1-1

Theory of Programming Languages

Normative & Descriptive

- What <u>should</u> X be like
- What <u>IS</u> X like?

Theories answer questions... what Qs are relevant for PLs?

- How to make a sound? (what libraries & fns are available?)
- Is it high-level? (ie not assembly)
- OO? Procedral? (paradigm? category?)
- Is it compiled or interpeted?
- Types? Strong or weak?
- Community of users?
- Universal vs platform specific

BNF/CFG

AE = 0 | 1 | ...
| AE + AE
| AE × AE

AE ↓ AE

Semantics of the PL

↳ What does program P mean? → ~~want to run them~~

Ex. PL = "middle school arithmetic"

"1+1" "2×3" "3×4+5" ∈ PL

↓ ↓ ↓

2 6 17 meaning = ℤ

A good semantics is compositional

$M(P_1 + P_2) = F(M(P_1), M(P_2))$

$f = P_1$

$\forall x. \langle f() x() \rangle = ?$ → very general form of testing

$f(5) = 10$?

$\forall P.\ M(P) = interp(P)$

$\forall P_x.\ M_x(P_x) = M_y(compile_{x \to y}(P_x))$

Does C have a semantics?

prices * items = cost

not an l-value

The semantics maps programs to meaning

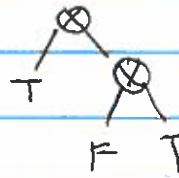
is a mathematical function relation

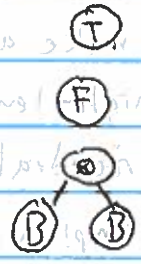
trees (as math objects)

math objects

$B = T \mid F \mid B \otimes B$

"$T \otimes F \otimes T$" $\in B$



$B =$ 

new TNode()

new FNode()

new Circ(B1, B2)

semantics $R : B \times V$

$T \; R \; \cup$ $\quad$ $F \; R \; \cap$

$\forall B.\quad F \otimes B \; R \; x \quad$ iff $\quad B \; R \; x$

meanings

$V = \cup \mid \cap$

$T \otimes T \; R$?

Operational semantics says that meanings are programs

i.e. $M = P \times P$

$1 + 1 + 2 \Rightarrow 2 + 2 \Rightarrow 4$