Why Grammar is required? (AFL)

Grammar It is defined by

$G = \{V, T, P, S\}$ (Set of 4 tuples)

- V → Finite set of non-terminal symbols → all should be written in capital letters.
- T → Set of terminal symbols.
- P → Production rule (formed by combination of non-terminal and terminal symbols).
- S → Starting non-terminal symbol

$G = \{\{S\}, \{a, b\}, P, S\}$

P: $S \to aSb \mid \varepsilon$ → or, → epsilon

Production rule always be written in this way

$\to \{S \to aSb$
$S \to \varepsilon$
Combining this two.

left hand side are mainly used for replacing the right hand side.

$S \to aSb$
$S \to \varepsilon$

$L(G) = \{\varepsilon, ab, a^2b^2, a^3b^3, \ldots a^nb^n\}$

$= \{a^nb^n, \mid n \geq 0\}$ where

$S \Rightarrow aSb$
$\to ab$
$\to aaSbb$
$\to a^2b^2$
$\to aaaSbbb$
$\to a^3b^3$

We can put any of this

$L(G_1) = \{b^na^n \mid n \geq 0\}$

For this the production will be

$S \to bSa \mid \varepsilon$

a^mb^n is regular but a^nb^n is not regular.

Regular language / Grammar → type-3

6) Consider the grammar G: ~~A and S~~

$G = \{\{A, S, B\}, \{a, b\}, P, S\}$

P consist of the following cases:-

Find the language generated by the grammar:-

G_1	G_2	G_3
$S \to aAb$ $A \to aAb \mid \lambda$	$S \to aA$ $A \to bS$ $S \to \lambda$	$S \to Aa$ $A \to B$ $B \to Aa$
$L(G_1) = ?$	$L(G_2) = ?$	$L(G_3) = ?$

G_1

$S \to aAb \to ab$

$\to aaAbb$

$\hookrightarrow a^2b^2$

$L(G_1) = \{ ab, a^2b^2, a^3b^3, \ldots\ldots, a^nb^n \}$

$= \{ a^nb^n, n \geq 1 \}$ or $n > 0$

G_2

$S \to aA$

$A \to bS$

$S \to \lambda$

$L(G_2) = \{ \varepsilon, ab, abab, ababab, \ldots\ldots \}$

$= \{ (ab)^n, n \geq 0 \}$

$S \to aA$

abS

$abaA$

$abab$

$L(G_3) = \{$ This will not generate anything

$= \phi$

G_3

$S \to Aa$

$\to Ba$ } This loop cannot be terminated

$\to Aaa$

:

:

$Aaaaa \ldots$

$S \to Aa$

$A \to B \mid \varepsilon \mid a$ → This will give a language.

$B \to Aa$

Regular Grammar (RG) → It has some specific pattern

$A \to XB$ (T: terminal)

$A \to Bx$

$A \to x$

T → Terminal

NT → Non Terminal.

3 production rule are in grammar

$NT \to T\ NT$

$NT \to NT\ T$

$NT \to T$

} → only then we can say that it is regular grammar (type 3)

a) $S \to aA$ (NT → T NT) ✓
$A \to bA \mid \lambda$ (NT → T NT, NT → T) ✓
$S \to Ba$ (NT → NT T) ✓
$B \to ab$ (NT → T T) ✓

So, all of them are regular

a → Terminal symbol

λ → Terminal symbol

$L = \{a,$

b) $S \to aA$ (NT → T NT) ✓
$A \to bA$ (NT → T NT) ✓
$A \to CB$ (NT → NT NT → NT) ✗
$C \to e$ (NT → T) ✓

It does not belong to type 3 (regular grammar)

→ Iske liye koi production rule nhi hai.

$L = \phi$

$S \to aA$
$\to abA$
$\to abbA$
$\to abbCB$
$\Rightarrow abbeB$ → There is no production rule. (Infinite loop)

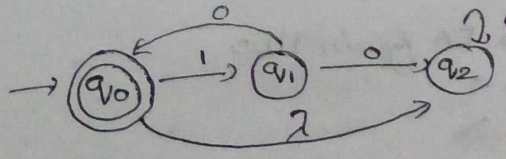
Regular grammar are of two types:-
i) Left linear grammar. (LLG)
ii) Right linear grammar. (RLG)

LLG (position of NT in the left hand side)
$A \to Bx$
1. (NT → NT T)
$A \to x \mid a \mid \lambda \mid b$
2. (NT → T)

RLG (position of NT in the right hand side)
$A \to xB$
1. NT → T NT
$A \to x \mid a \mid \lambda \mid b$
2. (NT → T)

a) Find the left linear and right linear grammar accepted in the following FA (for the following FA)

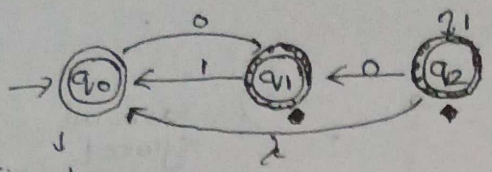
Minimum length String = ε



FA → RLG

$q_0 \to \lambda \mid 1q_1 \mid \lambda \cdot q_2 = q_2$ (T NT, T NT, T NT)
$q_1 \to 0 \cdot q_0 \mid 0 \cdot q_2$ (T NT, T NT)
$q_2 \to 1 \cdot q_2$ (T NT)

So, this is a RLG.



