Symbol - Smallest building block eg:(a, b, 0, 1)

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

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

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
eg:- $w = aab$, $w^R = baa$

Language - A language over Σ is a subset of Σ^* (or) collection of strings over Σ

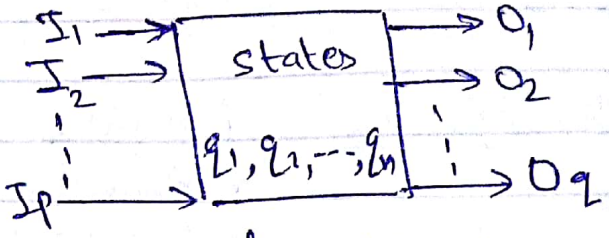
eg:- $L = \{0,1\}^*$: length of string is even
$= \{\epsilon, 00, 01, 10, 11, 0000, 0001, \dots\}$
↳ Infinite language

eg:- $\Sigma = \{a, b\}$, $L = \{a^n b^n : n >= 0\}$

$L = \{\epsilon, ab, aabb, aaabbb, \dots\}$
abb or aab 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

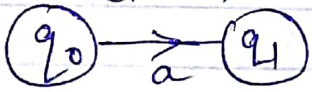


(Model of a discrete automata)

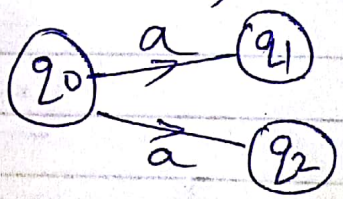
→ Finite Automata : Finite states.

FA

- Deterministic (DFA)



- Non-Deterministic (NFA)



→ DFA (Deterministic Finite Automata)

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

where

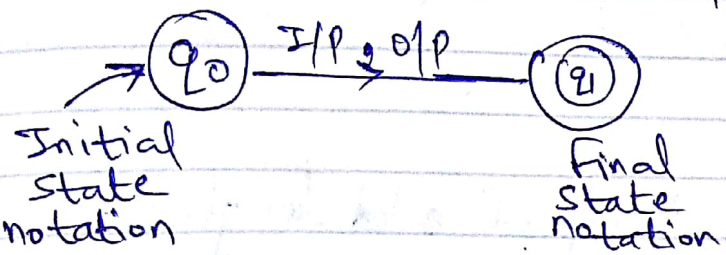
$Q \rightarrow$ finite non-empty set of states
$\Sigma \rightarrow$ I/P alphabet
$\delta \rightarrow$ Transition function
$q_0 \in Q$, initial state
$F \subseteq Q$, set of final states

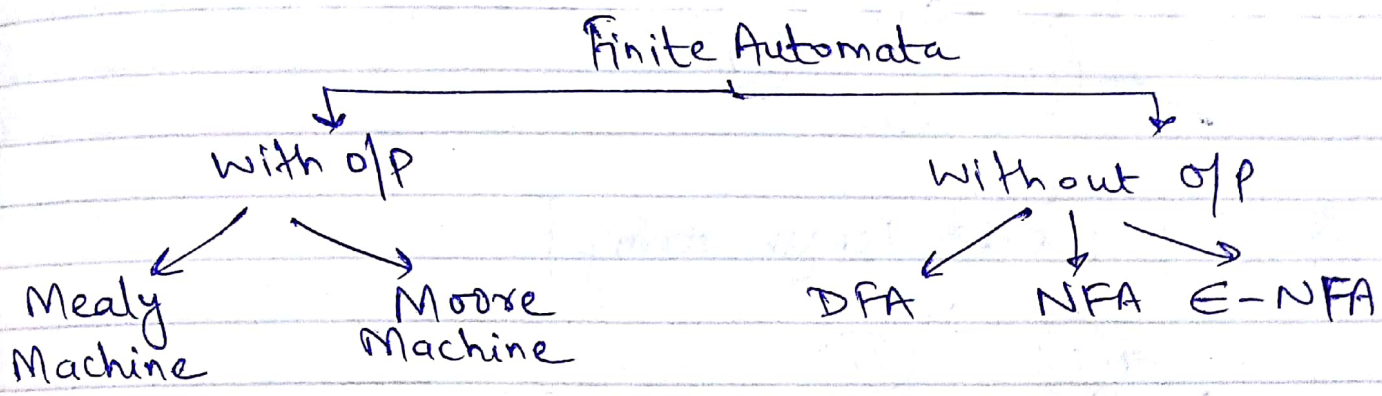
DFA transition function,

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

→ Transition System

It is a finite directed labelled graph

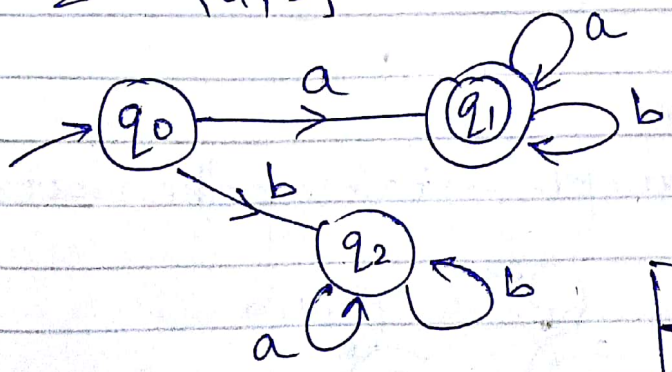




→ DFA

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

Q. L = { Set of all strings that starts with 'a' }
Σ = {a, b}

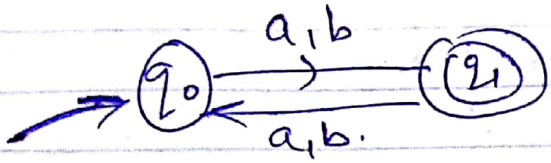


Transition Table

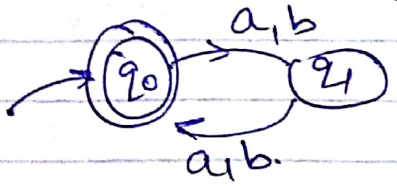
States	a	b
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

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

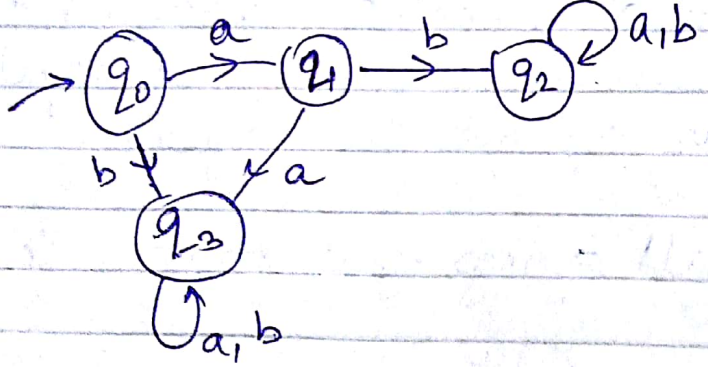
Q. L = { Set of all strings of ~~odd~~ length }, Σ = {a, b}



