

2-1/

- $r$  "base relation" of semantics  $B$  - boolean
- $\rightarrow_r$  compatible closure of  $r$  w.r.t. the P grammar
- $\rightarrow^*_r$  refl & transitive closure of  $\rightarrow_r$
- $=_r$  symmetric closure of  $\rightarrow^*_r$
- eval  $r$  evaluation function
- drop the  $r$  if obvious

$$(f \times (f \times f)) = f$$

What makes a good semantics? What are semantics for?

They answer questions about programs:

- What is its answer? - is it computable? decidable? efficient?
- I know the answer, check it! - can it be done faster than naive
- Function-like?  $(\forall A_1, A_2. (P, A_1) \in r \wedge (P, A_2) \in r \Rightarrow A_1 = A_2)$
- Are two programs the same? ~~is~~  $eval(P_1) = eval(P_2)$ ?
- $\forall x. eval(P_1(x)) = eval(P_2(x))$ ?

For Bool, prove that  $(\forall B_0. eval_r(B_0) = A_1 \wedge eval_r(B_0) = A_2 \Rightarrow A_1 =_r A_2)$  function-like proof

$H_1: B_0 =_r A_1$   $H_2: B_0 =_r A_2 \Rightarrow A_1 =_r A_2$   $=_r$  is the sym, trans, refl, compatible, cls  $r$

transitive (sym  $H_1$ ,  $H_2$ )

$A_1 =_r B_0$   $B_0 =_r A_2$

sym  $H_1$ :  $A_1 =_r B_0$   $A_1 =_r A_2$

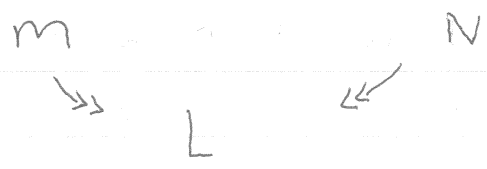
$r_3 (f \times B) \ r \ B \left( \begin{array}{l} \forall B_0. eval_r(B_0) = A_1 \\ \wedge eval_r(B_0) = A_2 \\ \Rightarrow A_1 = A_2 \end{array} \right)$

$\uparrow$   
no  $r$  meaning "objects are the same"

2-2/

$$M =_r N$$

$\Rightarrow$  not  $=_r$



$\rightarrow$   
only has  
refl + trans

$\rightarrow$   
includes  
sym

$\Rightarrow$  only applies in the  
"forward" direction  
"reduction"

If  $M$  ~~and~~ or  $N$  were T or F

then, they don't reduce

$$(t \times (f \times t)) \Rightarrow (f \times t)$$

$$T \Rightarrow T$$

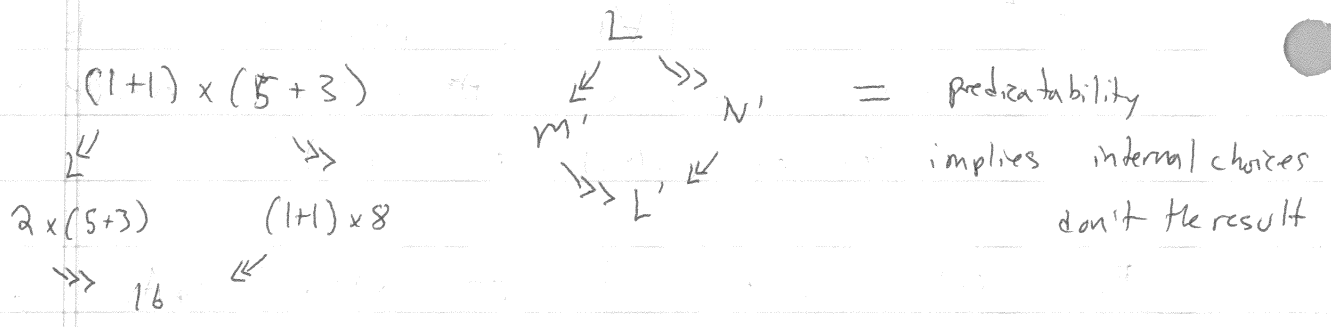
$$F \Rightarrow F$$

Church-Rosser:  $\forall M, N. \text{IF } M =_r N, \text{ exists } L,$

$$M \Rightarrow L \text{ and } N \Rightarrow L$$

Diamond Property: IF  $L \Rightarrow M'$  and  $L \Rightarrow N'$  then

exists  $L'$  where  $M' \Rightarrow L'$  and  $N' \Rightarrow L'$



$\forall P. \exists A. \text{eval}(P) = A$ , (There is always an answer)

— most languages don't have this because of infinite loops

$(\forall N. M \Leftrightarrow N \text{ and } \exists L. N \Leftrightarrow L.) = M \text{ is an infinite loop}$