UNIT-3

ONE SHOT VIDEO

T.A.F.L

MULTI ATOMS T

JOIN TELEGRAM FOR NOTES

CONTEXT FREE GRAMMER

- In formal language theory a context free grammar (CFG) is certain type of formal grammer
- A set of production rule that describle all possible string in a given formal language
- A CFG has 4 tuple

$G = \{V, T, P, S\}$

where V = Variable

T = Terminal

P = Set of production Rule

$\alpha \longrightarrow \beta$

where $\alpha \in V$ (Single variable)

$\beta \in (V+T)^*$ (β can be $\in$)

S = Starting Symbol

For ex :- $S \longrightarrow SAS$

$S \longrightarrow \in$

CONTEXT FREE LANGUAGE

The language generated by context free grammer is called context free language (CFL) which is accepted by PDA

Ex $S \longrightarrow aSb$
$S \longrightarrow ab$

or

$S \longrightarrow aSb$
$S \longrightarrow aaSb$
$S \longrightarrow aabb$

$L = \{a^n b^n \mid n \geq 1\}$

Derivation

The process of deriving a string is called as derivation.

Types of derivation: Generally there are two types of derivation

1] Left Most Derivation (LMD)
2] Right Most Derivation (RMD)

① LMD :- It is the process of deriving a string by expanding the left most non terminal at each step.

② RMD :- It is the process of deriving a string by expanding the right most non - terminal at each step.

Derivation tree : Derivation tree is the tree representation of the CFG.

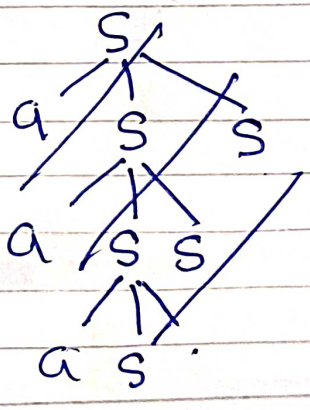
- **Ambigous grammar**

A CFG is said to ambigous if the grammar produces more than one parse tree.

Q Grammer G is given with the production $S \rightarrow aSS$, $S \rightarrow b$. compute the string $w = aababb$ with the LMD and RMD

Sol $S \rightarrow aSS$ $S \rightarrow b$

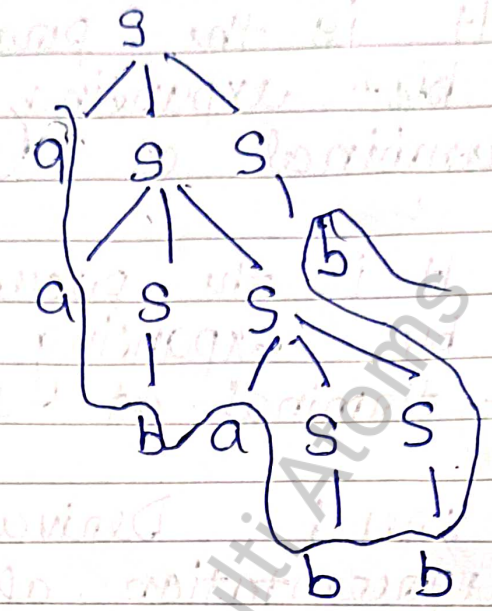
① LMD



• L.M.D

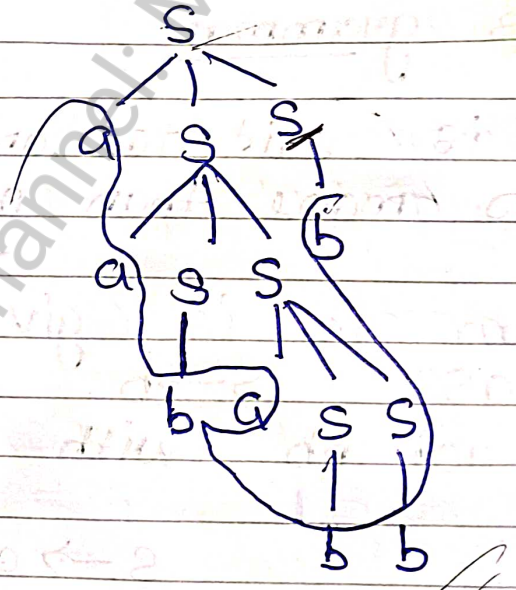
$S \rightarrow aSS$

$S \rightarrow b$



$w \rightarrow aababbb$ //

① R.M.D



Answer

Q check whether the grammar is ambigous or not.