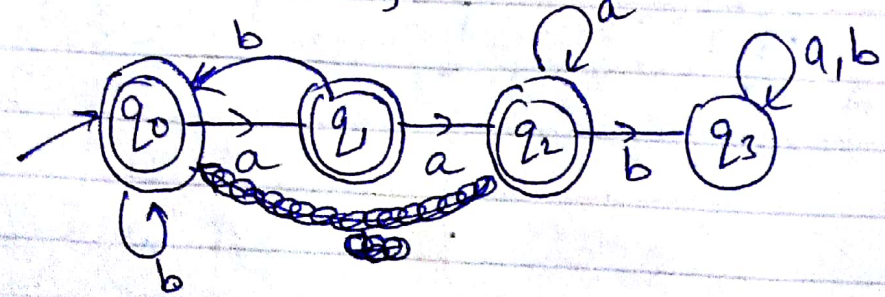
Q. L = { ~~odd~~ even length strings }



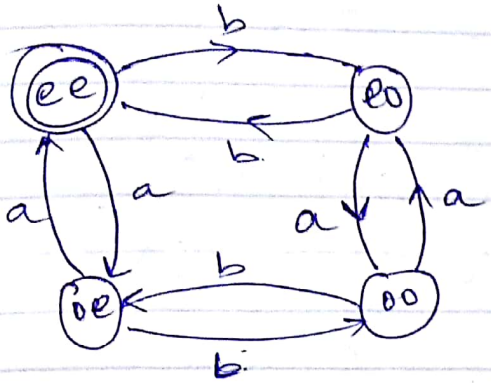
Q. L = { Set of all strings with prefix ab }



Q. L = { Strings except Containing the substring aab }

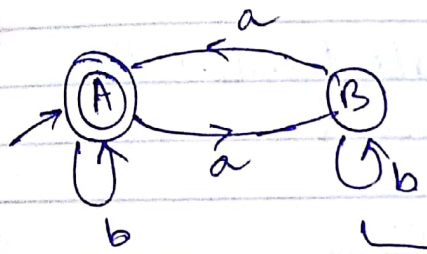


Q. $w = \{a, b\}^*$, $n_a(w) \cong 0 \bmod 2$
$n_b(w) \cong 0 \bmod 2$

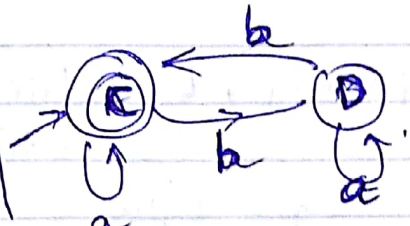


This can also be done using cross product

$n_A(w) \cong 0 \bmod 2$



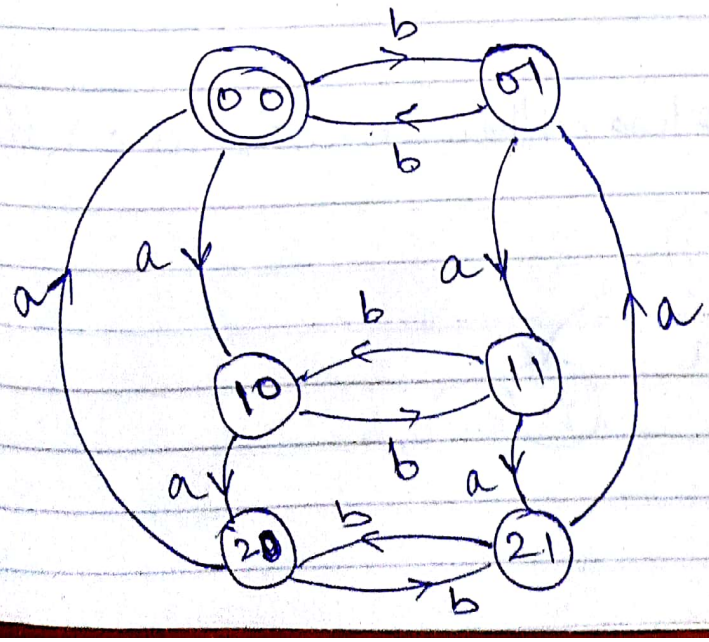
$n_B(w) \cong 0 \bmod 2$



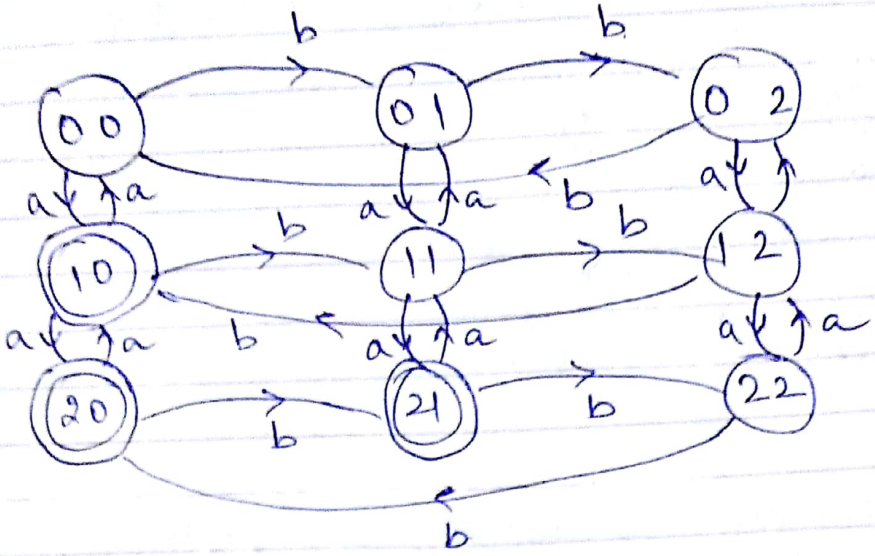
$\{A, B\} \times \{C, D\} = \{AC, AD, BC, BD\}$
ee eo oe oo