Since here initial and final states are same, they'll remain same.

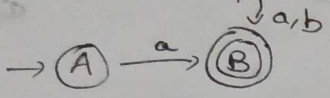
Reverse the FA to get LLG
Initial → final state
final → Initial state
Reverse the arrow heads

$q_0 \rightarrow \lambda \mid 0 \cdot q_1$
$q_1 \rightarrow 1 \cdot q_0$
$q_2 \rightarrow 1 \cdot q_2 \mid \lambda \cdot q_0 \mid 0 \cdot q_1$

↓ Reverse this

$q_0 \rightarrow \lambda \mid q_1 \cdot 0$
$q_1 \rightarrow q_0 \cdot 1$
$q_2 \rightarrow q_2 \cdot 1 \mid q_0 \mid q_1 \cdot 0$

This is the required LLG



Convert this to RLG and LLG

NT → T NT
NT → T
RLG

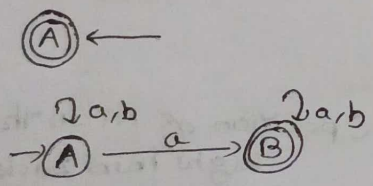
$a(a+b)^*$
Minimum length string = a

$A \rightarrow a \cdot B$ (T NT)
$B \rightarrow a \cdot B \mid b \cdot B \mid \lambda$ } RLG

we need some termination or else it will become infinite loop.

LLG

Reverse



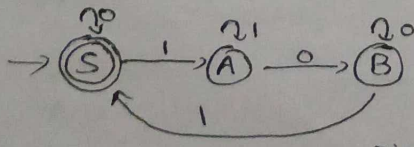
a

$A \rightarrow a \cdot A \mid a \cdot b \cdot A \mid a \cdot B$
$B \rightarrow a \cdot B \mid b \cdot B \mid \lambda$ ✗ not in A because A is not final state.

$S \rightarrow 0S \mid 1A \mid 0$
$A \rightarrow 1A \mid 0B \mid \varepsilon$
$B \rightarrow 0B \mid 1S \mid 1$

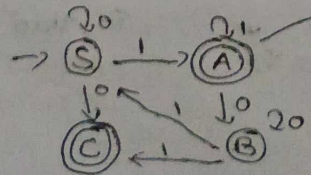
0, 101
10B
101

Construct a FA from this

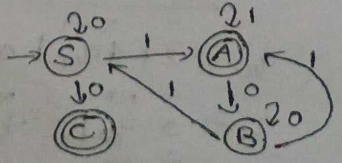


✗

Final state because ε



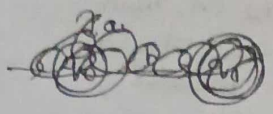
or

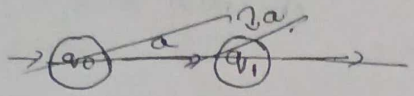


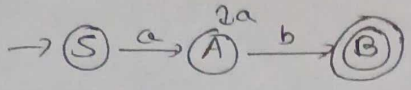
RE (anything) → FA → RLG → REV FA → RLG → LLG

L L L^R L^R L

$a^+b = aa^*b$





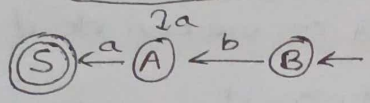


$S \to aA$
$A \to aA \mid bB$
$B \to \lambda$

RLG

or

$S \to aA$
$A \to aA \mid b$



RE

ba^*a

$S \to \varepsilon$
$A \to aA \mid aS$
$B \to bA$

or

$B \to bA$
$A \to aA \mid a$

$S \Rightarrow aA$
$\Rightarrow aaA$
$\Rightarrow aaaA$
$\Rightarrow aaab$

$S \to \varepsilon$
$A \to Aa \mid Sa$
$B \to Ab$

or

$B \to Ab$
$A \to Aa \mid a$

$B \to Ab$
$\to Aab$
$\to Aaab$
$\to aaaab$

If we reverse from this.

