

Symbol - Smallest building block e.g. (a, b, 0, 1)

Alphabet - Collection of symbols (Σ)
e.g.: $\Sigma = \{a, b\}$

String - A finite sequence of symbols from the alphabets. e.g.: $w = aab$, $|w| = 3$
 $w^3 = aab aab aab$.

Null String - ϵ (or λ or Λ)

Σ^* → set of strings of any length from Σ
 $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$
 Σ^k → set of strings of length k from Σ
 $\Sigma^1 = \{a, b\}$
 $\Sigma^2 = \{aa, ab, ba, bb\}$
 $\boxed{\Sigma^+ = \Sigma^* - \epsilon}$

w^R → Reverse of a string

e.g.: $w = aab$, $w^R = baa$

Language - A language over Σ is a subset of Σ^* \textcircled{OR} collection of strings over Σ

e.g.: $L = \{0, 1\}^*$: length of string is even
 $= \{\epsilon, 00, 01, 10, 11, 0000, 0001, \dots\}$

→ Infinite language

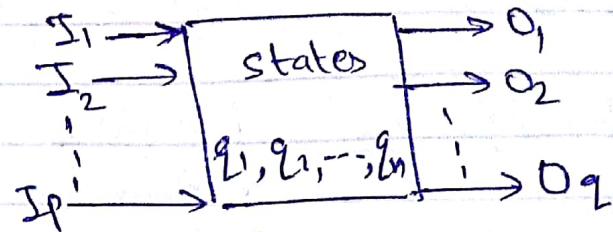
e.g.: $\Sigma = \{a, b\}$, $L = \{a^n b^n : n \geq 0\}$

$L = \{\epsilon, ab, aabb, aaabb, \dots\}$

abb or gab do not belong to language as they do not satisfy condition

→ Characteristics of Automata

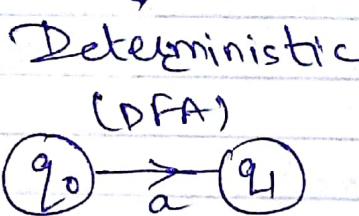
- ① I/P
- ② O/P
- ③ states
- ④ State Relation
- ⑤



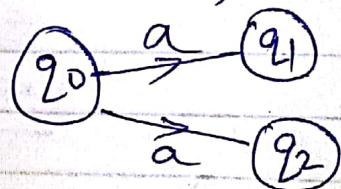
(Model of a discrete automata)

→ Finite Automata : Finite states.

FA



Non-Deterministic
(NFA)



→ DFA (Deterministic finite Automata)

- 5 tuple $(\emptyset, \Sigma, \delta, q_0, F)$
where

\emptyset → finite non-empty set of states

Σ → I/P alphabet

δ → Transition function

$q_0 \in \emptyset$, initial state

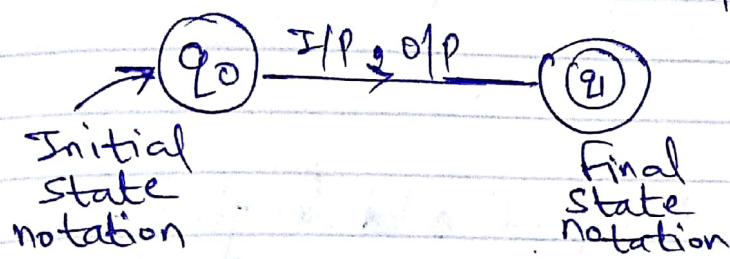
$F \subseteq \emptyset$, set of final states

DFA transition function,

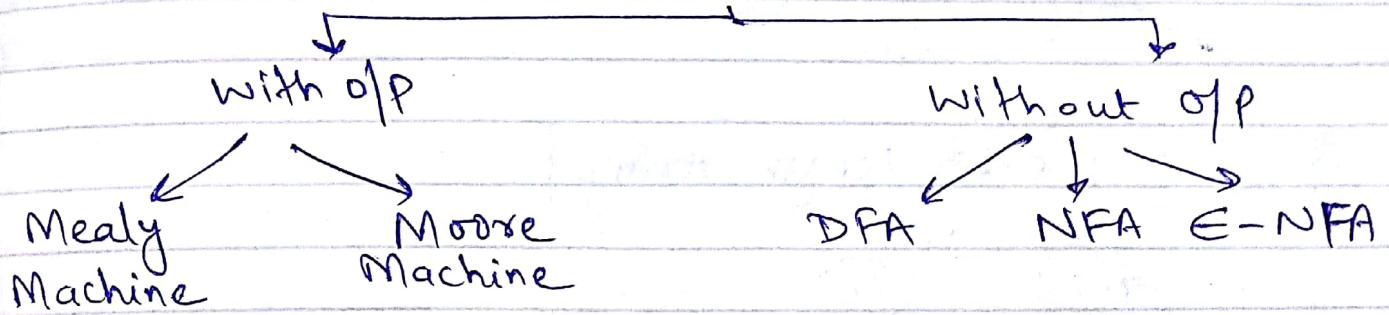
$$\delta : \emptyset \times \Sigma \rightarrow \emptyset$$

→ Transition System

It is a finite directed labelled graph



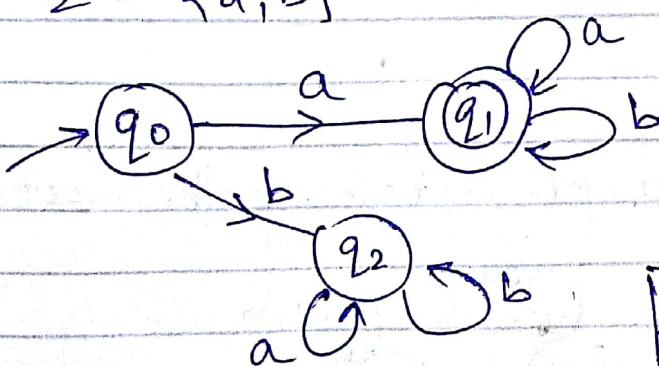
Finite Automata



→ DFA

Language acceptor when it accepts all the strings that are in language and rejects all the strings that are not in language

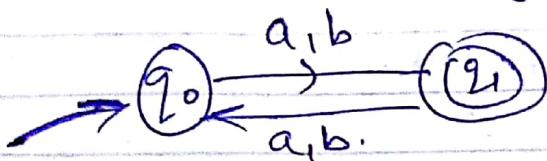
- L = { Set of all strings that starts with 'a' }
 $\Sigma = \{a, b\}$



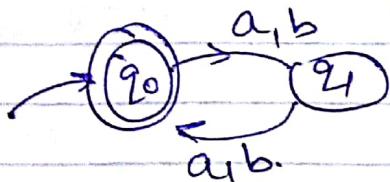
States	a	b
q0	q1	q2
q1	q1	q1
q2	q2	q2

* Finite language will always have finite automata but infinite language may or may not have finite automata

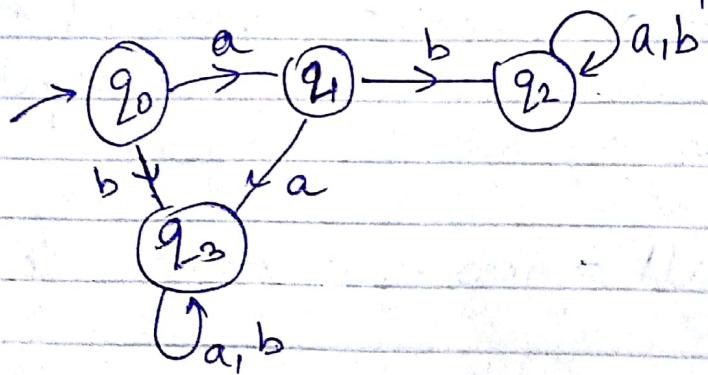
Q. $L = \{ \text{Set of all strings of odd length} \}, \Sigma = \{a, b\}$



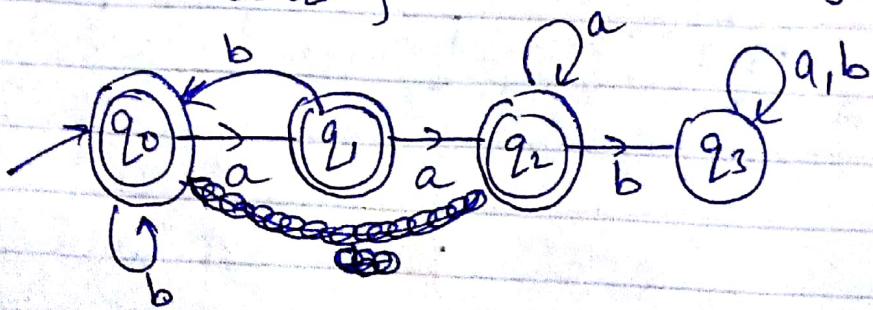
Q. $L = \{ \text{Set of even length strings} \}$



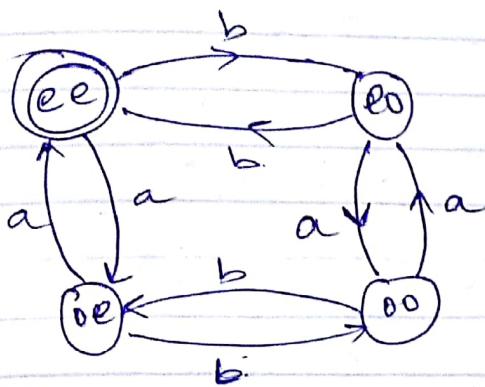
Q. $L = \{ \text{Set of all strings with prefix ab} \}$



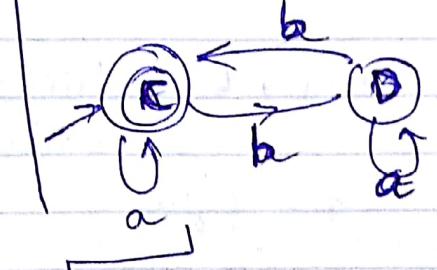
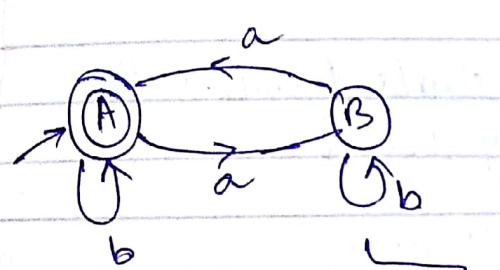
Q. $L = \{ \text{Strings except containing the substring aab} \}$



Q. $\omega = \{a, b\}^*$, $M_A(\omega) \cong 0 \pmod{2}$
 $M_B(\omega) \cong 0 \pmod{2}$



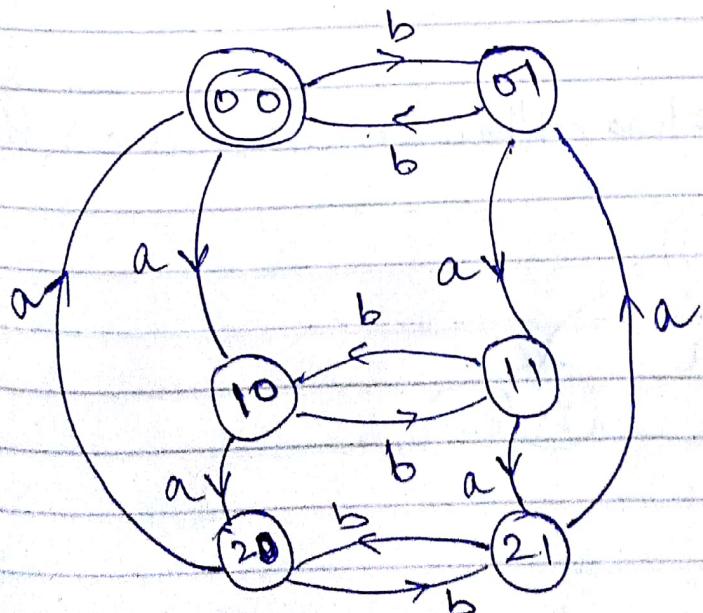
This can also be done using cross product
 $M_A(\omega) \cong 0 \pmod{2}$ $M_B(\omega) \cong 0 \pmod{2}$



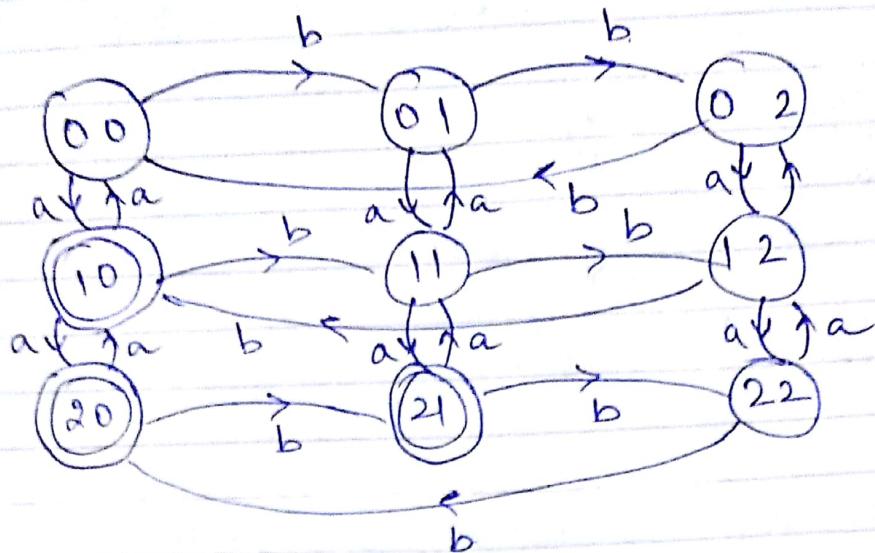
$$\{A, B\} \times \{C, D\} = \{AC, AD, BC, BD\}$$

AC will be final state because both A & C are final states. Similarly, with initial states.

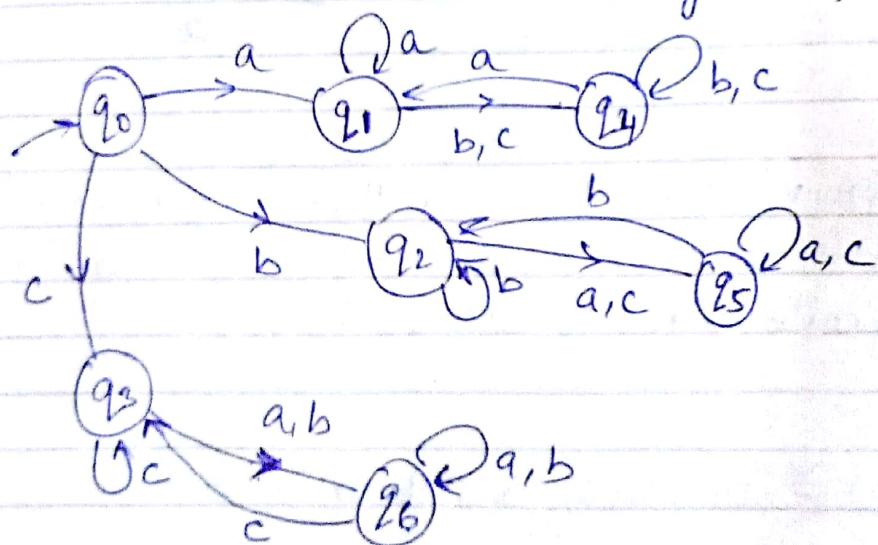
Q. $M_A(\omega) \cong 0 \pmod{3}$, $M_B(\omega) \cong 0 \pmod{2}$



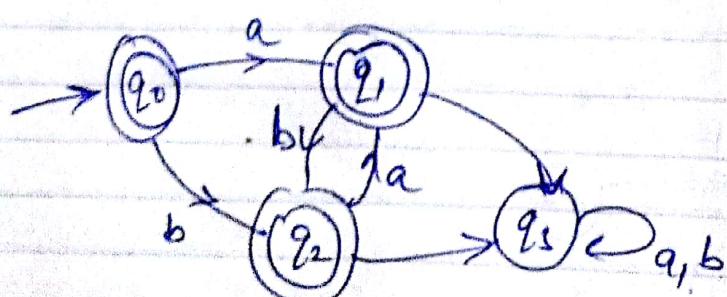
Q. $m_a(w) \bmod 3 > m_b(w) \bmod 3$



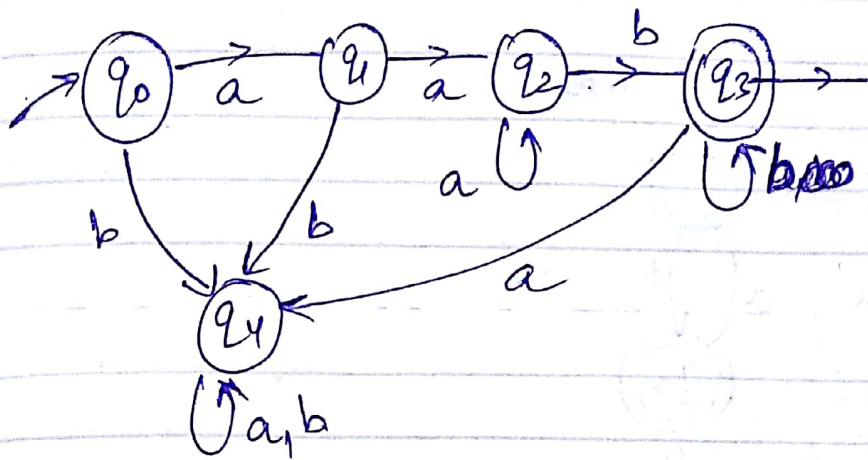
Q. $\Sigma = \{a, b, c\}$, $L = \{ \text{strings that starts & ends with different symbols} \}$



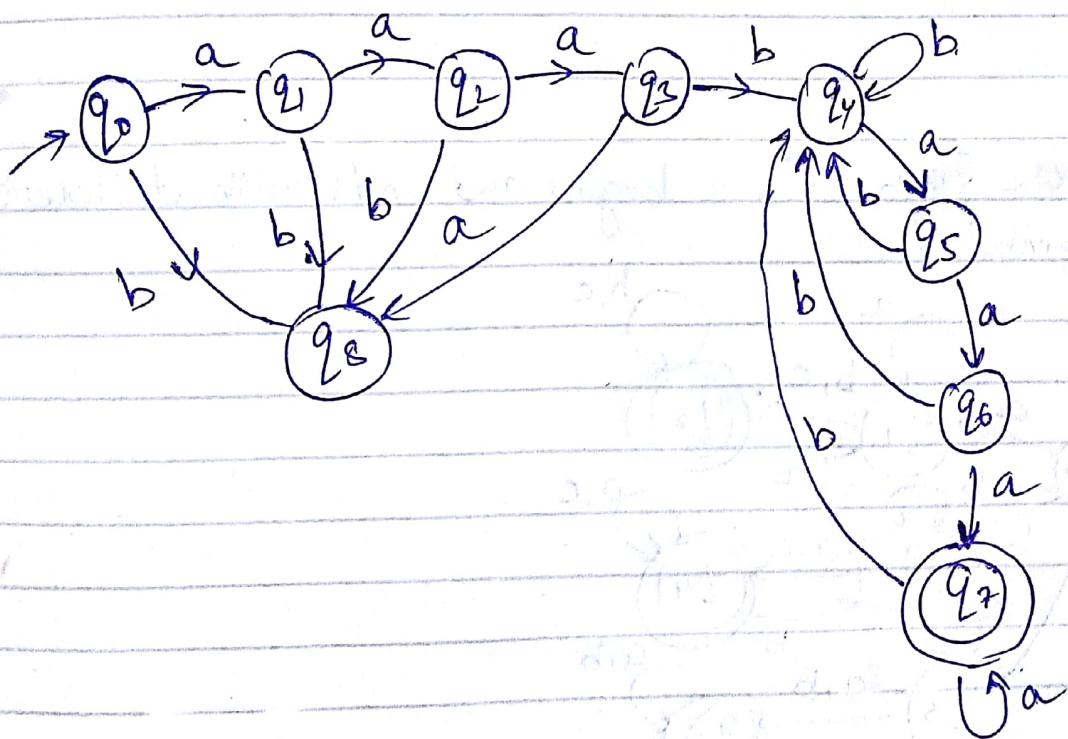
Q. $L = \{ x \in (a|b)^* : x \text{ has neither consecutive } a's \text{ nor } b's \}$



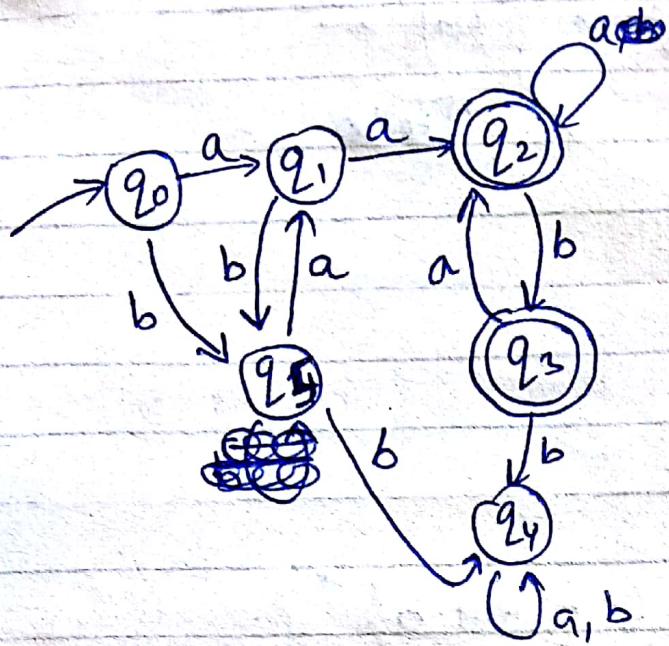
Q. $L = \{a^n b^m : n \geq 2, m \geq 1\}$



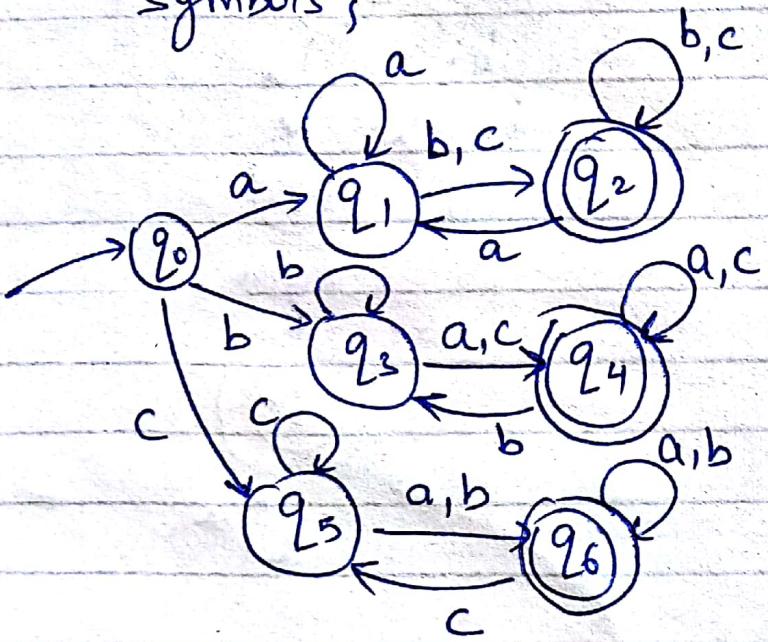
Q. $L = \{a^3 b^n a^3 | n \in \{a, b\}^*\}$



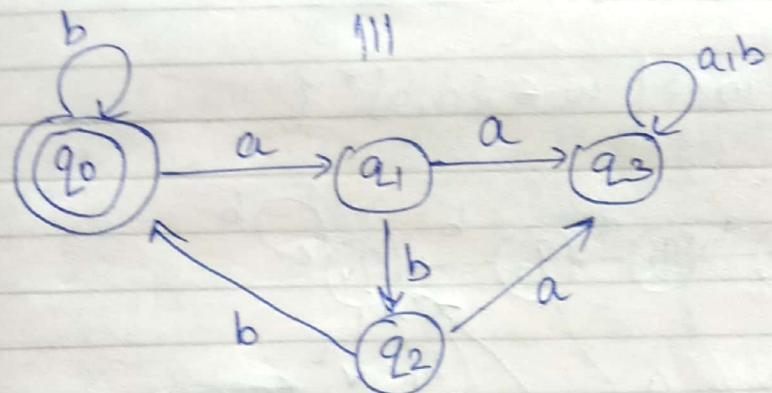
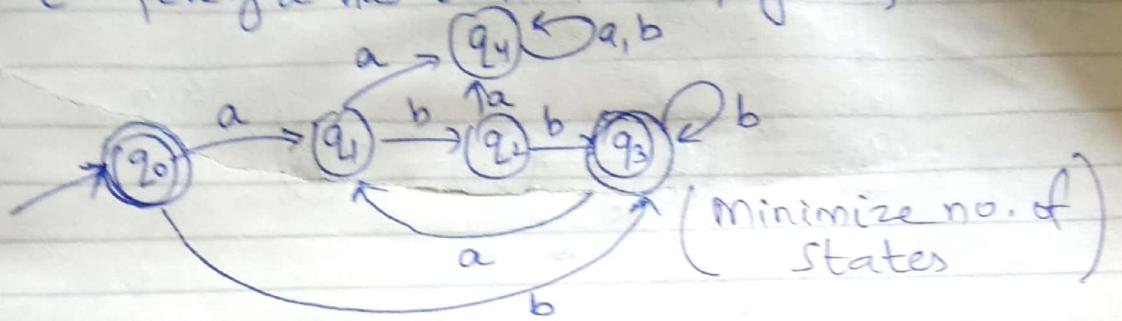
Q. $L = \{x \in \{a,b\}^*, x \text{ contains at least 2 consecutive } a's \text{ and } x \text{ does not contain 2 consecutive } b's\}$



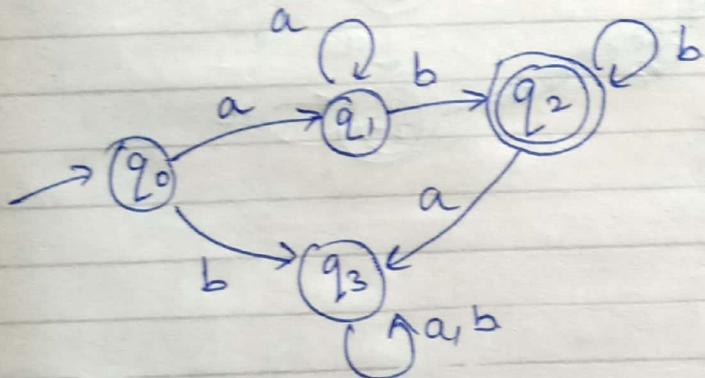
Q. $L = \{x \in \{a,b,c\}^*, x \text{ begins and ends with different symbols}\}$



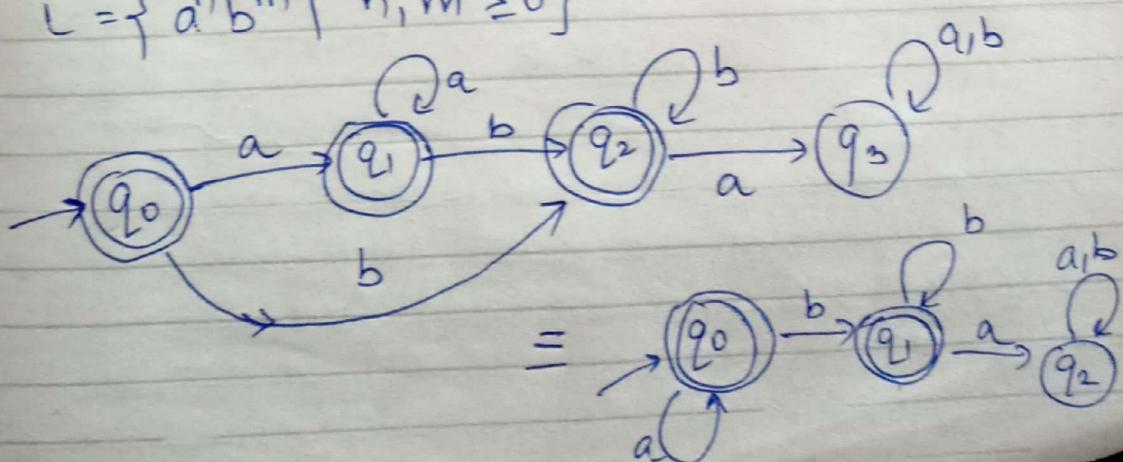
Q. $L = \{ \text{every } a \text{ has to be followed by } bb \}$