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

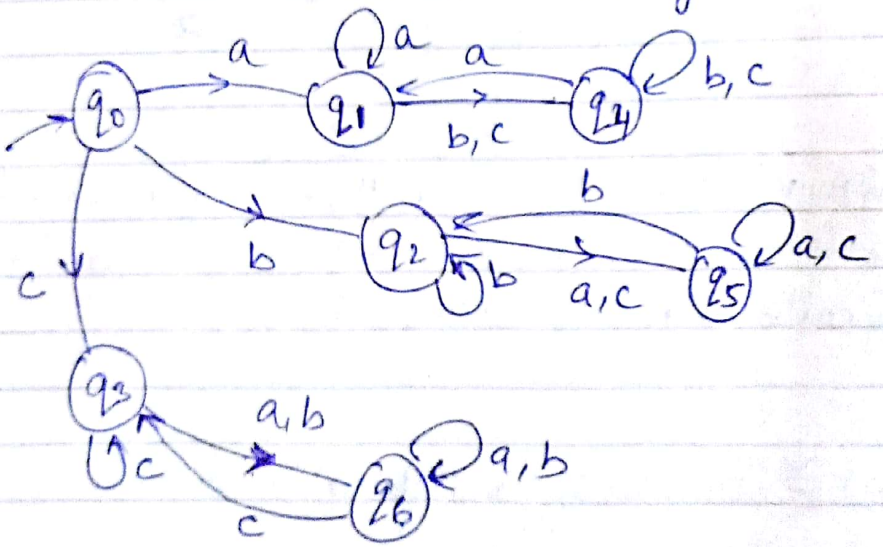
Q. $n_a(w) \cong 0 \bmod 3$, $n_b(w) \cong 0 \bmod 2$



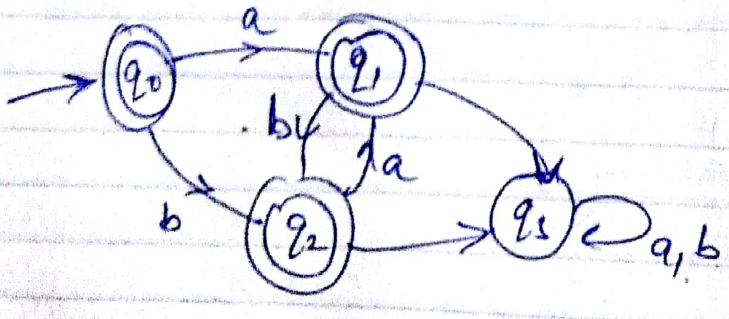
Q. $n_a(w) \bmod 3 > n_b(w) \bmod 3$



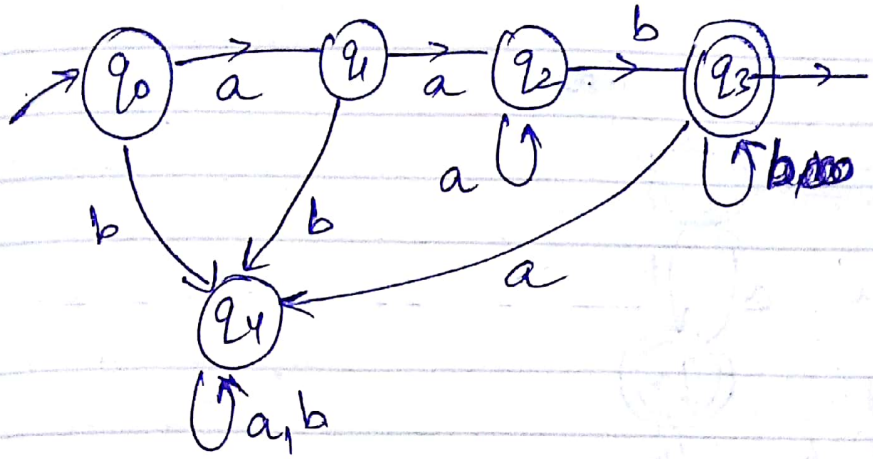
Q. $\Sigma = \{a, b, c\}$, $L = \{$ strings that starts & ends with different symbols $\}$



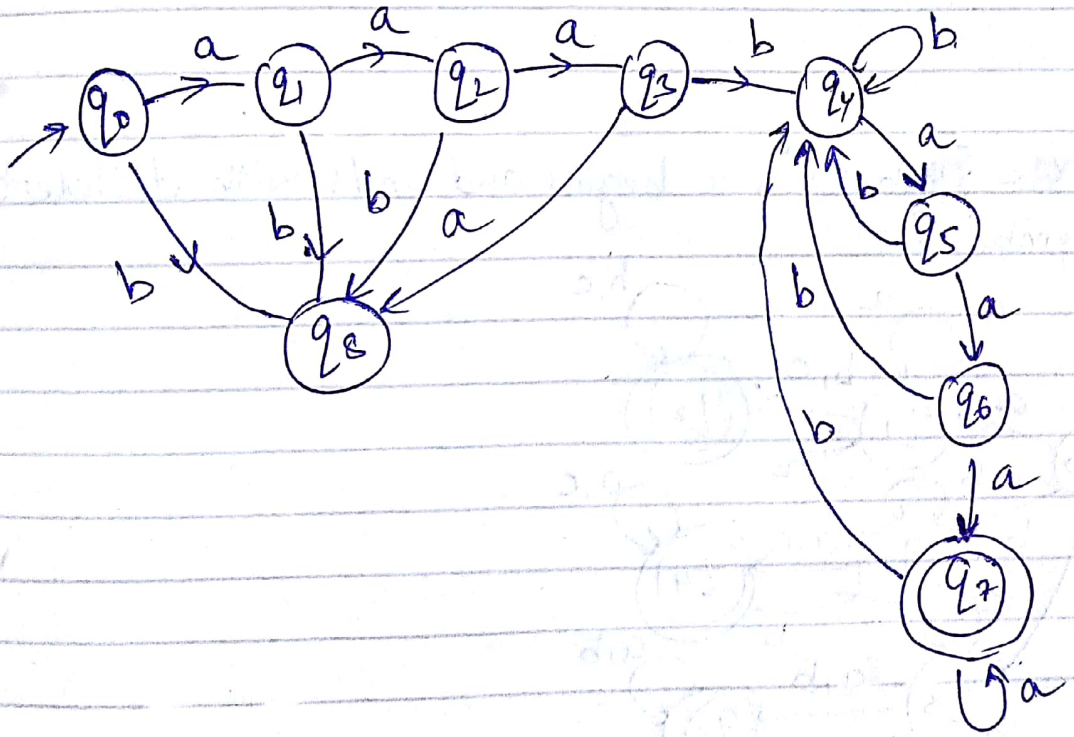
Q. $L = \{x \in (a|b)^* : x$ has neither consecutive a's nor consecutive b's$\}$