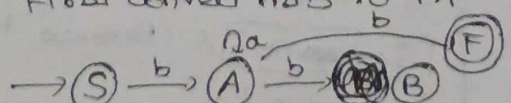
$S \to Aa$
$A \to Aa \mid b$

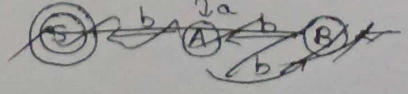
$S \to Aa$
$\to Aaa$
$\to Aaaa$
$= baaa$ → we are getting reversed (so, we can't directly find LLG)

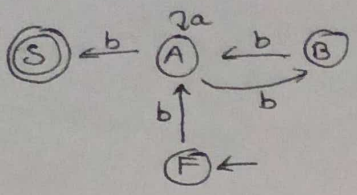
(a) Obtain LLG

$S \to bA$
$A \to aA \mid bB \mid b$
$B \to bA$

First convert this to FA







$F \to bA$

$A \to aA \mid bB \mid b$

$B \to bA$

If LLG is given, convert that to RLG, now construct the FA, then reverse the FA, now calculate the RLG.

$F \to Ab$

$A \to Aa \mid Bb \mid b$

$B \to Ab$

$F \to Ab$
$\to Aab$
$\to$

1) If there is a λ-NFA, convert this to NFA or DFA (Correct the question)

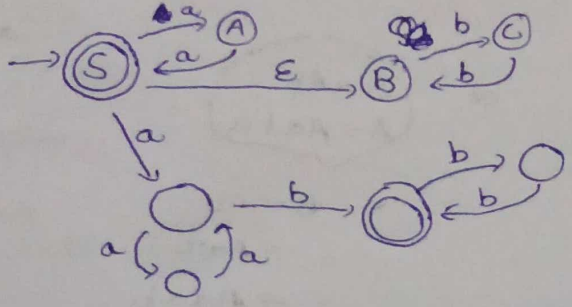
2) Construct a regular grammar when $\{a^n b^m \mid m+n \text{ is even}\}$

Either we can't construct a FA or we can do it directly.

$$RE = \underbrace{(aa)^*(bb)^*}_{E\,E} + \underbrace{a(aa)^* b (bb)^+}_{O\,O}$$

$O+O \to E$
$E+O \to O$
$O+E \to O$
$E+E \to E$

Now convert this to FA



Convert this ε-NFA to 1

n+m is even when

n and m both are even

$n = E, m = 0$
$m = E, n = 0$
$n \& m = E$
a+b

n and m both will be odd

n and m must have to add a & b

$S \to aB$
$B \to baaB \mid \lambda$

ε
aa
aaaa
aaaaaa

bb
bbbb

Starting symbol will be S

Even cases done

$S \to aaS \mid \varepsilon \mid bbA$

$S \to aabbS$ X (Because bbaa

$A \to bbA \mid \varepsilon$

$S \to aaS$
$\to aabbA$
$\to aabb$ ✓

Pattern for odd

a
aaa
aaaaa

b
bbb
bbbbb

$S \to aaS \mid \varepsilon \mid bbA \mid aB$

$S \to bbS$

$A \to bbA \mid \varepsilon$ → we can't include ε here.

$B \to aaB \mid \bullet \mid bC$

$C \to \bullet bbC \mid \varepsilon$

1+9

2+6

1+2 =

2+3=5

0+0 = E

E+E = E

0+E = 0

E+0 = 0

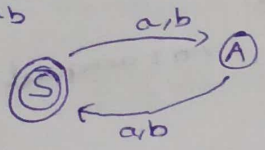
a) $n_a(w) + 3n_b(w)$ is even

a+3b

a+3b	ε
2+0	aa
0+6	bb
1+3	ab
2+6	aabb
3+3	aaab

5+3·1 = aaaaab

abbb = 4

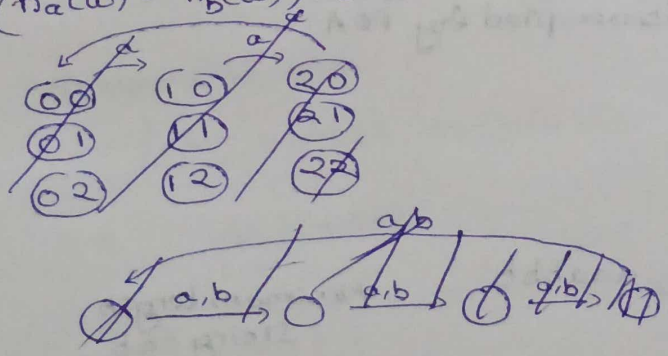


$S \to aA \mid bA \mid \varepsilon$

$A \to aS \mid bS$

0,1,2

b) $(n_a(w) - n_b(w)) \bmod 3 = 1$



(1-0)%3 = 1

a ✓

(0-1)%3 =

3-1 = 2%3 = 2 ✗

(0-2)%3

3-2 = 1%3 = 1 ✓

bb ✓

(0-3)%3

aaaa ✓

3-3 = 0%3 = 0

aaabb ✓

aab ✓

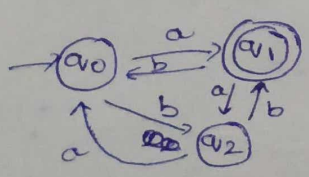
(0-4)%3

(3-4)%3

(-1%3)

