Unit-III - Context Free Grammer (CFG) & Context Free Language (CFL)

☆ Basic Elements of Grammer :-

A Grammer is a 4 tuple.

$G = (V, T, P, S)$

where

V → is a finite set of non terminals.
T → is a finite set of terminals
P → is a set of production rules
S → is a start symbol.

A> Def^n It is a formal grammer which is used to generate all possible patterns of string in given formal language.

B> Application of CFG :-

1) For defining programming Languages
2) For translation of programming lang.
3) For construction of compilers.
4) For parsing the program by constructing syntax tree.

☆ Formal Def^n of context Free Grammer :-

1) where each production rule is in the form of
   i) Non terminal ⟶ Non terminals.
   ii) Non terminal ⟶ Terminals
   S is a start symbol.
2) we are using following convertions.
   i) The capital letters are used to denote the non terminals.
   ii) The lower case letter are used to denote the terminals.

✯ Sentential Form:-

1) Consider $G = (V, T, P, S)$ be a context free grammer then, we can derive a string w from it.
2) This w can obtained from $(VUT)^*$ where V is non terminals and T denotes the terminal symbols.
3) The derivation of 'w' from start symbol S can be written as $S \overset{*}{\Rightarrow} w$ which is called as sentential form.
4) If $S \overset{*}{\underset{lm}{\Rightarrow}} w$ then w is a left sentential form.
5) If $S \overset{*}{\underset{rm}{\Rightarrow}} w$ then w is a Right sentential form.

✯ Derivation And Derivation Tree / parse Tree:-

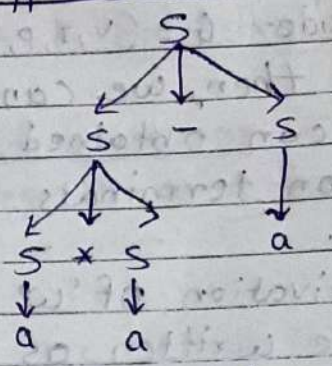
1) Derivation is a process of generating of a string in some finite steps.
2) while Generating a string, we replace a non-terminals by its production which is given in CFG
3) If the production rule consist of strings of two non-terminal on right hand side of production rule then question in which non-terminal should be replaced first to generate string or next sentential form
4) To solve above problem by using two types of derivation.
5) **Leftmost derivation:-**

A derivation of a string w in grammer G is left most derivation if at every step the left most non terminal is replaced.

Grammer - $S \rightarrow S+S \mid S-S \mid S*S \mid S/S \mid a$

o/p : a*a-a

$S \rightarrow S-S$
$\rightarrow S*S-S$
$\rightarrow a*S-S$
$\rightarrow a*a-S$
$\rightarrow a*a-a$

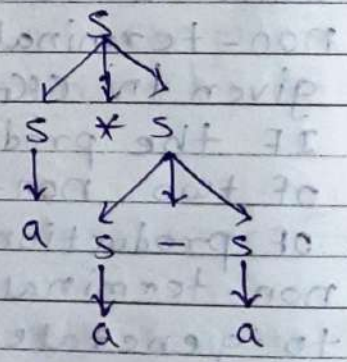


Leftmost Derivation. parse tree.

Right most Derivation :-

1) A derivation of a string w in a grammer G is right most derivation if at every step the right most non terminal is replaced.
2) It is all called canonical derivation.
3) Grammer : $S \rightarrow S+S \mid S-S \mid S*S \mid S/S \mid a$

S
$\rightarrow S*S$
$\rightarrow S*S-S$
$\rightarrow S*S-a$
$\rightarrow S*a-a$
$\rightarrow a*a-a$



Rightmost Derivation.

parse tree.

★ Context Free language :-

1) The lang generated by context free grammer is known as context free language.
2) It is basically used for arithmatic eqn generation.

3> A Language L is a context free lang (CFL) if there is a CFG G so that $L = L(G)$

★ Regular Grammer:-

A grammer $G = (V, \Sigma, S, P)$ is regular if every production takes one of the two forms.

$B \rightarrow ac$

$B \rightarrow a$

where B & C are non terminals and is terminal.

Ex> Find out the lang generated by the following CFG.

1> Construct CFG for the lang L which has all the string which are all palindrom $\Sigma = \{a, b\}$ (all: even or odd)

→ $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$

Ex 1> S → bSb → baSab → baaab

2> S → aSa → aεa → aa

3> S → b

4> S → a

2> construct CFG for the lang L which has all the string which are even palindrom over $\Sigma = \{a, b\}$

→ $S \rightarrow aSa \mid bSb \mid \epsilon$

Ex> S → aSa → aaSaa → aabSbaa → aabεbaa

3) Construct CFG for the following Lang having any number of a's over the set $\Sigma = \{a\}$

→ R.E = a^*

$S \rightarrow aS \mid \epsilon$

Ex) S
as
aas
aaas
aaaϵ your string - aaaa.

4) construct CFG for the following Lang L any number of 'a' & 'b' except null and ending with 'a'

→ $S \rightarrow aS \mid bS \mid a$

Ex) 1) S → a → a

2) S → bS → bbS → bba

3) S → aS → aa

4) S → aS → baS → abaa

5) construct CFG for any number of consecutive 'ba' and the string start and end with 'b'

→ $S \rightarrow SaS \mid b$

Ex) S → SaS → SaSaS → bSaSaS → babSaS → babab

2) S → b

3) S → SaS → bSaS → bSaSaS → bSabaS → bSabab → bababab

6) construct CFG for the following lang L having same number of 'a's & 'b's

→ $S \to aSb \mid bSa \mid \epsilon$

Ex) S → a S b → ε → ab

2) S → a S b → b S a → a S b → ε → ababab

3) S → ε → Null string

7) construct a grammer for the lang. containing strings of at least two a's

→ $S \to XaXaX$

$X \to aX \mid bX \mid \epsilon$

Ex) 1) S → XaXaX → ε ε ε → aa

2) S → XaXaX → aX ε ε → bX ε → ε → abaa

8) construct CFG for string containg any combinat^n of a & b but start with 'a' only

→ $S \to aX$

$X \to aX \mid bX \mid \epsilon$

Ex) S → a X → a ε → a

S	ababX
aX	ababaX
abX	ababaε
abaX	ababa

9) String containing even Length of 'a's

→ $S \to SS \mid XaXaX \mid \epsilon$

$X \to bX \mid \epsilon$

Ex) 1) S → ε — Null

2) S → SS → SSS → ε ε → XaXaX → ε ε ε → bX → bX → bbaa

3) S → X a X a X → ε bX ε → ε b ε → aba

10) Construct CFG for the Regular expression. Give the CFG which generates a string containing only a's

$R.E = a^+$

→ $S \to aS \mid a$

11) Construct CFG for the Regular Exp. a^*

→ $S \to aS \mid \epsilon$

Ex) aS → aS → aS → a ε → aaa

Page No:
Date:

imp
12) construct the CFG for the R.E $(a+b)^*$.

→ $S \to aS \mid bS \mid \epsilon$.

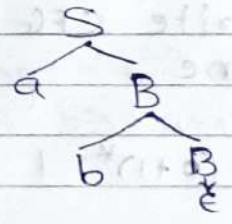
Ex)
S
aS
bS
ϵ
ab

13) construct the CFG for the R.E $(0+1)^*$.