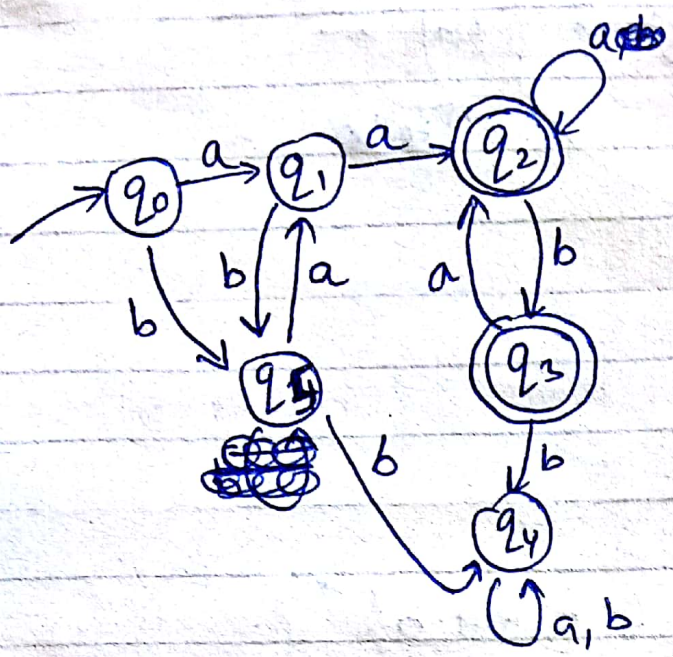
Q. $L = \{a^n b^m : n \geq 2, m \geq 1\}$



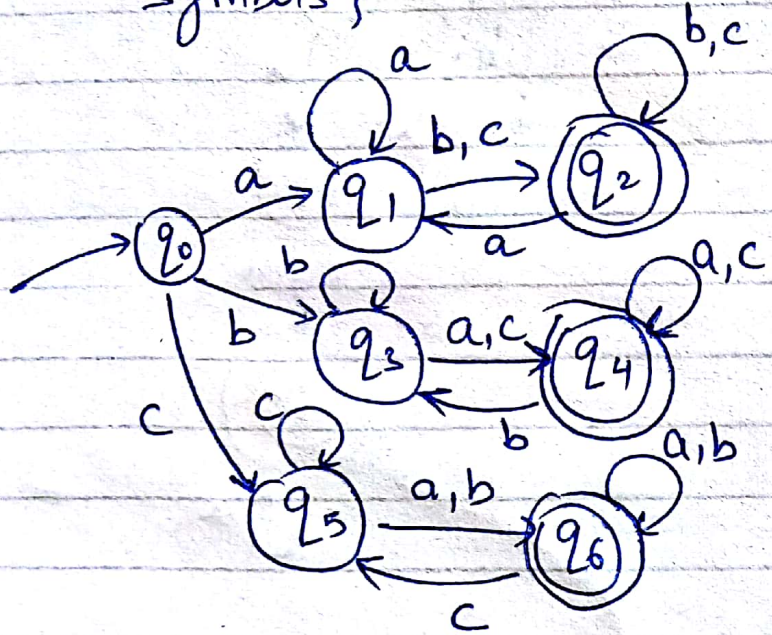
Q. $L = \{a^3 b w a^3 \mid w \in \{a,b\}^*\}$



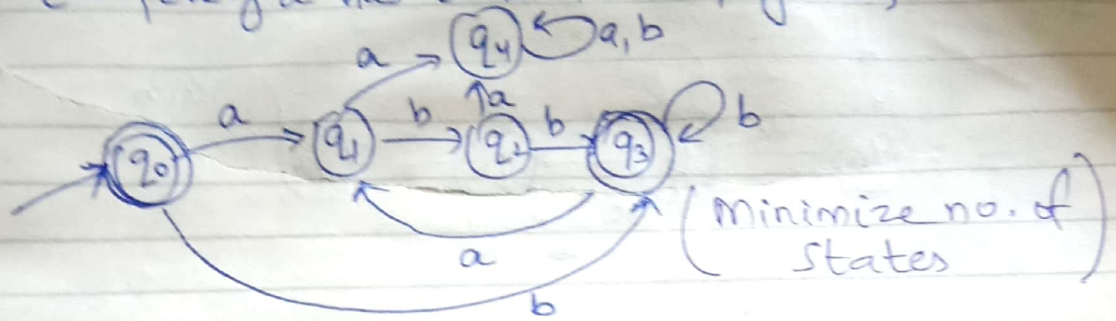
Q. $L = \{ x \in \{a,b\}^* , x$ contains atleast 2 consecutive a's and x does not contains 2 consecutive b's$\}$