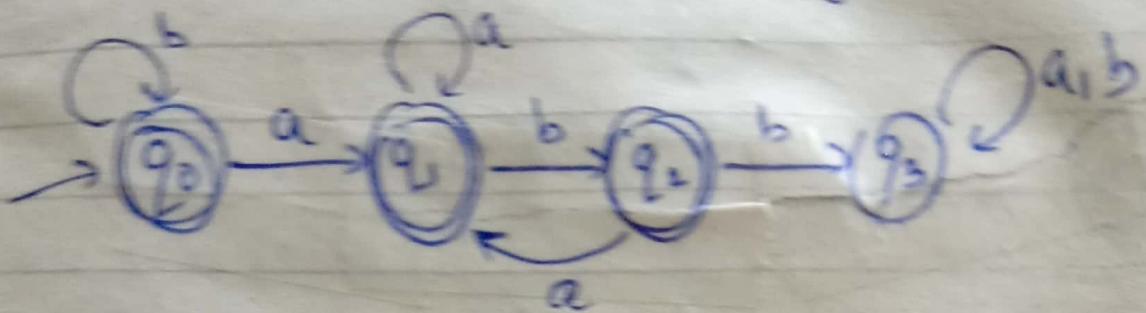
Q. $L = \{ a^n b^m \mid n, m \geq 1 \}$



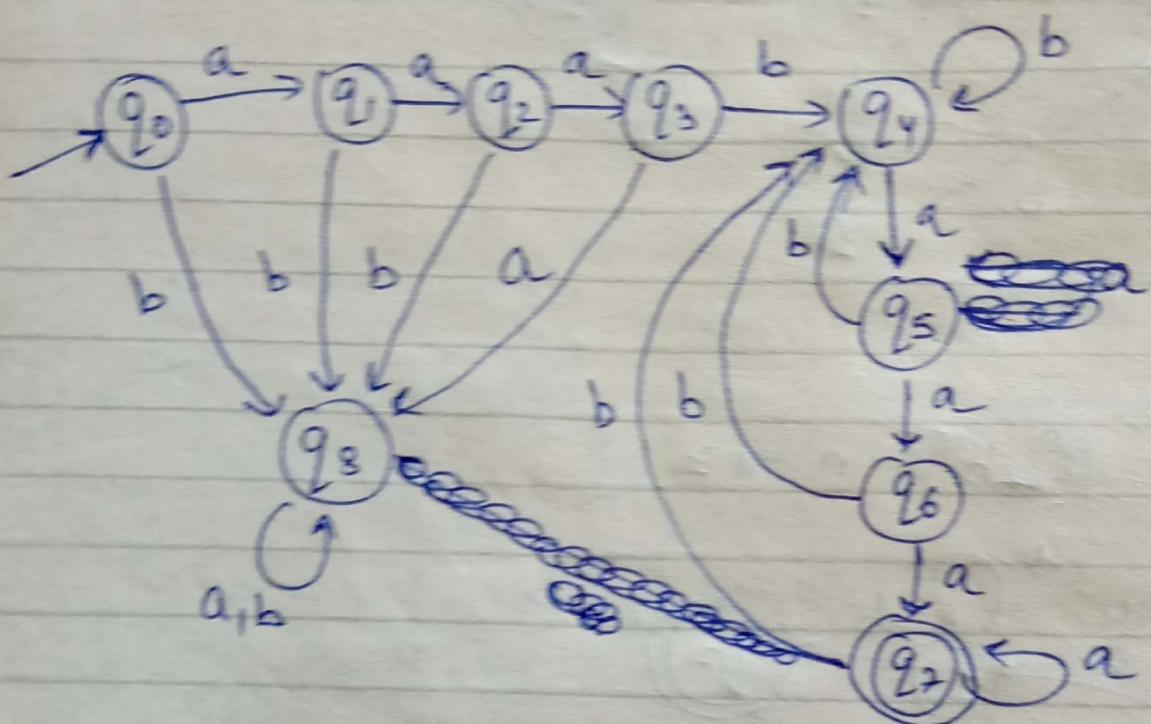
Q. $L = \{ a^n b^m \mid n, m \geq 0 \}$



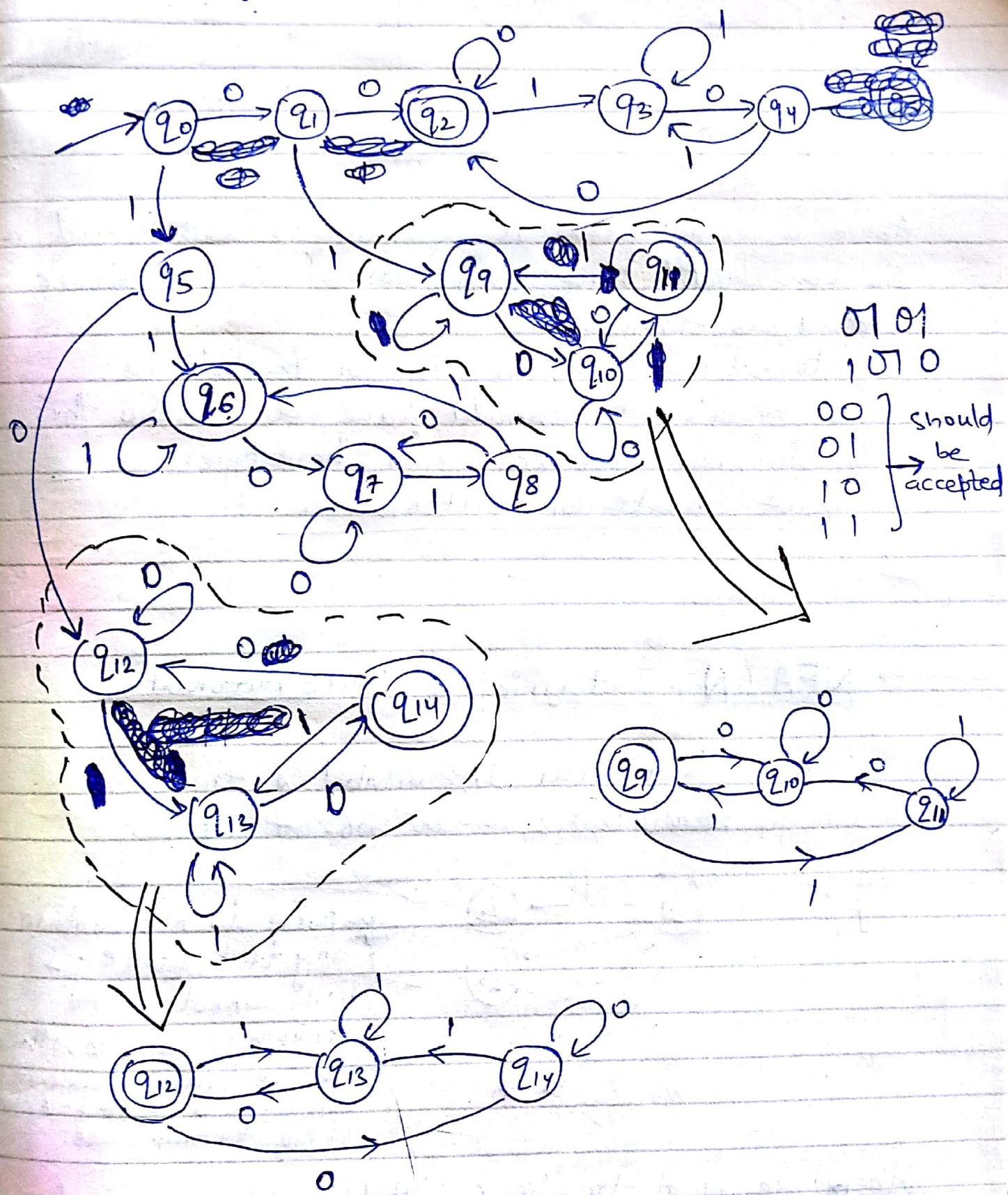
Q. $L = \{a \text{ never be followed by } bb\}$



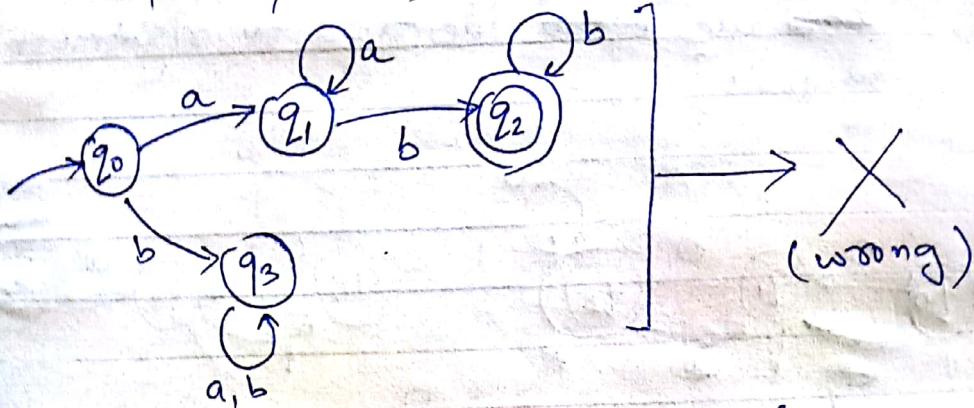
Q. $L = \{a^3bwba^3 \mid w \in \{a,b\}^*\}$



Q. $L = \{ \text{all strings in which the leftmost 2 symbols } \cancel{\text{are equal}} \text{ are identical to rightmost 2 symbols} \}$



Q. $L = \{a^n b^n / n \geq 1\}$

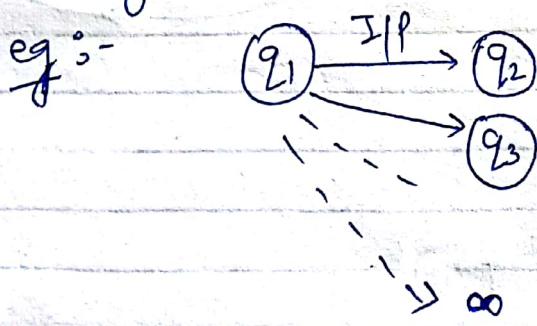


Above is wrong as n is common for both a and b .
So we should have same no. of a 's and b 's i.e.
if 4 a 's then 4 b 's.

Automata for above is not possible as
for ∞ a 's we should have ∞ b 's. So this
is infinite automata for these types, we
cannot create an automata..

→ NFA (Non Deterministic Finite Automata)

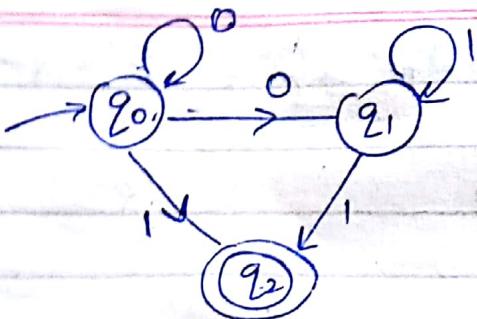
for a particular I/P symbol, transition
can go ~~to~~ to $0, 1, 2, \dots \infty$ no. of states



* NFA do not contain any trap state. If we cannot go to anywhere, i.e. \emptyset , then it is trap. because it will go to \emptyset state if \emptyset state is eqv. to trap state

When we want to check multiple ~~solutions~~ paths to a solution, we need NFA. We have to backtrack, so gives all paths to a solution and also the best path to a solution.

eg:-



	0	1
q0	q0, q1	q2
q1	∅	q1, q2

from q_0 , on inp 0, we can go to 2 states. Also, from q_1 , on inp 0, we cannot go anywhere & for 1, we have 2 states.

* NFA is 5 tuple $(\emptyset, \Sigma, q_0, \delta, F)$

↳ Transition fn

$$\delta: \emptyset \times \Sigma \rightarrow \underbrace{2^Q}_{\text{power set}}$$

* Every DFA is a NFA but every NFA is not a DFA

* Every DFA is a NFA because

$Q = \{q_1, q_2, q_3\}$, So all states in NFA are $\emptyset, \underbrace{q_1, q_2, q_3}, q_1q_2, q_2q_3, q_1q_3, q_1q_2q_3$

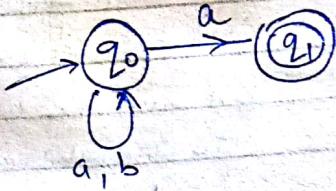
These 3 states are also present in DFA.

As, all states of NFA are also present in DFA, so every DFA is a NFA.

* An NFA can be converted to a DFA i.e. both can accept same language. So both are equivalent

* NFA is easier to design than DFA.

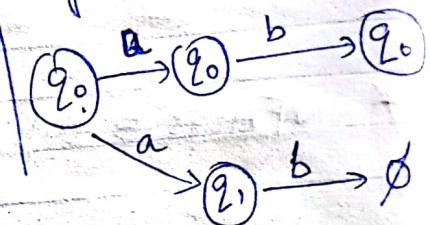
Q. $L = \{ \text{ends with } a^k \}, \Sigma = \{a, b\}$



eg:- ba



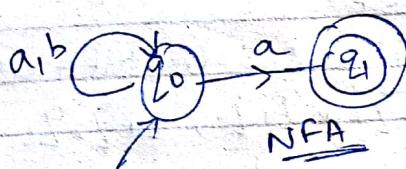
eg:- ab



→ Conversion of NFA to DFA

eg:- $L = \{ \text{Strings that ends with } a^k \}, \Sigma = \{a, b\}$

Transition Table:-



	a	b
q0	{q0, q1}	q0
q1	-	-

- To convert to DFA, make table of states and input states
- Now, these two states will be in DFA

	a	b
q0	q0	q0
q0, q1	q0	q0

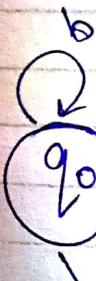
* The state which will be final state in NFA will also be final state in DFA if all states containing this state will also be final states

* In worst case, max no. of states in DFA can be 2^k .

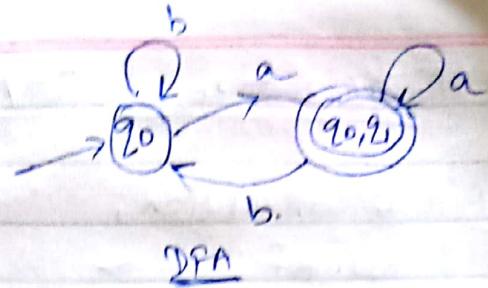
• So fina

Q. $L =$

Ans. L

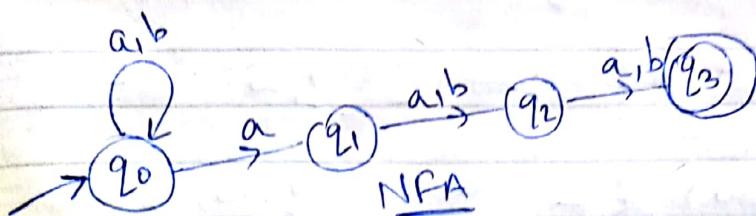


• So final DFA is :-



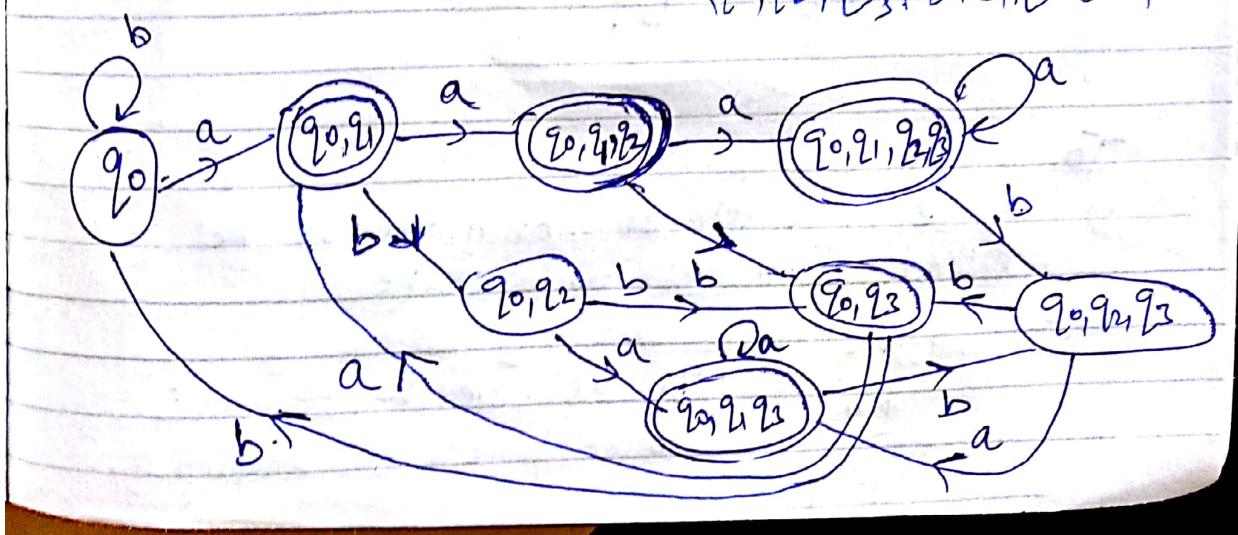
Q. Let Third symbol from RHS is 'a' } , $\Sigma = \{a, b\}$

$$\text{Ans. } L = \{ \sum^* \text{aaa}, \sum^* \text{aab}, \sum^* \text{aba}, \sum^* \text{abb} \}$$



Transition Table

Q	$\rightarrow q_0$	a	b
			Int
$\{q_0, q_1\}$	q_0, q_1	q_0	
$\{q_0, q_1, q_2\}$	q_0, q_1, q_2	q_0, q_2	
$\{q_0, q_1, q_3\}$	q_0, q_1, q_3	q_0, q_3	
$\{q_0, q_2\}$	q_0, q_2		
$\{q_0, q_1, q_2, q_3\}$	q_0, q_1, q_2, q_3	q_0, q_2, q_3	
$\{q_0, q_1, q_3\}$	q_0, q_1	q_0	
$\{q_0, q_1, q_2, q_3\}$	q_0, q_1, q_2	q_0, q_2, q_3	
$\{q_0, q_2, q_3\}$	q_0, q_2, q_3	q_0, q_3	



→ Minimization of DFA

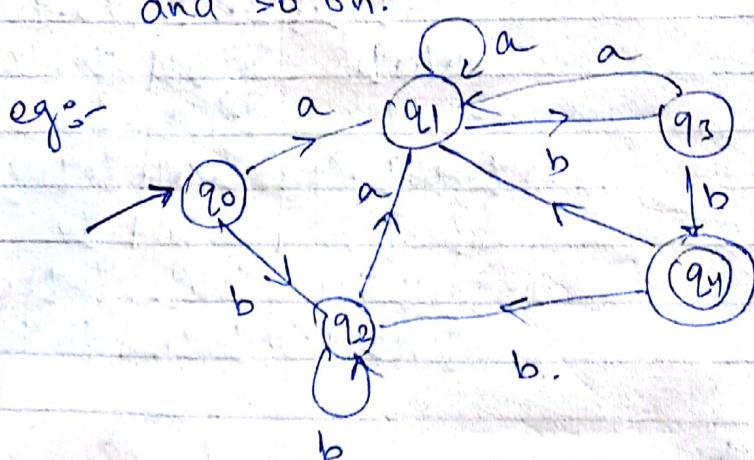
After conversion,

$$\text{No. of states (DFA)} \geq \text{No. of states (NFA)}$$

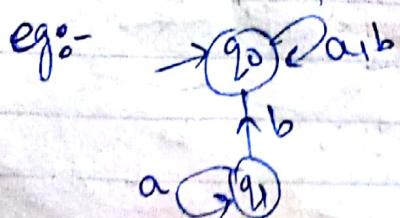
* Multiple correct DFA can exist for a language but we have to minimize no. of states by removing equivalent states.

* Two states p and q are said to be equivalent if $\delta(p, w) \in F \Rightarrow \delta(q, w) \in F$
or if $\delta(p, w) \notin F \Rightarrow \delta(q, w) \notin F$

* If $|w|=0$, then p and q are said to be 0-equivalent,
If $|w|=1$, then 1-equivalent
and so on.



To minimize,
Step 1 Remove those states which cannot be reached from starting state



q_1 can never be reached,
so no point of keeping q_1

Step 2 Check equivalence states with help of transition table

<u>0 equiv</u>	\emptyset	a	b
$[q_0, q_1, q_2, q_3] [q_4]$	$\rightarrow q_0$	q_1, q_2	q_3
	q_1	q_1	q_3
	q_2	q_1	q_2
	q_3	q_1	q_4
	q_4	q_1	q_2

Transition Table

1 equiv

$$[q_0, q_1, q_2] [q_3] [q_4]$$

If for adding, we go to ~~some states~~ states with same group in $(n-1)$ equivalent, then these states are n -equivalent

2 equiv

$$[q_0, q_2] [q_1] [q_3] [q_4]$$

3 equiv

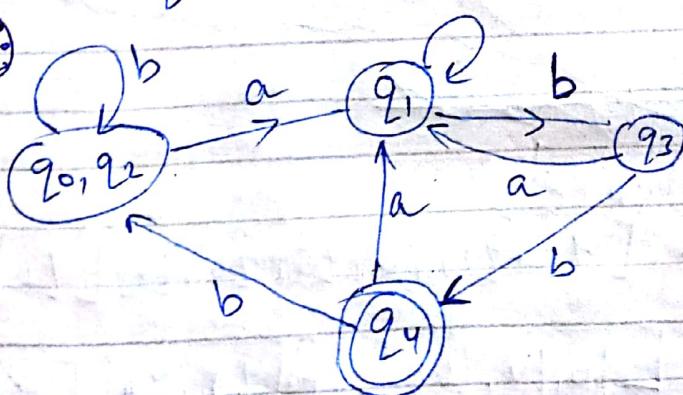
$$[q_0, q_2] [q_1] [q_3] [q_4]$$

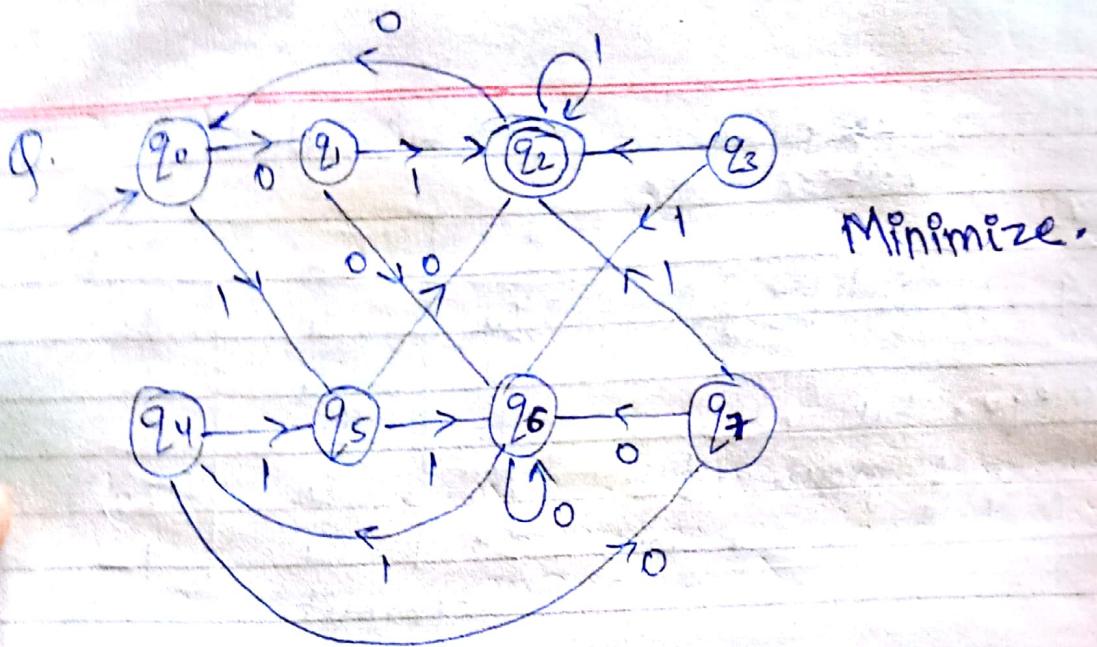
Now, no difference in groups, so stop.

From end point, we can see that we can merge

q_0 and q_2

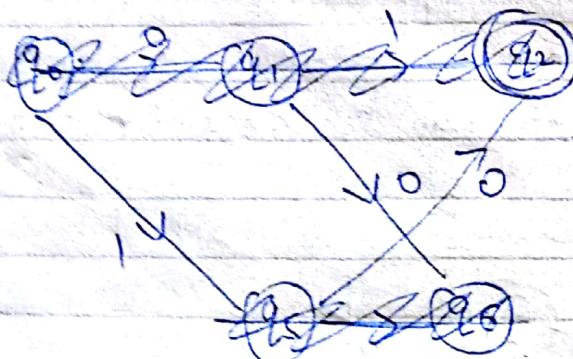
Step 3





① Remove q_3

②	q_0	q_1	q_2	q_4	q_5	q_6	q_7
	q_0	q_1	q_2				
	q_1	q_6					
	q_5	q_2					
	q_6	q_6					
	q_2	q_0					
	q_4	q_2					
	q_7	q_6					



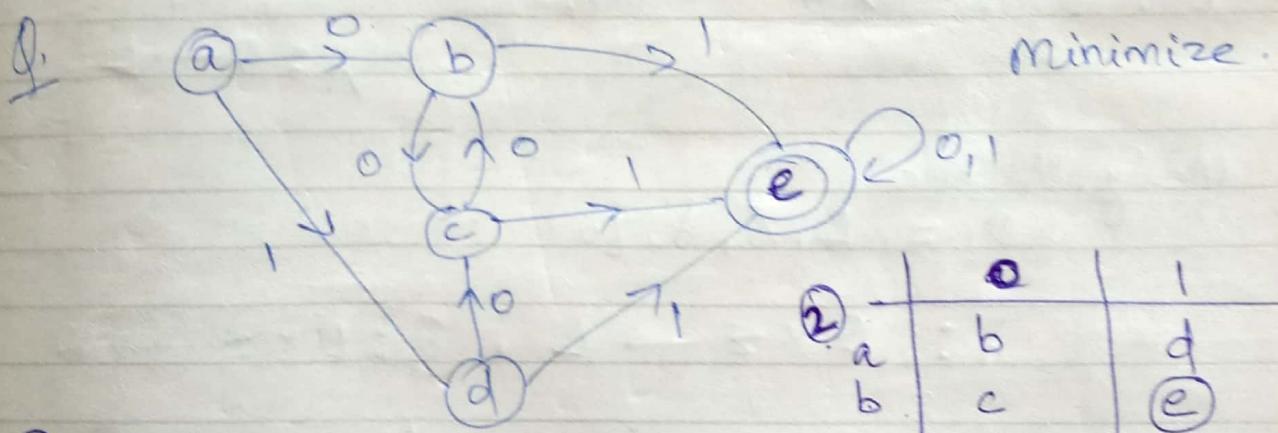
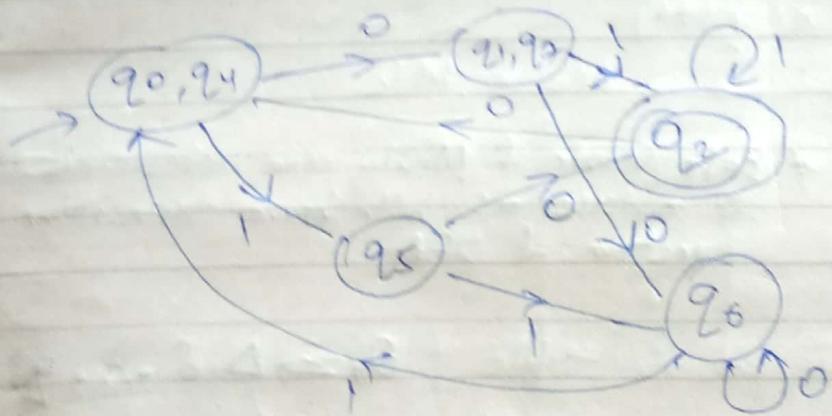
③ Test Equivalence

$$① \text{eqv} = [q_0, q_1, q_4, q_5, q_6, q_7] [q_2]$$

$$② \text{eqv} = [q_0, q_4, q_6] [q_5] [q_5, q_7] [q_2]$$

$$2\text{-equiv} - [q_0, q_4][q_0][q_5][q_1, q_3][q_0]$$

Therefore we can get 5 states.