→ $S \to 0S \mid 1S \mid \epsilon$.

14) construct the CFG for the ab^*

→ $S \to aB$
$B \to bB \mid \epsilon$

Ex)


ab.

15) construct the CFG for the ab^+

→ $S \to aB$
$B \to bB \mid b$

16) construct the CFG for the $a^* b^*$.

→ $S \to AB$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$.

17) construct CFG which consist of all string having atleast one occurrence of 000

A T A

Ex: (0,1) any 000 (0,1) any

→ RE = (0+1)* 000 (0+1)*.

S → ATA or A000A

A → 0A | 1A | ∈

T → 000

18) write a CFG for (011+1)*(01)*

→ S → AB

A → 011A | 1A | ∈

B → 01B | ∈.

19) write CFG which contain at least 3 times one.

→ R.E = (0+1)* 1 (0+1)* 1 (0+1)* 1 (0+1)*.

S → A1A1A1A

A → 0A | 1A | ∈.

20) write CFG that must start & end with same symbol ∈ = (0,1)

A A

→ RE = 0(0+1)*0 + 1(0+1)*1

S → 0A0 | 1A1

A → 0A | 1A | ∈.

21) Try to recognize the Lang L for given CFG

G = [{S}, {a,b}, P, {S}]

where P = { S → aSb, S → ab }

→ asb
aasbb
aaaSbbb
aaaabbbb

Thus we can have any number of a's first then equal number of b's following it.
$L = a^n b^n$ where $n \geq 1$

22) construct CFG for the Lang containing at least one occurrence of double aa

[SPPU - Dec-17, Mark-4]

→ R.E = (any a&b) (one occurrence of double a) (any a&b)

$S \rightarrow ATA$ or $A\,aa\,A$.
$A \rightarrow aA \mid bA \mid \epsilon$.
$T \Rightarrow aa$.

23) construct CFG for the Lang $a^n b^{2n}$ where $n \geq 1$

→ In this ex number of b's are double

$S \rightarrow aSbb \mid abb$

24) write CFG for set of odd length settings in $(0,1)^*$ with middle symbol '1'

[SPPU - May-14, Mark-4]

→
$S \rightarrow 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 1$

consider the string 01110 For derivation.

S
0S0
01S10
01110

write CFG for set of odd length setting in $(a,b)^*$ with middle symbol 'a'

→
$S \rightarrow aSa \mid aSb \mid bSa \mid bSb$
$S \rightarrow a$

25) write a CFG For generating identifiers in high level Lang such as 'c' Identifiers can be defined by the R.E (letter).(letter|digit)*

[SPPU-May-13, Mark-4]

→

S → LA

A → LA | DA | ∈

L → a|b|c ---z

D → 0|1|2 ---9.

26) write a CFG The set of even length strings (a,b)* with the two middle symbol equal.

[SPPU-Dec-07, May-08, Mark-8]

→

R.E = (a+b)*(aa+bb)(a+b)*

Hence CFG

S → aSa | aSb | bSb | bSa

S → aa

S → bb.

27) Give context Free Grammer for following Lang.

i) L = { S → aAb, A → aA|bA|∈, x|x ∈ (a,b)* with strings of starting with 'a' & end with 'b'

[SPPU-Dec-14, Mark-4]

→

R.E = a (a+b)* b.

∴ CFG

S → aAb

A → aA | bA | ∈.

ii) L = { x|x ∈ (a,b)* with strings of even length of palindrom → S → asa|bsb|∈

28) write a CFG for the R.E = $0(0+1)^* 01 (0+1)^* 1$ [SPPU - Aug-17, Mark-4]

→ $S \to 0A01A1$

$A \to 0A|1A|\epsilon$

Imp

29] Write CFG for $L = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$

→

For $i=j$	for $j=k$
$S \to AB$	$C \to DE$
$A \to aAb \mid ab$	$D \to aD \mid a$
$B \to cB \mid c$	$E \to bEc \mid bc$

Ex) 1) $S \to ab$

2) $S \to aSb$

$\downarrow ab$

3) aSb

$aaSbb$

$aaabb$

Imp

30) write a CFG for $L = \{a^i b^j c^k \mid j > i+k\}$

→ $S \to ABC$

$A \to aAb \mid \epsilon$

$B \to bB \mid b$

$C \to bCc \mid \epsilon$

Ex) aa bbbbb cc $\quad 4+1 = c$

$b^j > a^i c^k$

whenever 'a' is occurr then 'b' is also occurr whenever 'c' is occurr then 'b' is occure.

S

ABC

aAbbbCc

$a\epsilon bbb\epsilon c$ = abbbc

Imp

31) write a CFG for $L = \{0^i 1^j 0^k \mid j > i+k\}$

→

$S \to ABC$

$A \to 0A1 \mid \epsilon$

$B \to 1B \mid 1$

$C \to 1C0 \mid \epsilon$

[SPPU-Oct-18, Mark-4]

Imp 3>) write a CFG for the $L=\{a^n b^m c^n \mid n>=0, m>=1\}$

$S \rightarrow aSa \mid bB$

$B \rightarrow bB \mid \epsilon$

Ex> aSa
aaSaa
aabBaa
aabbBaa
aabbbBaa
aabbbEaa

I/P → aabbbaa

Imp 33) $0^i 1^{i+k} 0^k$ where $i,k >= 0$

$S \rightarrow AB$

$A \rightarrow 0A1 \mid \epsilon$

$B \rightarrow 1B0 \mid \epsilon$

Ex> S → AB
0A1 1B0
00A11 11B00
00ε11 11ε00
00111100

Imp 34) write a CFG for following. [SPPU-Dec-08, M-12]

$L = \{ a^n b^{2n} \mid n>1 \}$

$S \rightarrow aAbb$

$A \rightarrow aAbb \mid abb$

Ex> $a^2 b^4$
aabbbb

Ex> S → aAbb
aabbbb

2> $L = \{ a^m b^n \mid n>m \}$

$S \rightarrow aSbB \mid \epsilon$

$B \rightarrow bB \mid b$

Ex> $a^m b^n$ m=1, n=2
$a^1 b^2$
abb

aSbB
aSbbB
aεbbb
abb

✱ Parse Tree :-

1) It is a graphical representation for the derivation of the given production rules for a given CFG.
2) It is the simple way to show how the derivation can be done to obtain some string from given set of production rules.
3) Parse tree is called as derivation tree.

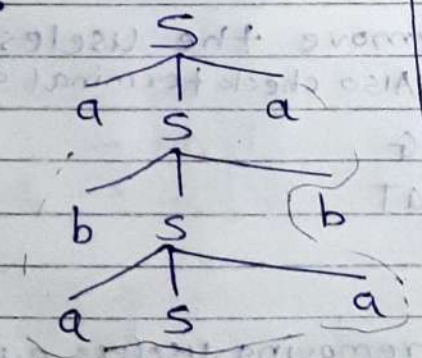
Properties of Parse Tree :-

1) The root node is always a node indicating start symbol.
2) The derivation is read from left to Right
3) The leaf nodes are always terminal nodes.
4) The interior nodes are always the non terminal nodes.