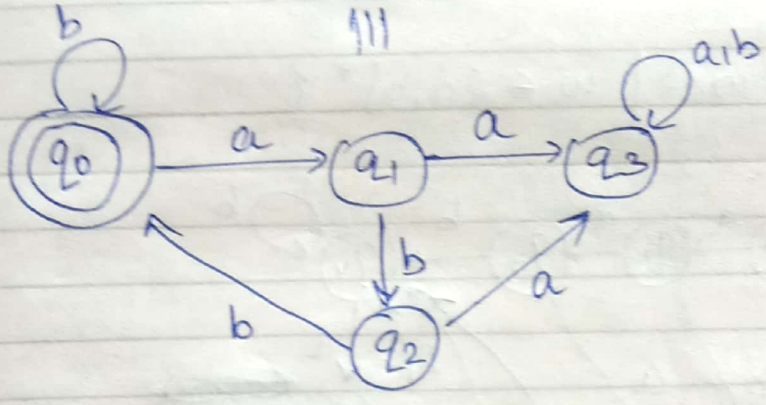


Q. $L = \{ x \in \{a,b,c\}^* , x$ begins and ends with different symbols$\}$

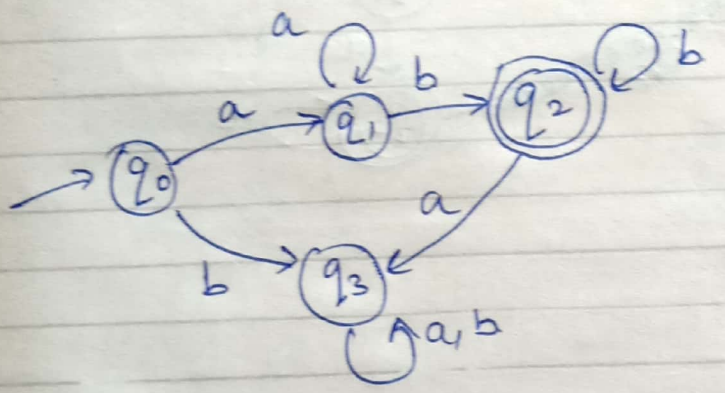


Q. L = {every a has to be followed by bb}

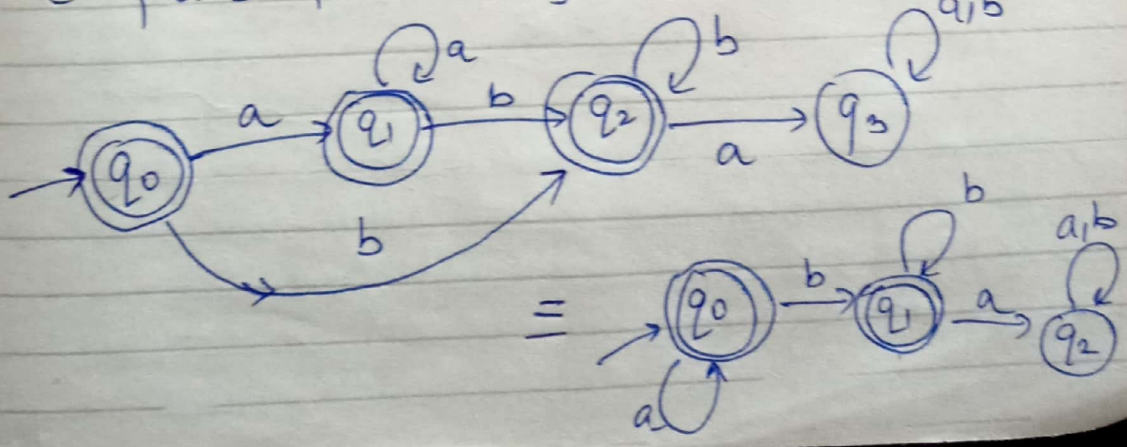




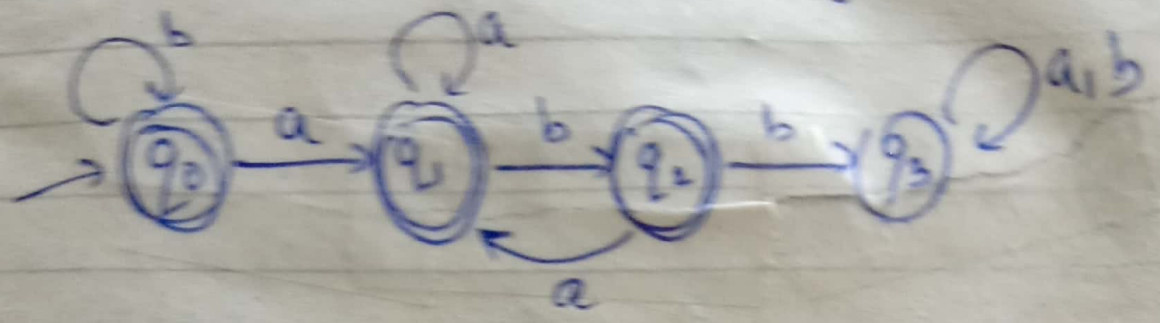
Q. $L = \{ a^n b^m \mid n, m \geq 1 \}$



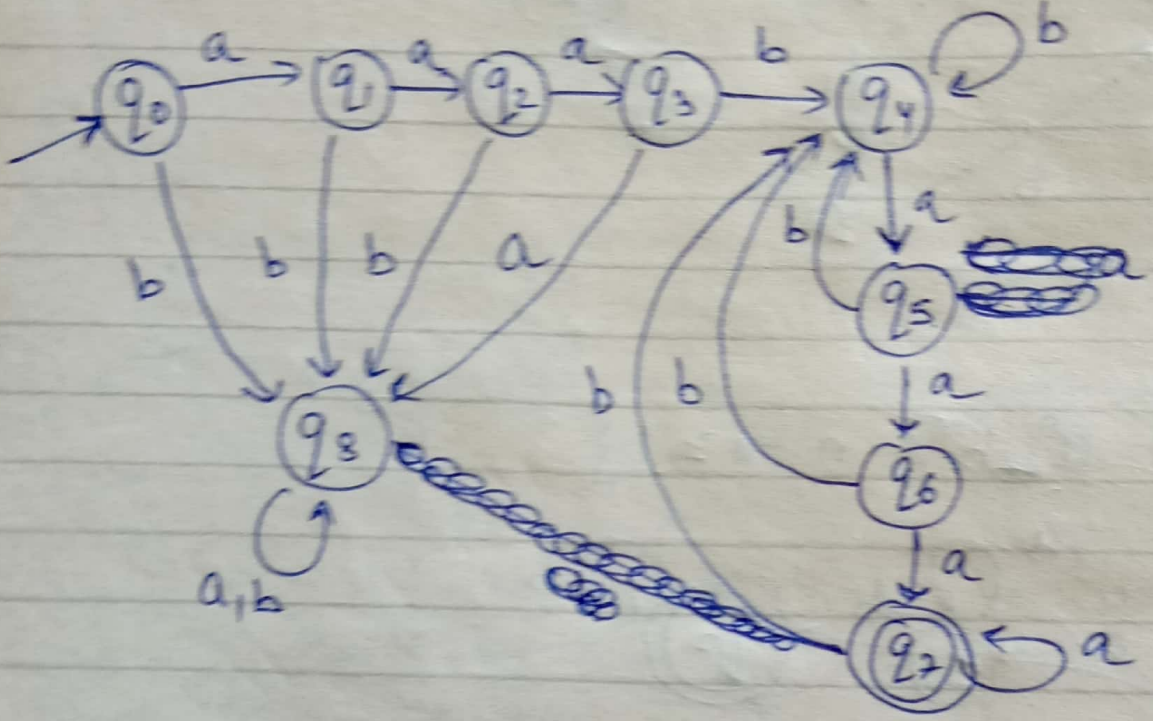
Q. $L = \{ a^n b^m \mid n, m \geq 0 \}$



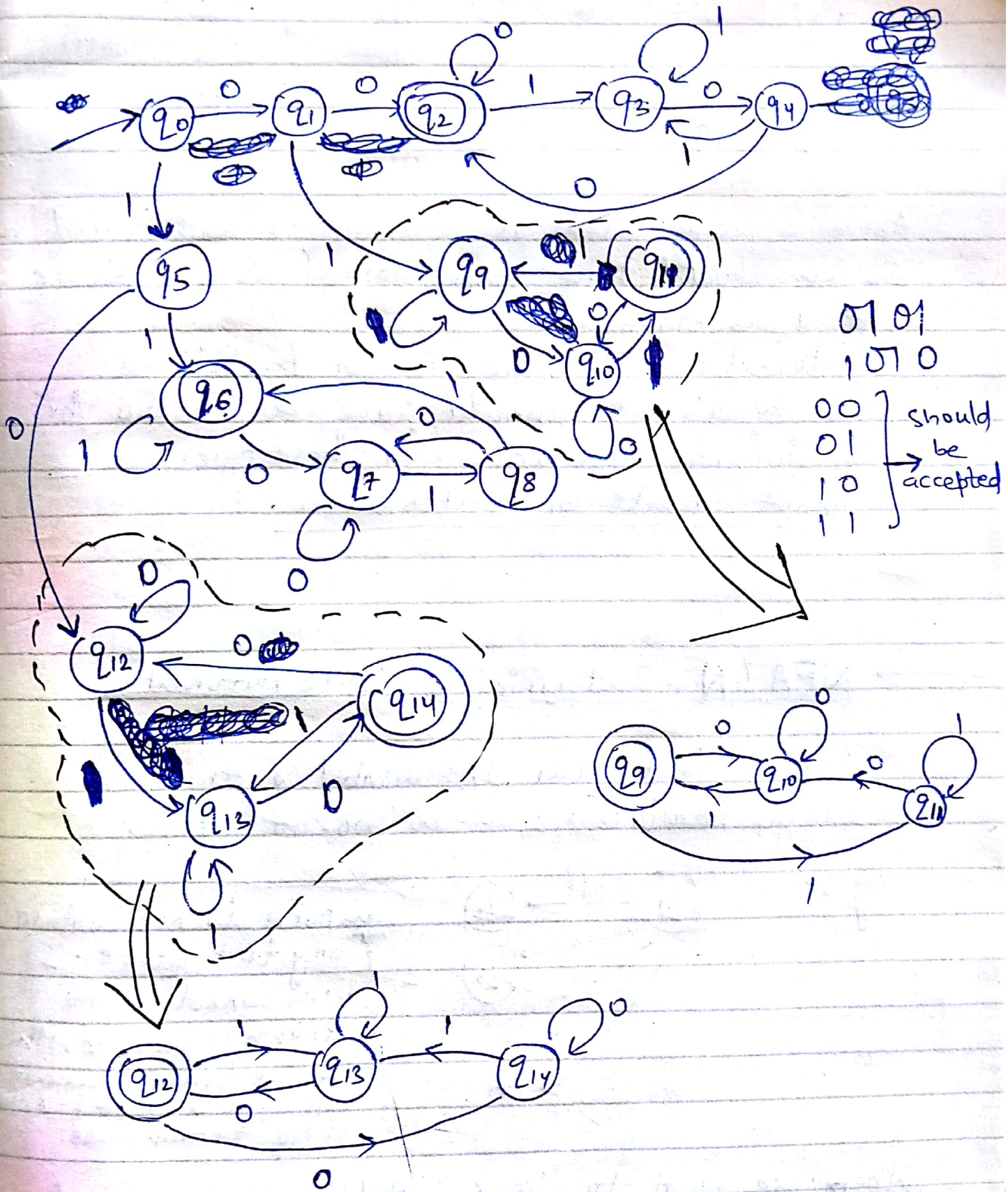
Q. $L = \{a$ never be followed by $bb\}$



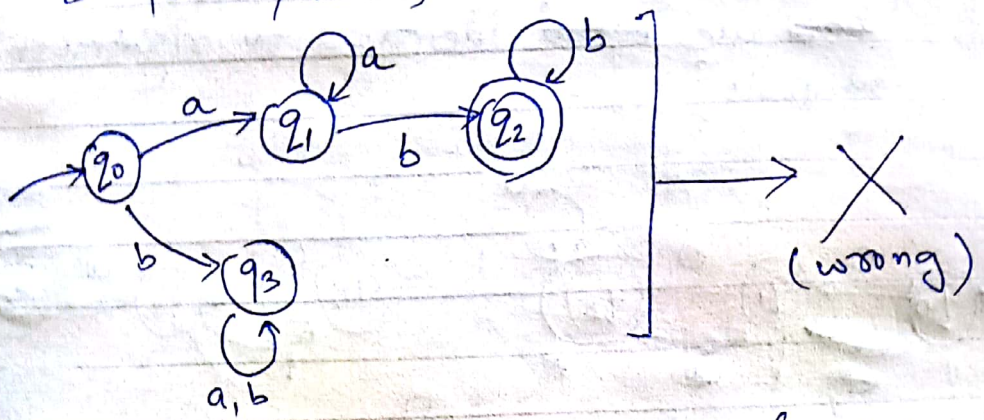
Q. $L = \{ a^3 b w a^3 \mid w \in \{a,b\}^* \}$



Q. L = { all strings in which the leftmost 2 symbols ~~[illegible]~~ are ~~equal~~ identical to rightmost 2 symbols }



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

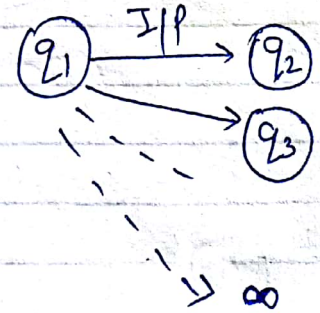


Above is wrong as n is common for both a and b. So we should have same no. of a's and b's i.e. if 4 a's then 4 b's.
Automata for above is not possible as for ∞ a's we should have ∞ b's. So this is infinite automata for these types, we cannot create an automata.

