2-1

$r$ "base relation" of semantics $B$ - boolean

$\rightarrow_r$ compatible closure of $r$ w.r.t. the P grammar

$\twoheadrightarrow_r$ / $\rightarrow^*_r$ refl + transitive closure of $\rightarrow_r$

$=_r$ symmetric closure of $\twoheadrightarrow_r$

eval$_r$ evaluation function

drop the $r$ if obvious

$$(t \;\; x \;\; (f \;\; x \;\; t)) = f$$

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? ~~the~~ $\text{eval}(P_1) = \text{eval}(P_2)$?

  $\forall x.\; \text{eval}(P_1(x)) = \text{eval}(P_2(x))$?

---

For Bool, prove that $\text{eval}_r(B_0) = A_x$ iff $B_0 =_r A_x$

$$\left(\begin{array}{l} \forall B_0.\;\; \text{eval}_r(B_0) =_r A_1 \\ \quad \wedge\; \text{eval}_r(B_0) =_r A_2 \\ \quad \Rightarrow\; A_1 =_r A_2 \end{array}\right)$$

function-like proof

$$\overset{H_1}{B_0 =_r A_1} \wedge \overset{H_2}{B_0 =_r A_2} \Rightarrow A_1 =_r A_2$$

$=_r$ is the sym, trans, refl, compatible, clos $r$

$H_1 : B_0 =_r A_1$

$H_2 : B_0 =_r A_2$

sym $H_1$ : $A_1 =_r B_0$

transitive (sym $H_1$, $H_2$)

$A_1 =_r B_0 \quad B_0 =_r A_2$

$A_1 =_r A_2$

$r_3$ $(f \;\; x \;\; B) \;\; r \;\; B$

$$\left(\begin{array}{l} \forall B_0.\;\; \text{eval}_r(B_0) = A_1 \\ \quad \wedge\; \text{eval}_r(B_0) = A_2 \\ \quad \Rightarrow\; A_1 = A_2 \end{array}\right.$$

no $r$ meaning "objects are the same"

2-2

$M =_r N$

$\twoheadrightarrow$ not $=_r$

only has refl + trans — includes sym

$M \twoheadrightarrow L \twoheadleftarrow N$

$\twoheadrightarrow$ only applies in this "forward" direction "reduction"

If M ~~and~~ or N were T or F then, they don't reduce

$T \twoheadrightarrow T$

$F \twoheadrightarrow F$

$(\lambda x (f\ x\ t)) \twoheadrightarrow (f\ x\ t)$

Church-Rosser: $\forall M, N.$ If $M =_r N$, exists $L$, $M \twoheadrightarrow L$ and $N \twoheadrightarrow L$

Diamond Property: If $L \twoheadrightarrow M'$ and $L \twoheadrightarrow N'$ then exists $L'$ where $M' \twoheadrightarrow L'$ and $N' \twoheadrightarrow L'$

$(1+1) \times (5+3)$

$\twoheadrightarrow 2 \times (5+3)$ and $\twoheadrightarrow (1+1) \times 8$

$2 \times (5+3) \twoheadrightarrow 16$ and $(1+1) \times 8 \twoheadrightarrow 16$

$L \twoheadrightarrow M'$, $L \twoheadrightarrow N'$, $M' \twoheadrightarrow L'$, $N' \twoheadrightarrow L'$

= predicatability implies internal choices don't the result

---

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

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

$(\forall N.\ M \twoheadrightarrow N$ and $\exists L.\ N \twoheadrightarrow L.)$ = M is an infinite loop