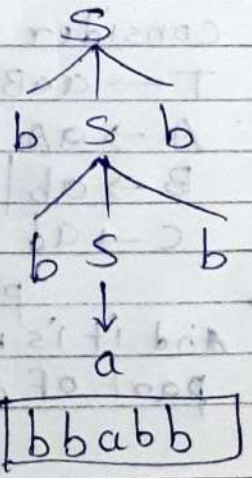
Ex) Draw a parse tree for the abaaba for the CFG given by.

$P = \{ S \to aSa$
$S \to bSb$
$S \to a|b|\epsilon \}$.

→



Ex) $S \to bSb|a|b$



bbabb

Simplification of CFG :-

1) As we have seen various Lang can effectively be represented by context free grammer
2) All the grammer are not always optimized.
3) That means grammer may consists of some extra symbols (non terminals)
4) Having extra symbol unnecessary increase the length of Grammer.

Properties of Reduced Grammer :-

1) Each variable (i.e non terminal) and each terminal of G appears in the derivation of some word on in L
2) There should not be any production as $X \rightarrow Y$ where X and Y are non terminals.
3) If ϵ is not in the Lang L then there need not be the production $X \rightarrow \epsilon$.

A) Removal of useless symbols :-

It is used to remove the useless production rule. Also check terminal symbol

Ex) consider the CFG

$T \rightarrow aaB \mid abA \mid aaT$

$A \rightarrow aA$

$B \rightarrow ab \mid b$

$C \rightarrow ad$

→ For removing useless symbol. production $A \rightarrow aA$ is a useless and it is not terminate. Hence it is not part of derivation.

to remove this useless production A → aA
This is not not terminate.

After Removal of useless symbols

T → aaB | aaT

B → ab | b

2) consider the CFG

S → A11B | 11A

S → 1B | 11

A → 0

B → BB using removing useless symbols.

→ In this case 'A' gives some terminal symbol as '0' but B does not gives the terminal symbol

After Removing

S → 11A

S → 11

A → 0

or

S → 11A | 11

A → 0

3) S → aA | bB consider the CFG.

A → aA | a

B → bB

D → ab | Ea

E → ac | d

→

S → aA

A → aA | a

B) Elimination of ε production from Grammer

1) we have seen in regular expression that ε or a null string indicates a string with no value.

2) If there is a ε production we can remove it without changing the meaning of the Grammer.

3) Ex)

$S \to 0S \mid 1S \mid \epsilon$

⟶

$S \to \epsilon$ Here there is null production we can Remove by without changing the meaning of Grammer.

If we place $S \to \epsilon$ in other rule we get $S \to 0$ when $S \to 0S$ and $S \to 1$ when $S \to 1S$

$S \to 0S \mid 1S \mid 0 \mid 1$

Ex 2) $S \to aSa$, $S \to bSb$, $S \to \epsilon$ Remove ε production From the following CFG.

⟶

~~$S \to aa$~~ ε replace by aa & bb.

$S \to aSa \mid bSb \mid aa \mid bb$

Ex 3) $A \to 0B1 \mid 1B1$

$B \to 0B \mid 1B \mid \epsilon$.

ε Replace by 01 & 11, 0 & 1

⟶

$A \to 0B1 \mid 1B1 \mid 01 \mid 11$

$B \to 0B \mid 1B \mid 0 \mid 1$

Ex4> $S \to XYX$
$X \to 0X \mid \epsilon$
$Y \to 1Y \mid \epsilon$

$\longrightarrow$

$S \to XY \mid YX \mid XX \mid X \mid Y$
$X \to 0X \mid 0$
$Y \to 1Y \mid 1$

Ex5> Eliminate ϵ production from the CFG.
$A \to aBb \mid bBa$ [SPPU-Aug-17, Mark-3]
$B \to aB \mid bB \mid \epsilon$

$\longrightarrow$

$A \to aBb \mid bBa \mid ab \mid ba$
$B \to aB \mid bB \mid a \mid b$

C] ☆ Removing Unit Production :-

The unit production are the production in which one non terminal gives another non terminal.

Step-1 - To remove $X \to Y$, add production $X \to a$ to the grammer rule whenever $Y \to a$ occurs in the grammer

Step-2 - Now delete $X \to Y$ for the grammer.

Step-3 - Repeat step 1 and step 2 until all unit production are removed.

Ex1> If the CFG is as below

$S \to 0A \mid 1B \mid C$

$A \to 0S \mid 00$

$B \to 1 \mid A$

$C \to 01$. then remove the unit. Produth

→

clearly $S \to C$ is a unit production then remove this rule. Replace 'C' by '01'

$S \to 0A \mid 1B \mid 01$

similarly $B \to A$

$B \to 1 \mid 0S \mid 00$

Final CFG.

$S \to 0A \mid 1B \mid 01$

$A \to 0S \mid 00$

$B \to 1 \mid 0S \mid 00$

$C \to 01$

Ex 2> Simplify the following Grammer.

i> $S \to Ab$,

$A \to a$

$B \to C \mid b$

$C \to D$

$D \to E$

$E \to a$

[SPPU - Dec-16, Oct-19 Marks - 4 or 8]

→

In this example $B \to C$, $C \to D$, $D \to E$ are unit production. These can be eliminated. and also $E \to a$ is eliminated.

we get $B \rightarrow a|b$.

In starting state non terminal B is non reachable. Hence eliminate it as useless symbol.

Obtain simplify Grammer is

$S \rightarrow Ab$

$A \rightarrow a$

3) consider the Grammer $G = \{(A,B), (a,b), P, A\}$

P consist of

$A \rightarrow B$

$B \rightarrow a|b$

Eliminate Unit production

[SPPU - Oct-18, Marks-3]

$\longrightarrow$ $A \rightarrow B$ is a unit production Hence remove it.

$\boxed{A \rightarrow a|b}$

4) simplify the CFG [SPPU - Dec-19, Mark-6]

$S \rightarrow ASB|\epsilon \quad B \rightarrow SbS|A|bb$

$A \rightarrow aAS|\epsilon$

$\longrightarrow$ we will eliminate unit production. $(B \rightarrow A)$

$S \rightarrow ASB|\epsilon$

$A \rightarrow aAS|\epsilon$

$B \rightarrow SbS|aAS|bb|\epsilon$

Now we will eliminate ϵ.

$S \rightarrow ASB|As|AB|SB$

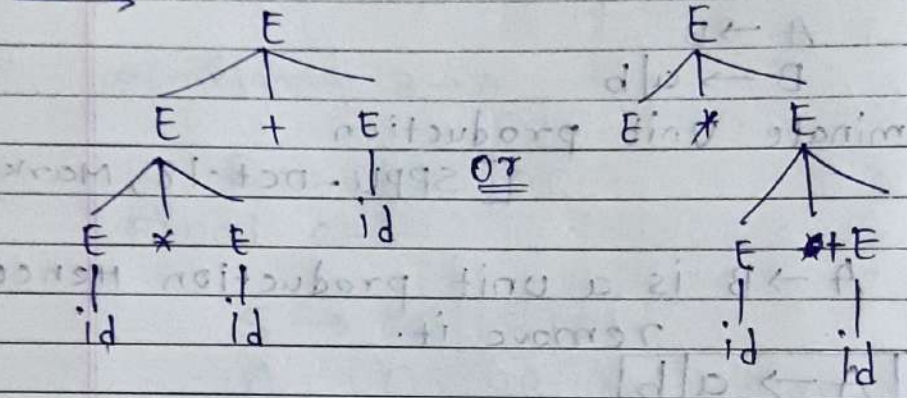
$A \rightarrow aAS|aA|a$

$B \rightarrow SbS|aAS|bb|sb|bS|b|aA|as|bb$

✱ Ambiguous Grammer :-

If there exists more than one parse tree for a given Grammer, that means there could be more than one leftmost or rightmost derivation possible and then that Grammer is said to be ambiguous Grammer.