→ NFA (Non Deterministic finite Automata)

for a particular I/P symbol, transition can go to 0, 1, 2, --- ∞ no. of states

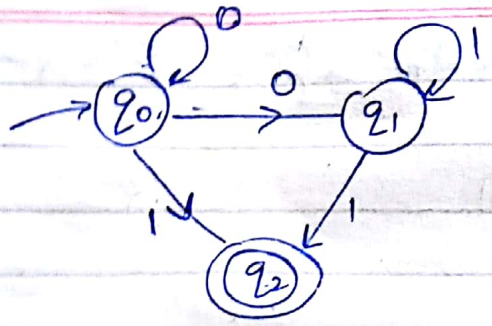
eg:-



* 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 & $\emptyset$ state is eqv. to trap state

When we want to check multiple 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
q_0	q_0, q_1	q_2
q_1	$\emptyset$	q_1, q_2
q_2		

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 $(Q, \Sigma, q_0, \delta, F)$
  - δ → Transition fn

$$\delta : Q \times \Sigma \longrightarrow 2^Q$$

2^Q → power set

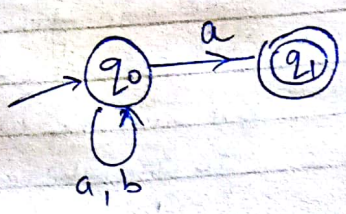
* Every DFA is a NFA but every NFA is not a DFA

* Every DFA is a NFA because
  $Q = \{q_1, q_2, q_3\}$, so all states in NFA are
  $\emptyset, q_1, q_2, q_3, q_1q_2, q_2q_3, q_1q_3, q_1q_2q_3$
  - q_1, q_2, q_3 → These 3 states are also present in DFA.

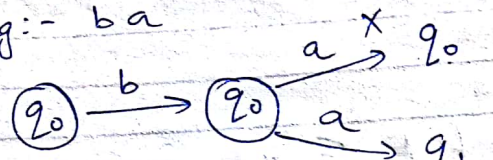
  As, all states of NFA are also present in DFA, so every DFA is a NFA.