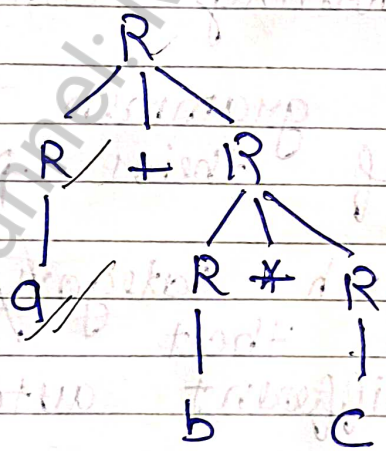
$R \rightarrow R+R / R-R / R*R / a/b/c$, obtain the string $w = a+b*c$

Ans

$R \rightarrow R+R / R-R / R*R / a/b/c$

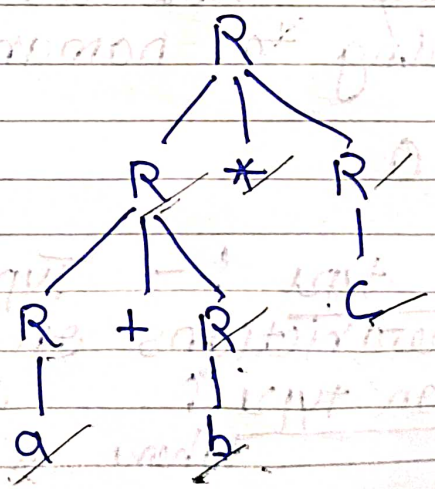
for ambigous ⟶ 2 · Left Most Derivation or 2 Right Most Derivation.

① L·M·D 1



a+b*c

① L·M·D 2



a+b*c

Here for the given grammer there is 2 L.M.D that exist. Therefore the given grammer is ambiguous.

NOAM CHOMSKY HIERARCHY

- Noam chomsky gave a mathematical model of grammar which is effective for writing computer language.
- Here the grammer are classified on the basis of their production.
- Here each category represent a class of language that can be recognized by a different automation
- Generally there are four type of grammer according to noman chomsky are :-

⇒ Type 0

- Grammer type :- Type 0 (most powerful no restrictions on production)
- Production type : $\alpha \rightarrow \beta$
  where $\alpha \in (V+T)^*$, $\beta \in (V+T)^*$

- Grammer Accepted : Unrestricted grammer
- Language accepted : Recursively Enumerable language
- Automation : Turing Machine

⇒ Type 1

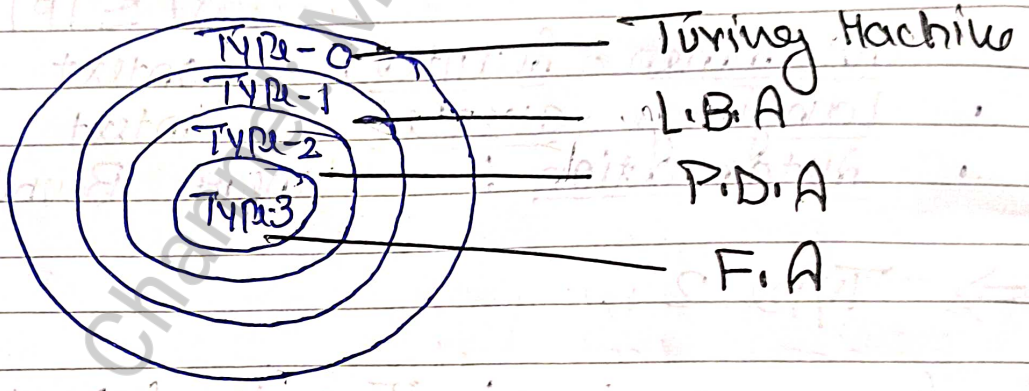
- Grammer type: Type 1 (Derived from type 0 with restriction)
- Production type : $\alpha \longrightarrow \beta$
  $|\alpha| \leq |\beta|$, $\alpha, \beta \in (V+T)^+$
- Grammer Accepted: Context Sensitive Grammes
- Language accepted : Context Sensitive language
- Automation : Linear Bounded Automaton

⇒ Type 2

- Grammer type: Type 2 (Derived from type 1 with more restriction)
- Production type : $\alpha \longrightarrow \beta$
  α is free of context
  $\beta \in (V+T)^*$
- Grammer Accepted : Context Free Grammes
- Language Accepted : Context free language
- Automation : Pushdown Automata.

⇒ Type 3

- Grammer type: Type 3 (derived from type 2 with more restriction)
- Production type: $A \to Ba$ (Left Linear)
  $A \to aB$ (Right Linear)
  $A \to \epsilon$ also
- Language Accepted: Regular language
- Grammer Accepted: Regular Grammer.
- Automation: Finite Automata.