Ex> $E \to E+E$
$E \to E*E$
$E \to id$

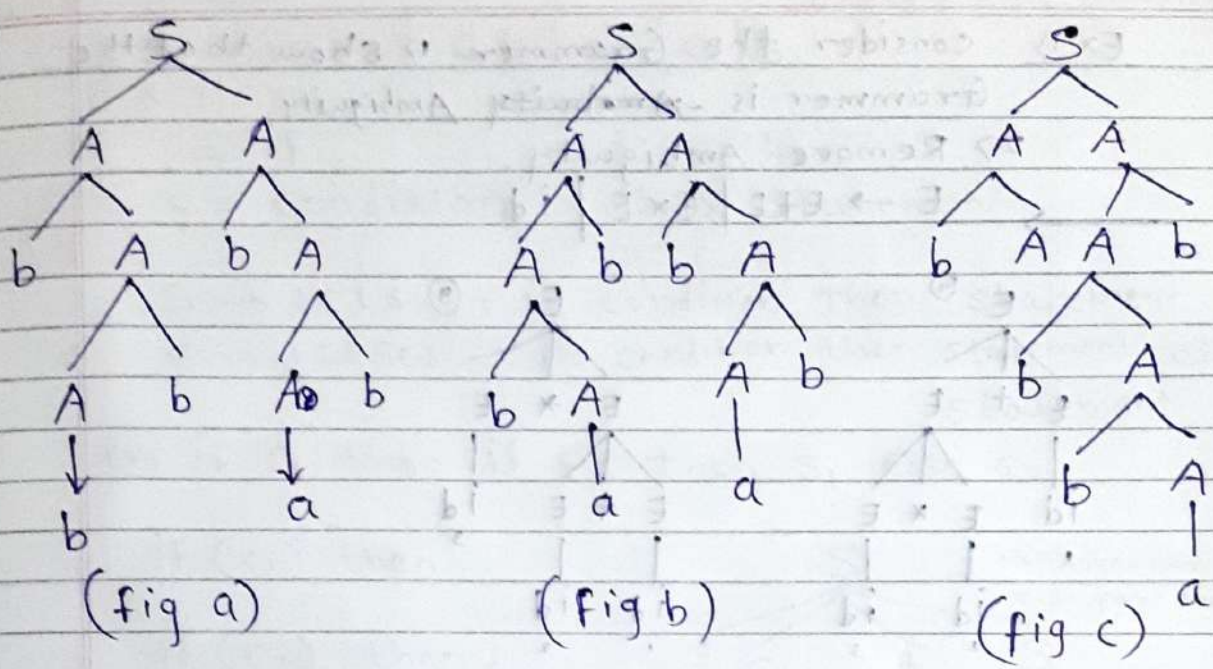


Left most Right most

Thus the above Grammer is an ambiguous Grammer.

Ex 2> consider the Grammer
$S \to AA$
$A \to AAA$
$A \to a$
$A \to bA$
$A \to Ab$ show that this is an ambiguous Grammer.

→ consider the string babbab



(fig a) (fig b) (fig c)

☆ Removal of Ambiguity :-

1) There are two problem occure regarding Precedance and Associative

2) (+, -) operator represent left associative.

Ex) (2+3)+4 ✓ 2+(3+4) ✗

3) If Grammer has left associative then induce the left recursion.

Left Recursion means.

R.H.S = L.H.S

$E \rightarrow E|a$

4) (+, -) operator is used for the term separator.

5) (*, /) operator is also n Left associative.

6) exponential operator is also Right associative then induce the Right Recursion.

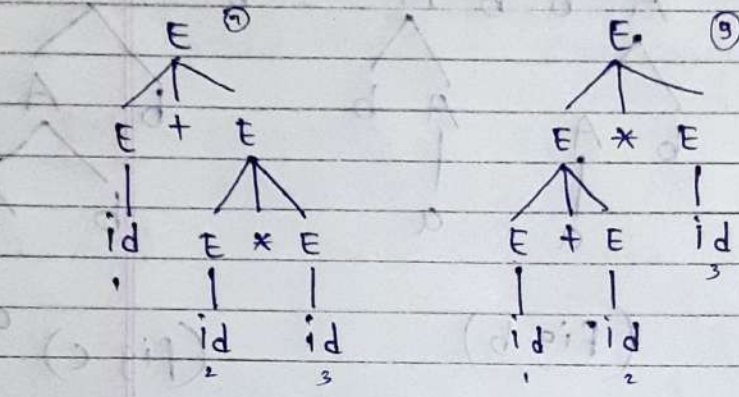
7) If precedance not define that case your Grammer o/p will be different or change.

8) To overcome these two problem by using Removal of Ambiguity.

Ex^n 1) consider the Grammer 1) show that the Grammer is ~~Amibuity~~ Ambiguity.

2) Remove Ambiguity.

$E \rightarrow E+E \mid E*E \mid id.$



Parse tree-1 Parse tree-2

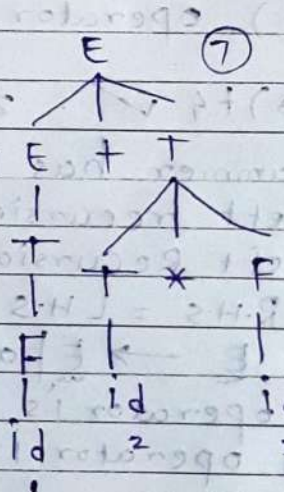
Fig:- Ambiguity Grammer.

2) Remove Ambiguity.

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow id.$



2) Give an ambiguous grammer for if then else statement and then re-write a equivalent unambiguous Grammer.

[SPPU - May-14, Mark-8]

Dangling Else problem

$S \rightarrow ictS \mid ictSeS \mid a \mid b$

$C \rightarrow c_1 \mid c_2$

i = if t → then

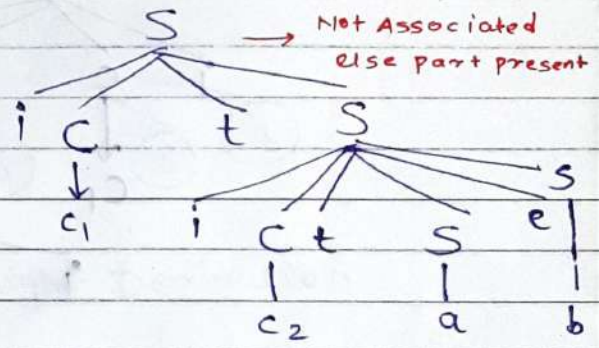
c = condition S → Statement

$S \rightarrow ictS$ = if condition then statement

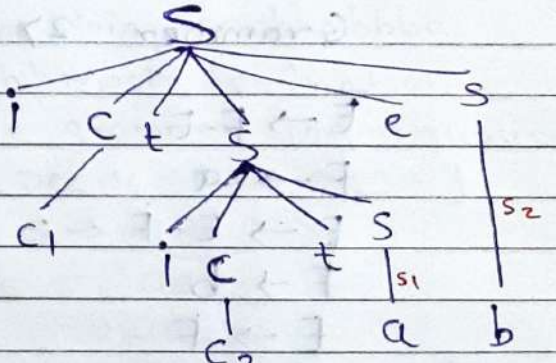
$S \rightarrow ictSeS$ = if condition then statement else statement.

