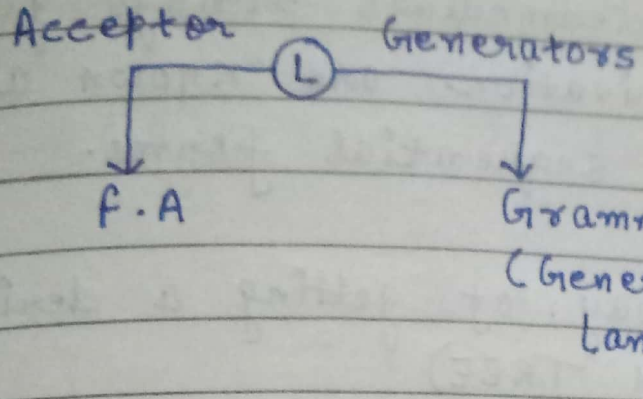
GRAMMARS



→ Definition :-

(V, T, S, P) → Productions

V → variables

T → Terminals

Eg :-

$$\left.\begin{array}{l} S \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{array}\right\} = P$$

Now,

V = {S, B}

T = {a, b}

S = {S}

↳ Start symbol

* Derivation :-

↳ generating a string from the grammar is known as derivation.

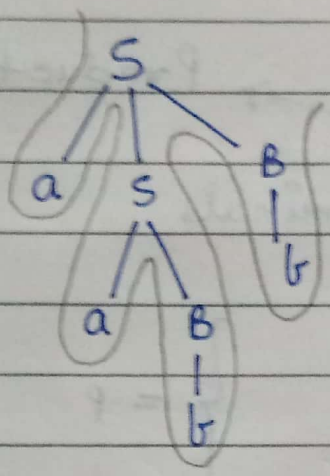
→ Deriving aaabbb :-

$$\left.\begin{array}{l} S \rightarrow aSB \\ \rightarrow aaSBB \\ \rightarrow aaaBBB \\ \rightarrow \# aaabbb \end{array}\right\}$$ Sentential / Sequential forms

↳ left-derivation

* Sentential forms :-
All the intermediate strings generated during derivation are known as sentential forms or sequential forms.

→ Another way of getting a derivation :-
(DERIVATION TREE)



Parse Tree

aabb
↳ yield

* yield of the tree is the string generated by the tree.

• $L = \{aa, ab, ba, bb\}$
↳ set of all strings of length 2

$S \rightarrow aa/ab/ba/bb$

Now,
if the language is finite, giving the grammar is quite simple.

Since,
$\overbrace{(a+b)}^{A}\overbrace{(a+b)}^{A}$
R.E for $L =$
↳ building block

Hence, $A \rightarrow a/b$
$S \rightarrow AA$ } P

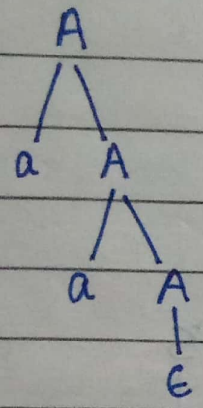
SUCCESS

Dt. _____ B+
Pg. _____

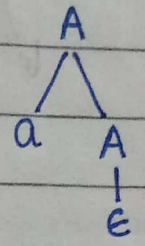
- $L = \{ a^n , n \geq 0 \}$

$\Rightarrow L = \{ a, aa, aaa, \ldots \} \cup \epsilon$

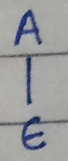
$A \rightarrow aA \mid \epsilon$ (∵ a is the building block)



D.T for {aa}



D.T for {a}



D.T for {ϵ}

* Note :-

Writing any production like $A \rightarrow aA$ is a^*.

