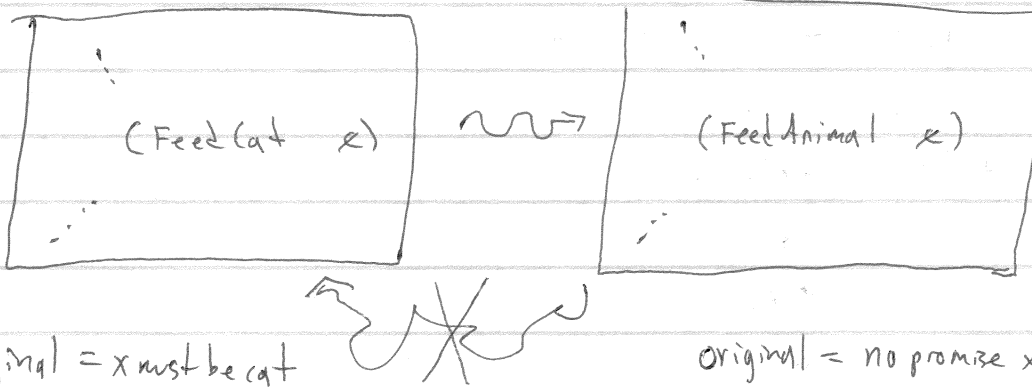
26-3

$x : D_x \to R_x$
$y : D_y \to R_y$

when is x compatible with y?
ie when can I replace all occurences of y with x and the program works?

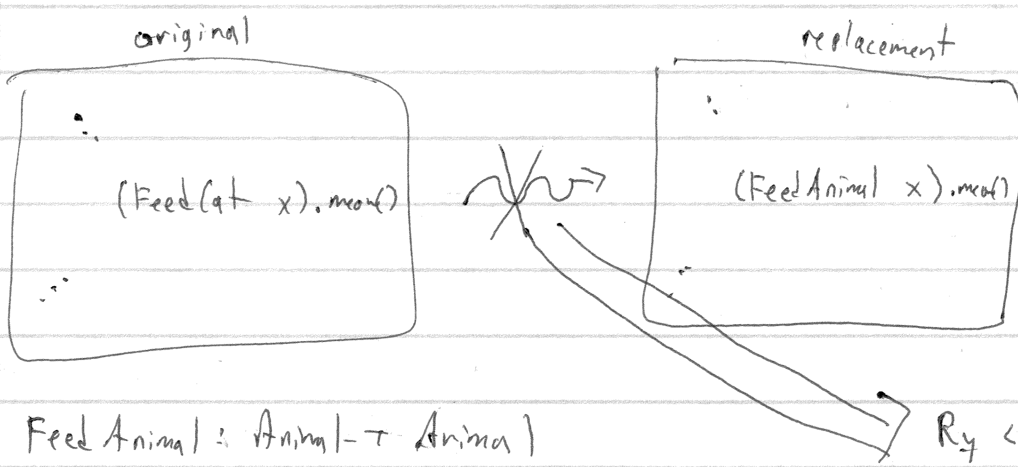
Liskov Substitution Principle
Barbara Liskov

FeedAnimal : Animal $\to$ Bool
FeedCat : Cat $\to$ Bool



original = x must be cat
original = no promise x is a cat

$$\boxed{\begin{array}{ll} D_y <: D_x & \text{[contra-variance]} \\ R_x <: R_y & \text{[co - variant]} \\ \hline D_x \to R_x <: D_y \to R_y & \\ \text{Animal} \qquad\qquad \text{Cat} & \end{array}}$$



$R_y <: R_x$

FeedAnimal : Animal $\to$ Animal
FeedCat : Cat $\to$ Cat

26-4

class List < X implents Ordered >

F-bounded Polymorphism

constrain types in $\forall$s

$\forall A.\ T$

$\forall A <: T\ ,\ T'$

List<X>

$\forall X.\ (\ \ldots \ )$

(FeedAnimal: $\forall A <: \text{Animal}.\ A \to A$)

$(\forall X.\ X \to X)$

Semantics → why to have them
↳ λ-calculus

ISWIM - defined
consistent
s.r.
machines
efficient implementation

G.C. → tail calls

control (exceptions + threads)

mutation

types → make guarantees about programs
add feature
show type sys
show the next flaw

---

printf : string × stuff → int

printf("%d", x);

"%s", x

printf : (fmt : String) × F(fmt) → int

dependent type

Calculus of Constructions

CoC → Coq

F-omega