Ex> if c_1 then if c_2 then s_1 else s_2

```
if (c1) then
{
 if (c2) then
 {
   s1
 }
 else
  {
   s2
  }
```




```
if (c1) then
{
 if (c2) then
 {
   s1
 }
}
else
{
  s2
}
```



→ else not show

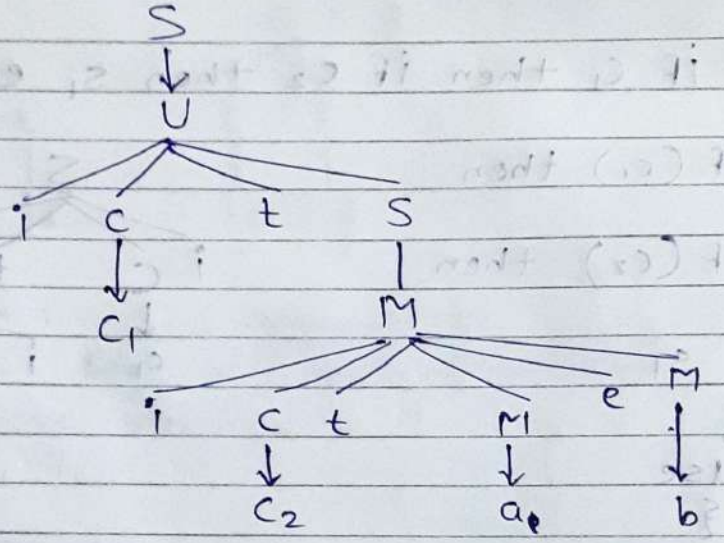
M - Matched statement.
U - un Matched statement

$S \rightarrow U \mid M$

$M \rightarrow iCtMeM \mid a \mid b$ condition for matched statement only means 'a' & 'b'

$U \rightarrow iCtS \mid iCtMeU$ 1) condition for if statement 2) condition for else statement

$C \rightarrow C_1 \mid C_2$



3) consider the Grammer 1) show that G is Grammer 2) Remove ambiguity.

$E \rightarrow E-E$

$F \rightarrow a$

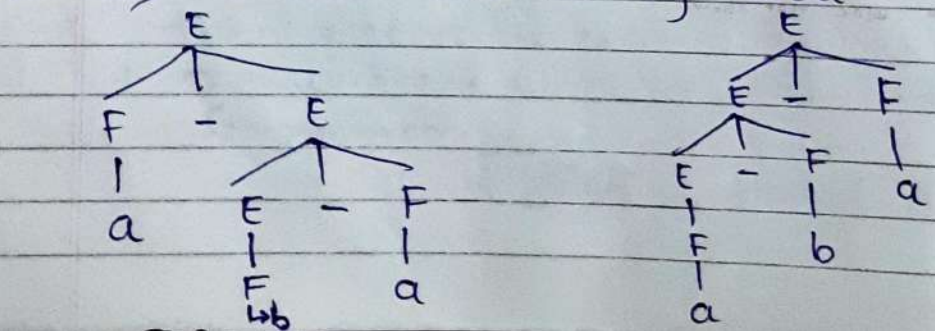
$E \rightarrow E-F$

$F \rightarrow b$

$E \rightarrow F$

[SPPU May-11, Mark-6]

→ consider the String aba



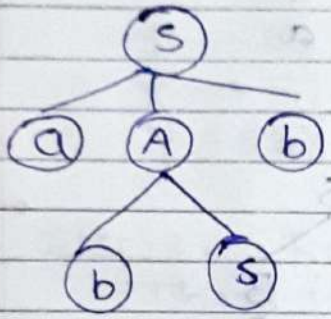
3> The unambiguous Grammer.

$E \rightarrow E-F \mid F$
$F \rightarrow a \mid b$

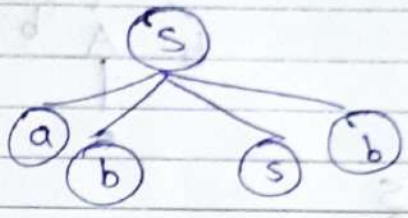
4> Show that following Grammer is ambiguous

$S \rightarrow a \mid absb \mid aAb$
$A \rightarrow bs \mid aAAb$ [SPPU-Dec-12, Mark-4]

⟶ Consider the string "absb"



Left Derivation.



Right Derivation

5> Let G be the grammer

$S \rightarrow aB \mid bA \quad A \rightarrow a \mid aS \mid bAA$
$B \rightarrow b \mid bS \mid aBB$ For the string aaabbabbba
Find 1> Leftmost & Right most derivation
3> Parse tree 4> Is the grammer unambiguous.
[SPPU - May-07, Dec-14, Mark-8 or 4]

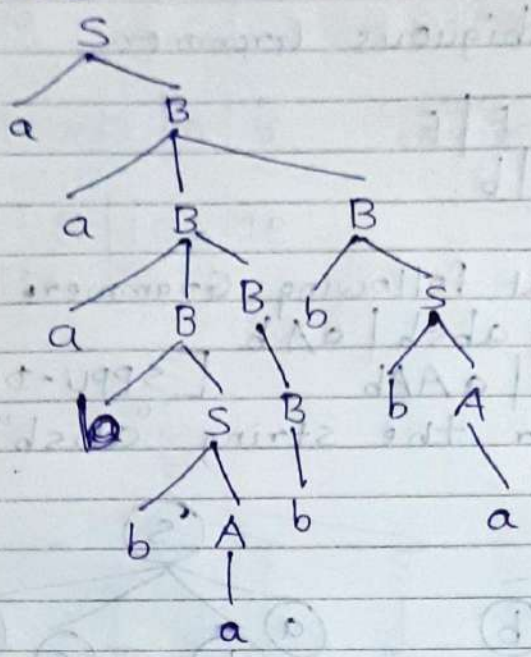
⟶ Left Most

$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aaaBBB$
$\rightarrow aaabSBB$
$\rightarrow aaabbABB$
$\rightarrow aaabbaBB$
$\rightarrow aaabbabB$
$\rightarrow aaabbabbS$
$\rightarrow aaabbabbbA$
$\rightarrow aaabbabbba$