2

bbbbb ✓



→ RLG

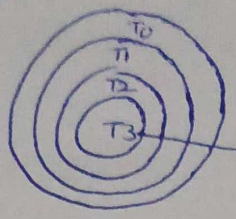
$q_0 \to aq_1 \mid bq_2$

$q_1 \to \bullet\varepsilon \mid aq_2 \mid bq_0$

$q_2 \to \bullet bq_1 \mid aq_0$

CFG → Type 2

(Context Free Grammar)



Chomsky hierarchy

→ Regular grammar used to generate regular language.

$a^n b^n$ → which is not regular

↓

This can be accepted by PDA → Push down automata

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

CFG T2	T3 RG
$S \to B$; $V \to V$ → $A \to x$	$V \to TV$ } Production rule
A → Non Terminal Symbol	$V \to VT$
$A \in V$, $\|A\| = 1$	$V \to T$
$x \in (VUT)^*$	⑤ —ε→ Ⓑ
	$S \to B$ ✗ → not accepted

All T3 grammar must be a part of T2

But all T2 must not be a part of T3

$L = \{a^n b^n \mid n \geq 1\}$ → Not regular [By pumping lemma]

But this is accepted by PDA.

↓ Example of context free language

$A \to x$

$|A| = 1 \quad x \in (VUT)^*$

$S = \{ab, aabb, aaabbb$

Minimum Length String = ab

① $S \to ab$ x

② $S \to aSb$

TVT

$S \to aSb$

$\to aabb$

$\to aaSbb$

$\to aaabbb$

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

$S \rightarrow aSb \mid \varepsilon$

→ Proved earlier that it is not regular.

a) $L=\{\omega c \omega^R \mid \omega \in (a,b)^*\}$ c is a terminal symbol.

CFL

Minimum Length string = c

{c, aca, bcb, abcba, bacab, ...

CFG

$S \rightarrow c \mid aSa \mid bSb$

• c, aca, bcb,
aSa
abSba
abcba

$S \rightarrow bSb$
$\rightarrow baSab$
$= bacab$

a) $L=\{\omega \omega^R \mid \omega \in (a,b)^*\}$

Minimum Length string = ε

{ ε, aa, bb, abba, baab, ...

$S \rightarrow \varepsilon \mid aSa \mid bSb$

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

Minimum Length string = abb

~~{abb, a^2b^4, a^3b~~

= {abb, aabbbb, aaabbbbbb

abb
aSbb ✓
aabbbb ✓

$S \rightarrow abb \mid aSbb$

a) $L=\{a^{2n} b^n \mid n \geq 3\}$

{ aaaaaabbb, aaaaaaaabbbb

$S \rightarrow aaaaaabbb \mid aaSb$

$a^3b^3c^2d^2$

a) $L=\{\underbrace{a^n b^n}_{A_1} \underbrace{c^m d^m}_{A_2} \mid n,m \geq 1\}$

{ abcd, aabbcd, abccdd, aabbccdd

$a A_1 b$
aabb
aaabbb A_2
$a^3b^3 c A_1 d$
a^3b^3ccdd

$S \rightarrow aSb \mid asd \mid A_1 A_2$

$A_2 \rightarrow cA_2d \mid cd$

$A_1 \rightarrow aA_1b \mid ab$

$L = \{ a^i b^j c^k \mid i=j \text{ or } j=k;\ i,j,k \geq 1 \} \rightarrow$ CFL

CFG ?

$L = \{$ abc, ~~abc~~, abcc, aabc, aabbc, ... $\}$

~~$S \rightarrow A_1$~~

abc $i=j$: aabbc (A_1 B_1)

abc $j=k$: abbcc (A_2 B_2)

$S_1 \rightarrow A_1 B_1$
$A_1 \rightarrow aA_1b \mid ab$
$B_1 \rightarrow cB_1 \mid c$

$S_2 \rightarrow A_2 B_2$
$A_2 \rightarrow aA_2 \mid a$
$B_2 \rightarrow bB_2c \mid bc$