Eg :- $A \rightarrow SA \mid \epsilon$

↳ for S^*

or

$A \rightarrow Aa \mid \epsilon$

↳ for a^*

Also,
$A \rightarrow Aa \mid a$
for a^+

- $L = \{ (a+b)^* \}$

$S \rightarrow aS \mid bS \mid \epsilon$

i.e it includes

$S \rightarrow aS \mid \epsilon$

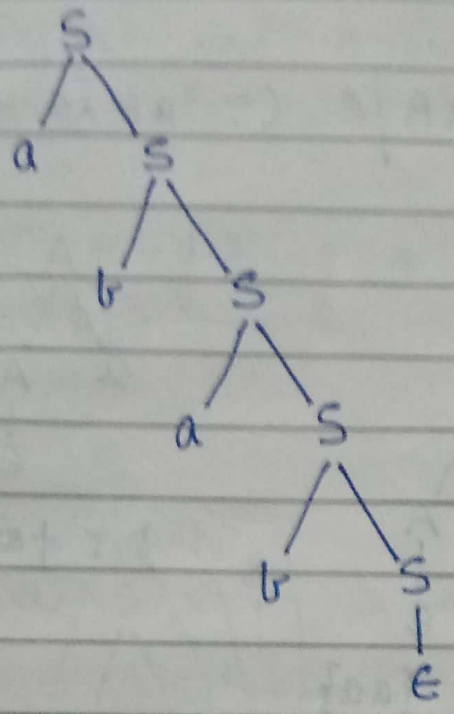
& $S \rightarrow bS \mid \epsilon$

Also, note :-

$(a+b)^* \neq a^* + b^*$

SUCCESS

Eg:- derivation tree for abab :-



- L = { set of all strings of length 2 }
  R.E = (a+b)(a+b) (a+b)* → at least

  B → aB/bB/ϵ [(a+b)*]
  A → a/b (for (a+b))
  S → AAB

  Due to the form of the above R.E.

- L = { set of all strings of ~~of~~ length at ~~least~~ most 2 }

  R.E = (a+b+ϵ)(a+b+ϵ)

  S → AA
  A → a/b/ϵ

Dt. B+
Pg

- L = { set of all strings starting with a and ending with b }

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

A → aA | bA | ∈
S → aAb

- L = { set of all strings starting and ending with different symbols }

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

S → aAb | bAa
A → aA | bA | ∈

Also, for
L = { starting & Ending with same symbol. }
S → aAa | bAb | a | b | ∈
A → aA | bA | ∈ (for (a+b)*)

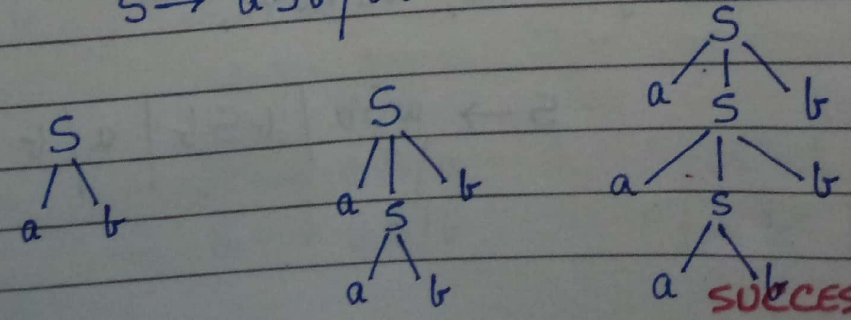
- L = { $a^n b^n$ / n ≥ 1 }
  → is not regular. Hence, R.E can't be given.

Hence, we need some other approach.

S → aSb → so that a and b are always levelled.
S → ab

i.e

S → aSb | ab



Also, note that :-

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

We have,

$$S \rightarrow aSb / \epsilon$$

(IMP) • $L = \{ww^R\} \cup \{waw^R\} \cup \{wbw^R\}$

$L \rightarrow$ set of all strings which are palindromes

where,

$w \in (a, b)^*$

$L_1 = \{ww^R\} \rightarrow$ set of all even length palindromes.