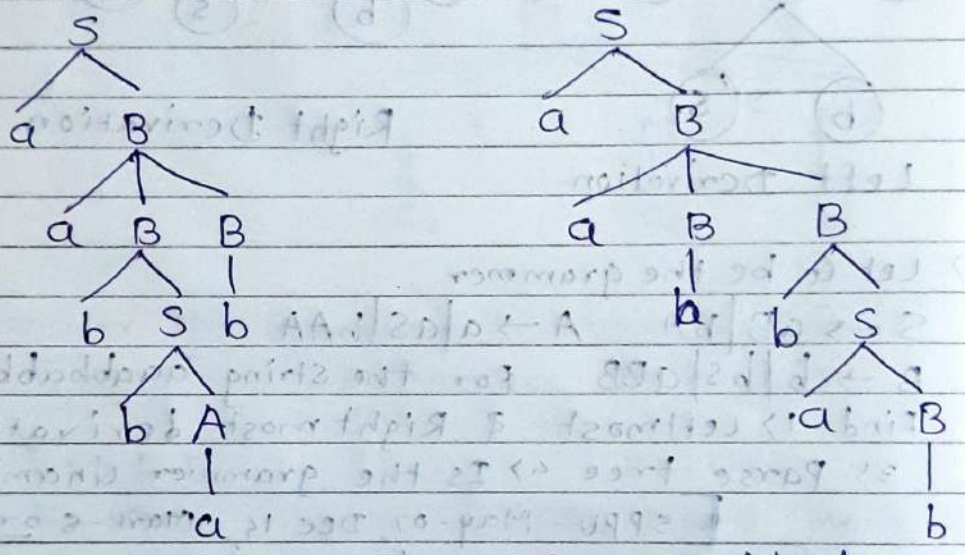
$S \rightarrow aB$
$\rightarrow aaBB$
$\rightarrow aaBbS$
$\rightarrow aaBbbA$
$\rightarrow aaBbba$
$\rightarrow aaaBBbba$
$\rightarrow aaaBbbba$
$\rightarrow aaabSbbba$
$\rightarrow aaabbAbbba$
$\rightarrow aaabbabbba$

Right most.

3) parse tree



4)



For string - aabbab.

No, the Grammer is ambiguous

5) Normal Forms :-

There are two important normal forms, Chomsky Normal Form (CNF) and Greibach Normal form (GNF)

A) Chomsky Normal Forms-

Rules

1) Non terminal → Non terminal. Non terminal
2) Non terminal → terminal

such that

$S \rightarrow AB$

$S \rightarrow a$

Ex1) convert the following CFG to CNF

$S \rightarrow aX \mid Yb$

$X \rightarrow S \mid \epsilon$

$Y \rightarrow bY \mid b$

→

Step-I:- Remove useless symbol

There is no useless symbol

Step-II:- Eliminate ϵ production

$S \rightarrow aX \mid a \mid Yb$

$X \rightarrow S$

$Y \rightarrow bY \mid b$

Step-III:- Eliminate unit production.

$S \rightarrow aX \mid a \mid Yb$

$X \rightarrow aX \mid a \mid Yb$

$Y \rightarrow bY \mid b$

Step-4:- Replace all mixed string with soid NT's

$S \rightarrow AX \mid a \mid YB$

$X \rightarrow AX \mid a \mid YB$

$Y \rightarrow BY \mid b$

$A \rightarrow a$

$B \rightarrow b$

The equivalent CNF for the given CFG's

Page No:
Date: / /

Step-5 - Shorten the String of NT to length 2
All NT strings on the R.Hs in the above CFG are already the required length. So CFG is in CNF.

2) convert the given CFG to CNF.

$S \to aB$ $\quad A \to bAA$
$S \to bA$ $\quad B \to b$
$A \to a$ $\quad B \to bS$
$A \to aS$ $\quad B \to aBB$

[SPPU - Dec-05, Dec-14, Dec-16, Mark-6]

	CNF Form
1) $S \to aB$	✓ $S \to R_1B$, $R_1 \to a$
2) $S \to bA$	✓ $S \to R_2A$, $R_2 \to b$
3) $A \to a$	✓ $A \to a$
4) $A \to aS$	✓ $A \to R_1S$
5) $A \to bAA$	$A \to R_2AA$, ✓ $A \to R_2R_3$, $R_3 \to AA$
6) $B \to b$	✓ $B \to b$
7) $B \to bS$	✓ $B \to R_2S$
8) $B \to aBB$	$B \to R_1BB$, ✓ $B \to R_1R_4$, $R_4 \to BB$

3) Convert the following CFG to CNF

$S \to ABA$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$

[SPPU - May-13, 14, Mark-6]

→ Step-I : Remove ∈ symbol

$S \to ABA \mid AB \mid BA \mid AA \mid A \mid B$
$A \to aA \mid a$
$B \to bB \mid b$

Step-II :- Remove unit production

$S \to ABA \mid AB \mid BA \mid AA \mid aA \mid a \mid bB \mid b$
$A \to aA \mid a$
$B \to bB \mid b$

	CNF Form
$S \to ABA$	✓ $S \to AR_1$
	$R_1 \to BA$
$S \to AB$	✓ $S \to AB$
$S \to BA$	✓ $S \to BA$
$S \to AA$	✓ $S \to AA$
$S \to aA$	✓ $S \to R_2A$
	$R_2 \to a$
$S \to a$	✓ $S \to a$
$S \to bB$	✓ $S \to R_3B$
	$R_3 \to b$
$S \to b$	✓ $S \to b$
$A \to aA$	✓ $A \to R_2A$
$A \to a$	✓ $A \to a$
$B \to bB$	✓ $B \to R_3B$
$B \to b$	$B \to b$

★ Greibach Normal Form :-
The rule for GNF is

Non-terminal → one terminal. Any number of non-terminals.

$NT \to t \cdot NT$

Ex1) convert the following Grammer to GNF

$S \to abSb$

$S \to aa$

→ **Step I :-** there is no ϵ, unit production and useless symbol.

Step-II :- convert Grammer in CNF

CNF form

$S \to abSb$ → $S \to XY$, $X \to AB$, $Y \to SB$, $A \to a$, $B \to b$

$S \to aa$ → $S \to AA$

Step-III :- Now rename non terminals in ascending order

$S \to XY$ → $A_1 \to A_2A_3$

$X \to AB$ → $A_2 \to A_4A_5$

$Y \to SB$ → $A_3 \to A_1A_5$

$A \to a$ → $A_4 \to a$

$B \to b$ → $A_5 \to b$

Step-IV Now we will varity the condition of A_i, A_j as if $A_i < A_j$ or $A_i \geq j$ Not req.

The rules are

$A_1 \to A_2A_3$

$A_2 \to A_4A_5$

$A_{3(i)} \to A_{1(j)}A_5$ Here $i > j$ i.e $3 > 1$

$A_4 \to a$

$A_5 \to b$

Therefore we Replace by A_1 on R.H.S

$A_3 \to \boxed{A_1}A_5$

$A_3 \to \boxed{A_2}A_3A_5$

$A_3 \to A_4A_5A_3A_5$

Now we get

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_4A_5$

$A_3 \rightarrow A_4A_5A_3A_5$

$A_4 \rightarrow a$

$A_5 \rightarrow b$

Try to convert into GNF. So Replace A_4 & A_5 by terminal 'a'

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow aA_5$

$A_3 \rightarrow aA_5A_3A_5$

$A_4 \rightarrow a$

$A_5 \rightarrow b$

Replace A_2 by its R.H.S in A_1 form.

Final GNF Form.