* 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.

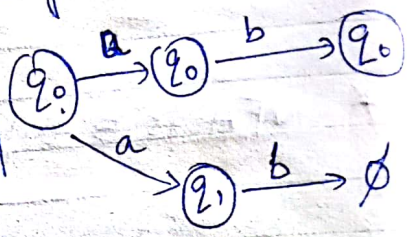
Q. $L = \{$ ends with a $\}$, $\Sigma = \{a, b\}$



eg:- ba

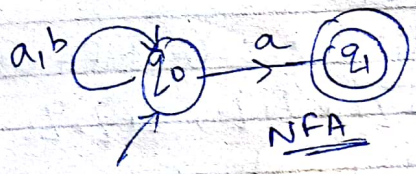


eg:- ab



→ Conversion of NFA to DFA

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



Transition Table:-

	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	—	—

- To convert to DFA, make table of states and inp~~ut~~ ~~transition table~~

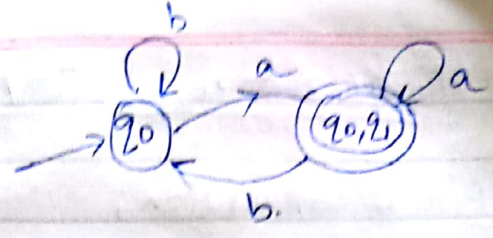
States	Input a	Input b
$\rightarrow q_0$	q_0, q_1	q_0
$\{q_0, q_1\}$	q_0, q_1	q_0

- Now, these two states will be in DFA

* The state which will be final state in NFA will also be final state in DFA & all states containg this state will also be final states

* In worst case, max no. of states in DFA can be 2^{q_0}.

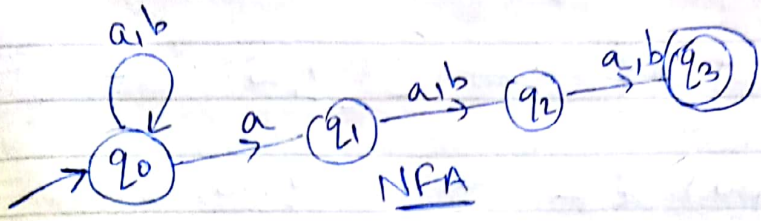
• So final DFA is :-



DFA

Q. L = { Third symbol from RHS is 'a' } , Σ = {a, b}

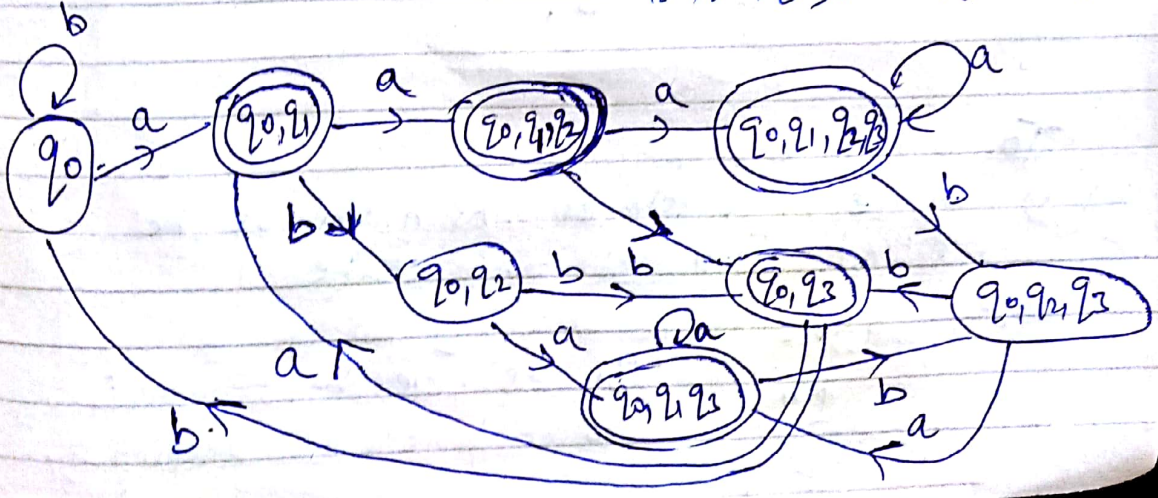
Ans. L = { aaa, aab, aba, abb. } (each $\in \Sigma^*$)



NFA

Transition Table

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



→ Minimization of DFA

After conversion,

No. of states (DFA) $\geq$ No. of states (NFA)

* Multible 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

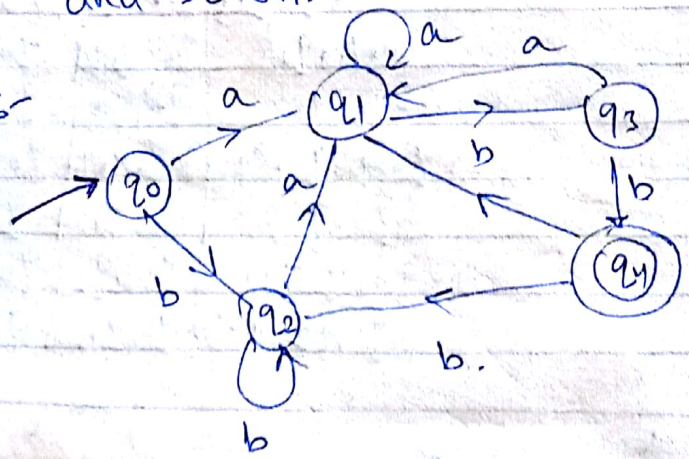
$$\delta(p,w) \in F \rightarrow \delta(q,w) \in F$$

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.