$L_2 = \{waw^R\} \cup \{wbw^R\} \rightarrow$ set of all odd length palindromes over $\Sigma = \{a, b\}$

$$S \rightarrow aSa / bSb / a / b / \epsilon$$

↓

If we generate 'a', then after 'S', there must be one more 'a'.

Also,

for L_1, we have,

$$S \rightarrow aSa / bSb / \epsilon$$

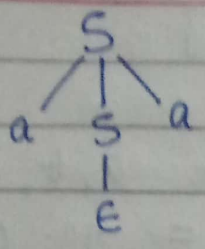
And, for L_2, we have,

$$S \rightarrow aSa / bSb / a / b$$

Dt. B+
Pg.

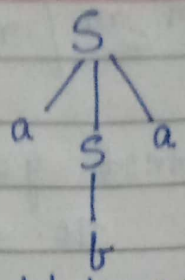
aa

Eg:-



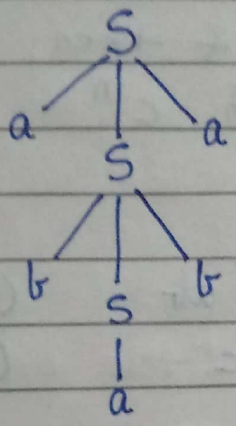
ww^R

aba

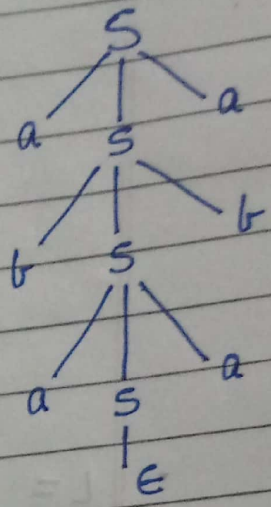


(odd length palindrome)

Also,



ababa



ababaaba

- $L = \{ w \mid |w| \bmod 2 = 0 \}$
  ↳ set of all even length strings
  $R.E = ((a+b)(a+b))^*$

$A \rightarrow a/b$ (for $(a+b)$)
$B \rightarrow AA$ (for $(a+b)(a+b)$)
$S \rightarrow BS/\epsilon$ (for Kleene Closure of B) i.e B^*

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

$S \rightarrow AB$
$A \rightarrow aA/a$ (for a^+ or $a^n, n \geq 1$)
$B \rightarrow bB/b$ (for b^+ or $b^m, m \geq 1$)

SUCCESS

- $L = \{ a^n b^n c^m \mid n, m \geq 1 \}$

Now,

we cannot generate a, b, c independently since, no. of a's = no. of b's

Hence,

We have to generate $a^n b^n$ & then, concatenate it with c^m.

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$ (for $a^n b^n$)

$B \rightarrow cB \mid c$ (for c^m)

- $L = \{ a^n c^m b^n \mid n, m \geq 1 \}$

Here, a's & b's have to be generated together because they have to be matched.

& then, c's have to be generated separately and then, put them in the middle.

VVI

Whenever we want to stop S, we stop it by using A. (STRATEGY)

$S \rightarrow aSb \mid aAb$

$A \rightarrow cA \mid c$ (for c^m, $m \geq 1$)

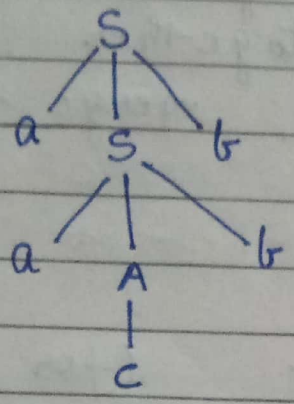
- Note :- $S \rightarrow aAb$

this is for inserting c in b/w.