$S = S_1 \mid S_2$

$L = \{ 0^m 1^n \mid m \neq n,\ m,n \geq 0 \}$

$m > n$ or $m < n$

~~{ab, aab, aaab, abbb,~~

{011, 001, 0001, 0111, ... }

~~$S \rightarrow A_1 B_1$~~
~~$A_1 \rightarrow 0A_1 \mid 0$~~
~~B_1~~

011
001
0001

$S_1 \rightarrow A_1 B_1$
$A_1 \rightarrow$ ~~0A1bb \| 0A111~~ $\mid 0A_1 \mid 0$
$B_1 \rightarrow 0B_1 1 \mid 01$

$m > n$: 001, 0001 (A_1 B_1)

$m < n$: 011, 0111 (A_2 B_2)

$S_2 \rightarrow A_2 B_2$
$A_2 \rightarrow 0A_2 \mid 01$
$B_2 \rightarrow 1B_2 \mid 1$

$S \rightarrow S_1 \mid S_2$

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

abcd, abbccd

{abcd, aabcdd, abbccd,

$S \rightarrow$ [illegible] $\mid aSd$

$A_2 \rightarrow$ [illegible] $bA_2c \mid bc$

Ambiguity / Ambiguous grammar → confusion type / Random

If we can generate more than 1 leftmost derivation tree / rightmost derivation tree / par

A grammar is ambiguous when more than one leftmost derivation / rightmost derivation tree can be generated.

$S \to aS \mid AS \mid A$

$A \to AS \mid a$

LD
Leftmost derivation

$S \to \underline{A}S$ (A will be replaced)

$\to \underline{A}SS$

$\to a\underline{S}S$ (leftmost non-terminal)

$\to a\underline{A}S$

$\to aa\underline{S}$ (leftmost non-terminal)

$\to aaa$

RD
Rightmost derivation.

If more than 1 RD, ambiguous

$S \to A\underline{S}$ (S will be replaced.)

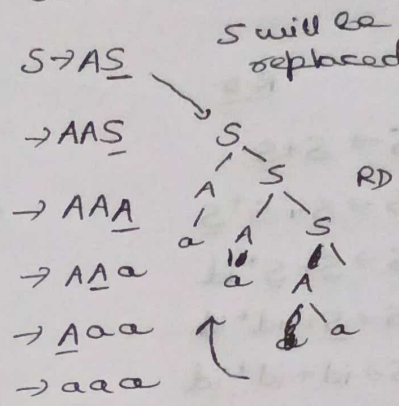
$\to AA\underline{S}$

$\to AAA$

$\to AA\underline{A}a$

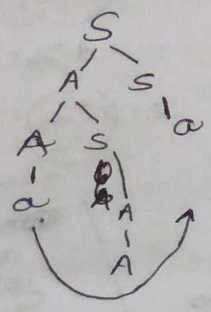
$\to A\underline{A}aa$

$\to aaa$



LD

If more than 1 RD, ambiguous.



Check whether $S \to aS \mid AS \mid A$, $A \to AS \mid a$ is ambiguous?

Case-1

L.D.

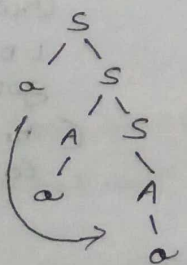
$S \to a\underline{S}$

$\to a\underline{A}S$

$\to aaS$

$\to aaA$

$\to aaa$



LD

Case-2 :

$S \to \underline{A}S$

$\to A\underline{A}S$

$\to aaS$

$\to aAS$

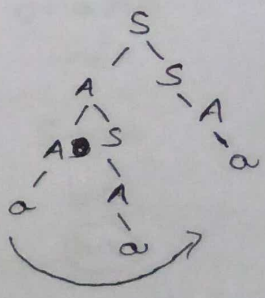
~~$\to aaaA$~~

~~$\to aaaa$~~

$\to aa\underline{S}$

$\to aa\underline{A}$

$\to aaa$



2 L.D. tree → more than one

So, the grammar is ambiguous.

Case-3 $S \to A$

$\to AS$

$\to ASS$

$\to aSS$

$\to aAS$

$\to aaS$

$\to aaA$

$\to aaa$

3rd L.D. is also possible.

So, the grammar is ambiguous.

$S \to S+S \mid S*S \mid id$

$T = \{+, *, id\}$

Construct

$id + id * id$

Terminal symbols = +, *, id

Non-Terminal Symbols = S

R.D.

$S \to S+\underline{S}$

$S \to S+S*\underline{S}$

$S \to S+\underline{S}*id$

$S \to \underline{S}+id*id$

$S \to id+id*id$

R.D.

~~$S \to S*\underline{S}$~~

~~$S \to S*S+\underline{S}$~~

~~$S \to S+\underline{S}$~~

~~$S \to \underline{S}+id$~~

~~$S \to \epsilon$~~

$S \to S*\underline{S}$

$S \to \underline{S}*id$

$S \to S+\underline{S}*id$

$S \to \underline{S}+id*id$

$S \to id+id*id$

So, the grammar is ambiguous.