Power: TM > LBA > PDA > FA

Subset: RG ⊂ CFG ⊂ CSG ⊂ UG

C.N.F

- CNF stands for Chomsky Normal Form
- A CFG is said to be in C.N.F if all of its production rule are of the form :

$A \rightarrow BC$ { A, B, C ∈ Variable }

or

$A \rightarrow a$ { a ∈ terminal }

{ Single Non. Terminal — Single terminal
Single Non. Terminal — Two N.T }

- No. of steps required to generate a string if length 'n' is "2n-1"
- It used in membership algo
- Derivation of parse tree obtain from CNF is always Binary tree
- length of each Production Restricted

G.N.F

① GNF stands for Greibach Normal form.

② A context free grammer C.F.G is in GNF if every production of the grammer is of the form

$$A \rightarrow a\alpha$$

(A: N.T, a: T, α: where $\{ x \in V^* \}$, $a \in T$)

$A \rightarrow aBAC$

[Single N.T → Single Terminal
Single NT → Single Terminal followed by the no. of NTS not include start symbol]

③ No. of steps required to generate a string of length 'n' is n

④ It's used to convert C.F.G to PDA

⑤ It is Not always formed in Binary Tree

⑥ It is not restricted

Q. Convert CFG to CNF

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

Sol Convert the grammer into simplified CFG.

(i) Removal of ε - production

$S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid$

$A \rightarrow B \mid S$

$B \rightarrow b$

(ii) Remove unit production

$A \rightarrow B$

$A \rightarrow S$

$A \rightarrow B \rightarrow S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$B \rightarrow b$

$S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow b \mid ASA \mid aB \mid a \mid AS \mid SA$

$B \rightarrow b$

(ii) Remove useless production

useful product {a, b}
There is no useless production

Replace terminal by introducing new terminal

$a \rightarrow C_1$
$b \rightarrow C_2$

Now the grammer become

$S \rightarrow ASA \mid C_1B \mid a \mid AS \mid SA$
$A \rightarrow b \mid ASA \mid C_1B \mid a \mid AS \mid SA$
$B \rightarrow b$
$C_1 \rightarrow a$
$C_2 \rightarrow b$

Reduce the no. of non terminals form production containing more two NT

$S \rightarrow ASA$

$S \rightarrow AD_1$

$B \rightarrow AA \rightarrow D^2 \mid AD^2$

$[SA \rightarrow D_1]$

Now the production in CNF

$S \rightarrow AD_1 \mid C_1B \mid a \mid AS \mid SA$
$A \rightarrow b \mid AD_1 \mid C_1B \mid a \mid AS \mid SA$
$B \rightarrow b$
$C_1 \rightarrow a$
$C_2 \rightarrow b$
$D_1 \rightarrow SA$

CONVERT C.F.G. TO C.N.F

Step1:- Convert the grammar into CNF. If the given grammar is not in CNF, convert it into CNF.

Step2:- If the grammar exists left recursion Eliminate it.

Step3:- In the grammar, convert the given production rule into C.N.F form:

Q

$S \rightarrow XA \mid BB$ (N.T)
$B \rightarrow b \mid SB$
$X \rightarrow b$
$A \rightarrow a$

Sol The given grammer G is already in CNF & there is no left recursion.

→ The production rule $B \rightarrow SB$ is not in GNF so, we substitute $S \rightarrow XA \mid BB$ in the production Rule.

$S \rightarrow XA \mid BB$
$B \rightarrow b \mid XAB \mid BBB$
$X \rightarrow b$
$A \rightarrow a$

The production rule $S \rightarrow XA$ & $B \rightarrow XAB$ are not in GNF so we substitute $X \rightarrow b$

$S \rightarrow bA \mid BB$

$B \rightarrow b \mid bAB \mid BBB$

$X \rightarrow b$

$A \rightarrow a$

So, we will remove left recursion $B = BBB$, we get

$S \rightarrow bA \mid BB$

$B \rightarrow b \mid bAB \mid bC \mid bABC$

$C \rightarrow BBC \mid BB$

$X \rightarrow b$

$A \rightarrow a$

⇒ ✱ whenever left recursion occur

$B = BBB$

introduced a new variable C and replace with and without a variable

Now production $S \rightarrow BB$ is not in GNF so we substitute

$B \rightarrow b \mid bAB \mid bc \mid bABC$

$S \rightarrow bA \mid bB \mid bABB \mid bcB \mid bABCB$
$B \rightarrow b \mid bAB \mid bc \mid bABC$
$C \rightarrow bBC \mid bABBC \mid bcBC \mid bABCBC \mid$
$bB \mid bABB \mid bcB \mid bABCB$
$X \rightarrow b$
$A \rightarrow a$

Now the grammer in G.N.F

Ans

⇒ **Closure Properties :-**

Already done in Unit-2.

⇒ **Decidable Properties :-**

Already done in Unit-2

UNIT-3 COMPLETE THNK YOU

@MULTIATOHST

LIKE SHARE SUBSCRIBE