5-1

ISWIM

$M, N, L, K ::= X$
$\quad | \ \lambda X.M$
$\quad | \ M\ N$
$\quad | \ b$
$\quad | \ o^n\ M_1 \dots M_m$

$V, U ::= \lambda X.M$
$\quad | \ b$

$\beta : (\lambda X.M)\ N \to M[X \leftarrow N]$

$\beta_v : (\lambda X.M)\ V \to M[X \leftarrow V]$

$\Delta : (o^n\ b_1 \dots b_n) \to \delta(o^n, b_1, \dots b_n)$

$\delta : O \times B \dots \times B \to V$

$\to_v$ one-step w/ compatible closure

$\to\!\!\!\!\to_v$ tran, refl closure

$=_v$ sym closure

$v = \beta_v \cup \Delta$

$eval(M) ::= b \quad$ if $\quad M =_v b$
$\quad\quad\ \ fun \quad$ if $\quad M =_v \lambda X.N$

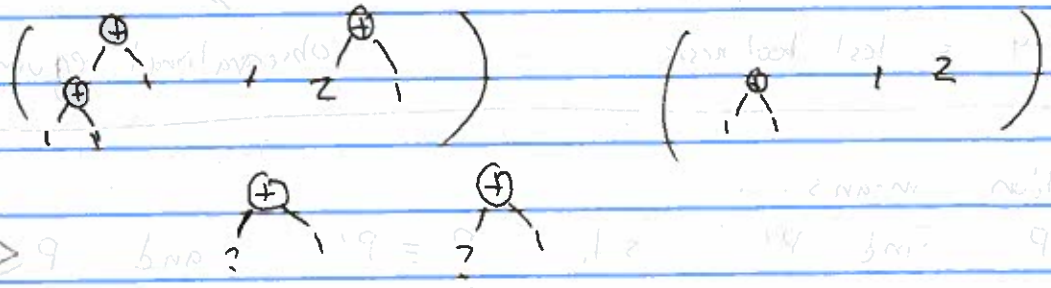
---

compatible closure of R $= cc(R)$

~~$(a,b) \in cc(R)$~~ if $(a \text{ prec } a,\ a \text{ prec } b) \in R$

$((1+1)+1,\ 2+1) \in cc(\text{Arith})$ $\quad ((1+1), 2) \in \text{Arith}$

"find a place to use the rule in a large program"



$C ::= ? \ | \ \lambda X.C \ | \ C\ N \ | \ M\ C \ | \ o^n\ M \dots C\ M \dots$

$(\lambda Y.(\lambda X.\ X+Y))\ (1+2) \to (\lambda Y.(\lambda X.\ X+Y))\ 3$

$(\lambda Y.(\lambda X.\ X+Y))\ ? \quad ? = (1+2) \to_v 3$

5-2 $C ::= \blacksquare \mid \lambda X.C \mid C\ N \mid M\ C \mid o^n\ M \ldots C\ M \ldots$

$C[M]$ = C where $\blacksquare$ is filled with M

$\blacksquare[M] = M$

$(\lambda X.C)[M] = \lambda X.\ C[M]$

$(C\ N)[M] = (C[M]\ N)$

$(M\ C)[M] = (N\ C[M])$

$(o^n\ M \ldots C\ N \ldots)[L] = (o^n\ M \ldots\ C[L]\ N \ldots)$

$$\frac{\exists C.\ M = C[L] \wedge N = C[K]}{M \not\rightarrow_v N}$$

"contexts" = C

ie programs w/ holes

"Does P do the right thing?"

$eval(P) = A$? (A is known)

"Is P the same as P'"

(same = want; P' = known is correct)

$\forall b.\ eval(P\ b) = eval(P'\ b)$

$eval(P) = eval(P')$?

$ffs_{want}(12) == ffs_{ref}(12)$?

$ffs_{want} == ffs_{ref}$

```
for (int i=0; i < INT_MAX; i++) {
  if(ffs_want(i) != ffs_ref(i)) exit(1);
```

contexts represent observations

$\blacksquare\ b$ = call func w/ b

$(\blacksquare\ b)\ d$ = call w/ b, call res w/ d

if $\blacksquare$ 3 4 = test boolness

$\forall C.\ eval(C[P]) = eval(C[P'])$

$\not\equiv$

observational equivalence $\equiv$

Optimization means...

given P find P' s.t. $P \equiv P'$ and $P \leq_m P'$

$\leq_m$: preference on programs (size, time, space)

if P doesn't terminate, iff $P \equiv \Omega$

$$\frac{M \equiv \Omega}{M\ N \equiv \Omega} \qquad \frac{N \equiv \Omega}{M\ N \equiv \Omega}$$

$(\lambda X.\ M)\ V \equiv M[X \leftarrow V]$

$(\lambda X.\ M)\ N \equiv (\lambda X.\ M)\ O$ iff $X \notin M$ and $N \not\equiv \Omega$

map-reduce

map: $(A \Rightarrow B) \times (\text{list } A) \Rightarrow (\text{list } B)$

reduce: $(A\ B \Rightarrow B) \times (\text{list } A) * B \Rightarrow B$

map f (map g l) ≡ map (f ∘ g) l

reduce (reduce) ≡ reduce $^{A(1,2)}$

map (reduce = ...

reduce (map = ...

---

consider the size of ≡ for diff languages

C: f ≡ ???

f(x) = 2 + x

```
C: int f(int x){
     ret x+2;
   }
```

ASM: ≡ is smaller than for C

```
asm: addq rax, $2
     ret
```

in C		asm
"x+2"	≡ "x=x+x+1+1"	"addq rax, $2"
	"y=x; y+=2;	"addq rbx, $2"
C[ ] = "int y= "	x=y"	"addq rax," C[ ] = ;

```
movq rbx, $3
movq rax, $1
addq rax, rbx
```