L.D.

$S \to \underline{S}*S$

$S \to \underline{S}+S*S$

$S \to id+\underline{S}*S$

$S \to id+id*\underline{S}$

$S \to id+id*id$

LD

$S \to \underline{S}+S$

$S \to id+\underline{S}$

$S \to id+\underline{S}*S$

$S \to id+id*\underline{S}$

$S \to id+id*id$

$(2+2)*2$

$2+2$

$4*2 = 8$

$2+(2*2)$

$2+2$

$= 4+2 = 6$

~~We~~ are getting different answers due to ambiguity

6 is the actual answer

But using L.D. we're getting 8.

(This is due to ambiguity).

Simplification of CFG

i) $P \to \varepsilon$ → Remove null production rule

ii) $P \to K$ → Remove unit production. $P \to KK$ → Not a unit production

(NT → NT)

iii) Removal of useless production rule:

$S \to aS \mid a$

$B \to a$ → useless production rule.

Without comprising the results, we need to delete production rule.

a) $S \to aA \mid aBB$
$A \to aaA \mid \varepsilon$
$B \to bB \mid bbC$
$C \to B$

$S \to aA$ → put ε = $S \to a$
$A \to aaA$ → put ε $S \to aa$

① ε (Removal of null production)

$S \to aA \mid aBB \mid a$
$A \to aaA \mid aa$
$B \to bB \mid bbC$
$C \to B$ → find if B implies anything exist → then copy

② (Remove the unit production)

~~$C \to bB \mid bbC$~~

$S \to aA \mid aBB \mid a$
$A \to aaA \mid aa$
$B \to bB \mid bbC$
$C \to bB \mid bbC$

③ (Remove the useless production)

$S \to aA \mid a$
$A \to aaA \mid aa$

$S \to aBB$
$\to abB\,B$
$\to abB\,bbC$ } infinite loop.
$\to$

So, we'll remove B and C

a) $S \to AB \mid CA$
$S \to BC \mid AB$
$A \to a$
$C \to aB \mid b$

No null and unit production exist

$S \to \underline{A}B$
$\to a$

$S \to \underline{C}A$
bA
(ba)

(iii) $S \to CA$
~~$S \to AB$~~
$A \to a$
$C \to b$

ba → resultant string

a) S → AB | ab
A → aA | B | a
B → D | E

No null production

(ii) S → AB | ab
A → aA | D | E | a
B → D | E

A → D | E

(iii) S → ab
~~A → aA | a~~

S → ab

b) S → aS
S → AB
A → ε
B → ε
D → b

(i) Remove Null production.

S → aS
S → ε | A | B
D → b

⇒

S → aS | a
~~D → b~~
S → AB | B | A
D → b

S → aS
S → AB

(iii) S → aS | a

c) S → AB
A → a
B → C | b
C → D
D → E
E → a

(ii) Remove Unit production

B → C → D → E → a

S → AB
A → a
B → a | b

d) S → A | bb
A → B | b
B → S | a

(ii) Remove unit production.

S → A → B → S →

S → A B | b | bb
A → S | a | b
B → S | a

⇒

S → S | a | b | bb
A → S | a | b

⇓

S → a | b | bb

S → bb | b
A → b | a | bb
B → a | bb | b → Useless production

⇓

S → a | bb | b

CNF (Chomsky normal form) : It is a type of CFG with some more restrictions.

CFG : NT → (T ∪ NT)*

CNF ⇒ NT → NT NT
NT → T

S → aSb → CFG but not CNF

GNF (Greybach normal form) NT → T is common in CNF and GNF

GNF ⇒ NT → TNT*

S → aABBC (example of GNF)

S → AB } CNF
S → a

CFG to CNF ~~or CFG to GNF~~

i) First Simplify the CFG
ii) Check whether it is following the pattern of CNF, if not update the production rule.

S → AB | aB
A → aab | ε
B → bbA

① ~~Simp~~ Remove null production. (ε Removal)

~~S → aaB~~
~~S → B~~
~~→ bbA~~

S → AB | aB | B
A → aab
B → bbA | bb

② Unit removal

S → AB | aB | bbA | bb
A → aab → only this is following CNF (nothing else)
B → bbA | bb

③ Remove useless production.
(No useless production) → Since, all of them have terminating conditions.

$X \to a \quad Y \to b$ → because it follows CNF pattern.

$S \to AB \mid XB \mid YYA \mid YY$

$A \to XXY$ → not following.

$B \to YYA \mid YY$

$C \to XX \quad D \to YY$

$S \to AB \mid XB \mid DA \mid YY$ → we'll not replace it with D, because it'll violate.