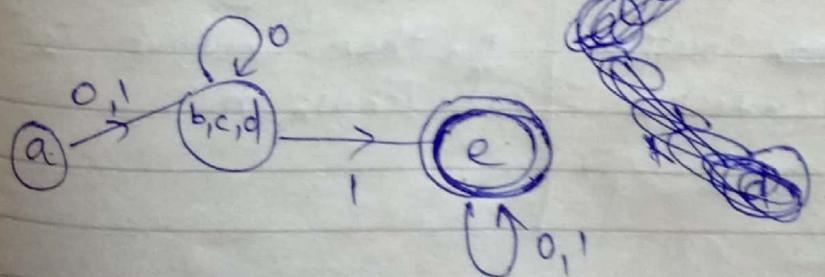


③ 0-equiv
[a, b, c, d] [e]
1-equiv

[a] [b, c, d] [e]

2-equiv

[a] [b, c, d] [e]



* Mealy and Moore does not contain any final state

→ Finite Automata with off

$$z(t) = \lambda\{q(t), x(t)\} \quad \rightarrow \text{Mealy M/c}$$

or

$$z(t) = \lambda(q(t)) \quad \rightarrow \text{Moore M/c}$$

Output function

* Both Moore & Mealy are deterministic in nature as both have transition function of DFA

Moore Machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

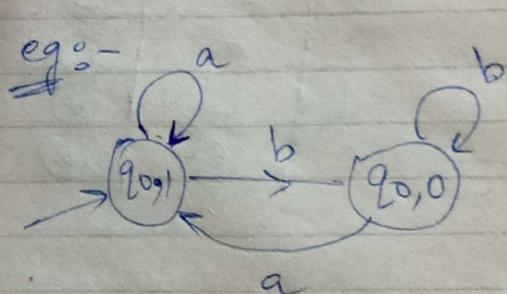
$$\delta : Q \times \Sigma \rightarrow Q$$

set of output alphabet \leftarrow output function
 $\lambda : Q \rightarrow \Delta$

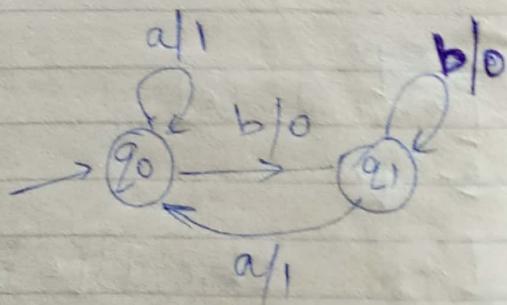
Mealy Machine

$$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

$$\lambda : Q \times \Sigma \rightarrow \Delta, \delta : Q \times \Sigma \rightarrow Q$$



Moore M/c.



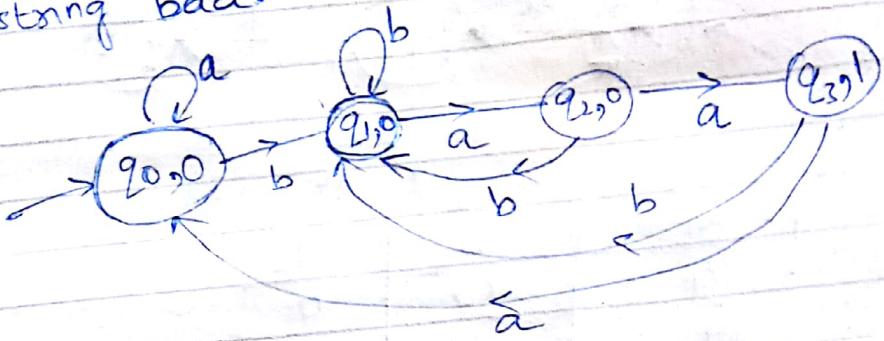
Mealy M/c

If string is of length n , then ~~length~~ output is of length $(n+1)$

If string is of length n , then output is of length n

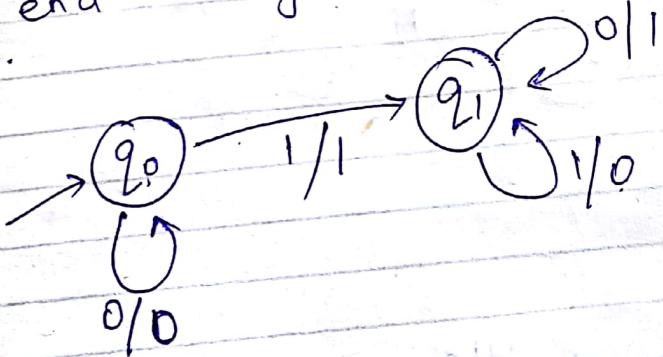
Q. Design a Moore M/c that takes set of all strings over $\{a, b\}$ and counts the no. of occurrences of substring baa .

Ans



Q. Design a Mealy M/C that takes binary no. as input & produce 2's complement of that no. Assume string is read from LSB to MSB and end carry is discarded.

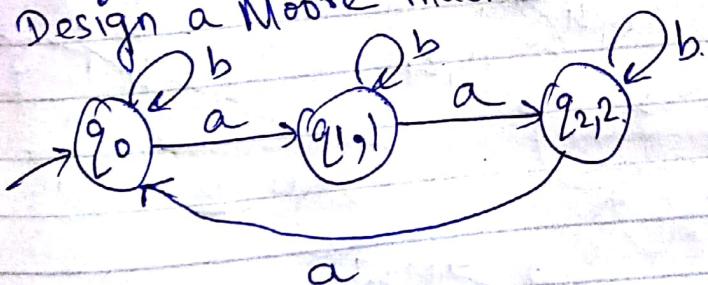
Ans



From LSB, as zero comes, they will remain zero till first one comes. First one will also remain 1. After that, 0 is converted to 1 and 1 is converted to 0.

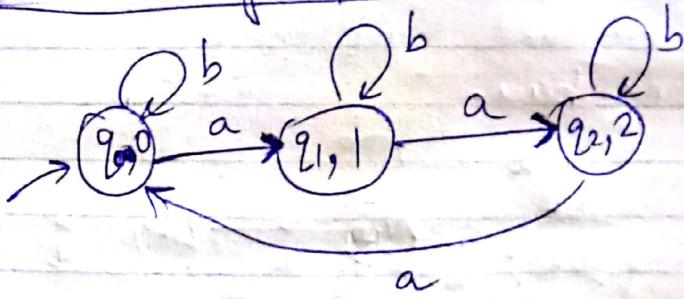
* Both Melay & Morse are equivalent i.e. they can be converted to each other.

Q. Design a Moore machine that count no. of a's % 3.



→ Moore to Mealy M/c

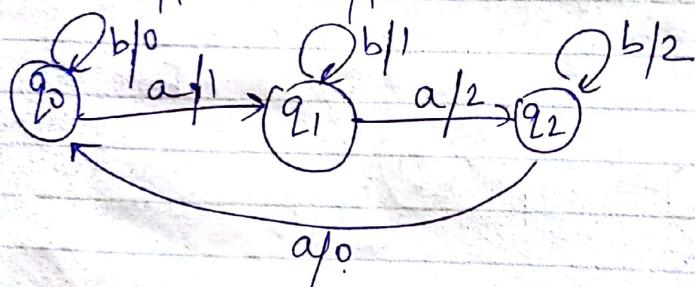
eg:-



Step 1 :-

States.	a	b	Δ
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

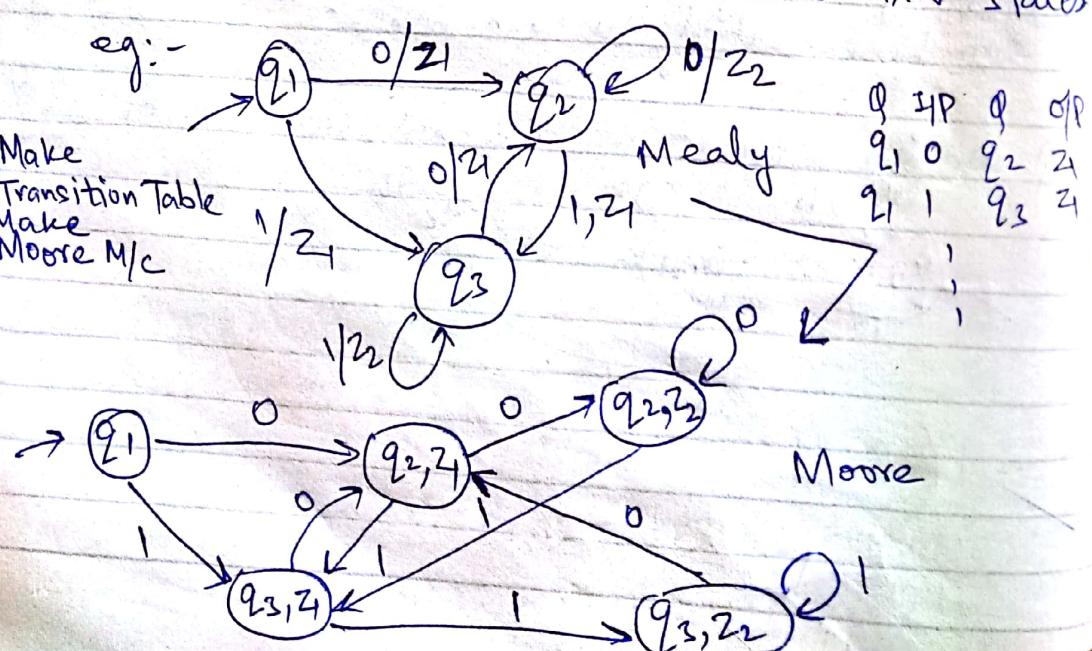
Step 2 :- Write o/p with y/p.



→ Mealy to Moore M/c

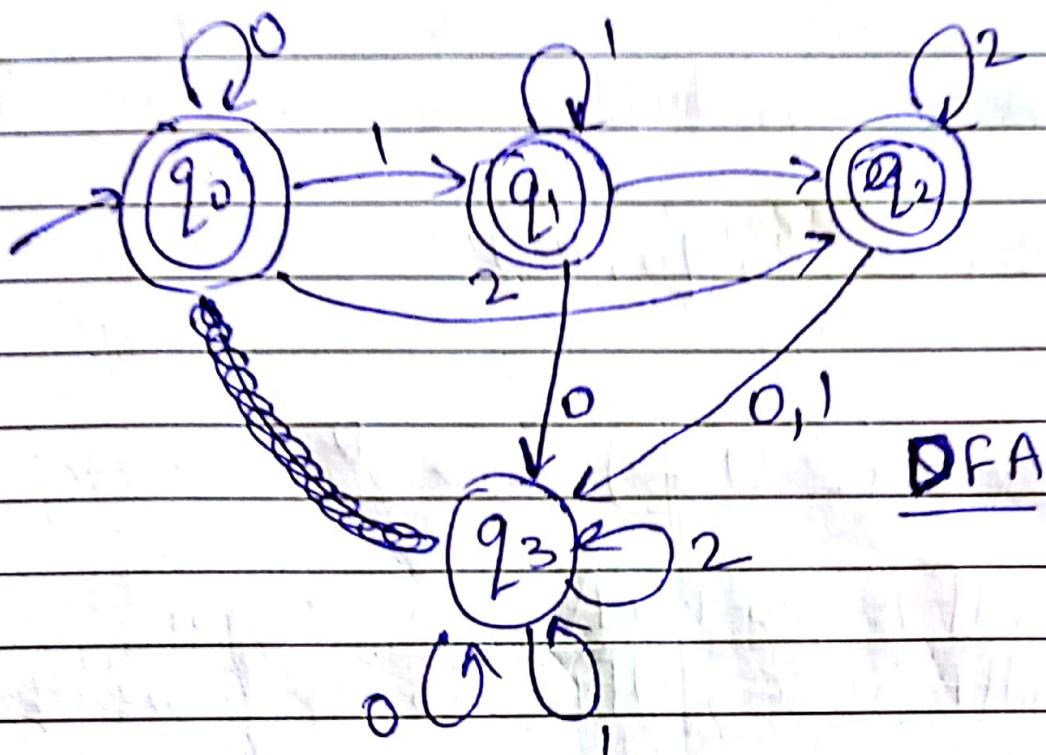
If mealy contains M states & N outputs, then
in worst case ~~M^N~~ Moore contains $M \times N$ states

- ① Make Transition Table
- ② Make Moore M/c

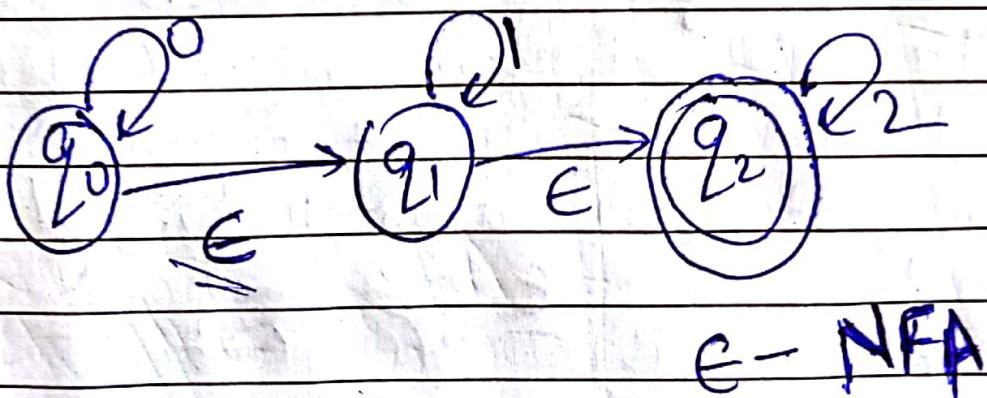


Epsilon NFA (ε-NFA)

Q. $L = \{0^m 1^n 2^k \mid m, n, k \geq 0\}, \Sigma = \{0, 1, 2\}$



ϵ -NFA allows transition from one ~~state~~^{state} to another even if no input is given



If you refuse to accept anything but the best, you very often get it.

* All ~~NFA~~, DFA, E-NFA are equivalent.

• E-NFA is five tuple

$$(Q, \Sigma, \delta, q_0, F)$$

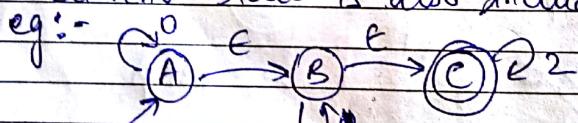
$$\delta: Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$$

→ E-NFA to NFA Conversion

* Every NFA is E-NFA. So to prove that both are equivalent, we need to convert E-NFA to NFA

* E-Closure of any state is set of all states that can be reached from that state using E only.

Current state is also included in set



$$E\text{-closure of } A = \{A, B, C\}$$

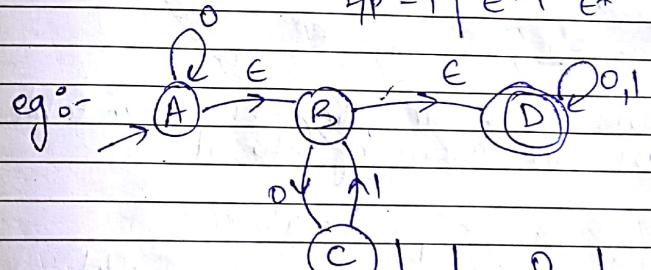


Man has ability to turn the stumbling blocks in to stepping stones.

* To convert, make table of NFA, remove all E edges and create new automata according to table. To remove E edge, use E closure.

$$E\text{-closure}(A), I/P \{$$

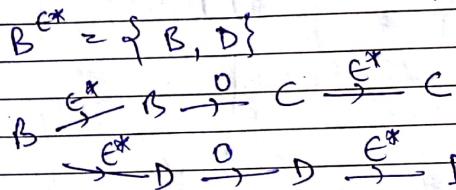
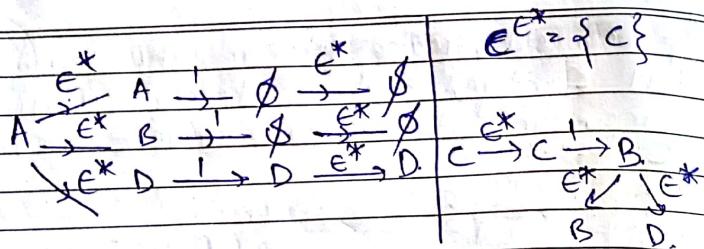
A	I/P - 0	$\epsilon^* 0 \epsilon^*$
	I/P - 1	$\epsilon^* 1 \epsilon^*$



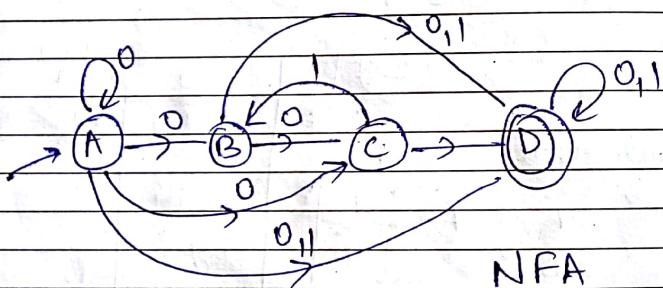
	0	1
A	{A, B, C, D}	D
B	{C, D}	D
C	∅	{B, D}
D	D	D



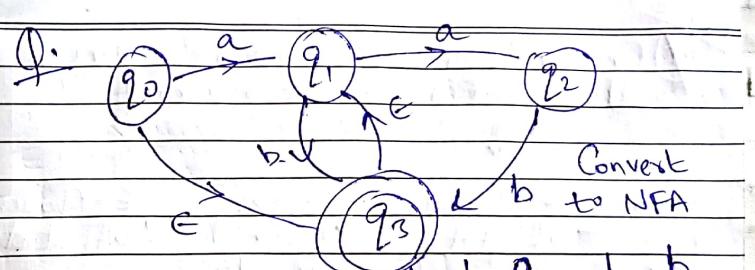
If you refuse to accept anything but the best, you very often get it.



Now, after completing transition table
draw NFA according to it:



Man has ability to turn the stumbling blocks in to stepping stones.



	a	b
$q_0 \xrightarrow{\epsilon^*} q_0$	$\{q_0\}$	$\{q_0\}$
$q_0 \xrightarrow{a} q_1$	$\{q_1\}$	$\{q_1\}$
$q_1 \xrightarrow{a} q_2$	$\{q_2\}$	$\{q_2\}$
$q_2 \xrightarrow{a} q_1$	\emptyset	$\{q_1, q_3\}$
$q_1 \xrightarrow{b} q_3$	\emptyset	$\{q_1, q_3\}$
$q_3 \xrightarrow{b} q_2$	\emptyset	$\{q_2\}$
$q_2 \xrightarrow{b} q_1$	\emptyset	$\{q_1\}$
$q_1 \xrightarrow{c} q_3$	\emptyset	$\{q_3\}$
$q_3 \xrightarrow{c} q_2$	\emptyset	$\{q_2\}$
$q_2 \xrightarrow{c} q_1$	\emptyset	$\{q_1\}$
$q_1 \xrightarrow{\epsilon} q_3$	\emptyset	$\{q_3\}$
$q_3 \xrightarrow{\epsilon} q_2$	\emptyset	$\{q_2\}$
$q_2 \xrightarrow{\epsilon} q_1$	\emptyset	$\{q_1\}$

	a	b	c
$q_1 \xrightarrow{\epsilon^*} q_1$	$\{q_1\}$	$\{q_1\}$	$\{q_1\}$
$q_1 \xrightarrow{a} q_2$	$\{q_2\}$	\emptyset	$\{q_2\}$
$q_2 \xrightarrow{a} q_1$	$\{q_1\}$	\emptyset	$\{q_1\}$
$q_1 \xrightarrow{b} q_3$	\emptyset	$\{q_3\}$	$\{q_3\}$
$q_3 \xrightarrow{b} q_2$	\emptyset	$\{q_2\}$	$\{q_2\}$
$q_2 \xrightarrow{b} q_1$	\emptyset	$\{q_1\}$	$\{q_1\}$
$q_1 \xrightarrow{c} q_3$	\emptyset	$\{q_3\}$	$\{q_3\}$
$q_3 \xrightarrow{c} q_2$	\emptyset	$\{q_2\}$	$\{q_2\}$
$q_2 \xrightarrow{c} q_1$	\emptyset	$\{q_1\}$	$\{q_1\}$



If you refuse to accept anything but the best, you very often get it.

~~Note -~~

* On giving an input, to check which all states can be reached

中

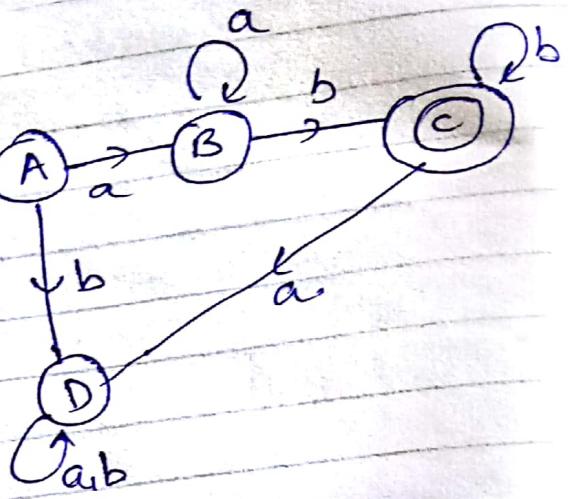


Man has ability to turn the stumbling blocks in to stepping stones.

Family of Languages

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

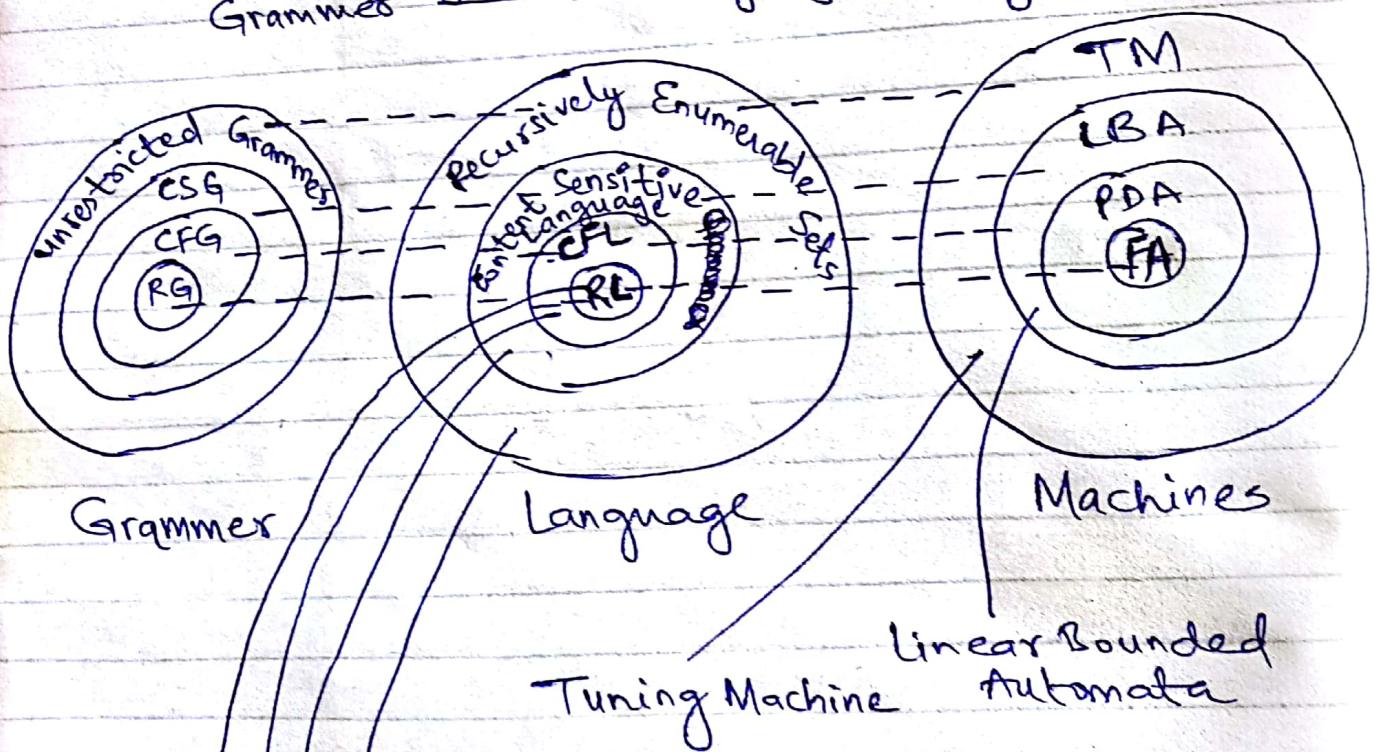
This automata is an acceptor as it accepts all strings like $a^n b^n$ but it is not rejecting $a^n b^m$.



F.A + memory = Push Down Automata

* Languages accepted by finite automata are regular languages.

Grammars generate language $\xrightarrow{\text{accepted by}}$ Machines



eg:- a^p , $p \rightarrow$ prime

eg:- $a^n b^n c^n \mid n \geq 1$

eg:- $a^n b^n \mid n \geq 1$

eg:- $a^n \mid n \geq 1$

→ Regular Language (RE)

A regular language (RL) can be represented by some mathematical expression called as regular expression (RE).

- ① ϕ , ϵ and $a \in E$ are primitive R.E
- ② If r_1 and r_2 are two RE then $r_1 + r_2$, r_1r_2 and r_1^* are also R.Es.
- ③ A string is a RE if and only if it can be derived from the primitive REs by a finite no. of the application of the rules in ②

e.g.: - $(\underbrace{a+b+c}_{RE}) \cdot (\underbrace{d+\phi}_{RE})$ → It is a R.E.

e.g.: - $(\underbrace{a+b+}_{RE} \dots)$ → It is not a R.E.

As $a+b$ is not a R.E, \dots is not a regular expression but \dots^* is a R.E.

Q. Design R.E given Language

① $\{ \epsilon, 0, 00, 000, 0000, \dots \}$ → 0^*

② $\{ 1, 11, 111, 1111, \dots \}$ → $1 \cdot 1^*$

If Σ_1 & Σ_2 are RE then

- ① $L(\Sigma_1 + \Sigma_2) = L(\Sigma_1) \cup L(\Sigma_2)$
- ② $L(\Sigma_1 \cdot \Sigma_2) = L(\Sigma_1) \cdot L(\Sigma_2)$
- ③ $L((\Sigma_1)) = L(\Sigma_1)$
- ④ $L(\Sigma_1^*) = [L(\Sigma_1)]^*$

- * Any language for which finite automata can be made is a finite language and so for it RE can be made.
- * Multiple RE can be made but we have to minimize RE.

Q. Given RE, generate language

$$\textcircled{1} \quad \Sigma = L(a^* \cdot (a+b))$$

$\hookrightarrow \{a, aa, aaa, aaaa, \dots, b, ab, aab, aaab, aaaaab\}$

$$\textcircled{2} \quad \Sigma = (a+b)^* (a+bb)$$

$$\textcircled{3} \quad \Sigma = \cancel{(aa)}^{\text{Don't care}} * (bb)^* b$$

Ans

$$\textcircled{1} \quad L(a^*) \cdot L(a+b)$$
$$\Rightarrow [L(a)]^* (a, b)$$

$$\Rightarrow (a^*) \cdot (a, b)$$

$$\Rightarrow \{ \in, a, aa, aaa, \dots \} \cdot (a, b)$$

$$\Rightarrow \{ a, aa, aaa, \dots, b, ab, aab, \dots \}$$

$$\textcircled{2} \quad \{ \epsilon, (a+b), (a+b)^2, \dots \} \cdot \{ a, bb \}$$

$$\Rightarrow \{ \epsilon, a, b, aa, bb, ab, ba, \dots \} \cdot \{ a, bb \}$$

$$\Rightarrow \{ a, aa, ba, \cancel{bb}, bba, aba, baa, \dots, \\ bb, abb, bbb, aabb, bbbb, abbb, babb, \dots \}$$

\textcircled{3} - It can be seen that Σ is a RE with even no. of a's & odd no. of b's.