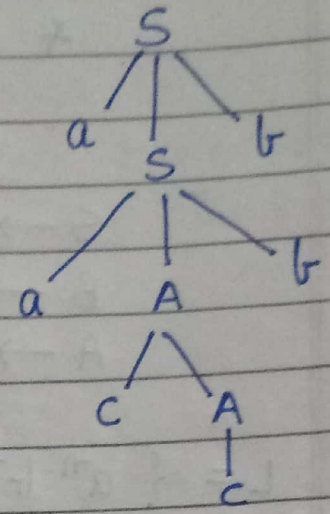
SUCCESS

Dt. B+
Pg.

Eg:- aacbb



aaccbb



- $L = \{ a^n c^m b^n \mid n, m \geq 0 \}$

$S \rightarrow aSb / \epsilon / aAb / A$

$A \rightarrow cA / \epsilon$

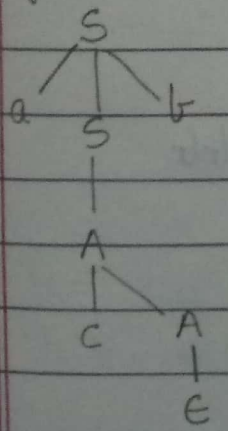
Eliminating redundant productions, We have,

$S \rightarrow aSb / aAb / A$

$A \rightarrow cA / \epsilon$

- Note:- $A \rightarrow cA / \epsilon$ → for $m=0$

Eg:- acb



$S \rightarrow A$ → for $n=0$

$S \rightarrow \epsilon$ → for $m, n=0$

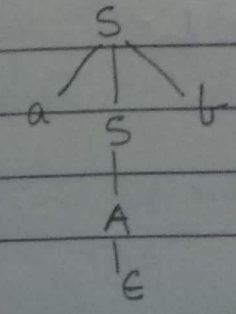
↳ but, $S \rightarrow A$

$\rightarrow \epsilon$ $(A \rightarrow \epsilon)$

Hence, redundant.

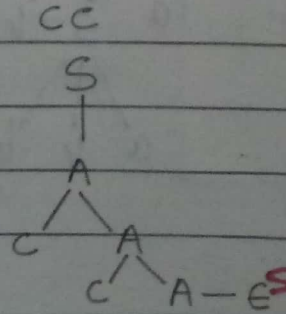
$(m=0)$

ab



$(n=0)$

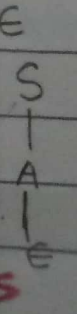
cc



SUCCESS

$(m, n=0)$

ϵ



- $L = \{ a^n b^n c^m d^m \mid n, m \geq 1 \}$

Generate a & b together
& c & d together
& then, merge them.

$S \rightarrow AB$
$B \rightarrow cBd \mid cd$
$A \rightarrow aAb \mid ab$ (for $a^n b^n$, $n \geq 1$)

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

Generate all together.

↳ Grammar is a little bit difficult to design.

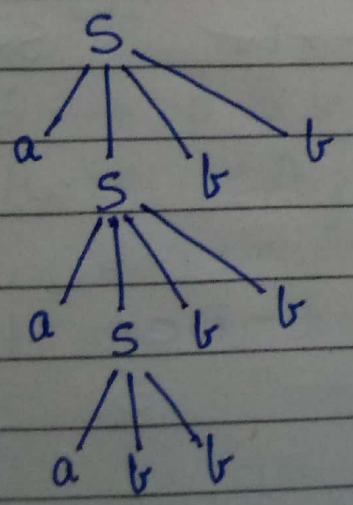
↳ CONTEXT SENSITIVE GRAMMAR

Hence, context-free grammar for the above language is not possible.

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

$S \rightarrow aSbb \mid abb$

Eg :-



aaabbbbbb

Dt. Pg. B+

• $L = \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$

Generate a's & d's together
& generate b's & c's together
then, put b's & c's in between.