$A \to CY$

$B \to DA \mid YY$

$X \to a$

$Y \to b$

So, actually we'll replace it with non-terminal symbols.

CFG to CNF

b) $S \to bA \mid aB$

$A \to bAA \mid aS \mid a$

$B \to aBB \mid bS \mid b$.

bA

~~ba~~ ~~bbAA~~

① No null production

② No unit production.

③ No useless production.

So, it is already simplified.

$X \to a$, $Y \to b$ (Here, we can't take this because $NT \to T$ is present in this question)

$S \to$ ~~XB XB~~ $BA \mid AB$

$A \to BAA \mid AS \mid a$

$B \to ABB \mid BS \mid b$

$C \to AA$, $D \to BB$

$S \to BA \mid AB$

$A \to BC \mid AS \mid a$

$B \to AD \mid BS \mid b$

or

$S \to BA \mid AB$

$A \to SA \mid AS \mid a$

$B \to SB \mid BS \mid b$

a) $S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \varepsilon$

① Remove null production → No need to write: $S \to S$

$S \to ASA \mid aB$ ~~$aB \mid B$~~ $\mid S \mid SA \mid AS \mid a$

$A \to B \mid S$ ~~$\mid \varepsilon$~~

$B \to b$ ~~$\mid \varepsilon$~~

② Remove unit production

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

$A \to b \mid ASA \mid aB \mid BSA \mid AS \mid a$

⑤ Remove useless production
(No useless production)

~~S → A | A~~

S → (ASA) | AB | SA | AS | a
A → b | (ASA) | SB | SA | AS | a
B → b

→

S → AA | AB | SA | AS | a
A → b | SA | SB | SA | AS | a
B → b

⇓

S → AA | AB | SA | AS | a
A → b | SA | SB | AS | a
B → b

(It comes in exam with left recursion)

GNF (Greybach normal form)

A → Aα | β

Left recursion
Aαααα.....

If there is any left recursion in GNF, remove left recursion.

CFG to GNF
- A → βA'
- A' → ε | αA'
- No ⇒ the left recursion concept is broken

Step-1
: Simplify the CFG

Step-2
Check whether it follows the pattern NT → TNT*
If it doesn't follow, introduce new variables and convert it.

Step-3
Remove left recursion.

⑥ S → ABb | a
A → aaA | B | b
B → bAb

Step-1
① No null production
② Remove unit production
S → ABb | a
A → aaA | bAb | b
B → bAb
(only these two are following the pattern.)
③ Remove useless production
No useless production.

Step-2
~~NT → T NT*~~

A → Aα | β → First check left recursion.
No left recursion

Step-3

Check the pattern $NT \rightarrow TNT^*$

$S \rightarrow aAABb \mid bAbBb \mid bBb \mid a$

$A \rightarrow aA \mid bAb \mid b$

$B \rightarrow bAb$

$S \rightarrow aSABA \mid bAABA \mid bBA \mid a$ → Now everything follows the pattern

$A \rightarrow aSA \mid bAA \mid b$

$B \rightarrow bAA$

q) $S \rightarrow AB$

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

Step-1

① No null production

② No unit production

③ No useless production.

Step-II - Check and remove left recursion.

No left recursion.

Step-III Check for pattern $NT \rightarrow TNT^*$

$S \rightarrow aAB \mid bBB \mid bB$ → We need to check for every possibilities.

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

6) $S \rightarrow CA \mid BB$

$B \rightarrow b \mid SB$

$C \rightarrow b$

$A \rightarrow a$

Step-1

① No null production.

② No unit production.

③ No useless production.

Step-II Check for pattern.

$S \rightarrow bA \mid BB$

$B \rightarrow b \mid SB$

$C \rightarrow b$ ⟶ useless production

$A \rightarrow a$

⇓

$S \rightarrow bA \mid BB$

$B \rightarrow b \mid SB$

$A \rightarrow a$

Case-1

Replace B

$S \rightarrow bB \mid SBB$ → left recursion

Case-2

Replace S

$B \rightarrow bAB \mid BBB$

Remove the left recursion.

we can chose anyone of the two cases:-

$A \to a$

$S \to bA \mid BB$

$B \to b \mid bAB \mid BBB$

$\beta \quad \beta_1 \quad \beta_2 \quad \alpha \quad \alpha$

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \varepsilon \mid \alpha A'$

$\Downarrow$

$B \to b\beta' \mid bABB' \mid$

$\beta_1 \quad \alpha' \quad \beta_2 \quad \alpha'$

$B' \to \varepsilon \mid BBB'$

$\downarrow$ Remove null production.