So,

$$L = \{ a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0 \}$$

① $L = \{ w \in \Sigma^*, w \text{ has at least one pair of consecutive 0's} \}$
 $\Sigma = \{0, 1\}$

$$\Rightarrow (0+1)^* 00 (0+1)^*$$

② $L = \{ \text{set of all strings of exactly two length} \}$

$$\Rightarrow (a+b)^2 (a+b)$$

③ $L = \{ \text{at least two length} \}$

$$\Rightarrow (a+b) \cdot (a+b) \cdot (a+b)^*$$

④ $L = \{ \text{atmost two length} \}$

$$\Rightarrow (a+b+\epsilon)^* + (a+b)(a+b)^* +$$

⑤ $L = \{ \text{all strings of even length} \}$

$$\Rightarrow [(a+b)(a+b)]^*$$

⑥ $L = \{ \text{that has exactly two a's} \}$

$$\Rightarrow b^* a b^* a b^*$$

⑦ $L = \{ \text{at least 2 a's} \}$

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

⑧ $L = \{ \text{at most two a's} \}$

$$\Rightarrow b^* + \cancel{ab^*} b^* a b^* + b^* a b^* a b^* \\ \text{OR} \\ b^* (a+\epsilon) b^* (\epsilon+a) b^*$$

⑨ $L = \{ \text{No. of a's are even} \}$

~~$$b^* a b^* a b^*$$~~
$$\Rightarrow (b^* a b^* a b^*)^* + b^*$$

⑩ $L = \{ \text{starting \& ending with same symbol} \}$

$$\Rightarrow a(a+b)^* a + b(a+b)^* b + (a+b) + \epsilon$$

⑪ $L = \{ \text{starting \& ending with diff. symbol} \}$

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

⑫ $L = \{ \text{all strings that do not contain two a's together} \}$

$$\Rightarrow (b+ab)^* (\epsilon+a)$$

⑬ $L = \{ \text{Set of all strings in which two a's and b's come together} \}$

~~$$\Rightarrow (a+\epsilon+b)(ab)^*(\epsilon+ab)$$~~

$$\begin{aligned}
 & \cancel{(a+b)}(ba)^* (b+\epsilon) \\
 & + (b+\epsilon)(ab)^* \cancel{(a+\epsilon)} \\
 & (a+\epsilon)(ba)^* b \rightarrow (ab)^* (a+\epsilon) \\
 \Rightarrow & (ab)^* a + (ab)^* + \cancel{(ab)^*} \\
 & + b(a,b)^* a + b(ab)^*
 \end{aligned}$$

OR

$$(b+\epsilon)(ab)^* (a+\epsilon)$$

$$\textcircled{14} \quad L = \{ a^m b^n c^k \mid m, n, k \geq 1 \} \\
 \Rightarrow a \cdot a^* b \cdot b^* c \cdot c^*$$

$$\textcircled{15} \quad L = \{ a^n b a^{2m} b^2 \mid m \geq 0, n \geq 1 \} \\
 \Rightarrow a \cdot a^* b (aa)^* b b$$

$$\textcircled{16} \quad L = \{ w \mid |w| = 3k_1 + 5k_2, \begin{cases} k_1 \geq 0 \\ k_2 \geq 0 \end{cases} \} \\
 \subseteq \{ a, b \}^*$$

$$\Rightarrow ((a+b)(a+b)(a+b))^* \cdot ((a+b)(a+b)(a+b)(a+b)(a+b))^*$$

→ Arden's Theorem

let P and Q be two RE's over Σ . If P does not contain ϵ , then the following equation in R , namely

$$R = Q + RP$$

~~has~~ has a unique solution given by $R = QP^*$

Proof :-

Put $R = QP^*$ in the RHS of

$$\begin{aligned} R &= Q + RP \quad \text{--- (1)} \\ &= Q + (QP^*)P \\ &= Q(\epsilon + P^*P) \\ &= QP^* \end{aligned}$$

Hence (1) is satisfied by $R = QP^*$

This means $R = QP^*$ is a solution of (1)

To prove uniqueness :-

Replace R by $Q + RP$ on RHS of (1)

$$\begin{aligned} &= Q + RP \\ &= Q + (Q + RP)P \\ &= Q + QP + RPP \\ &= Q + QP + RP^2 \\ &= Q + QP + \dots + QP^i + RP^{i+1} \\ &= Q(\epsilon + P + P^2 + \dots + P^i) + RP^{i+1} \quad \text{--- (2)} \end{aligned}$$

Thus

$$R = Q(\epsilon + P + P^2 + \dots + P^i) + RP^{i+1}, \text{ for } i \geq 0$$

Now, we show that any solution of (1) is equivalent to QP^*

Let w is a string of length i in R . Then w belongs to the set ~~the set~~ $Q(\epsilon + P + P^2 + \dots + P^i) + RP^{i+1}$ as P does not contain ϵ , so $w \notin R.P^{i+1}$ and thus

$w \in Q(\epsilon + P + P^2 + \dots + P^k)$ and hence to QP^* .

Consider, Now, $w \in QP^*$, then w is in the set QP^k for some $k \geq 0$ and hence in $Q(\epsilon + P + P^2 + \dots + P^k)$, so w is in RHS of ②. Therefore $w \in R$ of ②. Thus R and QP^* represents the same set. As,

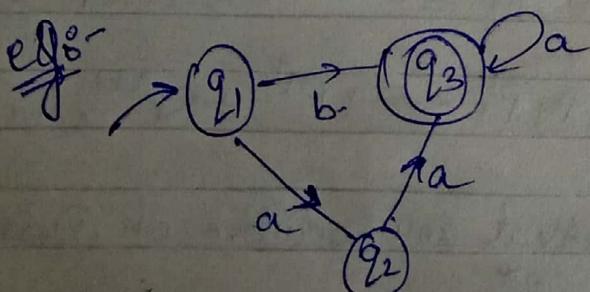
$w \in R$, then QP^*

$w \in QP^*$, then R .

→ FA to RE

Rules :-

- ① The diagram does not have ϵ moves
- ② It has only one initial state
- ③ Its vertices are q_1, \dots, q_n
- ④ Solution is the RE for final state
- ⑤ If there are multiple final states then we will take the union of RE of all final states.
- ⑥ Initial states always have ϵ move added to it.



for q_1 ,

$$q_1 = \epsilon - ①$$

for q_2 ,

$$q_2 = q_1 \cdot a - ②$$

for q_3 ,

$$q_3 = q_1 \cdot b + q_2 \cdot a + q_3 \cdot a - ③$$

From ①, ② and ③

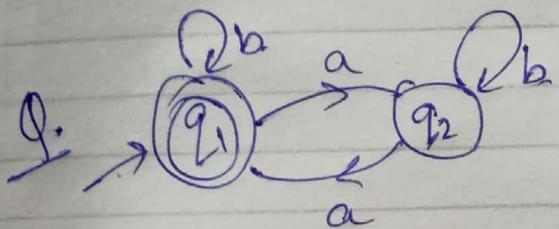
$$q_2 = \epsilon \cdot a = a$$

$$\begin{aligned} q_3 &= \epsilon \cdot b + a \cdot a + q_2 \cdot a \\ &= b + aa + qa \end{aligned}$$

By Arden's Theorem, $R = Q + RP$.

$$\text{So, } R = q_3, Q = b + aa, P = a$$

$$\therefore \boxed{q_3 = (b+aa)a^*}$$



$$q_1 = \epsilon + q_1 \cdot b + q_2 \cdot a \quad \boxed{1}$$

$$q_2 = q_1 \cdot a + q_2 \cdot b. \quad \boxed{2}$$

Using Arden's Theorem on q_1 ,

$$q_1 = (\epsilon + q_2 \cdot a) \cdot b^* \quad \boxed{3}$$

Using Arden's Theorem on q_2 ,

$$q_2 = (q_1 \cdot a) \cdot b^* \quad \boxed{4}$$

Put ④ in ③

$$q_1 = b^* + ab^* q_2 \quad q_2 = ab^*$$

$$q_1 = b^* + q_1 a b^* ab^*$$

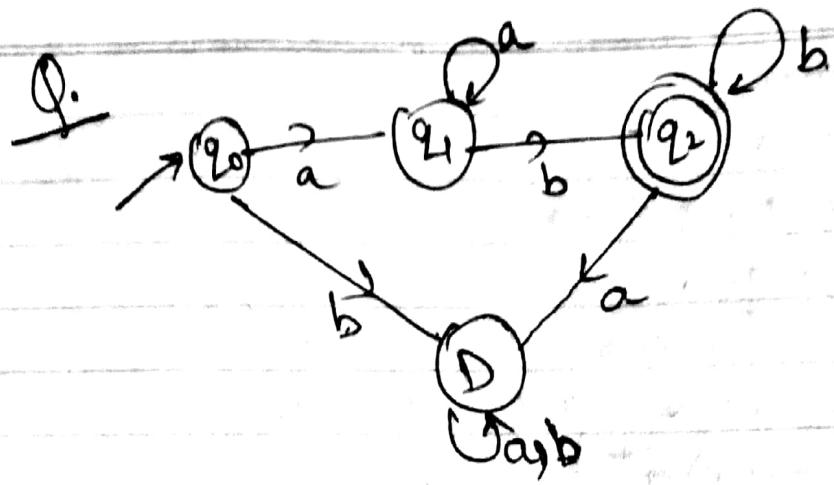
$$\therefore q_1 = b^* (ab^* ab^*)^* \text{ on } q_1.$$

Put ③ in ①

$$\begin{aligned} q_1 &= \epsilon + q_1 \cdot b + (q_1 \cdot a) \cdot b^* a \\ &= \epsilon + q_1 (b + ab^* a) \end{aligned}$$

Using Arden's Theorem,

$$q_1 = \epsilon \cdot (b + ab^* a)^*$$



$$\Rightarrow q_0 = \epsilon.$$

$$q_1 = q_0 \cdot a + q_1 \cdot a \\ = a\epsilon + q_1 \cdot a$$

$$\Rightarrow q_1 = a \cdot a^*$$

$$q_2 = q_1 \cdot b + q_2 \cdot b$$

$$= aa^*b + q_2 \cdot b$$

$$\Rightarrow q_2 = (aa^*b)b^*$$

Regular Expression to FA

• Kleene's Th^m

If R is a regular expression over Σ representing $L \subseteq \Sigma^*$, then there exists an NFA with ϵ -moves such that,

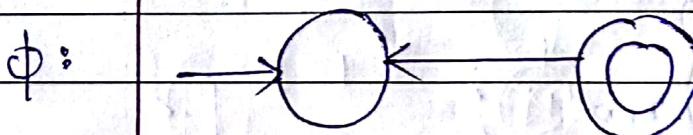
$$[L = T(M)] \Leftarrow$$

NFA or FA

$$R = (a+b) \Rightarrow \text{character} = \textcircled{3}$$

Proof: Using induction.

$$\underline{n=1} \quad R = \epsilon, \phi \text{ or } a.$$



Regular expressions possible of R of length less than equal to n .

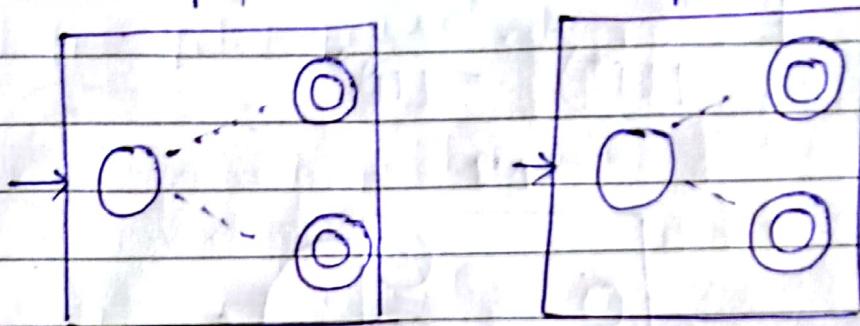
$$R = P + Q \xrightarrow{M_1} M_2$$

$$R = \overline{PQ} \xrightarrow{M_1} M_2$$

$$R = P^* \xrightarrow{M_1} M_2$$

M1

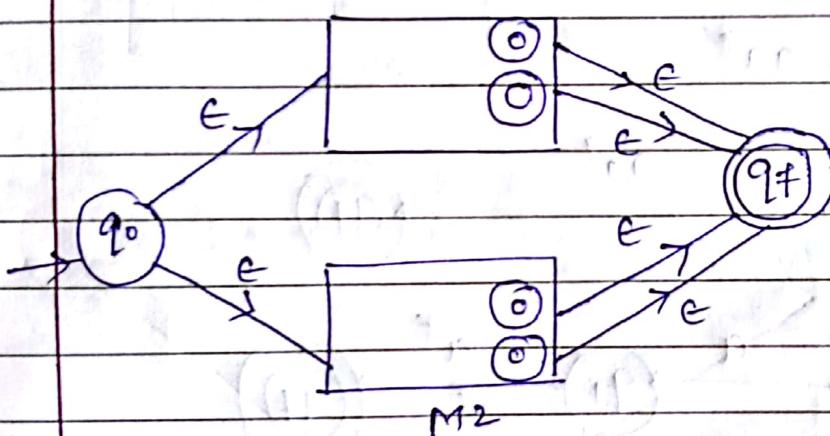
M2



$$\text{1). } L(R) = L(P \oplus Q) \\ = [L(P) \cup L(Q)]$$

M1

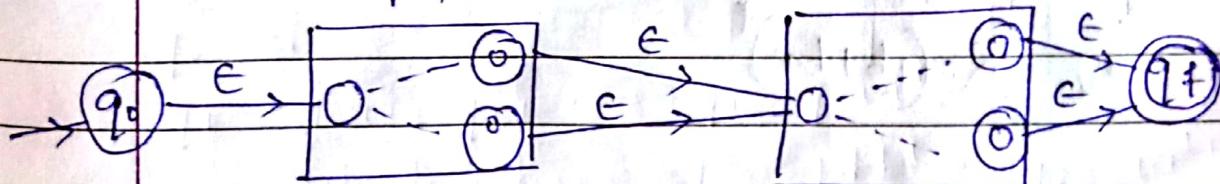
M2



$$\text{2). } L(R) = L(P \cdot Q) = L(P) \cdot L(Q)$$

M1

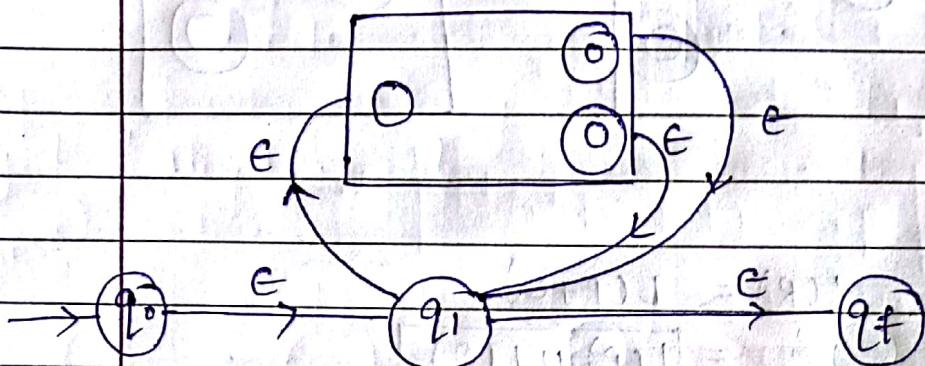
M2



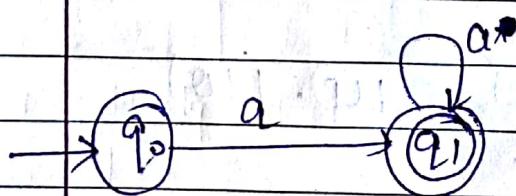
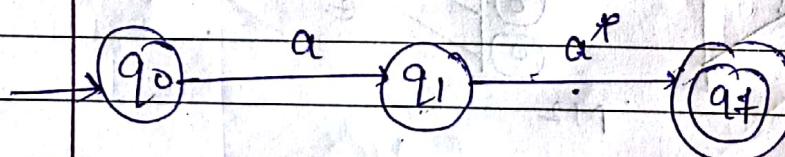
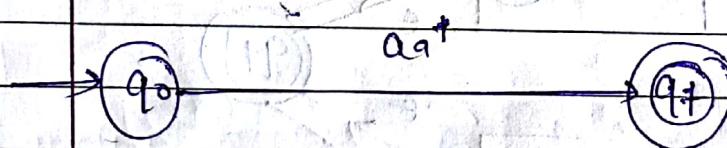
3). P^*

$$L(P^*) = L(P)^*$$

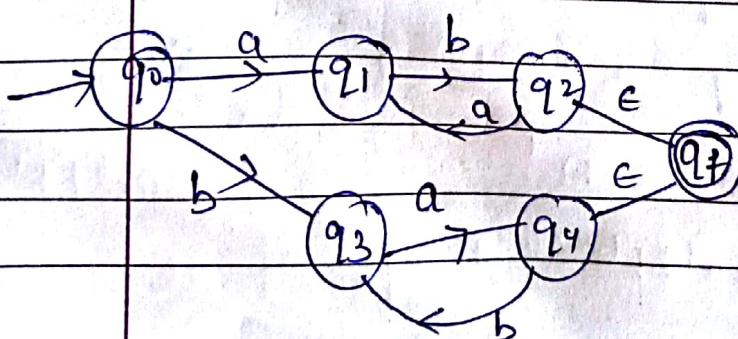
M1



(Q), $R = aa^*$



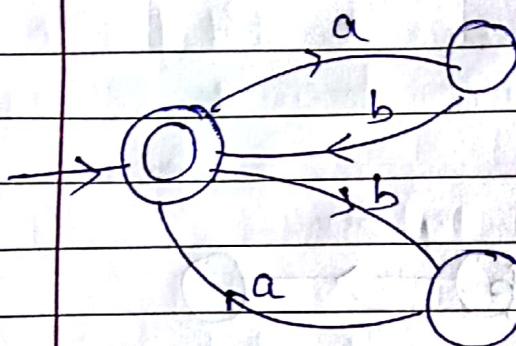
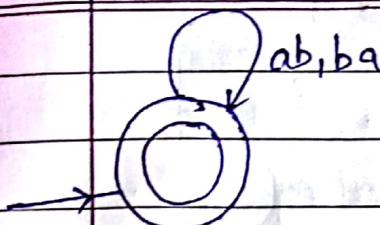
(Q), $R = (ab + ba)^*$



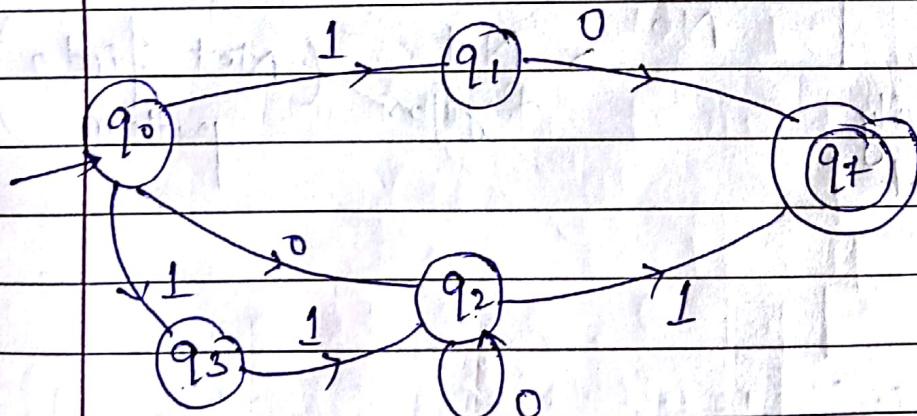
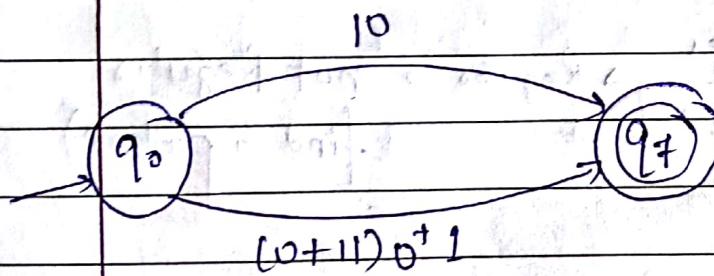
IMP $(a+b)^*$ mean all combo accept 0.

i.e. $n \geq 1$ but $n \neq$ _____

Page No. _____	Date _____	_____	_____	_____
----------------	------------	-------	-------	-------



Q. $R = 10 + (0+11)0^*1$



Pumping Lemma Theorem

language

1



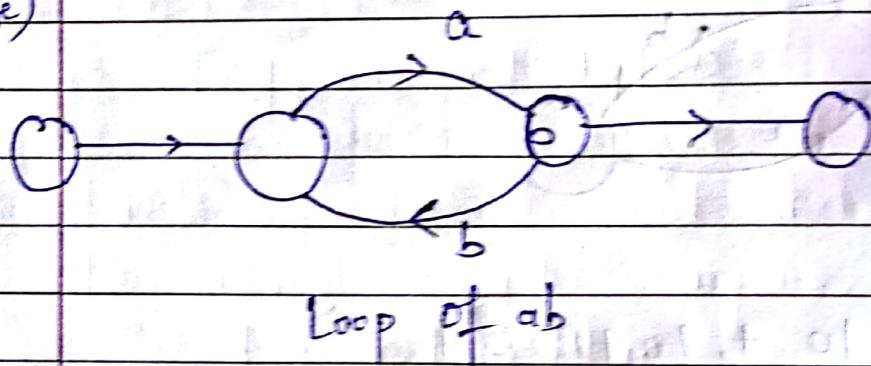
finite



Infinite

e.g., $L = \{ab, abab, \dots\}$

(infinite)



Pumping
Lemma
Test

Yes

→ Regular or not Regular
(find a pattern)

Negativity
test

No

→ Not
Regular
(Not find a
pattern)

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata with n states.

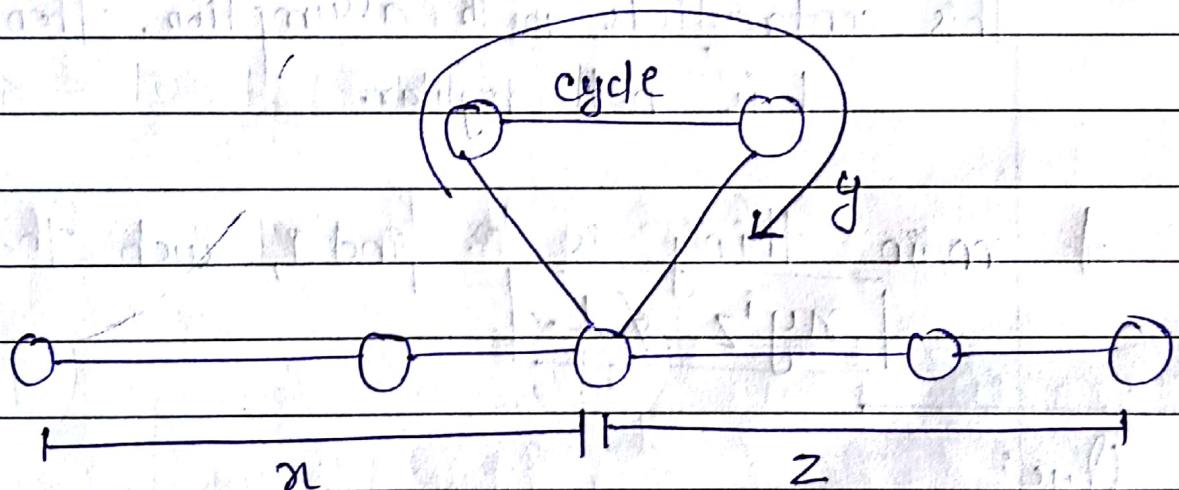
Let L be the regular set accepted by M .

Let, $w \in L$ and $|w| \geq n$.

If, $m \geq n$, then there exist x, y, z such that,

$w = xyz$, $xy \neq \emptyset$ & $xy^iz \in L$ for each $i \geq 0$

$\text{or, } |y| > 0$ $i \geq 0$
 $\text{or, } |y| \geq 1$



$$|y| \leq n$$

Application of Pumping Lemma

Step 1: Assume that L is regular. Let ' n ' be the no. of states in corresponding FA.

Step 2: choose a string w such that $|w| \geq n$ using pumping lemma to write, $w = xyz$ with, $|xy| < n$ and $|y| > 0$.

Step 3: find a suitable i , such that, $xyz^i \notin L$. This contradicts with assumption. Hence, L is not regular.

main thing is to find i , such that,
 $xyz^i \notin L$

Ques:

1). $a^n | n \geq 1$ (R)

2). $a^n b^m | n, m \geq 1$ (R)

3). $a^n b^n | n \leq 10^{10}$ (R)

4). $a^n b^n | n \geq 1$ (NR)

5). $w w^R | |w| = 2, S = \{a, b\}$ (R)

6). $w w^R | w \in (a, b)^*$ (NR)

7). $w w | w \in (a, b)^*$ (NR)

8). $a^n b^m c^t | n, m, t \geq 1$ (R)

9). $a^i b^{2j} | i, j \geq 1$ (R)

IMP.

10). $a^i b^j \mid i, j \geq 1 \text{ (R)}$

If pattern is
not there

11).

(a) $a^n \mid n \text{ is even (R)}$

or, pattern is
in gp.

(b) $a^n \mid n \text{ is odd (R)}$

=

(c) $a^n \mid n \text{ is prime (NR)}$

if (A.P. is
there, then

(d) $a^{n^2} \mid n \geq 1 \text{ (NR)}$

(e) $a^{2^n} \mid n \geq 1 \text{ (NR)}$

12). $a^i b^{j^2} \mid i, j \geq 1 \text{ (NR)}$

pattern is
there

13). $a^i b^{2^j} \mid i, n \geq 0 \text{ (NR)}$

14). $a^i b^p \mid i \geq 1, p \text{ is prime (NR)}$

15). $L = \{w \mid n_a(w) = n_b(w)\} \text{ (NR)}$

$$\rightarrow n_a(w) > n_b(w)$$

$$\rightarrow n_a(w) < n_b(w)$$

IMP.
 $\underbrace{100111001}_{w} \underbrace{000}_{w.R.}$

we can
make it

16). $a^n b^{n+m} c^m \mid n, m \geq 1 \text{ (NR)}$

as,
starting &
ending

17). $wxw^f \mid w, x \in \{0,1\}^*$ (R)

symbol
are
same.

18). $wxw^R \mid w \in \{0,1\}^* f \mid |x|=5$

or $|x|$ is

not
fixed.

(NR)

$\because |x|$ is fixed

so for all
string.

so, if length exceeds,
we'll need to
check.

$|w|=1 \neq$
rest will be
taken under
 $x \sim$

Q. $L = \{a^p \mid p \text{ is prime}\}$

1). Let us assume, 'L' is regular.

Let 'n' be the no. of states in finite automata accepting 'L'.

2). $w = a^p$, where p is prime & $p > n$
Acc.

$w = a^p$, where, p is prime & $p > n$.

acc. to Pumping lemma,

$$w = xyz \quad |xy| \leq n \\ |y| > 0$$

Let,

$$y = a^m, m \geq 1 \quad \& \quad m \leq n$$

$$xyz = w = \underbrace{a^k a^m a^0}_{x} \rightarrow z \\ k + m + 0 = p$$

$$ay^iz = a^p$$

$$(y^i)^m$$

$$|ay^iz| = |xyz| + |y^{i-1}|$$

$$= p + (i-1)m$$

for $i = p+1$

$$= p + (p+p-1)m$$

$$= p + pm = p(1+m)$$

product of two

no's can't

be a prime

one prime is the

Q. $L = \{a^i b^i \mid i \geq 1\}$

1). assume 'L' is regular.
let 'n' be the no. of states.

2). $w = a^i b^i$, where, $i \geq 1$

acc. to pumping lemma

$$w = xyz \quad |y| \leq n \quad |y| > 0$$

let

$$y = (ab)^m, m \geq 1 \text{ & } m \leq n$$

$$xyz = w = (ab)^k (ab)^m (ab)^l$$

$$\boxed{(k+m+l)} = \boxed{2i}$$

$$|xyz| = |yz| + |y|^2$$

$$= p + (p-1)m$$

~~$\cancel{= p + (p-1)m}$~~

~~$\cancel{= p + p(m-1)}$~~

~~$\cancel{= p + pm =}$~~

~~$\cancel{\text{Put repeat } p = i+1}$~~

$$= 2i + (i+1-1)m$$

$$= 2i + im = i(2+m)$$

i.e. L

Here, for $m = \text{odd}$,
result will be odd.
∴ Not regular.

~~*Correct Proof~~

$$\text{Let } w = a^n b^n$$

$$|w| = 2n > n$$

$$w = xyz$$

$$|xy| \leq n, |y| > 0$$

$$\text{let, } y = a^k, 1 \leq k < n$$

$$\text{let, } n = a^q, 0 \leq q \leq n$$

$$xyz = a^q a^k a^{n-q-k} b^n$$

$$(ny^2z) = a^q a^{2k} a^{n-q-k} b^n$$

$$= a^{q+2k+n-k} b^n$$

$$= a^{q+n+k} b^n$$

$$n+k \neq n, k \geq 1$$

$$Q. L = \{ wwww^* \mid w \in (a, b)^* \}$$

$$\text{Let } w = \underline{a^n b}$$

i). Let, L is regular.

$$www = \underline{a^n b a^n b}$$

$$\text{Let, } y = a^k, \quad k \geq 1 \quad \& \quad k < n$$

$$nyz = a^m a^k b a^n b \quad | \quad m = n-k$$

$$= \underbrace{a^{n-k}}_n \underbrace{a^k}_y \underbrace{b a^n b}_z$$

$$\left\{ \begin{array}{l} |nyz| = 2n+2 \\ |yz| = n+k \end{array} \right.$$

$$|ny^i z| = |nyz| + |y^{i-1}|$$

$$= (2n+2) + (i-1)k$$

$$= 2n+2 + ((n+1)-1)k \quad | \quad \text{put } (i=n+1)$$

$$= 2n+2 + (nk)$$

$$= 2n+nk+2$$

$$= n(2+k)+2$$

~~for i=2~~

Put i=2

$$ny^2 z = a^{n-k} a^{2k} b a^n b$$

$$= a^{n-k+2k} b a^n b$$

$$= a^{n+k} b a^n b \neq \underline{a^n b a^n b}$$

If L_1 and L_2 are two regular languages, then,

$\underline{L_1 \cup L_2}$, $\underline{L_1 \cap L_2}$, $\underline{L_1 \cdot L_2}$, $\underline{\overline{L_1}}$, $\underline{L_1^*}$, $\underline{L_1/L_2}$

Proof
simply prove that these also regular.