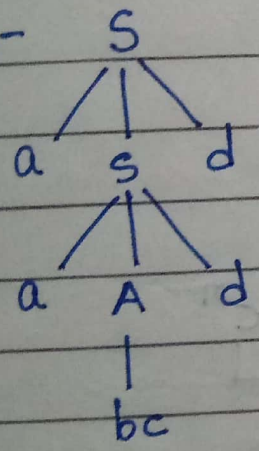
$$S \rightarrow aSd \mid aAd$$
$$A \rightarrow bAc \mid bc$$

(for $b^m c^m / m, n \geq 1$)

Now, note :- $S \rightarrow aSd$
(for generating equal no. of a's & d's)

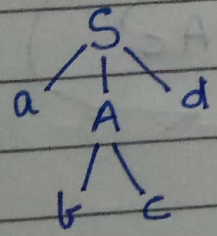
aabcdd

Eg :-



Also,
$S \rightarrow aSd \mid A$
cannot be taken
$\because$ $S \rightarrow A$
$\rightarrow bc$ $(A \rightarrow bc)$
but, the smallest string we require is abcd.

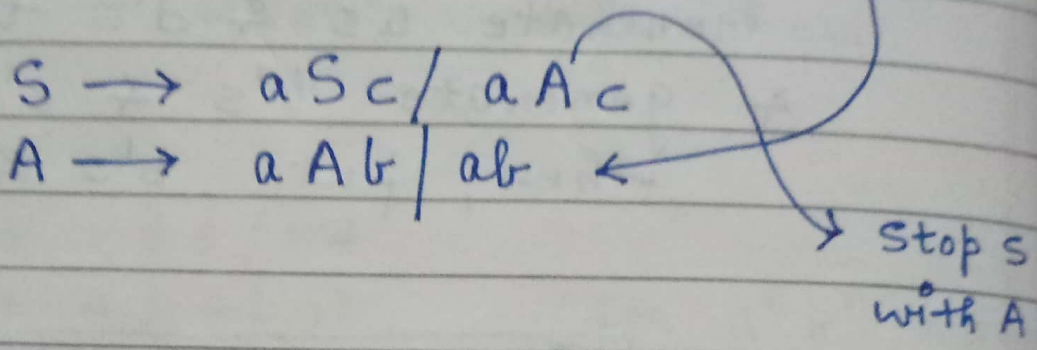
abcd



Hence,
$S \rightarrow aSd \mid aAd$
is taken.

* • $L = \{ a^n b^m c^n d^m \mid n, m \geq 1 \}$
L → difficult to give a context-free grammar.
(To be dealt Later)

- $L = \{ a^{m+n} b^m c^n \mid m, n \geq 1 \}$
  - rewriting it as $a^n a^m b^m c^n$



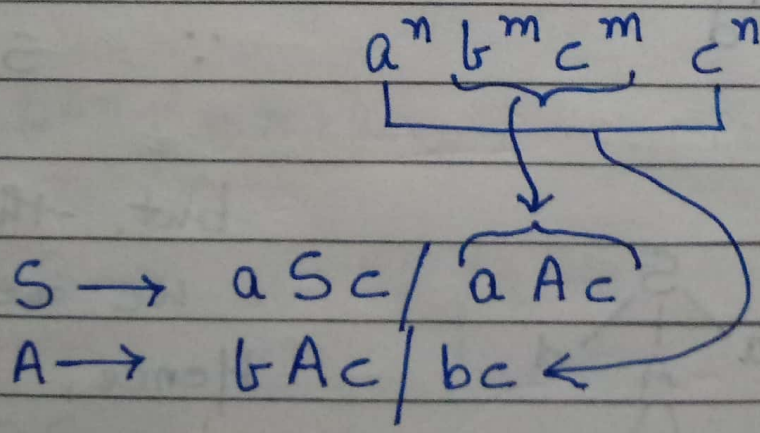
- $L = \{ a^n b^{m+n} c^m \mid n, m \geq 1 \}$
  - i.e $a^n b^n b^m c^m$

$S \rightarrow AB$

$B \rightarrow bBc / bc$

$A \rightarrow aAb / ab$

- $L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$
  - i.e $a^n b^m c^m c^n$



