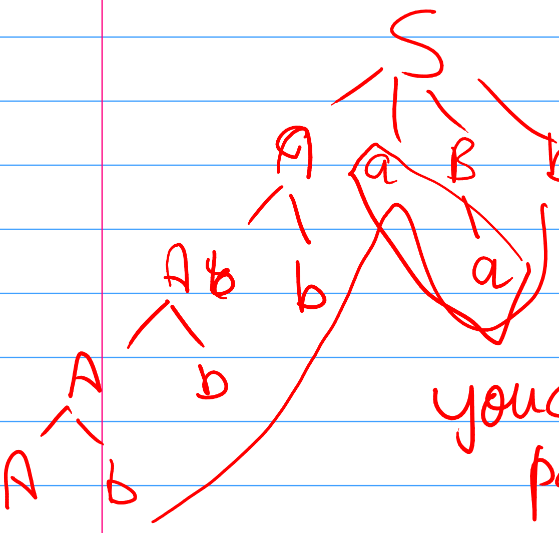
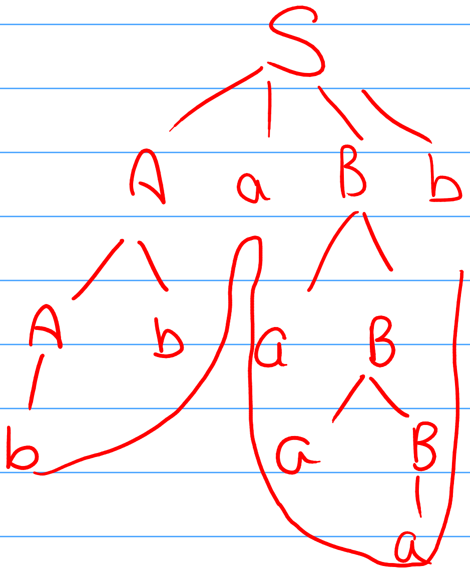
S → AaBb
A → Ab| b
B → aB| a

bbaaaa

S → AaBb
AbaBb
bbaBb
bbaaaab

bbbab





AaBb
AbaBb → AbbaBb
bbbaBb
invalid

you cant construct parse tree of invalid string

Left recursive grammar

E → E+T

Right recursive grammar

E → T+E

Extended BNF :

(1) Optional part of RHS →

<if_stmt> → if (<expr>) <statement> [else <st2>]

(2) Repitition

<idnt_list> → <identifier> {, <identifier>}

(3) ~~<Term> →~~ Alternation

<term> → <term> (*|/|%) <factor>

But in BNF

<term> → <term>*<factor> | <term>/<factor> |
<term> % <factor>

Attribute grammar

→ device used to describe the structure of a PL that can be described by a context-free grammar

→ Extension of CFG, an attribute grammar these extensions follow certain language rules to be described conveniently

eg : type compatibility

→ We cannot describe if a variable is declared before being used in BNF.

→ But we can check it by looking at the program

→ These kind of rules are called static semantic rules.

→ Attribute grammar can check both syntax and static semantics.

→ Features

↳ A(x) = Attributes

Synthetic attributes

Inheritance attributes

Intrinsic: Synthesized attributes of leaf nodes determined outside of parse tree

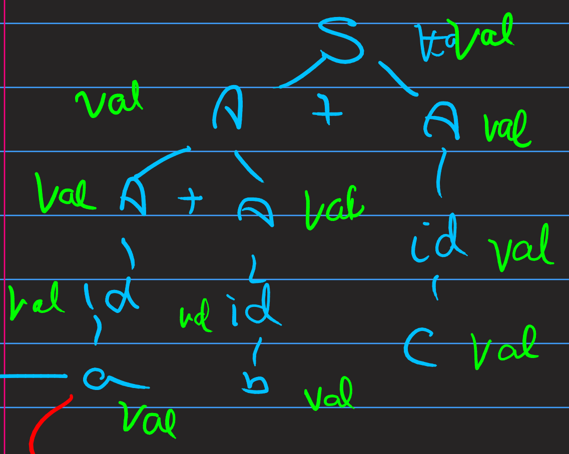
$S \rightarrow A+A$

$A \rightarrow id$

$id \rightarrow a|b|c$ $a+b+c$ is value

$a=5$, $b=3$, $c=2$

So if "5+3+2" is given

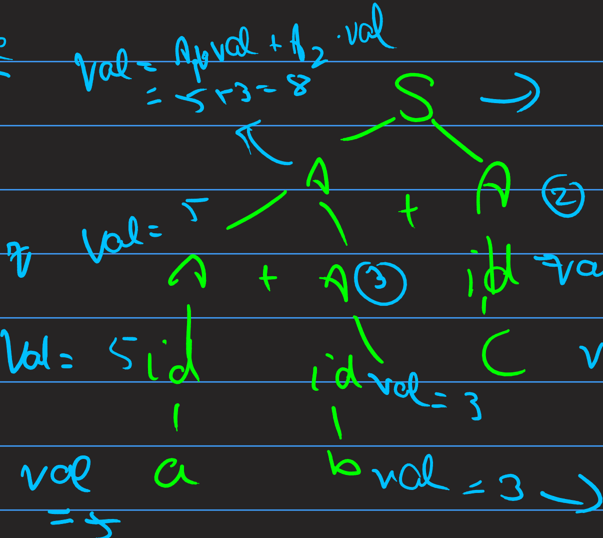


synthetic attribute

intrinsic ~~val~~ attribute

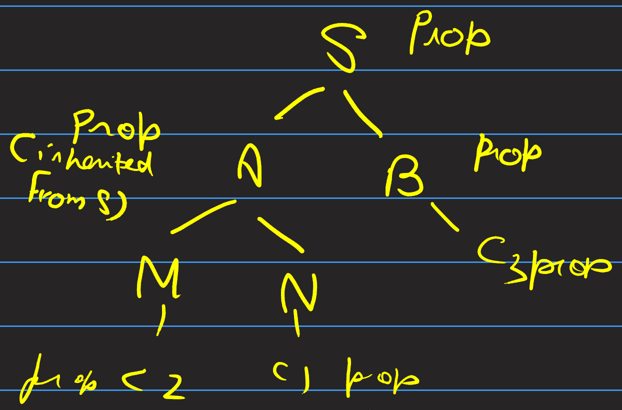
output parse tree

- id → a ⇒ id.val = a.val
- A → id ⇒ A.val = id.val
- A.val = A_1.val + A_2.val



S → S = A_1.val + A_2.val
= 8+2 = $\boxed{10}$

→ symbol table



S → AB | A.prop = S.prob/2
B.prop = S.prob/2
A → M | m.prop = A.prop

inherited attribute
(you need parent values)