By prop.
using F.A.

↓
Right
Quotient

→ By making finite automata.

→ By generating regular expression.

→ By " regular grammar

i) UNION ($L_1 \cup L_2$)

$$L_1 \rightarrow M_1 = \{ S_1, \Sigma, \delta_0, S_1, F_1 \}$$

$$L_2 \rightarrow M_2 = \{ S_2, \Sigma, \delta_0, S_2, F_2 \}$$

$$L(M_1) = L_1, L(M_2) = L_2$$

We can define an,

F.A.,

$$M = \{ S, \Sigma, \delta, S, F \}$$

and,

$$\delta = (S, \cup S) \times \Sigma \rightarrow \{ (S, \cup S) \}$$

$$S_0 = S_0 \cup S_0$$

$$S = S_1 \cup S_2$$

$$F = F_1 \cup F_2$$

$$\delta(\varsigma, a) = \begin{cases} \delta_1(\varsigma, a) & \text{if } \varsigma \in S_1 \\ \delta_2(\varsigma, a) & \text{if } \varsigma \in S_2 \end{cases}$$

for all $\varsigma \in S_1 \cup S_2$ and for all

$a \in \Sigma$.

$\Rightarrow (L_1 \cup L_2)$ is regular.

II) CONCATENATION ($L_1 \cdot L_2$)

$$L_1 \rightarrow M_1 = \{ \varsigma_1, \Sigma, S_{01}, \delta_1, F_1 \}$$

$$L_2 \rightarrow M_2 = \{ \varsigma_2, \Sigma, S_{02}, \delta_2, F_2 \}$$

$$S_1 \cap S_2 \neq \emptyset$$

$$M = \{ \varsigma, \Sigma, S_0, \delta, F \} \quad | \quad F = \{ F_2 \text{ if } \varsigma \in \emptyset^* L_2 \}$$

$$S_0 = S_{01} \cup S_{02}$$

$$\text{if } \varsigma \in \emptyset^* L_1 \quad S_0 = S_{01}$$

$$F_1 \cup F_2 \text{ if } \varsigma \in L_2$$

or,

$$S_0 \text{ and if } \varsigma \in \emptyset^* L_1 \quad S_0 = S_{01}$$

and,

$$\delta : (S_1 \cup S_2) \times \Sigma \rightarrow S_1 \cup S_2$$

$$\in \emptyset^* L_1$$

$$\delta(\varsigma, a) = \begin{cases} \delta_1(\varsigma, a) & \text{if } \varsigma \in S_1 \end{cases}$$

$$\left. \begin{cases} \delta_1(\varsigma, a), \delta_2(\varsigma, a) & \text{if } \varsigma \in F_1 \\ \delta_2(\varsigma, a) & \text{if } \varsigma \in F_2 \end{cases} \right\}$$

$$\delta_2(\varsigma, a) \text{ if } \varsigma \in F_2$$

III) COMPLEMENT (IV)

$$L \rightarrow M = (Q, \Sigma, S_1, q_0, F)$$

$$\bar{L} \rightarrow M = (Q, \Sigma, S_1, q_0, Q - F)$$

IV) INTERSECTION (L1 ∩ L2)

$$M_1 = (S_1, \Sigma, S_{01}, \delta_1, F_1)$$

$$M_2 = (S_2, \Sigma, S_{02}, \delta_2, F_2)$$

$$L_1 = L(M_1), L_2 = L(M_2)$$

We define a DFA :-

$M = (S_1 \times S_2, \Sigma, S_{01}, \delta, F)$ such that,

$$S = S_1 \times S_2$$

$$S_0 = (S_{01}, S_{02})$$

\downarrow
Pair

$$F = F_1 \times F_2$$

$$\delta : (S_1 \times S_2) \times \Sigma \longrightarrow S_1 \times S_2$$

if defined by,

$$\delta((q_1, p), a) = (\delta_1(q_1, a), \delta_2(p, a)) = (s_1, s_2)$$

for all $q \in S_1$ and $p \in S_2$ for all $a \in \Sigma$

$$\text{ori } \delta((q,p), a) = (s_1)$$

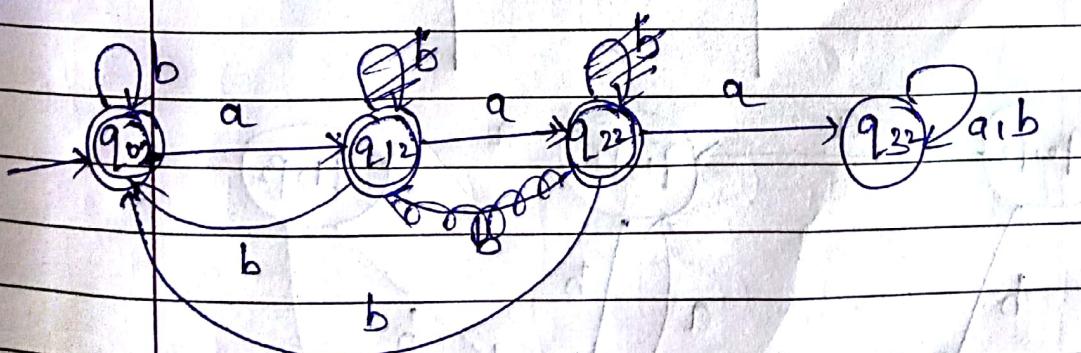
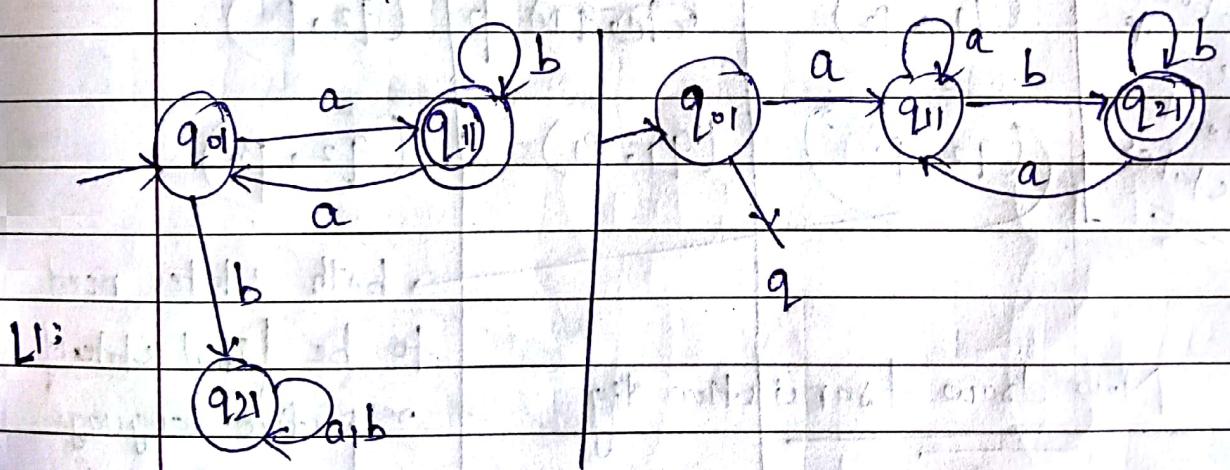
where $s \in S_1$ and $t \in S_2$

Q1

Q. Intersection

$\Sigma(a|b)$, L_1 is the language that begins with a a and ends with b .

$L_2 \rightarrow$ triple a is not a substring.



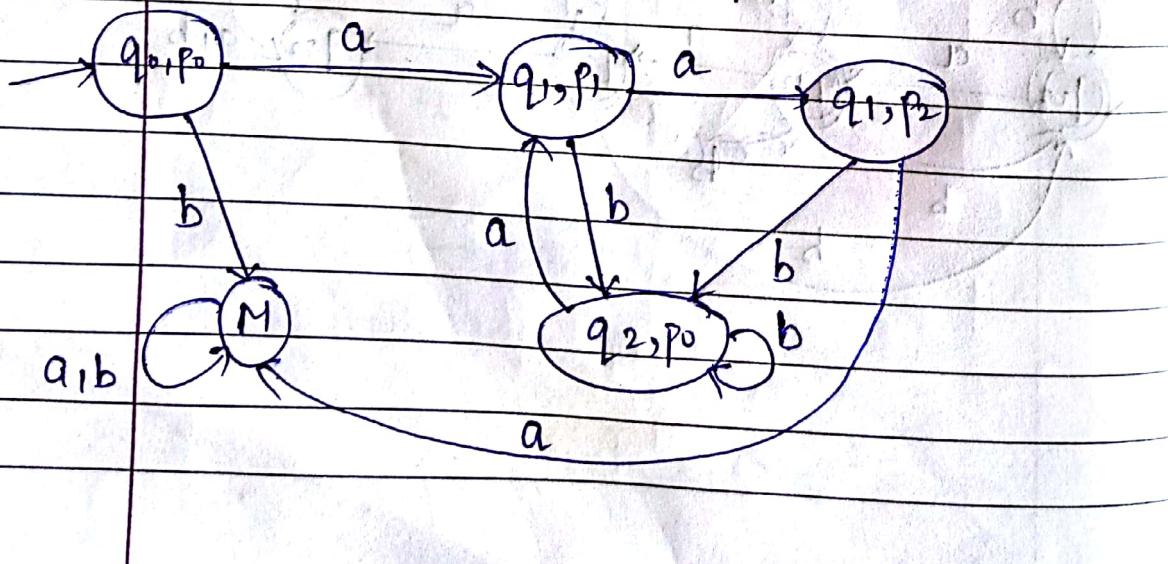
Transition table of

(L)

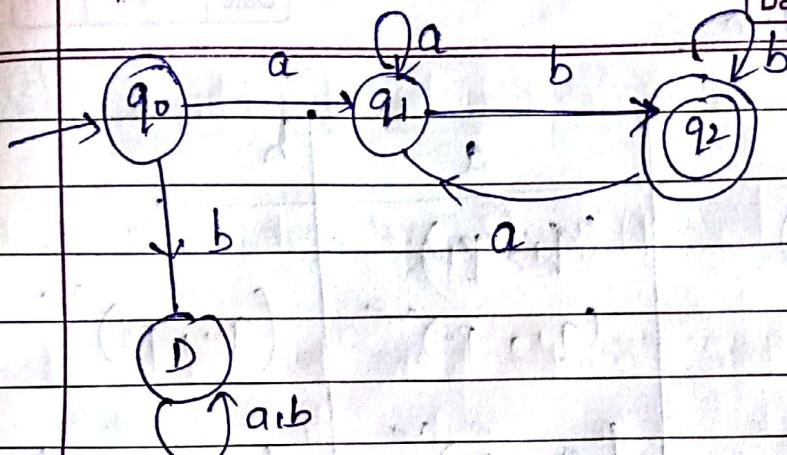
	a	b
(q_{01}, q_{02})	(q_{11}, q_{12})	(q_{21}, q_{02})
(q_{11}, q_{12})	(q_{01}, q_{12})	(q_{21}, \emptyset)
$\overbrace{\text{IMP}}^{\text{if any}} \rightarrow \text{Pairs}$	a	b
one leads to dead states	(q_1, p_1)	\emptyset
whole p lead dead states	(q_1, p_1)	(q_2, p_0)
(q_1, p_2)	(q_2, \emptyset)	(q_2, p_0)
(q_2, p_0)	(q_1, p_1)	(q_2, p_0)

→ both states need
to be final states of
respective language.

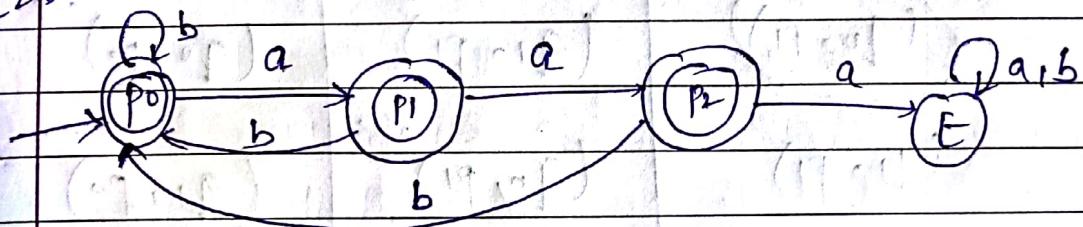
Now draw transition dig



L1:

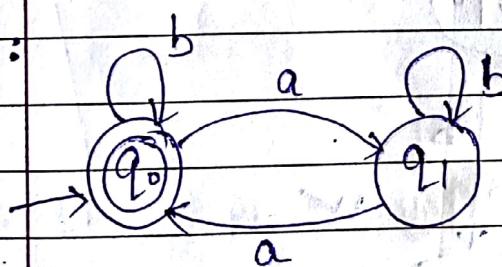


L2:

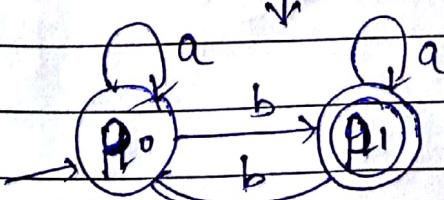


- Q. Draw for, no. of a's - even (Intersection)
- no. of b's - odd

L1:

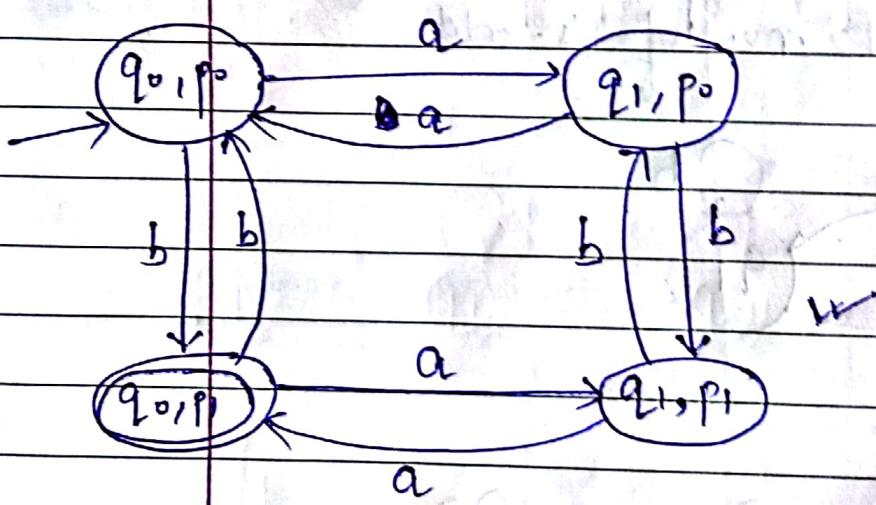


L2:



Transition

	a	b
$\rightarrow (q_0, p_0)$	(q_1, p_1)	(q_0, p_1)
(q_1, p_0)	(q_0, p_0)	(q_1, p_1)
(q_0, p_1)	(q_1, p_1)	(q_0, p_0)
(q_1, p_1)	(q_0, p_1)	(q_1, p_0)



IV) RIGHT QUOTIENT. (L_1/L_2)

$$L_1/L_2 = \{ n : xy \in L_1 \text{ for some } y \in L_2 \}$$

①. $L_1 = \{ \text{Fish, dog, carrot} \}$

$L_2 = \{ \text{rotl} \}$ then,

$$L_1/L_2 = \{ \text{car} \}$$

IMP.
from L_1 , we take
all those strings
whose suffix is
in L_2 after
removing their
suffix part.

②. $L_1 = \{ \text{Carrot, Parrot, rot} \}$

$$L_2 = \{ \text{rot} \}$$

$$L_1/L_2 = \{ \text{Car, Par, } \epsilon \}$$

$$L_3 = \{ \text{rot, cheese} \}$$

$$L_1/L_3 = \{ \text{Car, Par, } \epsilon \}$$

③. $L_1 = \{ xab, Jab \}$, $L_2 = \{ b, ab \}$ then,

$$L_1/L_2 = \{ x, J, x, J \}$$

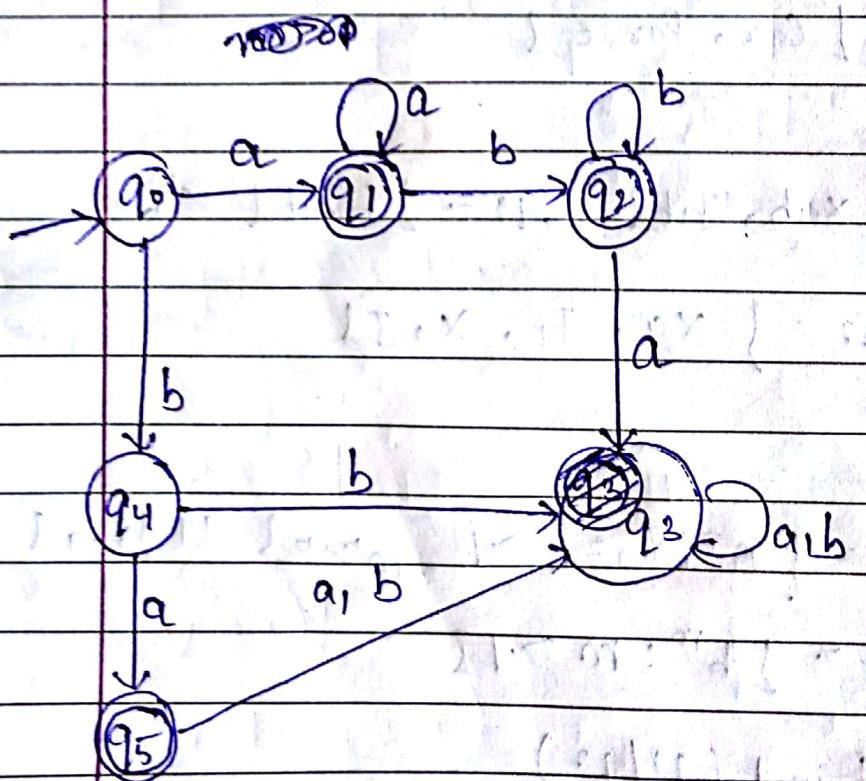
Q. If, $L_1 = \{ a^n b^m : n \geq 1, m \geq 0 \} \cup \{ b^m \}$

$$L_2 = \{ b^m : m \geq 1 \}$$

Find (L_1/L_2)

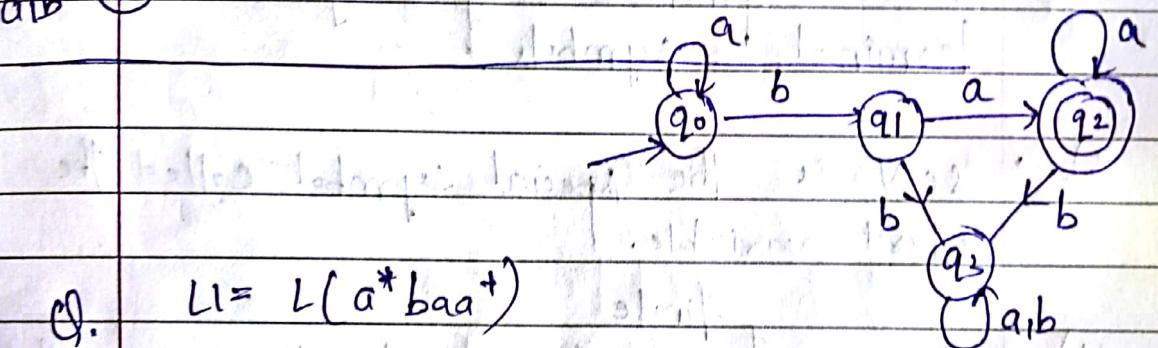
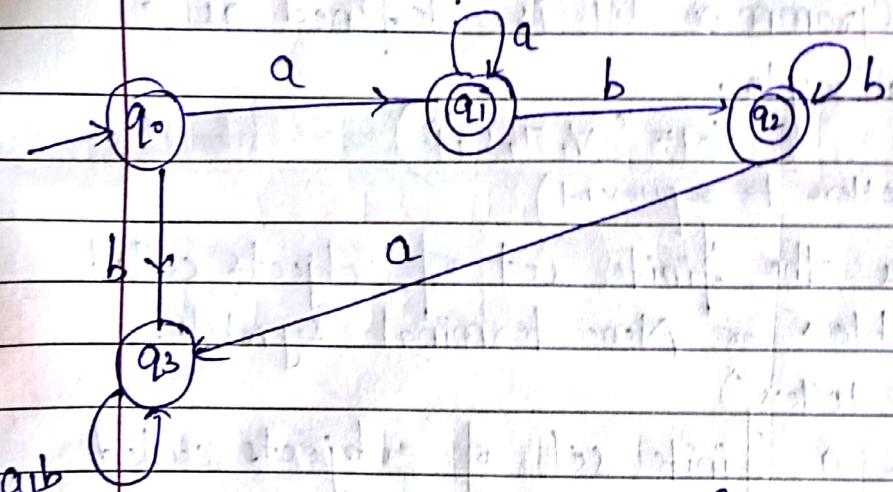
1. Draw finite automata of L1.
2. Try to apply ② on each state of L1.
3. If, there exist a path to any final state then make that state as a final state.
- 4). Repeat ③ for all states.
- 5). At last, all final states received are final states and others are non-final.

C). $(a^n b^m : n \geq 1, m \geq 0) \cup L_2 b \{ \}$



Proof is → Peter Linke
given
 $L_1 = \{a^* b a a^*\}$

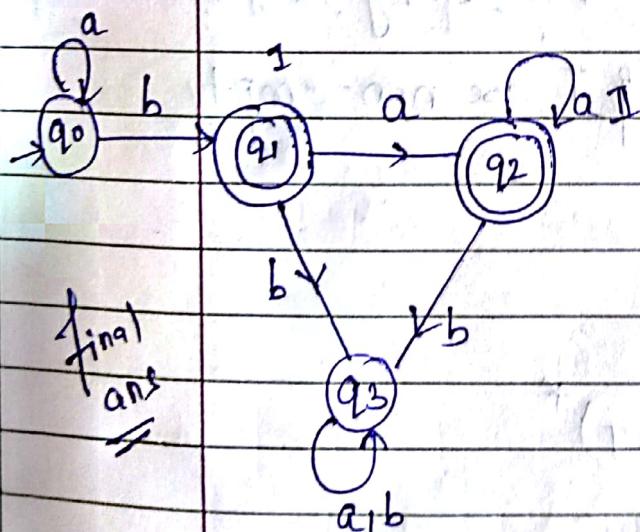
Page No.	_____
Date	_____



$$L_1 = L(a^* b a a^*)$$

$$L_2 = L(ab^*)$$

$$L(M_0) \cap L_2 = \emptyset$$



$$L(M_1) \cap L_2 = \{a\} \neq \emptyset$$

$$L(M_2) \cap L_2 = \{a\} \neq \emptyset$$

$$L(M_3) \cap L_2 = \emptyset$$

$$\begin{aligned}
 & \text{to reach final state } \text{II} \\
 & a^* b + a^* b a a^* \\
 & = a^* b (\epsilon + a a^*) \\
 & = a^* b a a^*
 \end{aligned}$$

GRAMMAR :-

→ Called generators of the language.

→ A Grammar G is defined as a quadruple,

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

G (Capital letter to represent)

1). → V is the finite set of objects called variables or Non-terminal symbols.

(Small letter)

2). → T is a finite set of objects called terminal symbols.

3). → $S \in V$ is the special symbol called the start variable.

4). → P is a finite set of productions called rules, how string is generated

5). → $V \cap T = \emptyset$ i.e. V & T are non-empty disjoint sets.

$$n \rightarrow y$$

$$(V \cup T)^*$$

$(V \cup T)^+$
 (not epsilon)

* T is the superset of Σ .

Page No.			
Date			

$G_1 = \{ \Sigma, A, B! , \{a, b\}, S, L : S \rightarrow AB, A \rightarrow^q, B \rightarrow b \}$

Derivation from a grammar

$$w = u \cdot v$$

$n \rightarrow y$ is applicable to this string.

the can
be replaced

by $L(T)$
in derivation.

$$z = u \cdot y \cdot v$$

$$w \Rightarrow z$$

$$w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w_n \quad \text{by applying multiple derivations.}$$

$$\text{or, } w_1 \xrightarrow{*} w_n$$

+ from another
string.

$$L(G_1) = \left\{ w \in T^* \mid S \xrightarrow{*} w \right\}$$

If, $w_0 \in L(G_1)$ then the sequence,

$$(\Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w)$$

derivation of the sentence w_0 .

final string

intermediate forms
= sentential form.

Eg. $G_1 = \{ S, S, S, S, a, b, \epsilon, P \}$ where,

$$P = \{ S \rightarrow aSb, S \rightarrow \epsilon \}$$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb = \underbrace{\dots}_{\text{Sentential form.}} \underbrace{a^n b^n}_{n \geq 0} \quad \text{Sentence.}$

II. HOW WE CONVERT GRAMMAR TO A LANGUAGE

Q. 1. $\{ A, S \}, \{ a, b \}, S, P \}$

$$P = \{ S \rightarrow aAb | e \}$$

$$A \rightarrow aAb | e$$

$S \Rightarrow aAb \Rightarrow aaAbb \Rightarrow aaaAbbbb = \underbrace{\dots}_{(n \geq 0)} \underbrace{a^n b^n}_{n \geq 0}$

Q. 2. $G_1 = \{ S, S, S, S, \{ S \rightarrow ss \} \}$

$$\begin{aligned} S &\Rightarrow ss \Rightarrow s|ss \Rightarrow ss|ssss \Rightarrow \cancel{ssss} \\ &\Rightarrow ss - s|ss \quad \left(+ \right)^n ; n \geq 1 \end{aligned}$$

language is ϕ , \because no terminal symbol

Q.3. $G = \{S, C\}, \Sigma_{a,b}, P$ विद्युत,

$$P = \{S \rightarrow aCa, C \rightarrow (aCa) | b\}$$

$$\begin{aligned} S &\Rightarrow aCa \Rightarrow a a C a a = aa \\ &\Rightarrow a a C b a \end{aligned}$$

$$\begin{aligned} S &\Rightarrow aCa \Rightarrow a a C a a \\ &\Rightarrow aba \quad \text{or} \quad a a b a a \\ &= (a^n b^n a^n) \quad | n \geq 1 \end{aligned}$$

Q.4. $S \rightarrow aSB$

$$\begin{aligned} S &\rightarrow aB \\ B &\rightarrow b \end{aligned}$$

$$S \Rightarrow aSB = a a SB B$$

$$a a SB B = a a b b \quad a^n b^n \quad | n \geq 1$$

Q.5. $S \rightarrow aSB | ab$

$$\begin{aligned} S &\rightarrow aSB = a a S B B \\ &\rightarrow ab \quad a^n b^n \quad | n \geq 1 \end{aligned}$$

11 GENERATE GRAMMAR FOR A LANGUAGE

Q.1. (All strings of length 2)

$$\begin{array}{l}
 S \rightarrow AB \\
 \quad\quad\quad A \rightarrow aa \\
 \quad\quad\quad A \rightarrow ab \\
 \quad\quad\quad A \rightarrow ba \\
 \quad\quad\quad B \rightarrow bb
 \end{array}$$

$$S \rightarrow aa | ab | ba | bb.$$

~~$S \rightarrow AA$~~

$$\left. \begin{array}{l}
 S \rightarrow AA \\
 A \rightarrow aAb
 \end{array} \right\} \begin{array}{l}
 \text{all strings of length } 2 \\
 (a+b)(a+b)
 \end{array}$$

Q.2. $L = \{a^n \mid n \geq 0\}$

$$\begin{array}{l}
 S \rightarrow aS | e \\
 S \not\rightarrow \alpha
 \end{array}$$

* building block for *, $S \rightarrow S|e$ S^*

③. $L = (a+b)^*$

$$\begin{array}{l}
 S \rightarrow ABS | e \\
 A \rightarrow aAb \\
 B \rightarrow aBb
 \end{array}
 \quad S \rightarrow aSbS | e$$

④. $L = \{ \text{set of all strings of at least 2 length} \}$

✓

$$S \rightarrow AAB$$

$$A \rightarrow a/b$$

$$B \rightarrow aB/bB/e$$

e

$$S \rightarrow ACB$$

$$A \rightarrow a/b$$

$$B \rightarrow a/b$$

$$C \rightarrow ACB/e$$

$$S \rightarrow (ACB)$$

$$CACB \quad \underline{CEE}$$

⑤. $L = \{ \text{at most 9 length} \}$

$$S \rightarrow AB/A/e$$

$$A \rightarrow a/b$$

$$B \rightarrow a/b$$

$$S \rightarrow AA$$

$$A \rightarrow a/b/e$$

⑥. $L = \{ \text{starting with 'a' and ending with 'b'} \}$

$$S \rightarrow aSb$$

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

$$S \rightarrow aACBb$$

$$A \rightarrow a/b/e$$

$$B \rightarrow a/b/e$$

$$C \rightarrow aACBb/e$$

$$S \rightarrow aBb$$

$$B \rightarrow aB/bB/e$$

⑦. $L = \{ \text{starting and ending with same symbol} \}$

$$a(a+b)^+a + b(a+b)^+b$$

$$S \rightarrow aBa/bBb/a/b/e$$

$$B \rightarrow aB/bB/e$$

even length odd length palindrome

1. $L = \{ wwo^R \cup woow^R \cup wbw^R \}$ and,
 $w \in \{a, b\}^*$

$$S \rightarrow asa \mid bsb \mid a \mid b \mid e$$

$$S \rightarrow asa \mid bsb \mid e \rightarrow \text{even length}, S \rightarrow asa \mid bcb \mid a \mid b \mid e \rightarrow \text{odd length}$$

2. $L = \{ \text{Even length strings} \}$
 $((a+b)(a+b))^*$

$$S \rightarrow AS \mid e$$

$$A \rightarrow BB$$

$$B \rightarrow ab \mid a \mid aa \mid \dots$$

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

$$S \rightarrow AB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB \mid b$$

4. $\cancel{a^n b^n c^m} \mid n, m \geq 1$

$$S \rightarrow AC$$

$$A \leftarrow aA \mid a \quad A \rightarrow aAb \mid ab$$

$$C \rightarrow cc \mid c$$

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

~~$$A \rightarrow acb \mid b$$~~

~~$$C \rightarrow cab \mid c$$~~

$$S \rightarrow asb \mid aAb$$

$$A \rightarrow ca \mid c$$

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

$$A \rightarrow aAb \mid ab$$

$$B \rightarrow cBd \mid cd$$

$$S \rightarrow AB \quad \checkmark$$

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

$$S \rightarrow AB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bbB \mid bb$$

$$S \rightarrow aSbb \mid abb$$

$$\checkmark$$

8. $a^{m+n} b^m c^n \mid m, n \geq 1$

$$a^m a^n b^m c^n$$

$$\underline{a^n a^m b^m c^n}$$

$$A \rightarrow aAb \mid ab$$

$$S \rightarrow aSc \mid aAc$$

$$\checkmark$$

Types Of Grammars

- Chomsky hierarchy of Grammars

Type 0 or unrestricted grammar

Type 1 or context sensitive

Type 2 or context free

Type 3 or regular grammar

- Type 0

$$\alpha \rightarrow \beta$$

$$\alpha \in (VUT)^+$$

$$\text{and } \beta \in (VUT)^*$$

e.g. $aAb \rightarrow bB$

$$aA \rightarrow e \quad \text{or} \quad CB \rightarrow DB$$

$$Bc \rightarrow acB$$

- Type 1

$$aAb \rightarrow \beta$$

\downarrow left context \rightarrow right context
of A

context free, $\epsilon \in A \epsilon \rightarrow \beta$.

context sensitive, $aA\beta \rightarrow \beta$

→ only changed to β , when

A is preceded by
'a' if succeeded by
'b'

$\alpha \rightarrow \beta$

$|\alpha| \leq |\beta|$ i.e. length of $\alpha \leq \beta$

$\alpha, \beta \in (V \cup T)^+$

→ length increasing grammars.