Chomsky's Classification of Grammar :-

* **Type - 3 :-** (Regular Grammars)

$A \rightarrow \alpha B / \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

Right linear Grammar

$A \rightarrow B\alpha / \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

Left linear Grammar

(Regular Grammars)

Eg :-

$A \rightarrow aB / a$ (R.L.G) TYPE-3

$B \rightarrow aB / bB / a / b$

$A \rightarrow Ba / a$ (L.L.G) TYPE-3

$B \rightarrow Ba / Bb / a / b$

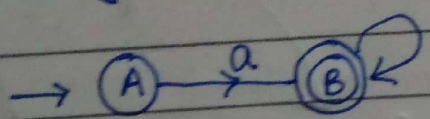
NOT TYPE-3

$A \rightarrow Ba / a$ → Right linear

$B \rightarrow aB / a$ → Left linear

• Conversion of a F.A to a Regular Grammar :-

Eg :-



$A \rightarrow aB$

$B \rightarrow aB / bB / \in$

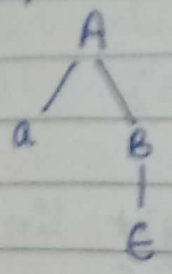
A on seeing 'a' goes to B.

Start symbol is initial state.

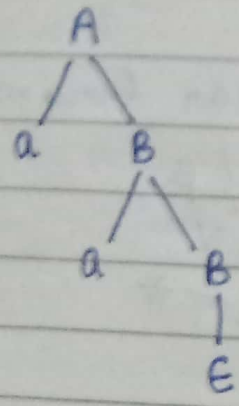
B on seeing 'a' or 'b' goes to B.

IMP → This is to end the derivation. ∵ B is the final state, hence, ∈ is added to it. → SUCCESS

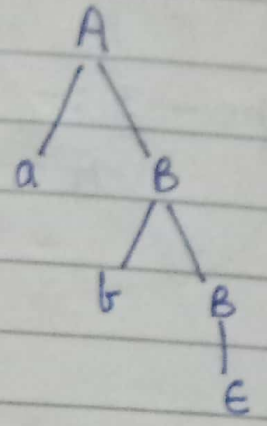
Now, a



aa



ab



Also, we constructed a R.L.G.

* What if we construct a L.L.G for this grammar?

Approach ① :-

We tried to reverse the productions of our R.L.G that we generated to form a L.L.G for the F.A.

Then,

$A \rightarrow aB$

$B \rightarrow aB / bB / \epsilon$

R.L.G

$\xRightarrow{R}$

$A \rightarrow Ba$

$B \rightarrow Ba / Bb / \epsilon$

L.L.G

→ Conclusion :-

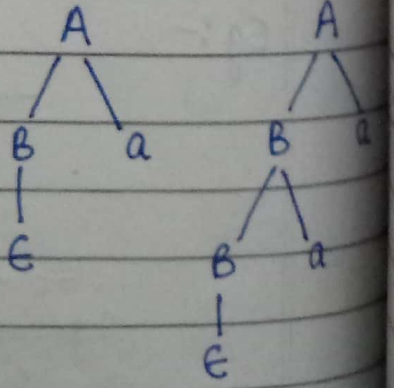
FA (L) → RLG (L)

↓ Reverse

LLG (L^R)

Eg :-

This L.L.G does not represent L.



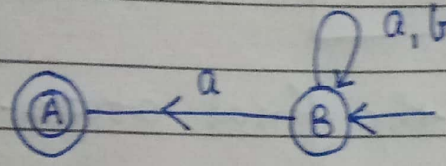
Set of all strings ending in 'a'.

SUCCESS

Now, what should we do to generate a L·L·G for a F·A?