$B \to bB' \mid bABB' \mid b \mid bAB$

$B' \to BBB' \mid BB$

$S \to bA \mid BB$

$A \to a$

check the pattern

$B \to bB' \mid bABB' \mid b \mid bAB$

$B' \to bB'BB' \mid bABB'BB' \mid bBB' \mid bABBB'B \mid bBB' \mid bABB$

All of them are following the pattern ✓

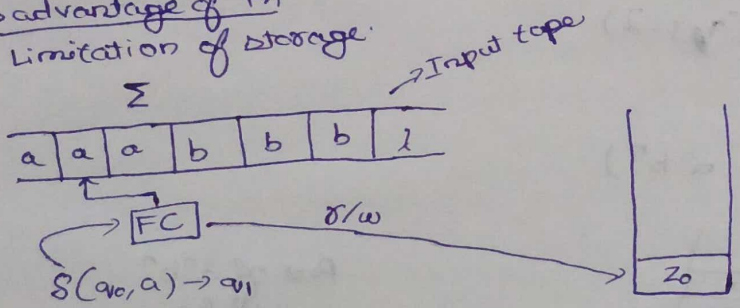
$S \to bA \mid bBB' \mid bABB'B \mid bBB \mid bABB$

$A \to a$

PDA (Push down automata) :- Context free Language is accepted by PDA.

Disadvantage of FA

Limitation of storage.



PDS (Push down store)

FC will read one alphabet from the input tape and checks the topmost element of the stack and perform two operations push or pop.

$PDA = \{Q, \Sigma, q_0, Z_0, F, \delta, \gamma\}$ ($\gamma \to$ Push down store symbol)

$Z_0 \to$ Default push down store symbol

Total 7 tuples

For FA

$Q \times \Sigma \to Q'$ → new state or same state

$\delta \to$

$Q \times \{\Sigma \cup \varepsilon\} \times Y \to Q' \times Y$

PS → a/ε → stack element

→ because when we processed the whole string, we need epsilon

q_0 q_1

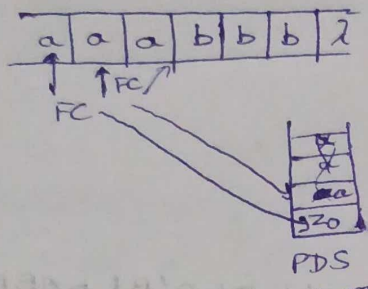
$\delta(q_0, a, Z_0) \rightarrow (q_1, aZ_0)$

Q) Design a PDA to accept $\{a^n b^n / n \geq 1\}$ → Context Free language.

CFG = ?

$S \rightarrow ab \mid aSb$

$= \{ab, aabb, aaabbb, \ldots\}$



$\delta(q_0, a, Z_0) \rightarrow (q_1, aZ_0)$ → topmost element not pop

$\delta(q_1, a, a) \rightarrow (q_1, aaZ_0)$ → we can write or avoid it.

a^{n-1}

~~$\delta(q_1, a, a) \rightarrow$~~

→ pop

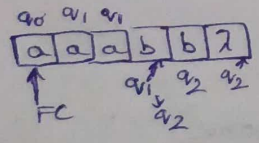
$\delta(q_1, b, a) \rightarrow (q_2, \lambda)$

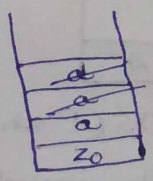
$\delta(q_2, b, a) \rightarrow (q_2, \lambda)$ → loop

$\delta(q_2, \lambda, Z_0) \rightarrow (q_f, \lambda)$

Q) aaabb

a^3b^2 (It does not follow $a^n b^n$)





Rest of $a^n b^n$ will be rejected.

q_{dead}

$\delta(q_2, \lambda, a) \rightarrow$ not able to reach to the final state.

Q) bbaa

$\delta(q_0, b, Z_0) = ??$ → It will not accept anything because we have not defined such transition function.

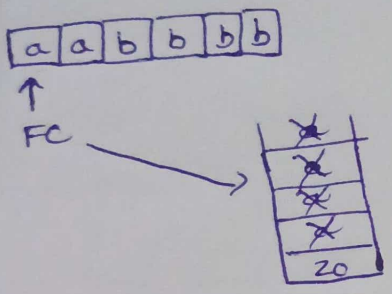
The main advantage of this PDA is memory.

n-elements can be pushed at a time but only 1 element can be popped at a time.

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

~~aaabbbb~~

aabbbb



$\delta(q_0, a, Z_0) \rightarrow (q_1, aZ_0)$

$\delta(q_1, a, a) \rightarrow (q_1, aaaaZ_0)$

$\delta(q_1, b, a) \rightarrow (q_2, \lambda)$

$\delta(q_1, b, a) \rightarrow (q_2, \lambda)$ → loop.

$\delta(q_2, \lambda, Z_0) \rightarrow (q_f, \lambda)$

- pop ~~a~~ can be done on