$AB \rightarrow AbBc$

$S \rightarrow aSbc$

$S \rightarrow abc$

$CB \rightarrow Bc$

$bB \rightarrow bb$

$\Leftrightarrow \epsilon$ cannot be there on RHS. of any production

of $S \rightarrow \epsilon$, then S should not belong to RHS of any production

If language
is accepting
then follow
 ϵ , then this.

Type 2:

$$A \rightarrow a$$

$$a \rightarrow (VUT)^*$$

$$\text{and, } A \in V$$

$$A \rightarrow C$$

$$A \rightarrow B+C$$

can have many variables on RHL

but can have only one variable

on LHS (that's how it's understood).

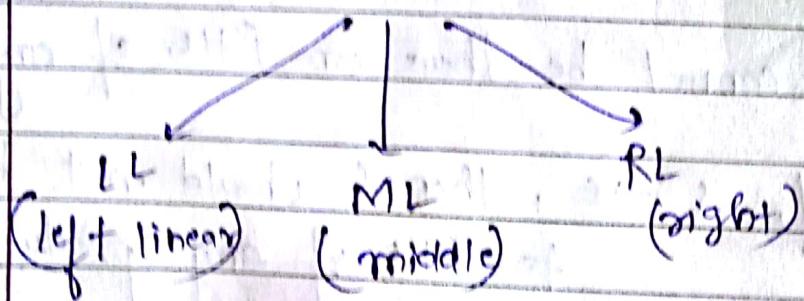
(context
in terms
of small
obj.)

3) LINEAR GRAMMAR:

any grammar in which, there exist,

exactly one variable on LHS

almost one variable on RHL.



$$S \rightarrow aSb \quad (\text{ML})$$

$$A \rightarrow aBb \quad (\text{RL})$$

$$S \rightarrow Ba|b \quad (\text{LL})$$

Linear Grammar = $LL + ML + RL$

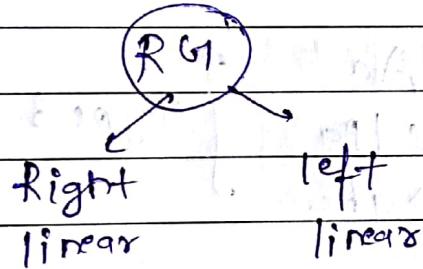
any combination

Type 3: Regular Grammar

$RG = LL + RL$

any one. (either LL or RL)
(not together)

$RG \subset LG$

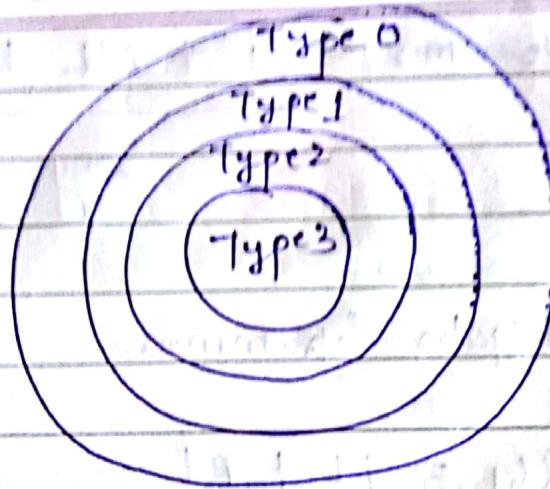


$V \rightarrow T^* / T^* V$ $V \rightarrow T^* / VT^*$

Eg. $A \rightarrow nB/n$
where $A, B \in V$
(R.L.)

$A \rightarrow Bn/n$ $n \in \Sigma^* \text{ or } T^*$
(L.R.)

$\left. \begin{array}{l} \text{LHS is context free} \\ \text{RHS has only one} \\ \text{variable.} \end{array} \right\}$



$[Type 3 \subset Type 2 \subset Type 1 \subset Type 0]$

Highest form of the grammar?

Eg:

$$\begin{array}{l} S \rightarrow Aa \\ A \rightarrow C \mid Ba \\ B \rightarrow abc \end{array} \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} Type 3$$

$\rightarrow Type 2$

Eg:

$$\begin{array}{l} S \rightarrow A \cup B \cup d \\ A \rightarrow aA \end{array} \Rightarrow \left. \begin{array}{l} S \rightarrow A \cup B \cup d \\ A \rightarrow aA \end{array} \right\} Type 3$$

(Combined
Type 2)

CONVERSION OF FA to RG

Page No. _____

Date _____

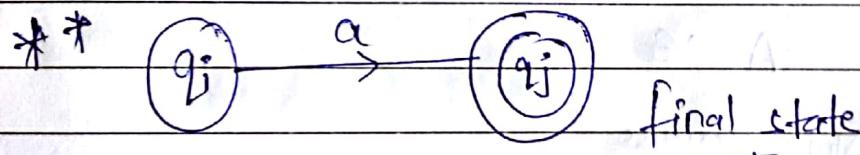
We can convert an automata to a grammar

$$G = \{ S, A_0, A_1, \dots, A_n \}, T, P, A_0 \}$$

where, P is defined by the following rules.

(i) $A_i \rightarrow a A_j$ is included in P if $\delta(q_i, a) = q_j \notin F$

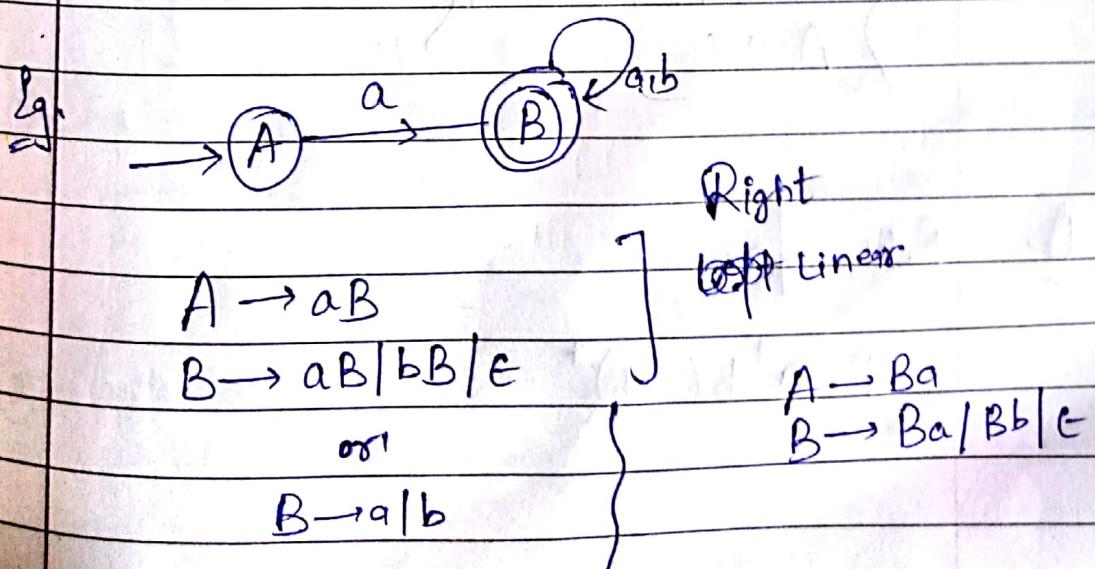
(ii) $A_i \rightarrow a A_j$ and $A_i \rightarrow a$ if $\delta(q_i, a) = q_j \in F$
or we can also, $A_i \rightarrow e$ for final state
remove $(A_i \rightarrow a)$. Put



$$A_i \rightarrow a A_j \quad \text{&} \quad A_i \rightarrow a$$

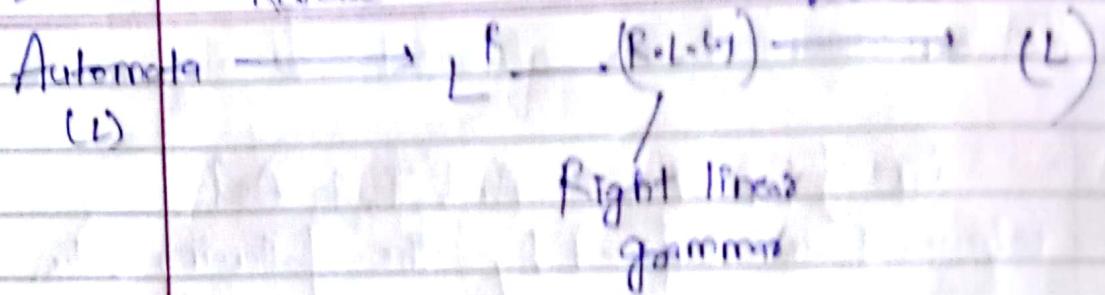
or

$$A_i \rightarrow a A_j \quad \text{&} \quad A_j \rightarrow e$$

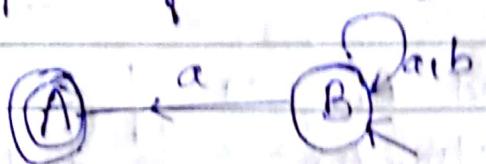


VII.

Reverse

~~Q.~~

to find left linear grammar



$$B \rightarrow aA$$

$$B \rightarrow aB \mid bB$$

$$A \rightarrow G$$

Now reverse,

$$B \rightarrow Aa$$

$$B \rightarrow Ba \mid Bb$$

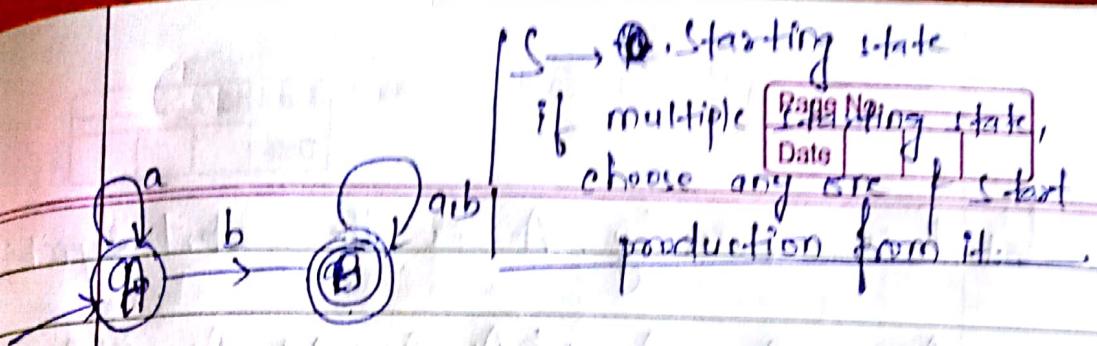
$$A \rightarrow \epsilon$$

$$\left\{ \begin{array}{l} B \rightarrow Aa \mid Ba \mid Bb \\ A \rightarrow \epsilon \end{array} \right. \times$$

~~Q.~~

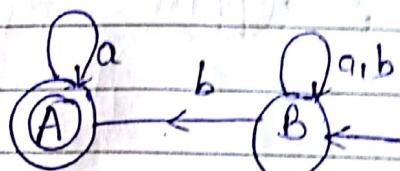
R.GI. for,

$$P = a^* b (a+b)^*$$



$A \rightarrow aA \mid bB$ Right linear.

$B \rightarrow aB \mid bB \mid \epsilon$



$A \rightarrow aA$
 $B \rightarrow bA \mid aB \mid bB$

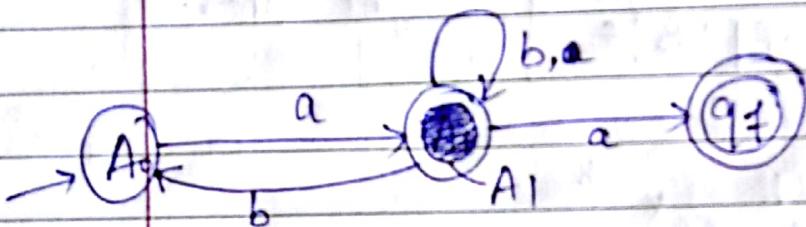
~~Reverse~~
 $\left. \begin{array}{l} A \rightarrow Aa \\ B \rightarrow Ab \mid Ba \mid Bb \end{array} \right\}$

Regular Grammars to Automata

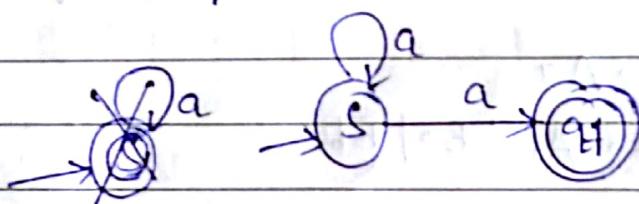
- ① Each production $A_i^* \rightarrow a A_j$ indicates a transition from q_i to q_j with label a .
- ② Each production $A_k \rightarrow a$ induces a transition from q_k to q_f with label a where, $q_f \in F$.
- ③ Make a new state from A_k to q_f as final with label a .

Q.1. P is given by. A₀ is the start symbol.

$$A_0 \rightarrow aA_1, A_1 \rightarrow bA_1, A_1 \rightarrow a, A_1 \rightarrow bA$$



Q.2. $S \rightarrow aS/a$



IMP

* Automata from Left Linear grammar.

1). Reverse it to get (R.L.G)

2). Make automata for (R.L.G) i.e. automata for

3). Now, Reverse automata. L^R .

i.e. Automata of (\bar{L}).

Context Free Grammar

Type-2 Grammar

Page No. _____
Date _____

Derivation Tree or Parse Tree

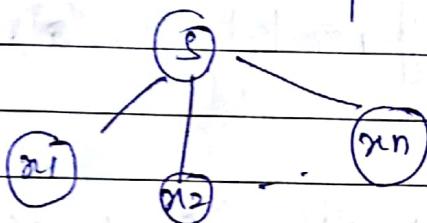
$$A \rightarrow n$$

where, AεV and

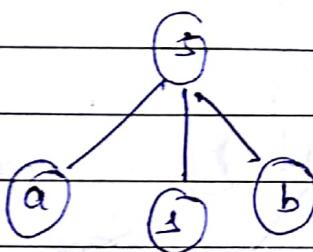
$$n \in (V \cup T)^*$$

- ① Root vertex (starting symbol)
- ② Vertex (variables - capital letters)
- ③ leaves (terminal symbols) $\in T$
- ④ $\Sigma \Rightarrow S \rightarrow n_1 n_2 n_3 \dots n_r$

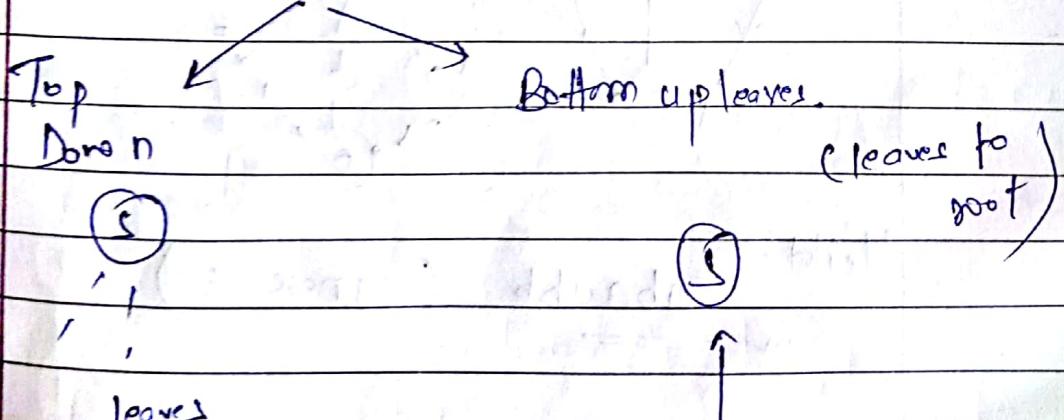
is production rule then the
 ④ Derivation Tree or parse tree,



$$S \rightarrow a S b$$



Free construction



(Start from top
move towards the leaves)

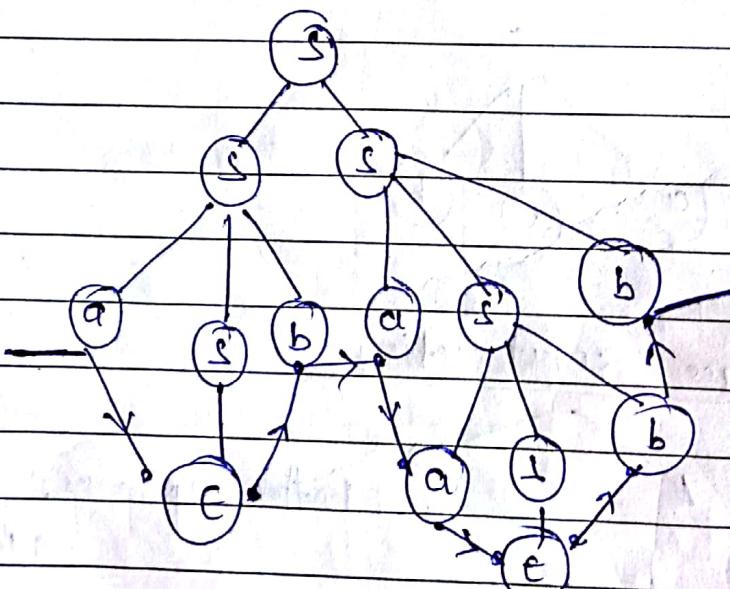
- Tree is not for grammar.
- It is drawn for a particular string.

• Yield of Tree :- String generated from tree
 Starting from leftmost side & check the leaves, ignoring ϵ)

Eg. → If not terminal symbol is present in leaves & only non-terminal & ϵ then language generated is, (ϵ) .

Eg. $G = \{ S \in \{a, b\}, S \rightarrow S \cup S^2, P = S \rightarrow aSb | \epsilon \}$

$S \Rightarrow S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abaab \epsilon \Rightarrow abaabb$



Yield:-

abaabb (ignore ϵ)

leftmost f right most derivation

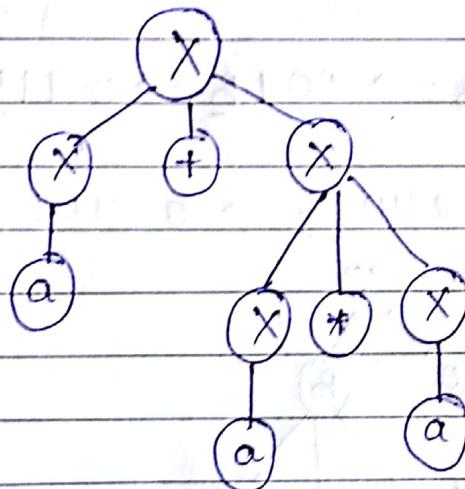
$\Sigma = \{a\}$

Eg. $X \rightarrow X + X \mid X * X \mid X \mid a$

* The leftmost derivation of a string "a+a*a" may be,

$$X \Rightarrow X + X \Rightarrow a + X \Rightarrow a + X * X \Rightarrow a + a * X \Rightarrow a + a * a$$

↑ ↑
leftmost leftmost



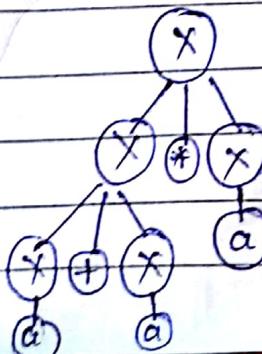
right most

$$X = X + X \Rightarrow X + a \Rightarrow X * X + a \Rightarrow a * X + a \Rightarrow a + a * a$$

or

$$X = X * X \Rightarrow a * X \Rightarrow a *$$

$$\Rightarrow X * a \Rightarrow X + X * a \Rightarrow X + a + a \Rightarrow (a + a + a)$$



Q. Consider the grammar

$$S \rightarrow 0B \mid IA$$

$$A \rightarrow 0 \mid 0S \mid IAA$$

$$B \rightarrow 1 \mid 1S \mid 0BB$$

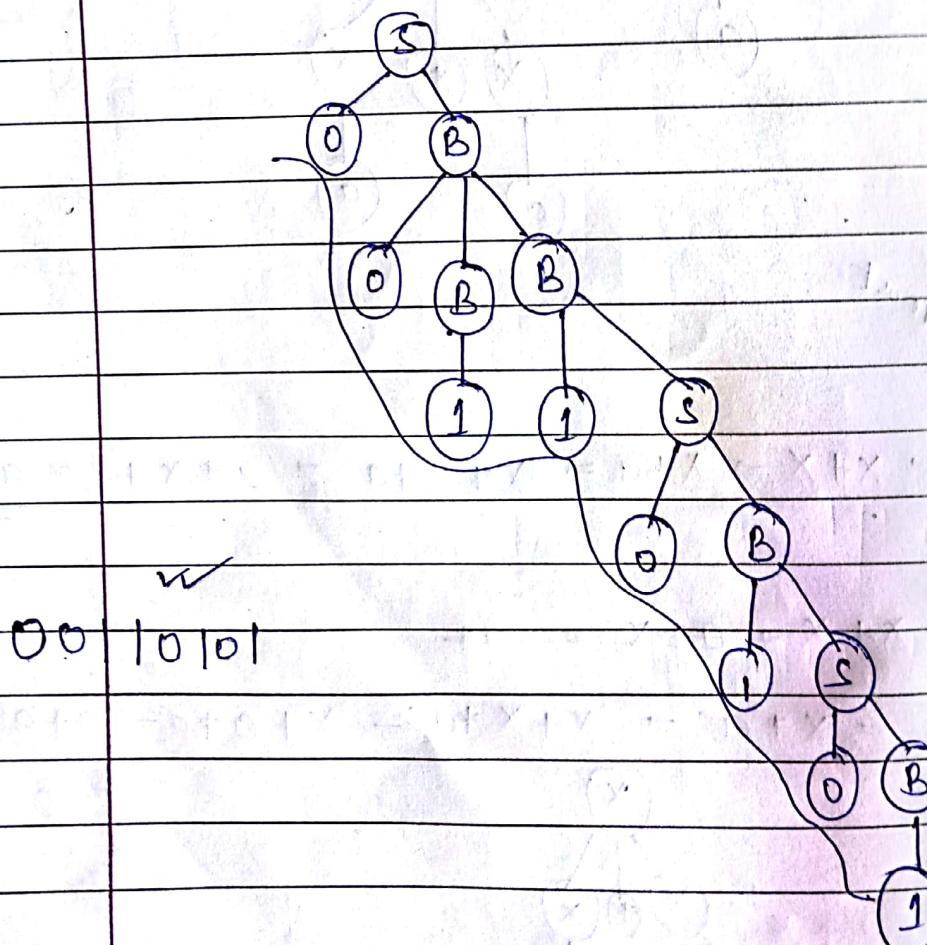
find left most & right most derivation

left most

$$\text{String} = "00110101"$$

$$S \rightarrow 0B \rightarrow 00\underline{BB} \rightarrow 001\underline{B} \rightarrow 0011\underline{S} \rightarrow 00110\underline{B}$$

$$\rightarrow 001101\underline{S} \rightarrow 0011010\underline{B} \rightarrow 00110101$$



5 - DB → DO BB → DO BI → DO

Ambiguity in CFG

($w \in L(G)$) derivation

- If more than one tree can be drawn for a particular string, it is said to be an ambiguous grammar.

$$S \rightarrow S+S \mid S*S \mid a \mid b$$

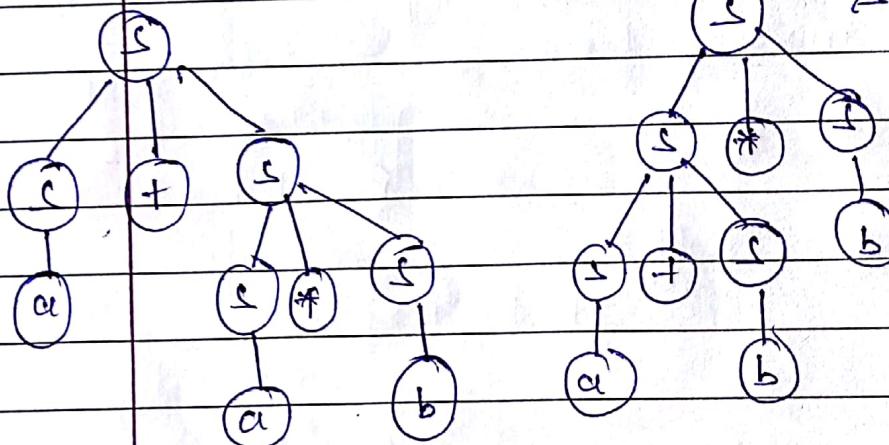
~~Two left derivations~~

$$a + a^*$$

$$S \Rightarrow S+S \Rightarrow a+S \Rightarrow a+S*S \Rightarrow a+a^*S \Rightarrow a+a^*$$

$$S+S*S \Rightarrow S+S*S \Rightarrow a+S*S + a+a^*S = a+a^*$$

 same string.