$A_1 \rightarrow aA_5A_3$

$A_2 \rightarrow aA_5$

$A_3 \rightarrow aA_5A_3A_5$

$A_4 \rightarrow a$

$A_5 \rightarrow b$

2) convert the given CNF to CFG.

$S \rightarrow ABA$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

⟶ Step-I Remove ϵ

$S \rightarrow ABA \mid BA \mid AB \mid AA \mid A \mid B$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Now eliminate unit production

$S \rightarrow A \mid B$

✓

S → ABA | BA | AB | AA | aA | a | bB | b
A → aA | a
B → bB | b

Step-II In above Grammer the rules for A & B non-terminals are already in GNF some rules of S also GNF only handle
S → ABA | BA | AB | AA
by simple put the val A & B.

Step-III

S → ABA	S → aABA | aBA
S → BA	S → bBBA | bA
S → AB	S → aAB | aB
S → AA	S → aAA | aA

Step-IV Final GNF.

S → aABA | aBA | bBA | bA | aAB | aB | aAA | aA
S → aA | a | bB | b
A → aA | a
B → bB | b

Pumping Lemma For CFG :-

1) Pumping Lemma is used to prove that a Language is not context free.
2) IF A is a context free Lang. then A has a pumping length 'p' such that any string 's' where |s| ≥ p may be devided into 5 pieces s = uvxyz such that the following conditions must be true.
   1) uv^ixy^iz is in A for every i≥0
   2) |vy| > 0
   3) |vxy| ≤ p

Ex 1) Show that $L = \{a^N b^N c^N \mid N > 0\}$ is not context free

→ Step-I Assume that L is context free

Step-II L must have a pumping length (say P)

Step III Now we take a string s such that

$$S \Rightarrow a^P b^P c^P$$

Step-IV we devide s into parts uvxyz

consider $P = 4$ so $S = a^4 b^4 c^4$

$S = aaaabbbbcccc$

case-I v and y each contain only one type of symbol

aaaa bbbb cccc

u v x y z

$\therefore uv^i xy^i z$ consider $i = 2$

$uv^2 xy^2 z$

$\therefore$ aaaaaa abbbbc cccc

$a^6 b^4 c^5 \notin L$

It is not a context free.

IF 1st condition fail then not to check 2 & 3 cond^n

Ex 2) Show that $L = \{ww \mid w \in \{0,1\}^*\}$ is not context free.

→ 1) Assume that L is context free

2) L must have a pumping length (say P)

3) Now we take a string s such that

$S = 0^P 1^P 0^P 1^P$

4) we devide s into parts uvxyz

consider $P = 5$. so; $S = 0^5 1^5 0^5 1^5$

$S = 00000\ 11111\ 00000\ 11111$

case-1 vxy does not straddle a boundary.

00000 11111 00000 11111

u vxy z

$\therefore uv^i xy^i z$

consider $i = 2$

$\therefore uv^2xy^2z$

000001111111000001111

$0^5 1^7 0^5 1^5$

1st part ≠ 2nd part $\therefore$ not a context free.

✡ Closure Properties of CFL :-

1) The context free Languages are closed under union.

⟶ we will consider two Language L_1 & L_2 which are context free Language. we can give these lang. using CFG G_1 & G_2 such that $G_1 \in L_1$ & $G_2 \in L_2$. The G_1 can be given as $G_1 = (V, \Sigma, P_1, S_1)$ where P_1 can be given as.

$L_1 = \{a^n b^n, n>0\}$

$L_2 = \{a^m b^m, m>0\}$

Ex)

$S_1 \to aS_1b \mid \epsilon$ $\quad S_1 = aAb \mid ab$

$S_2 \to bS_2c \mid \epsilon$ $\quad S_2 = aBb \mid \epsilon$

$L_1 \cup L_2 = \{a^n b^n\} \cup \{a^m b^m\}$

o/p:- $S_1 \mid S_2$ $\quad S = S_1 \mid S_2$

Hence satisfy the closed under union.

2) The context free Languages are closed under concatenation.

⟶ we will consider two Lang L_1 & L_2 which are CFG. we can give these lang using CFG G_1 & G_2

$L_1 = \{a^n b^n, n>0\}$

$L_2 = \{c^m d^m, m>0\}$

$S \to S_1 \mid S_2$

$S_1 \to aS_1b \mid \epsilon$ $\quad S_1 = aAb \mid ab$

$S_2 \to bS_2c \mid \epsilon$ $\quad S_2 = cBd \mid \epsilon$

$L_1 \cdot L_2 = a^n b^n c^m d^m$

$S \rightarrow S_1 \cdot S_2$

o/p $S_1 \cdot S_2$

Hence satisfy the colsed under concatenation.

3) The contex free lang are closed under kleen closure.

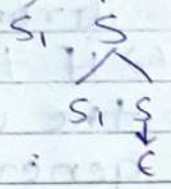
→ Let L_1 be a CFL represented by G_1 such that $G_1 \rightarrow \in L_1 \therefore L_1^* \rightarrow G_1$

$L = \{a^n b^n, n \geq 0\}$

$G: S \rightarrow aAb/\epsilon$

$L_1 = \{a^n b^n\}^*$

Ex) $S \rightarrow S_1 S / \epsilon$.



Hence It is proved that the CFG are closed under kleen closure.

4) The CFL are not closed under intersection

→ consider two lang L_1 & L_2

$L_1 = a^n b^n c^m$

$L_2 = a^m b^n c^n$

$L_1 \cap L_2 \rightarrow a^n b^n c^n \rightarrow$ CSL (In this case two comparison are perform)

But in CFL contain only one comparision.

Then we get sometimes CFG(L) and sometimes non CFL. Thus we can say that CFL are not closed under intersection.

5) The CFL are not closed under complement.

→ 1) consider two lang L_1 & L_2 are two CFL.

2) The complement of L_1 & L_2 are L_1' & L_2' respectively. Both are the CFL. $(L_1' \cup L_2')$

3) But $(L_1' \cup L_2') = L_1 \cap L_2$ may or may not be CFL.

4) we can say that CFL is not closed under complement.

☆ Decision properties of CFL:-

Deddable problem means problem solve by using any exists algorithm.

unDecidable problem means problem solve by using any not existing algorithm.

1) Emptyness Problem

1) It is decidable problem.
2) checking weather the given CFG generated empty lang or not.

Algo → 3) Eliminate useless var from the given Grammer.

4) if the starting symbol also includes the useless variable then the grammer generate empty lang otherwise non empty lang.

Ex)

$S \rightarrow asb \mid sb \mid Sa$

(× × × — all posible productions are useless. Hence it is empty)

2) Finiteness problem :-

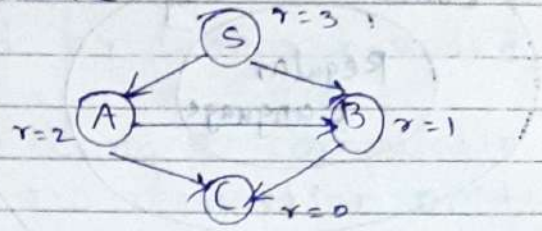
1) checking whether the given grammer generate finite or infinite lang.
2) It is decidable problem in CFG.

Algo 3) simplify the Grammer
4) convert it into CNF form
5) construct the CNF graph
6) If the graph contains any loop or cycle then the grammer generate Finite lang.