(VVI) F·A (L) $\xrightarrow{R}$ F·A (L^R) $\longrightarrow$ R·L·G (L^R) $\xrightarrow{R}$ L·L·G $(L^R)^R = L$

Now, Step-① :-
Reversing the F·A :-



Step-② :-

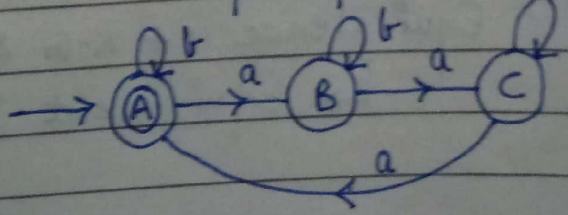
R·L·G giving L^R { $B \rightarrow aB/bB/aA$
$A \rightarrow \epsilon$
→ ∵ it is the final state.

Step-③ :-
Reversing all productions.

L·L·G giving L { $B \rightarrow Ba/Bb/Aa$
$A \rightarrow \epsilon$

- Conversion of R·G to F·A :-

Eg :- $A \rightarrow aB/bA/b$
$B \rightarrow aC/bB$
$C \rightarrow aA/bC/a$
} R·L·G



* How to find the final state?

SUCCESS

(VVI)

** * Method to find final state?

We are given,

$A \rightarrow aB/bA/b$

$B \rightarrow aC/bB$

$C \rightarrow aA/bC/a$

- Just see where the string is going to end.

In place of A, if we place an ε, we get, b.

Hence, A is the final state.

Eg:- If the given R.G is R.L.G, we can directly convert it to F.A.

But, if L.L.G is given then,

L.L.G (L) $\xrightarrow{R}$ R.L.G (L^R) $\rightarrow$ F.A (L^R)

F.A (L^R) $\xrightarrow{R}$ F.A $(L^R)^R$

Hence,

RG $\rightarrow$ F.A

F.A $\rightarrow$ RG

Equivalence of R.G & F.A.

↳ Equivalent in power.

Dt. B+
Pg

If a R.L.G is given,
then, what / how to find the language represented by it?

APPROACH :-

* Convert to f.A & then, check it.

As in our previous example,

$$L = \{ w / n_a(w) \bmod 3 = 0 \}$$

→ NOW,

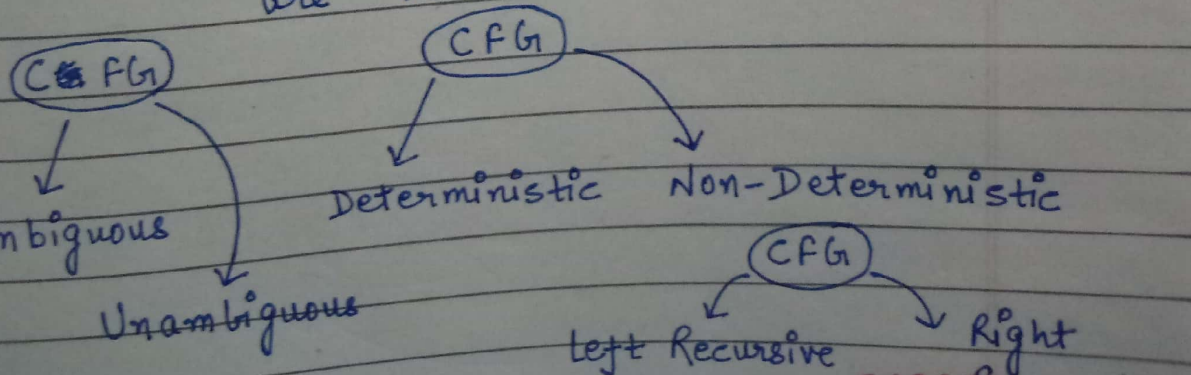
* TYPE-2 :- (Context-free Grammars)

$A \rightarrow \alpha$, $A \in V$
$\alpha \in (V \cup T)^*$

→ It is called context free because wherever we see A in a sentential form, we do not worry about it's context (On which it appears)
↓A

Till now, all grammars we discussed are context-free. They generate context free languages.

- Machine to accept such languages are PDA.



SUCCESS