- To check ambiguity \Rightarrow use two left most or two right most

Q. Show that G is ambiguous.

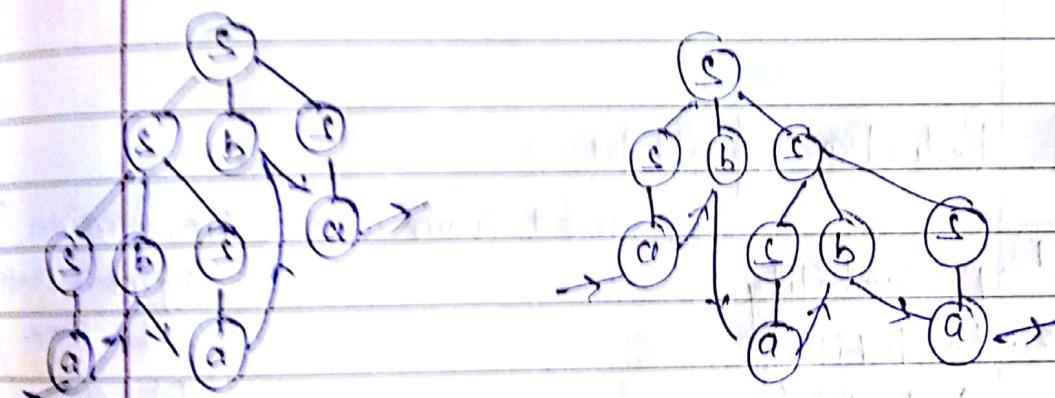
$S \rightarrow SbS \rightarrow abba=aaba$

$SbSbS \rightarrow abbaS \rightarrow ababa \rightarrow \text{ababa}$

$S \rightarrow SbS \rightarrow abS \rightarrow abcba \rightarrow ababa$

ababa

(same string)



SIMPLIFICATION OF CFG

Page No. _____

Date _____

- (1). Reduction of CFG
- (2). Removal of unit productions
- (3). Removal of null productions.

unit production eg :-

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \\ C \rightarrow a \\ A \rightarrow Aa \end{array}$$

↑ unnecessary
only

null productions

$$\begin{array}{l} A \rightarrow \epsilon \\ B \rightarrow \epsilon \end{array}$$

Reduction of CFG

- (1). Desirability
- (2). Reachability

→ what is our aim (i.e. terminal)

Symbol should
be there

Not only
variable to
variable

If terminal is reachable
from start
that state

$$E.g. S \rightarrow Aa$$

$$A \rightarrow a$$

$$B \rightarrow CD$$

$$C \rightarrow D$$

true

false

Measurable
from S.

$$P = S \rightarrow A c \mid B$$

$$A \rightarrow a$$

$$C \rightarrow c \mid B c$$

$$E \rightarrow a A \mid e$$

TS(a, c, E)

$$V = \{S, A, B, C, E\}$$

check
Desirability

$$W_1 = \{S, A, C, E\}$$

set
that
generate
terminal
symbol.



$$W_2 = W_1 \cup \{ \text{ } \}$$

all those variables
whose RHS is in
 $\{W_1, U\}$

$$W_2 = \{A, C, E, S\}$$

$$W_3 = W_2 \cup \phi$$

stop

\Rightarrow remove all production of B.

$$P = S \rightarrow A c$$

$$A \rightarrow a$$

$$C \rightarrow c$$

$$E \rightarrow a A \mid e$$

check
Desirability

$$Y_1 = \{S, A, C, E\} \nearrow \text{Variables}$$

$$Y_2 = Y_1 \cup \{ \text{symbols and terminals that can be reached from } S \}$$

$$Y_2 = \{A, S, C\}$$



$$Y_3 = \{S, A, C, a\}$$



Hence, C & E are not reachable.

Reduced
Grammar
G1

$$\left[\begin{array}{l} S \rightarrow Ac \\ A \rightarrow a \end{array} \right]$$

Q.

$$S \rightarrow ABC \mid BaB$$

$$A \rightarrow aA \mid BAC \mid aaa$$

$$B \rightarrow bBb \mid a$$

$$C \rightarrow CA \mid AE$$

$$T = \{a, b\}$$

$$W_1 = \{ \textcircled{A}, B \}$$

$$W_2 = \{ A, B \} \underset{S \rightarrow BaB}{\overset{i}{\longrightarrow}}$$

$$\Rightarrow P = A \rightarrow aA \mid aaa$$

$$B \rightarrow bBb \mid a$$

Reachability

$$\{ S \} \xrightarrow{a} \{ S, B \}$$

$$\xrightarrow{} \{ S, B, a, b \}$$

$$\left(\begin{array}{l} S \rightarrow BaB \\ B \rightarrow bBb \mid a \end{array} \right)$$

Q. $S \rightarrow aAa$

$A \rightarrow Sb \mid bCC \mid DaA$

$C \rightarrow abb \mid DD$

$E \rightarrow aC$

$D \rightarrow aDA$

$$T = \{a, b\}$$

$$W_1 = \{abb, C\}$$

$$W_2 = \{C, A, E\}$$

$$W_3 = \{S, C, A, E\}$$

$S \rightarrow aAa$

$A \rightarrow Sb \mid bCC$

$C \rightarrow abb$

$E \rightarrow aC$

$$\{S\} \rightarrow \{S, aA\} \rightarrow \{S, aA, b, C\}$$

$$\rightarrow \{S, aA, b, C\}$$

E not reachable.

$$P = \left. \begin{array}{l} S \rightarrow aAa \\ A \rightarrow Sb \mid bCC \\ C \rightarrow abb \end{array} \right]$$

Elimination of unit Productions

$S \rightarrow A, A \rightarrow B, B \rightarrow C, C \rightarrow a$

$$L(G) = \{a\}$$

- ①. Not increasing the length
- ②. Not introducing the terminal symbol to the length.

RULES: To remove $A \rightarrow B$, add

- ①. Production, $A \rightarrow X + P$ where, $B \rightarrow X$ occurs in the grammar [not ϵ] \cup $\{\epsilon\}$.
- ②. Delete $A \rightarrow B$ from G .
- ③. Repeat step 2 until all unit productions are removed.

$$\begin{array}{l}
 \text{① } S \rightarrow XY \\
 \quad X \rightarrow a \\
 \quad Y \rightarrow Z \mid b \\
 \quad Z \rightarrow M \\
 \quad M \rightarrow N \\
 \quad N \rightarrow a
 \end{array}
 \qquad
 \begin{array}{l}
 M \rightarrow a \\
 S \rightarrow XY \\
 X \rightarrow a \\
 Y \rightarrow M \mid b
 \end{array}
 \qquad
 \boxed{
 \begin{array}{l}
 S \rightarrow XY \\
 X \rightarrow a \\
 Y \rightarrow a \mid b
 \end{array}
 }$$

ELIMINATION OF ϵ -PRODUCTIONS.

- ①. find all null productions [$A \rightarrow \epsilon$]
- ②. find all nullable variables, which are moving towards ϵ i.e. directly or by applying various derivations i.e.

$$\begin{array}{l} X \xrightarrow{*} \epsilon \\ \text{or, } X \Rightarrow \cdot \cup \cdot \Rightarrow \cdot \Rightarrow \epsilon \end{array}$$

We can't remove all ϵ moves.

Eg. if, starting condition includes ϵ i.e.
language accepting ϵ , we can't remove it
but it should
not come on RHS.

$$S \rightarrow aSb \mid aAb$$

$$A \rightarrow \epsilon$$

$$\text{Nullable set} = \Sigma A \Sigma$$

$$S \rightarrow aSb \mid aAb \mid ab$$

↑ A has been removed.

to remove $A \rightarrow \epsilon$.

∴ A is not reachable.

$$\boxed{S \rightarrow aSb \mid ab}$$

Q. $S \rightarrow A b a C$

$A \rightarrow B C$

$B \rightarrow b | c$

$C \rightarrow D | E$

$D \rightarrow d$

Nullable set = {A, B, C}

$S \rightarrow A b a C | b a C | A b a | b a$

$D \rightarrow d \quad , A \rightarrow B C | B | c$

$C \rightarrow D$

$\boxed{B \rightarrow b} \checkmark$

$S \rightarrow A b a d | b a d | A b a | b a$

$A \rightarrow$

Q. $S \rightarrow A \sqcup A | a B | b$

$A \rightarrow B$

$B \rightarrow b | e$

Nullable = {A, B, D}

$S \rightarrow A \sqcup A | A \sqcup | \sqcup A | \sqcup | a B | a | b$

$A \rightarrow B$

$B \rightarrow b$

↓ now can be reduced.

Normal forms Of CFG

- ①. Chomsky Normal form (CNF)
- ②. Greibach Normal form (GNF)

CNF

Length of RHS and nature of symbols on the RHS.

A CFG is in chomsky Normal form if every production is of the form,

$A \rightarrow a$ or $A \rightarrow Bc$ and $S \rightarrow e$ if, 'e' $\in L(G)$. When e is in $L(G)$, we

assume that e does not appear on the RHS of any production.

Here, $(A, B, C) \in V$ & $a \in T$.

Eg. ①. $S \rightarrow ABc$

$A \rightarrow a$

$B \rightarrow b$

③. $S \rightarrow A \underline{S} | A \underline{A} S$

x Not chomsky

$A \rightarrow \underline{A} a a$ ②

if, $e \notin L(G)$ then for CNF all functions
one of

$A \rightarrow a$ or $A \rightarrow Bc$

How To Convert CFG to CNF

Page No. _____
Date _____

- (1) If start symbol occurs on RHS of some production, create a new start symbol S' and a new production

$S' \rightarrow S$ (unit production so that we can remove all null productions)

- (2) Remove all null productions [after that, we'll never

use them for this grammar]

- (3) Remove all unit productions.

- (4) Replace each production $A \rightarrow B_1, \dots, B_n$ (where, $n > 1$) with $A \rightarrow B_1 C, \dots, B_n C$. In this, $C \rightarrow B_2 - \dots - B_n$. Repeat this step for all productions having more than 2 symbols on RHS.

- (5) If RHS of any production is in the form $A \rightarrow aB$, where a is the terminal of $A, B \in V$, then the production is replaced by,

$$A \rightarrow X B$$

$$X \rightarrow a$$

Repeat this step for every production which is in the form,

$$A \rightarrow aB.$$

Q. $S \rightarrow aSb$
 $S \rightarrow ab$

$$\begin{aligned} A &\rightarrow a \\ B &\rightarrow b \\ S &\rightarrow A \cup B \\ c &\rightarrow AB \end{aligned}$$

$S \rightarrow AX \mid AB$ CNF
 $X \rightarrow CB$
 $A \rightarrow a$
 $B \rightarrow b$

Q. $S' \rightarrow S$

$S \rightarrow aSb$
 $S \rightarrow ab$

$$S' \rightarrow aSb \mid ab$$

$$S \rightarrow aSb \mid ab$$

$A \rightarrow a$
 $B \rightarrow b$

$$S' \rightarrow ASB \mid AB$$

$$S \rightarrow ASB \mid AB$$

$S' \rightarrow AX \mid AB$
 $X \rightarrow SB$
 $S \rightarrow AX \mid AB$

$$A \rightarrow a$$

$$B \rightarrow b$$

Page No. _____
Date _____

$$Q. S \rightarrow A \sqcup A \sqcup AB$$

$$A \rightarrow B \sqcup$$

$$B \rightarrow b \sqcup c$$

$$\{B\}$$

A

$$S \rightarrow A \sqcup A \sqcup SA \sqcup A \sqcup AB \sqcup A \sqcup S$$

$$A \rightarrow B \sqcup S$$

$$B \rightarrow b$$

$$S \rightarrow A \sqcup A \sqcup SA \sqcup A \sqcup AB \sqcup A \sqcup S$$

$$A \rightarrow b \sqcup S, B \rightarrow b$$

$$S \rightarrow A \sqcup A \sqcup SA \sqcup A \sqcup AB \sqcup A \sqcup S$$

$$A \rightarrow b$$

$$A \rightarrow S, B \rightarrow b$$

$$S \rightarrow A \sqcup A \sqcup SA \sqcup A \sqcup AB \sqcup A \sqcup S$$

$$A \rightarrow A \sqcup A \sqcup SA \sqcup A \sqcup AB \sqcup A \sqcup S$$

$$A \rightarrow b, B \rightarrow b$$

$$S \rightarrow AX \sqcup SA \sqcup A \sqcup YB \sqcup a$$

$$A \rightarrow AX \sqcup SA \sqcup A \sqcup YB \sqcup a$$

$$A \rightarrow b$$

$$X \rightarrow SA$$

$$Y \rightarrow a$$

$$B \rightarrow b$$

GREIBACH NORMAL FORM

A CFG is in GNF if every production is of the form,

$$A \rightarrow \alpha,$$

where, $\alpha \in V^*$ and $\alpha \in \Sigma$, α may be ϵ , and $S \rightarrow \epsilon$ is in GNF if $\epsilon \in L(G)$ i.e. ϵ is in $L(G)$

$$S \rightarrow aAB \mid \epsilon$$

$$A \rightarrow bc$$

$$B \rightarrow b, C \rightarrow c$$

Context free to GNF :-

- If start symbol S occurs on some RHS create a new start symbol S' and add a new production.

$$S' \rightarrow S \quad (\text{in } \epsilon \text{ case})$$

- Remove null productions.

- Remove unit productions.

- Remove all direct & indirect left recursion.

- Do proper substitutions of productions to convert it into the proper form of GNF.

Q. $S \rightarrow AB$
 $A \rightarrow \alpha A | bB | b$

$B \rightarrow b$

$S \rightarrow \alpha AB$
 $S \rightarrow bBB$
 $S \rightarrow bB$

$A \rightarrow \alpha A | bB | b$
 $B \rightarrow b$

②.

$A \rightarrow a$
 $B \rightarrow b$

$S \rightarrow aBb | ab$

$A_3 \rightarrow A_2 A_3 A_2 | b$

$\checkmark A_3 \rightarrow A_3 A_1 A_3 A_2 | b | a A_3 A_2$ (Substitute.
A₂ complete)

* left recursion

formula : to convert left recursion to
right recursion.

$\text{if } A \rightarrow A\alpha | \beta \quad A \xrightarrow{\sim} A \text{ right recursion.}$
then convert it to
(is in UNF)

③. $S \rightarrow AB$
 $A \rightarrow B\alpha | \beta$ CNF form

$B \rightarrow aB | b$

~~INF~~
~~CNF~~
~~for CINF~~

convert to CINF

Step 1: Rename the variables in
order A₁, A₂, A₃ starting
symbol is the first.

$A' \rightarrow A_1 A_3 A_2 A'_1 | A_1 A_3 A_2$
 $A_3 \rightarrow \alpha A_3 A_2 A'_1 | b A'_1 | a A_3 A_2 | b$

$A_1, A_2 \notin A'$ are not in GFL

Page No. _____
Date _____

No. 21

$$\nwarrow A_2 \rightarrow aA_3A_2A' A_1 | bA'A_1 | aA_3A_2A_1 | bA_1 | a$$

$$A_1 \rightarrow aA_3A_2A' A_1 A_3 | bA'A_1 A_3 | aA_3A_2A_1 A_3 | bA_1 A_3 | aA_3$$

$$A_1 \rightarrow aA_3A_2A' A_1 A_3 A_3 A_2 A' | bA'A_1 A_3 A_3 A_2 A' | aA_3A_2A_1 A_3 A_2 A' | bA_1 A_3 A_3 A_2 A'$$

(some) with
last A_3A_2 .

Q.

$$S \rightarrow AA | a$$

$$A \rightarrow S | b$$

$$A_1 \rightarrow A_2 A_2 | a$$

$$A_2 \rightarrow A_1 A_1 | b$$

α β

$$A' \rightarrow A_1 A'_1 | A_1$$

$$A_1 \rightarrow bA'_1 | b$$

$$A_1 \rightarrow A_2 A_2 | a$$

$$A_1 \rightarrow bA'_1 | b$$

$$A'_1 \rightarrow A_1 A'_1 | A_1$$

Page No. _____
Date _____

Page No. _____
Date _____

$$A_1 \rightarrow A_2 A_2 | a$$

$$A_2 \rightarrow A_1 A_1 | b$$

$$A_2 \rightarrow \underbrace{A_2 A_2}_\alpha A_1 | b | \underbrace{a A_1}_\beta$$

$$A' \rightarrow A_2 A_1 A' | A_2 A_1$$

$$\nwarrow A_2 \rightarrow bA'_1 | aA_1 A'_1 | b | aA_1$$

∴

$$\nwarrow A_1 \rightarrow bA'_1 A_2 | aA_1 A'_1 A_2 | b A_2 | a A_1 A_2 | a$$

$$\nwarrow A' \rightarrow bA'_1 A_1 A' | aA_1 A'_1 A_1 A' | b A_1 A'_1 | a A_1 A_1 A'$$

$b A'_1 A_1 | a A_1 A'_1 A_1 | b A_1 | a A_1 A_1$

PUMPING LEMMA for CFL (context free language)

$$A_1 \rightarrow A_2 A_2 | a$$

$$A_2 \rightarrow A_1 A_1 | b \quad \checkmark \quad \checkmark$$

$$A_2 \rightarrow \underbrace{A_2 A_2}_{\alpha} \underbrace{A_1}_{\beta} | b | a A_1$$

$$A' \rightarrow A_2 A_1 A' | A_2 A_1$$

$$\rightsquigarrow A_2 \rightarrow b A' | a A_1 A' | b | a A_1$$

$$\rightsquigarrow A_1 \rightarrow b A' A_2 | a A_1 A' A_2 | b A_2 | a A_1 A_2 | a$$

$$\rightsquigarrow A' \rightarrow b A' A_1 A' | a A_1 A' A_1 A' | b A_1 A' | a A_1 A_1 A' | \\ b A' A_1 | a A_1 A' A_1 | b A_1 | a A_1 A_1$$

PUMPING LEMMA FOR CFL (context free language)

Let 'L' be a context-free language. Then
 i.e. can find a natural number 'n' such
 that,

- (1). Every $z \in L$ with $|z| \geq n$ can be written
 as $uvwxy$ for some string u, v, w, x, y

(2). $|vwx| \geq 1$

(3). $|vwx| \leq n$

(4) If $uv^kwy^k \in L$ for all $k \geq 0$
 then, n is called the pumping length.

* Proof (by yourself)

Ex {
 Then, a^n is not a fn.}

Ex. Show that, $L = \{a^p\}^p$ is prime? is
 not a CFL.

(1). Let, let, $L = L(G)$ is a context free language.

(2). Let, p be a prime no. greater than n .

Then, $z = a^p \in L$ & hence ($p > n$) or,
 $|z| > n$,

we write $z = uvwxy$

(3). By pumping lemma,

$z^k = uv^k w^k y^k$ for, $k=0$ as, $k \geq 0$,
 we get,

$z^0 = uv^0 w^0 y^0 = uwy \in L$ so,

$|uwy|$ is a prime no. say (q)

Let $|uwy| = r$ so,

$$z^q = uv^q w v^q y$$

$$\Rightarrow |z^q| = |uwyl| + q \times |v^n| \\ = q + q \times (x) = q(x+1)$$

\because it is prime.

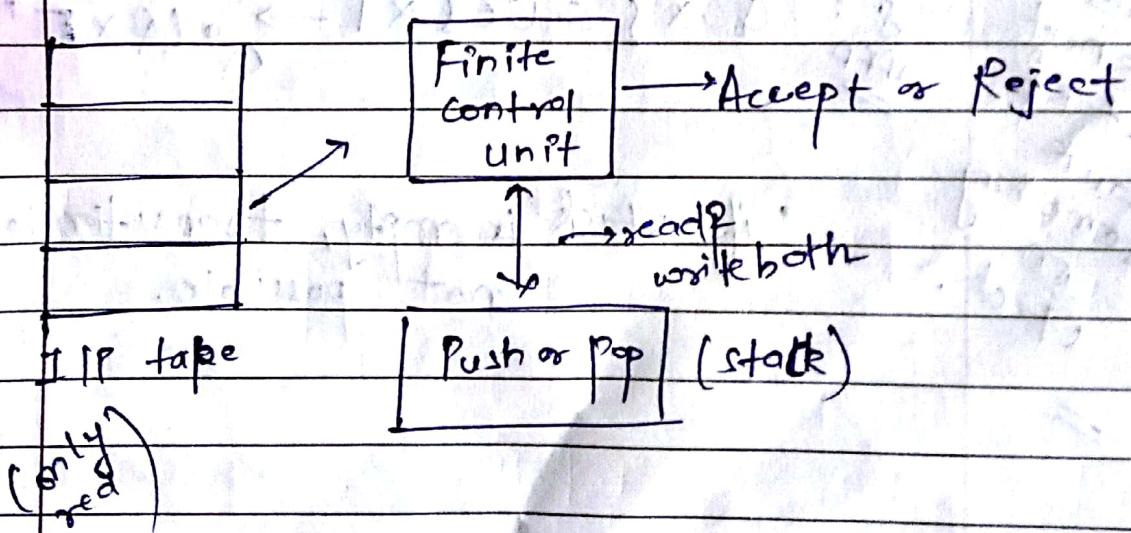
$\therefore q(x+1)$ is not a prime.

Hence failed.

Push Down Automata (PDA)

PDA = "FA" + Stack

- ①. An IIP tape (Read only)
- ②. A control unit
- ③. A stack with infinite size.



A PDA is a 7-tuple $(Q, \Sigma, \delta, q_0, z_0, F, \Gamma)$
where,

$Q \rightarrow$ states

$\Sigma \rightarrow$ input symbol

$\delta \rightarrow$ Transition function

$q_0 \rightarrow$ initial state

$z_0 \rightarrow$ called initial symbol denoted by z_0

$F \rightarrow$ Set of final states (FCQ)

$\Gamma \rightarrow$ Stack alphabet

$\{F \text{ is a subset}$
 $\text{of } Q\}$

PDA

Deterministic

Non-Deterministic

DFA

read stack (mandatory)

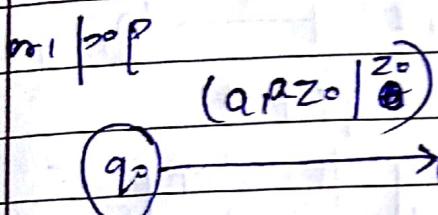
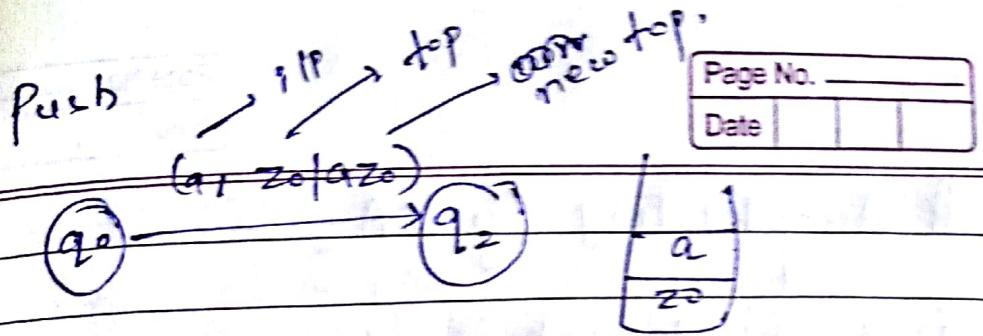
$$\delta : Q \times \Sigma \times \Gamma \times F \rightarrow (Q \times \Gamma^*)$$

NDFA

$$\delta : Q \times \Sigma \times \Gamma \times F \rightarrow Q \cdot (Q \times \Gamma^*)$$

leads
to next
state
write
something
in
stack.

- if stack is empty, transition is not possible.



Instantaneous Description (ID)

{ q, w, z } where,

$q \rightarrow$ is the state

$w \rightarrow$ unconsumed i/p

w = what is left now,
 z = stack contents.

$$\text{Ex. } (q_1, aba, a z_0) \Rightarrow (q_2, ba, z_0) \\ \Rightarrow (q_3, a, b z_0)$$

Transitive Notation \Rightarrow (we can also replace instead)

$(p, aw, Tb) \vdash (q, w, qb)$ of push or pop.

We can use * to skip states in b/w.

$$(q_1, aba, a z_0) \vdash (q_2, ba, z_0)$$

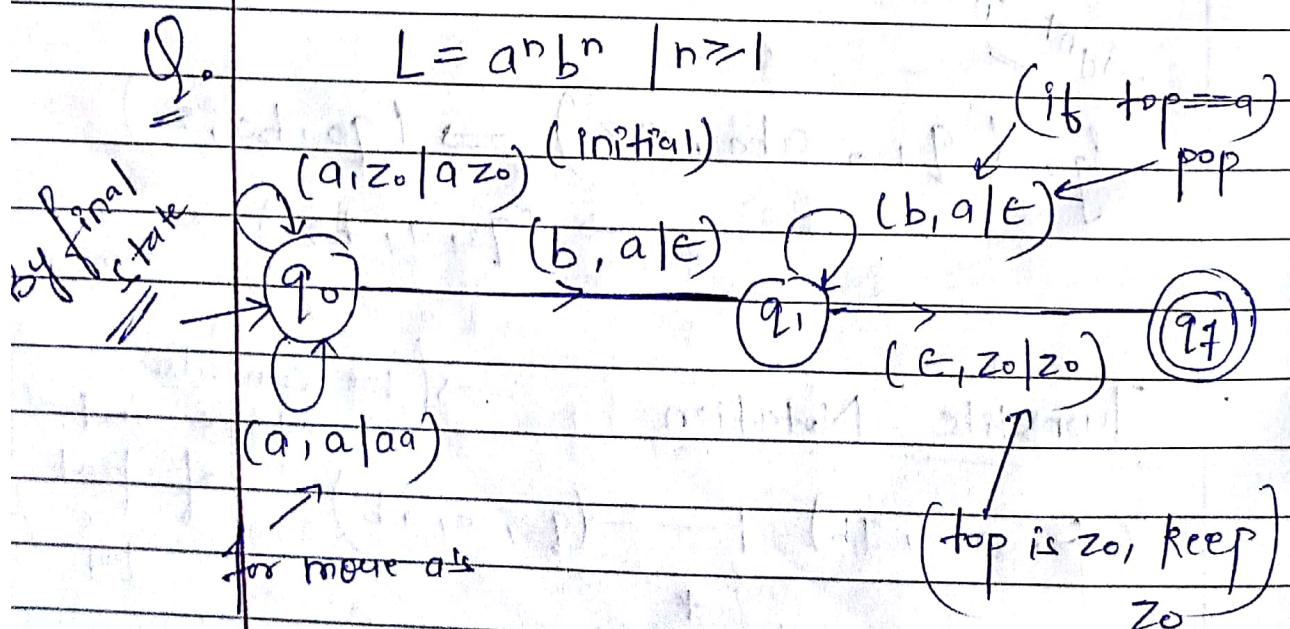
Acceptability by PDA

① By Final State :-

$$L(PDA) = \left\{ w \mid (q_0, w, I) \xrightarrow{*} (q_f, \epsilon, z), \begin{array}{l} q_f \in F \\ \text{for any i/p string } w. \\ \Rightarrow (q_f \text{ is final state}) \end{array} \right\}$$

② Empty stack :-

$$L(PDA) = \left\{ w \mid (q_0, w, I) \xrightarrow{*} (q_f, \epsilon, \epsilon), \begin{array}{l} q_f \in F \\ (\text{Here, } q_0 \text{ may not be a final state.}) \end{array} \right\}$$



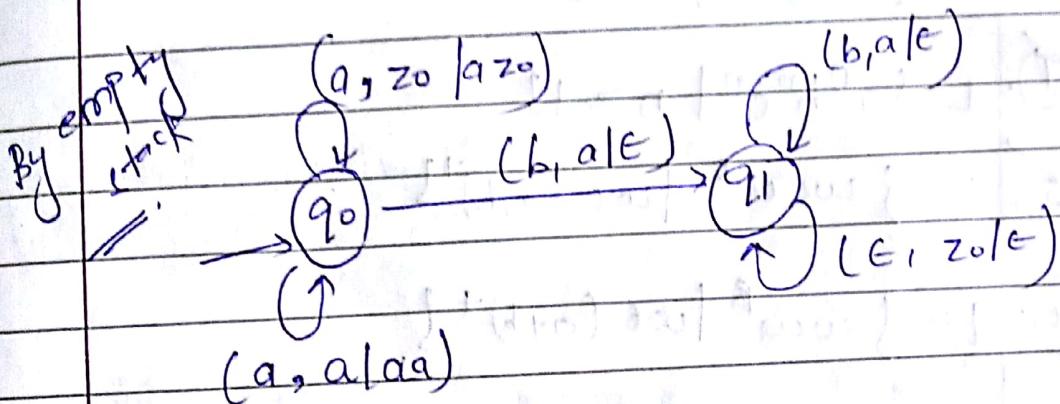
$$\delta(q_0, a, z_0) = (q_0, a, a z_0)$$

$$\delta(q_0, a, a) = (q_0, aa)$$

$$\delta(q_0, b, a) = (q_1, \epsilon)$$

$$\delta(q_1, b, a) = (q_1, \epsilon)$$

$$\delta(q_1, \epsilon, z_0) = (q_1, z_0)$$



* By default N DFA

✓ ①. $L = \{ \omega \mid n_a(\omega) = n_b(\omega) \}$

✓ ②. $L = \{ a^n b^m c^m \mid n, m \geq 1 \}$

✓ ③. $L = \{ a^n b^m c^n \mid n, m \geq 1 \}$

✓ ④. $L = \{ a^{m+n} b^m c^n \mid m, n \geq 1 \}$

✓ ⑤. $L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$

✓ ⑥. $L = \{ a^n b^{2n} \mid n \geq 1 \}$

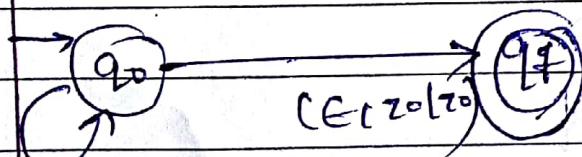
✓ ⑦. $L = \{ a^n b^n c^n \mid n \geq 1 \}$

✓ ⑧. $L = \{ \omega \omega^R \mid \omega \in (a, b)^+ \}$

✓ ⑨. $L = \{ \omega \omega^R \mid \omega \in (a+b)^+ \}$

✓ ⑩. $L = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^{2n} \mid n \geq 1 \}$

⑪.