→ If the grammer generates finite lang then we can define Rank of the var.

The Rank of variable 'r' i.e the length of the Longest path in the CNF graph starting From that variable

$S \rightarrow AB$
$A \rightarrow Bc \mid a$
$B \rightarrow CC \mid b$
$C \rightarrow a$



★ Membership :-

There exists an algorithm which tells whether given string belongs to given Grammer To check whether the given Grammer generates desired string the derivation tree can be drawn.

$S \rightarrow PQ \mid a \mid b$
$P \rightarrow Xb$
$Q \rightarrow bPP$
$X \rightarrow aaa$

I/P - aaabb
O/P - aaabb generated.

★ Chomsky Hierarchy :-

1) It is used to represent the class of Lang that are accepted by different Machines.
2) The category of Lang in chomsky Hierarchy is given below.

Lang Class	Language	Grammer	Machine	ex
Type-3	Regular	Regular Grammer	FSM: DFA NFA	a^*b^*
Type-2	Context Free	CFG	PDA	a^nb^n
Type-1	Decidable Language	Context sensitive grammer	Linear bounded Automata.	$a^nb^nc^n$
Type-0	Computable lang.	unrestricted Grammer	Turing Machine	$n!$

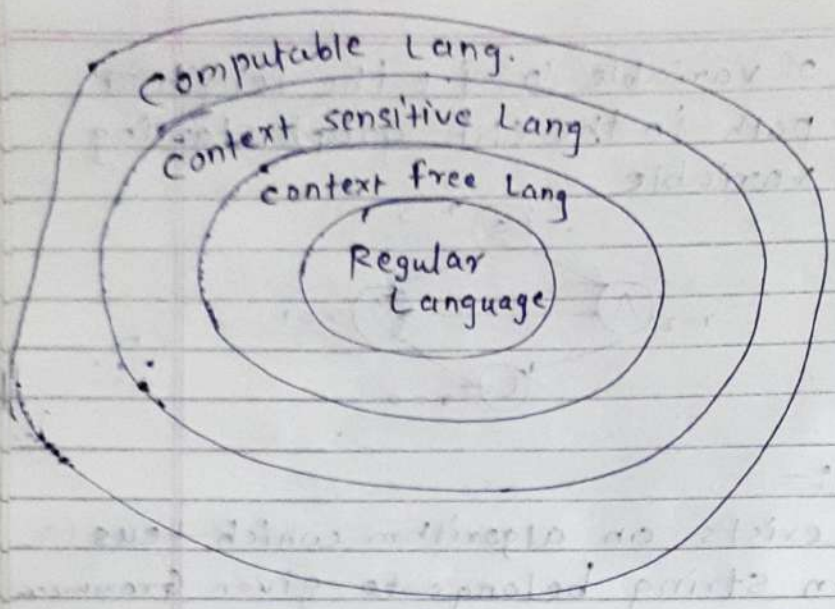


fig Chomsky Hierarchy.

Type-3 Regular Lang.

Those Lang which can be described using regular expressions.

These Lang can be modelled by NFA or DFA.

Type-2 :- context free Grammer

The Lang can be represented by CFG.

The production rule is of the form.

$$A \longrightarrow \alpha$$

These Lang can be modelled by Push down Automata.

Type-1 :- context sensitive Grammer

The context sensitive Grammer are used to represent context sensitive Lang. The following rules.

1) The CSG may have more than one symbol on the left hand side of their production rules

2) The num of symbols on the left hand side must not exceed the number of symbols on the right hand side.

3) It is modelled by Linear bounded automata

Type-0 Unrestricted Lang or computable Lang.
There is no restriction on the Grammer rules of these type of Lang.
These Lang can be effectively modeled by Turing Machine.

★ Application of CFG :-

1) Parsing Techniques :-
   1) CFG is used for parsing the programming constructs and finding the syntatical errors from source program.
2) There are two types.
   A) Top-down parser :-
      When parse tree is constructed from root and expanded to leaves then such type of parser is called as Top-down parser.
   B) Bottom-up parser :-
      When parse tree is constructed from leaves to root is called as Bottom-up parser.

2) XML & Document Type definitions :-
   1) The XML is a special kind of Lang can using which user can define his own Tags.

2) The purpose of XML is not to describe the formatting HTML but to describe the semantics of the text.
3) The document Type definition (DTD) is a kind of context free grammer used to define the structure of XML Document.
4) The DTD has its own notations & variables.

3) Markup languages :-
1) The markup lang. is a family of Lang in which the certain strings have special Meaning.
2) These strings are reffered as 'tags'
3) The most commonly used example of markup lang. is Hyper text markup Lang. (HTML)
4) The CFG is used to describe the structure of HTML document.

★ Cock - Younger kasami Algorithm :-
1) CYK Algorithm is used to decide wheather a string belongs to lang of grammer or not.
2) It is also called as membership algorithm of CFG.
3) CYK algorithm works only on CFG which are in ~~CFG~~ (CNF)

Ex) Check wheather a string "abbb" is a valid member of following CFG.

$S \rightarrow AB$
$A \rightarrow BB \mid a$
$B \rightarrow AB \mid b$

→

Step-I :- Check given Grammer is in form of CNF or not. If it is not then It is converted.
∴ Here Grammer is form of CNF.

Step-II :- string size is '4' then prepare Half matrix for 4×4

	4	3	2	1
1		A	S,B	A
2	S,B	A	B	
3	A	B		
4	B			

a b b b
1 2 3 4

Step-III - Check (11), (22), (33) & (44) position value.
(11) → that position A is fill because 'a' is present at 1 position and also rule is used represent. ('a' is derived from A Grammer)
(22) → B, (33) → B, (44) → B.

Step-IV - check (12), (23) (34) position value.
(12)
(11) (22) ← perform concatenation
A B ← check AB return by which rule ∴ S & B

(23)	(34)
(22)(33)	(33)(44)
B B	B B
A	A

Step-V - Check the position value (1,3), (2,4)

(13) ← means 123

123

(11)(23)

A A ← If it is not present then mark ϕ

[ϕ]

(12)(33)

(SB)(B) ← perform concatination

SB BB

[ϕ A] ← All possible Ans perform union → ϕ∪ϕ∪A → A.

(24) → 2,3,4

(22)(34)

B A

[ϕ]

(23)(44)

A B

[S ϕ B]

Step-VI Check the position value (14)

(14)

1234

(11)(24)

A SB ← perform concatination

As AB ← As no rule is present.

[Xϕ S,A]

(14)

(12)(34)

SB A

SA BA

X X

[ϕ ϕ]

(14)

(13)(44)

A B

[S,B]

The starting symbol present in (14) so it is called as member of CFG.

Case studies:-

1) CFG for palindrome :-

The palindrome is a condition in which we get the same string when read from left to right or from right to left.

CFG

$S \rightarrow aSa \mid bSb$

$S \rightarrow a \mid b \mid \epsilon$

S
aSa
aaSaa
aaϵaa
aaaa

2) CFG for Parenthesis Match :-

Parenthesis match is a match made between opening and closing parethesis.

CFG

$S \rightarrow SS \mid (S) \mid ()$

Ex) string (())()

S
SS
S()
(S)()
(())()