(b, a | e)

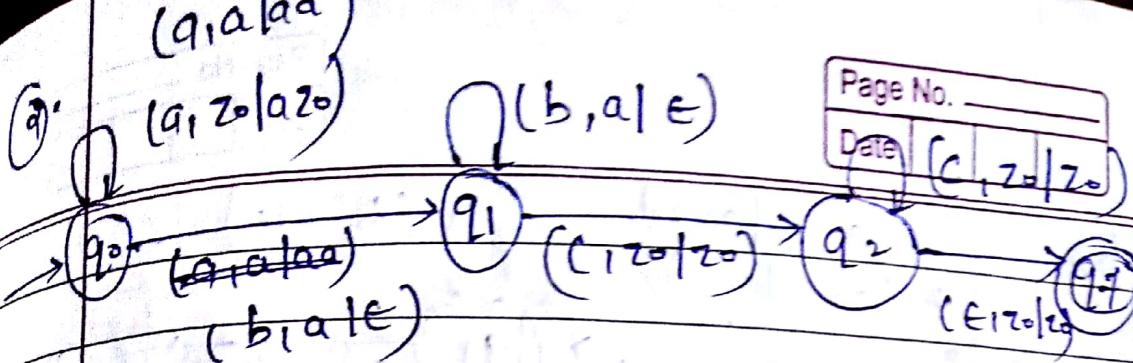
(a, b | e)

(b, z0 | b z0)

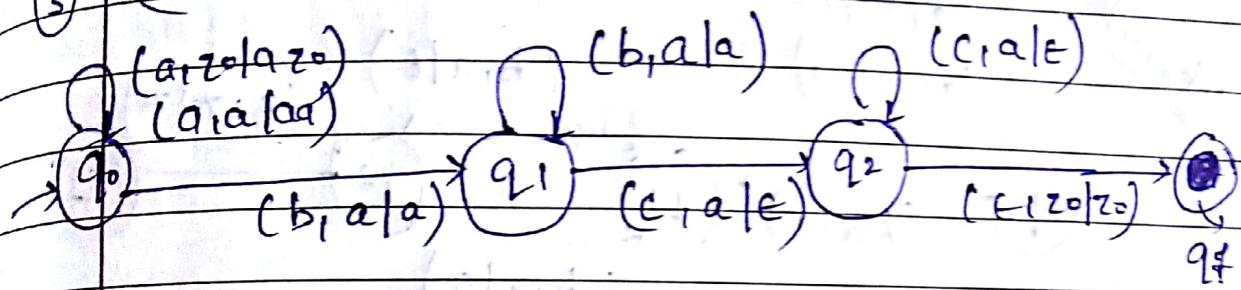
(b, b | b b)

(a, z0 | a z0)

(a, a | a a)

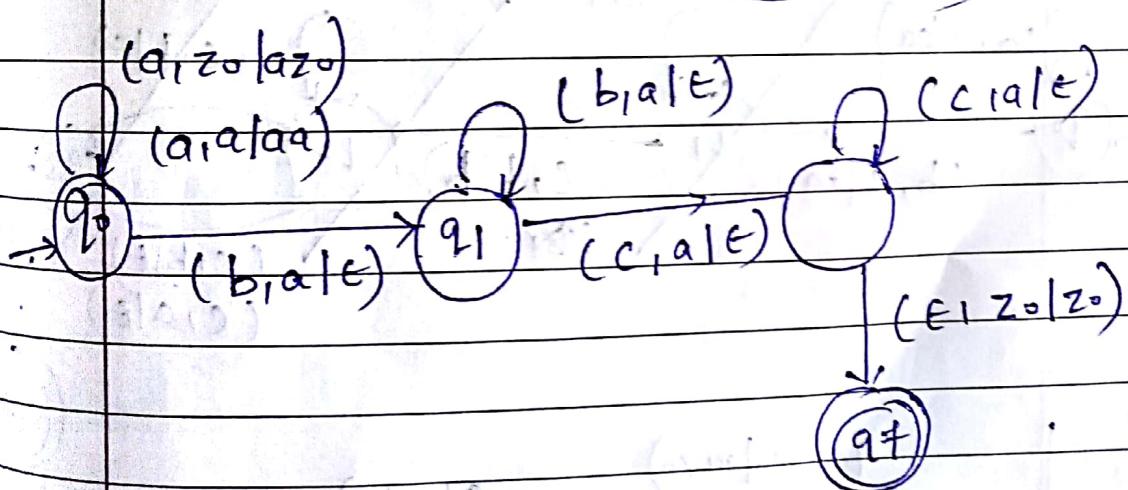


③. $(a^n b^m c^n)$

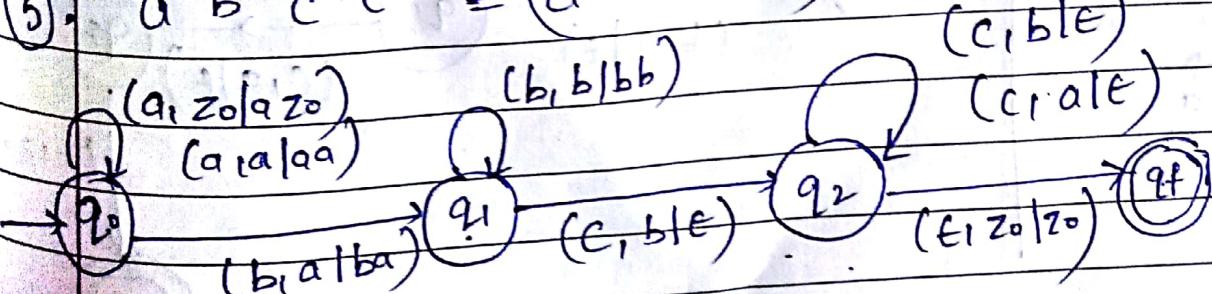


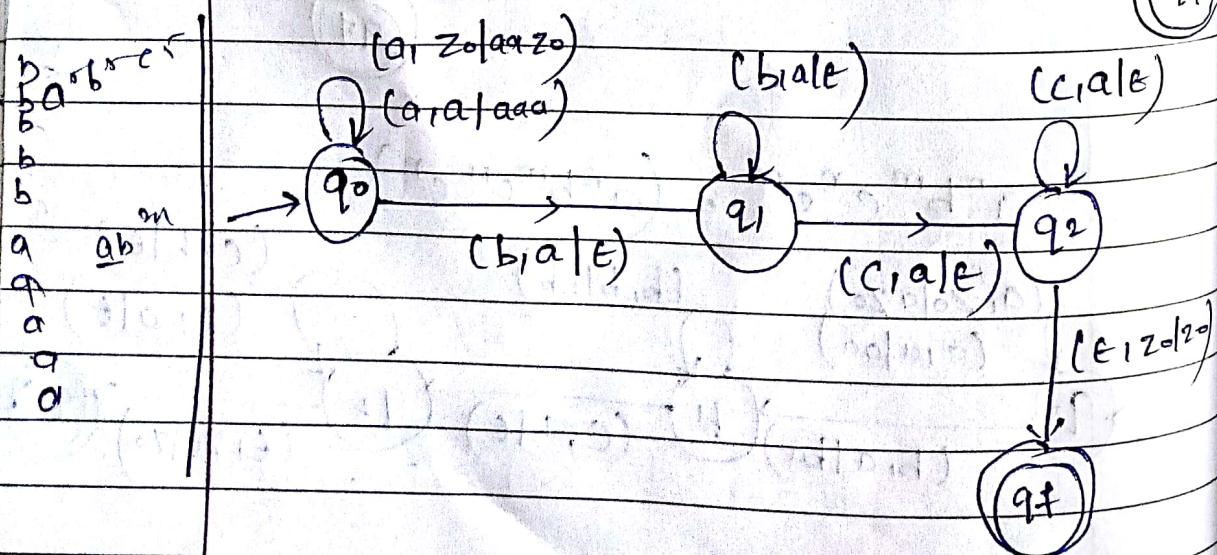
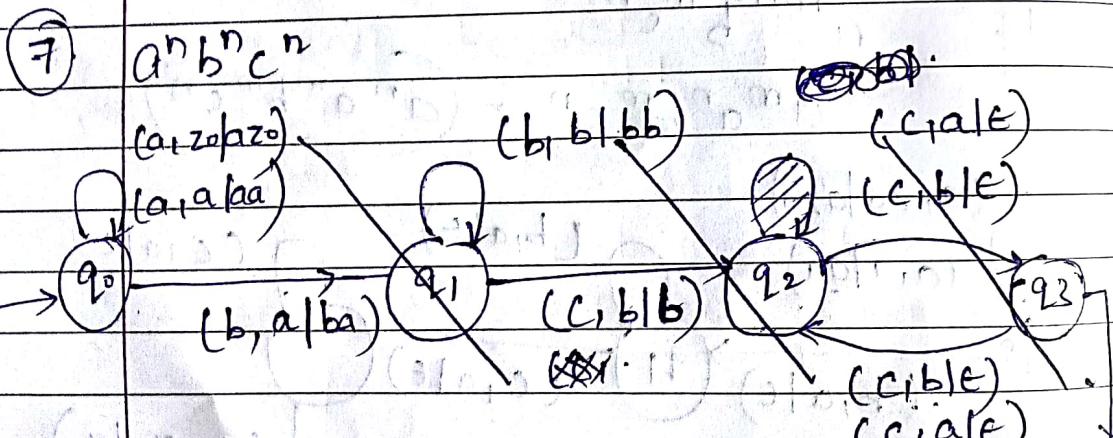
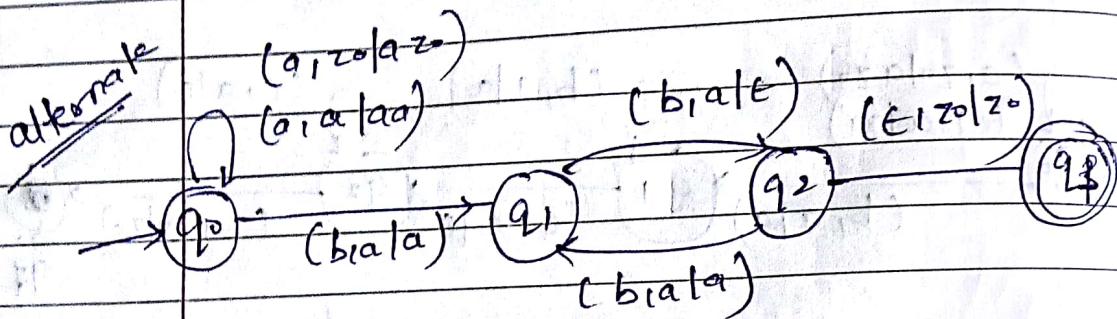
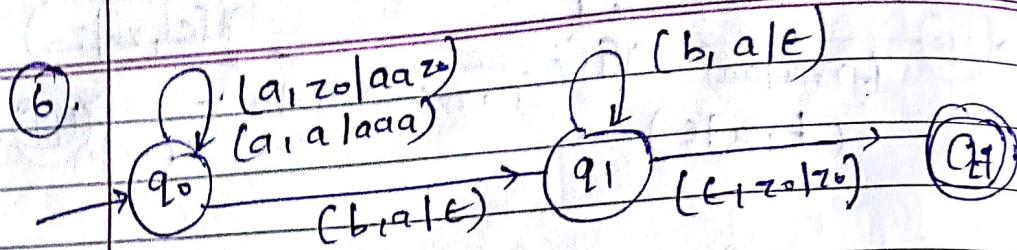
④. $a^{m+n} b^m c^n$

$$a^m a^n b^m c^n = (a^n a^m b^m c^n)$$



⑤. $a^n b^m c^n c^m = (a^n b^m c^m c^n)$

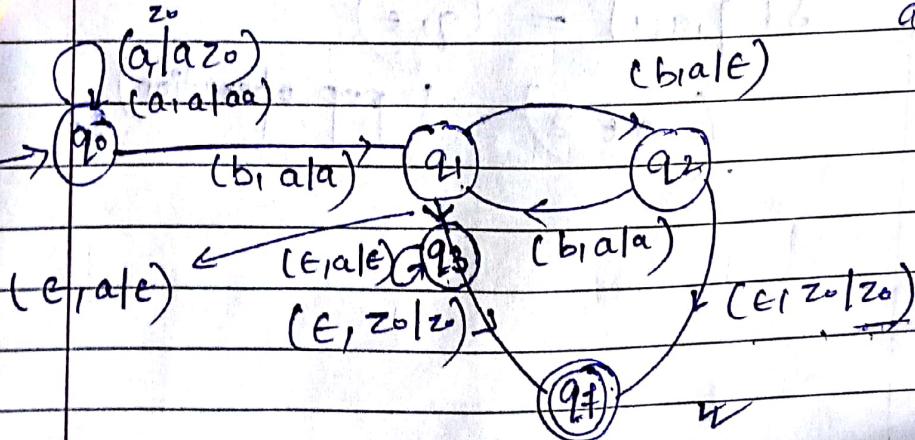




(8)

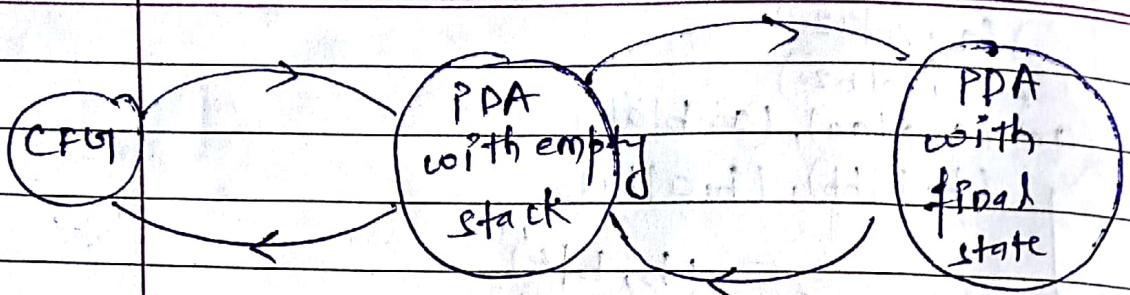
 $(wCwR)$ $(a, z|az|z)$ $(b, z|b|bz)$ $(a, a|aa), (a, bla|b)$ $(b, b|bb), (b, alba)$ q₀ $(c, bl|b)$
 (c, ala) q₁ $(b, b|e)$
 (a, ale) $(c, z|z)$ q_f

(9)

 $(wCwR)$ $(e, z|z)$ $a^n b^n$ (10). $[a^n b^n]$ $a^n b^{2n}$ $aabb$ 

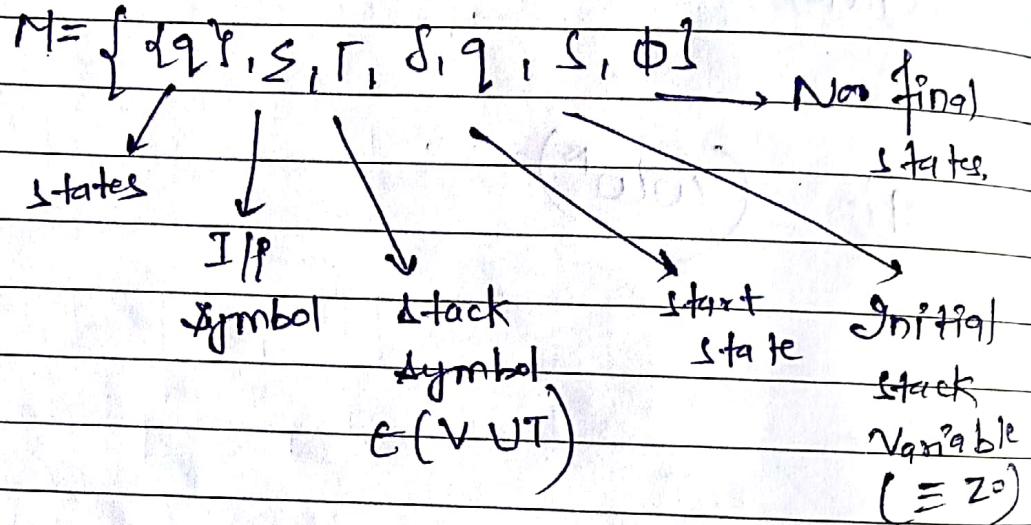
CFG to PDA Conversion

Page No.	
Date	



A CFG can be converted to a PDA

* since converting
converting
to a PDA
empty stack
if state
is equal



$$①. \quad \delta(q_1, \epsilon, A) \xrightarrow{\text{Top of the stack}} (q_1, \alpha) \quad (\text{RHS of transition})$$

if there is α ,
LHS production $A \rightarrow \alpha$ needs top
of the stack

$$②. \quad \delta(q_1, a, A) = (q_1, e) \quad (a \in \Sigma) \quad (\text{pop operation})$$

$s \rightarrow asb$
 $s \rightarrow a$

$$T = \{ s(a,b) \}$$

$$\textcircled{1}. \quad \delta(q_1, e_1, s) = (q_1, asb)$$

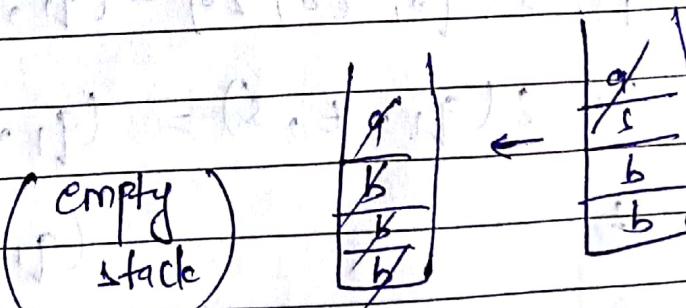
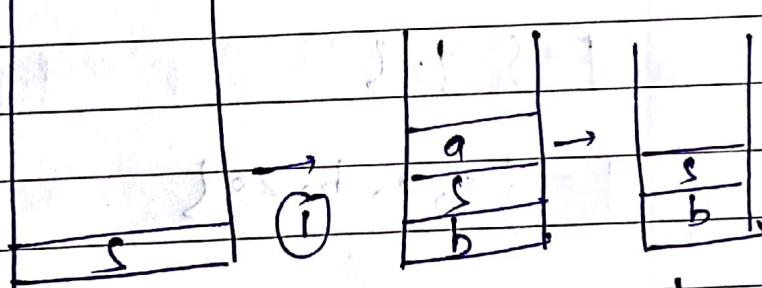
$$\textcircled{2}. \quad \delta(q_1, e_1, s) = (q_1, ab) \quad \leftarrow \begin{array}{l} \text{write for all} \\ \text{variables \&} \\ \text{terminals.} \end{array}$$

$$\textcircled{3}. \quad (q_1, a, a) = (q_1, e)$$

$$\textcircled{4}. \quad \delta(q_1, b, b) = (q_1, E)$$

(aaa bbb)

check for it.



after consuming
whole string.

Acceptance by final state

$$\textcircled{1}. \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2}. \delta(q_1, \epsilon, A) = (q_1, q') \quad \text{for all } A \rightarrow q$$

\textcircled{3}. for every $a \in T$, we define,

$$\delta(q_1, a, a) = (q_1, \epsilon)$$

$$\textcircled{4}. \delta(q_1, \epsilon, z_0) = (q_2, z_0), q_2 \in F$$

$\Sigma \rightarrow a|aa|b|ss|ssb|sbs$

$$Q = \{q_0, q_1, q_2\}$$

$$F = \{q_2\}$$

$$T = \{\Sigma, a, b, z_0\}$$

$$\textcircled{1}. \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2}. \delta(q_1, \epsilon, \Sigma) = (q_1, q')$$

$$(q_1, q_2)$$

$$(q_1, bss)$$

$$(q_1, ssb)$$

$$(q_1, sbs)$$

$$s(q_1, a, q) = (q_1, t)$$

$$s(q_1, b, b) = (q_1, t)$$

$$s(q_1, e, z^0) = (q_2, z^0)$$

$q_2 \in F$

Check (abbaa)

$$(q_0, abbaa, z^0)$$

T

$$(q_1, abbaa, s, z^0)$$

T

$$(q_2, z^0)$$

$$(q_1, abbaa, sb, s, z^0)$$

T

$$(q_1, abbaa, ab, s, z^0)$$

T

$$(q_1, abbaa, ab, s, z^0)$$

T

$$(q_1, baa, s, z^0)$$

T

$$(q_1, baa, b, s, z^0)$$

T

$$(q_1, aa, ss, z^0)$$

T

$$(q_1, aa, ss, z^0)$$

T

$$(q_1, a, s, z^0)$$

T

$$(q_1, a, a, z^0)$$

T

$$(q_1, a, e, z^0)$$

CFG in GNF to PDM.

Page No. _____

Date _____

empty stack

$$M = \{ S, T, \Gamma, \delta, q_1, S, \phi \}$$

Here, $T = V$ only - (only variable, no terminal)

and S is on the top of the stack.

①. If $A \rightarrow a\alpha$ is a production, where, $a \in T$ & $\alpha \in V^*$ then,

$$\textcircled{1}: \delta(q_1, a, A) = (q_1, \alpha)$$

Top of
the stack

initially, S is on the top of the stack.

Eg. $S \rightarrow aSS$

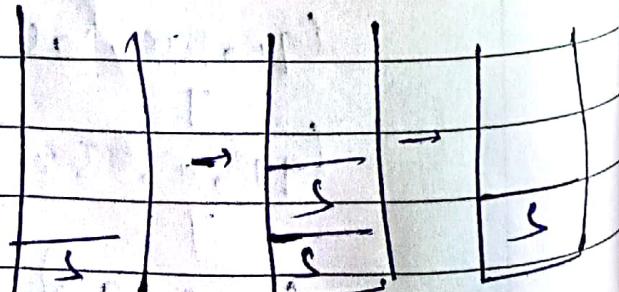
$$S \rightarrow b$$

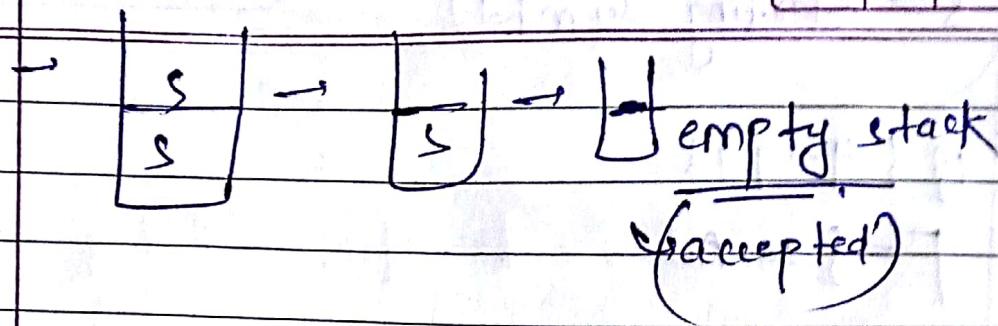
$$\Gamma = \{ S \}$$

$$\textcircled{1}: \delta(q_1, a, S) = (q_1, SS)$$

$$\textcircled{2}: \delta(q_1, b, S) = (q_1, \epsilon)$$

"ab ab"





PDA to CFG conversion

Let $M = (\mathbb{Q}, \Sigma, \Gamma, \delta, q_0, z_0, \phi)$ be a PDA accepting by empty store,

construct a CFG.

$G_1 = (V_1, T, P, S)$ such that

$$L(G) = L(PDA)$$

$$\delta(q_1, \epsilon, z_0) \rightarrow (q_2, z_0)$$

to change final state

to empty stack form.

$$G_2 = (V, T, P, S)$$

$(q_1, \epsilon, \emptyset)$ pushed down automata.
states of

$$V = \Sigma \cup \{ [q_1, A, p] \} \rightarrow \text{Triplet}$$

push down symbol

If, $|\mathbb{Q}| = k$ and $|\Gamma| = m$

then, no. of variables,

$$|V| = k \times k \times m + 1 = \boxed{k^2 m + 1}$$

$T = \text{Terminal symbols}$
 $S = \text{Starting symbol}$

Page No. _____
 Date _____ / _____ / _____

$q_0, Q = \{q_0, q_1, \dots\}$
 $F = \{z_0, A\}$

$S, [q_0, z_0, q_1], [q_0, z_0, q_1], [q_0, A, q_1],$
 $[q_0, A, q_1], \dots$

$S \rightarrow [q_0, z_0, p]$ for each $p \in Q$.
 fixed.

Then, no. of productions will be equal to
 no. of variables.

(1). If, $\delta(q, x, A) = (p, B_1, \dots, B_m)$
 where $n \in \{ \in, U, \Sigma \}$

Then we write a production - should be same.

$[q, A, q_{m+1}] \rightarrow x [p, B_1, q_2] [q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$
 ↑
 any state
 ↓
 any stack

(2). If, $\delta(q, x, A) = (p, \epsilon)$

$[q, A, p] \rightarrow x$

$M = \{ [q_0, q_1, 1011, 1, z_0, x_1, \dots, z_1, \epsilon] \}$

(1). $\delta(q_0, 0, z_0) = (q_0, x_0)$

(2). $\delta(q_0, 0, x) = (q_0, xx)$ Push

(3). $\delta(q_0, 1, x) = (q_1, \epsilon)$

(4). $\delta(q_1, 1, x) = (q_1, \epsilon)$

(5). $\delta(q_1, \epsilon, x) = (q_1, \epsilon)$

(6). $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$S, [q_0, z_0, q_0], [q_0, z_0, q_1]$

$[q_1, z_0, q_0],$

$[q_1, z_0, q_1],$

$[q_0, x, q_0]$

Pop

1

(1). $S \rightarrow [q_0, z_0, q_0]$

(2). $S \rightarrow [q_0, z_0, q_1]$

(3). $[q_0, x, q_1] \rightarrow 1$

(4). $[q_1, x, q_1] \rightarrow 1$

(5). $[q_1, x, q_1] \rightarrow G$

(6). $[q_1, z_0, q_1] \rightarrow \epsilon$

$$\{q_1 \times A\} = \{p \cdot t\}$$

$$[q_1 A p] \rightarrow n$$

Pub for $\delta(q_0, 0, z) = [q_0 \times z]$

$$1) [q_0, z, q_0] \rightarrow 0, [q_0 \times z, q_0] [q_0, z_0, q_0]$$

$$2) " q_0 " q_1 z_1 " q_0$$

$$3) " q_1 " q_2 z_2 " q_1$$

$$4) " q_2 " q_0 z_0 " q_2$$

Pub for $\delta(q_0, 0, x) = (q_0 \times x)$

Number all the triplets and reduce it

$\left. \begin{array}{l} \text{reducibility} \\ \text{derivable} \end{array} \right\} \text{red.}$

$$[q_0, z_0, q_0]$$

$$[q_0, z_0, q_1]$$

$$[q_0, x, q_1]$$

$$[q_1, x, q_1]$$

$$[q_1, z_0, q_1]$$

$$[q_1, x, q_0]$$

Page No. _____
Date _____

$$a_1 = 2 \quad a_2 = 3$$

Page No. _____
Date _____

$$M = m_1 m_2$$

$$M_1 = 2 \cdot 1 / 17 \quad (13) \quad \left\{ \begin{array}{l} m_1 = 17 \\ m_2 = 13 \end{array} \right.$$

$$M_2 = 2 \cdot 1 / 13 = (17)$$

$$\min M_2 = \min (17, 13) =$$

$$(13, 17) \checkmark$$

$$\gcd(17, 13) = \gcd(7, 10)$$

$$\min (13, 17) = (0, 1) \quad ; \quad 13/17 = 0$$

$$\min (13)$$

$$13/17$$

$$1 \quad ?$$

$$2 \quad 45$$

$$3 \quad 7$$

$$4 \quad 11$$

nongun



Trivium M/C

→ T/T Tape

Symbol	
Data	

$$\begin{array}{l} L \rightarrow AB \\ A \rightarrow A1B \\ B \rightarrow 2B \end{array}$$

Page No.	
Date	

Trivium M/C
T/T Tape

$$R/\# Head \quad \delta(q_1, q_2) = (q_{new}, a_{new}, L/R)$$

finite control

0 1 2 3

• Tuple $(\Sigma, \Gamma, \delta, q_0, b, F)$

- ①. Σ is a finite non-empty set of states.
- ②. Γ is a finite non-empty set of tape symbols.
- ③. $b \in \Gamma$ is the blank symbol. Except T/T
- every thing is blank on the tape.
- ④. Σ is a non-empty set of T/T symbols and is q subset of Γ and $b \notin \Sigma$.
- ⑤. $\delta(q, r) = (q', y, D)$ $D = L/R$. \rightarrow Right [left or right]
 \rightarrow left [move either]
- ⑥. $q_0 \in Q$ is the initial state.
- ⑦. $F \subseteq Q$ is the set of final states.