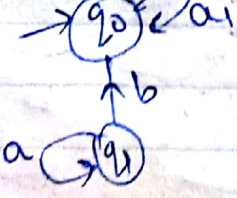
eg:-



To minimize,

Step (1) Remove those states which cannot be reached from starting state

eg:-



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

Step (2) Check equivalence states with help of transition table

0 eqv

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

* For 0 eqv, put final states in 1 group and non-final in another group

Q	Input a	Input b
→ q_0	q_1	q_2
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 eqv

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

If for all inp, we go to ~~same states or~~ states with same group in (n-1) equivalent, then these states are n-equivalent

2 eqv

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

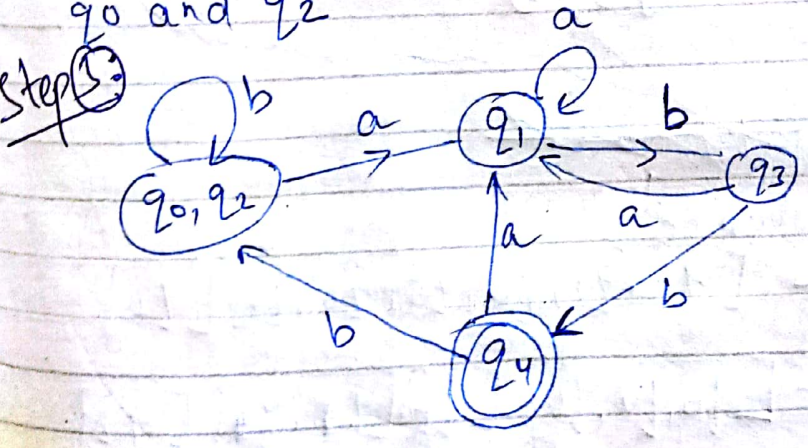
3 eqv

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

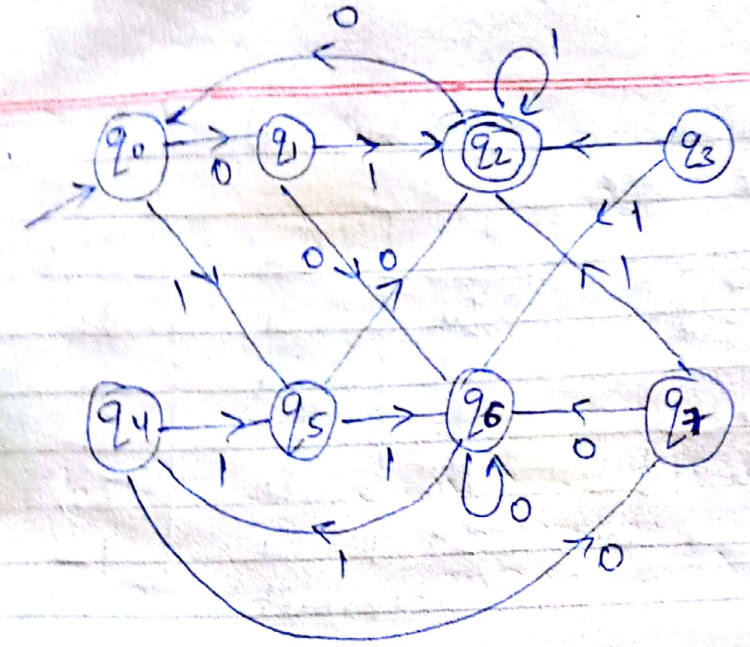
Now, no difference in groups, so stop.

From end point, we can see that we can merge q_0 and q_2

Step (3)



Q.

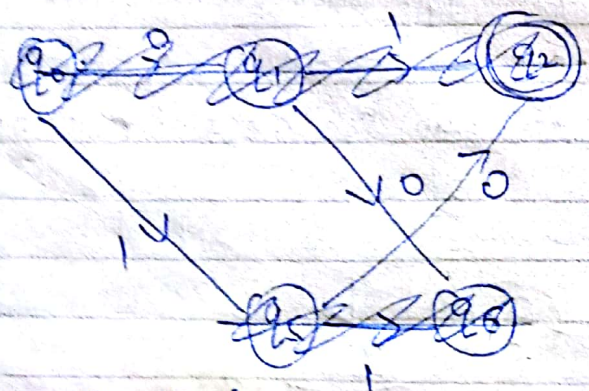


Minimize.

① Remove q_3

②

Q	0	1
q_0 ✓	q_1 ✓	q_5 ✓
q_1	q_6 ✓	q_2 ✓
q_5	q_2 ✓	q_6 ✓
q_6	q_6 ✓	q_4 ✓
q_2	q_0 ✓	q_2 ✓
q_4	q_2 ✓	q_5 ✓
q_2	q_6 ✓	q_2 ✓



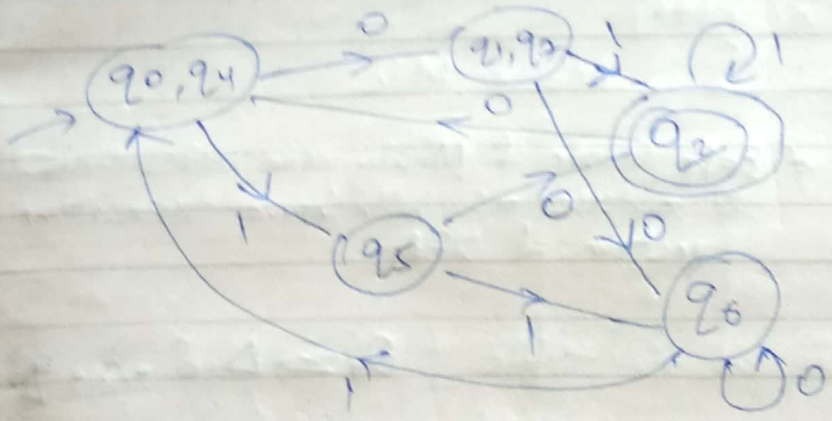
③ Test Equivalence

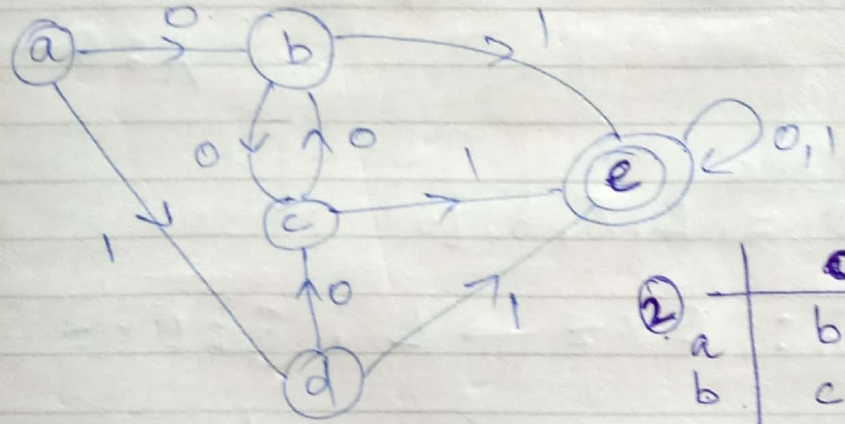
① eqv - $[q_0, q_1, q_4, q_5, q_6, q_7]$ $[q_2]$

1 eqv - $[q_0, q_4, q_6]$ $[q_5]$ $[q_1, q_7]$ $[q_2]$

2 eqv - [q0, q4] [q6] [q5] [q1, q7] [q2]

Therefore we can get 5 states



Q. 

Minimize.

②

	0	1
a	b	d
b	c	(e)
c	b	(e)
d	c	(e)
e	(e)	(e)

③ 0-eqv

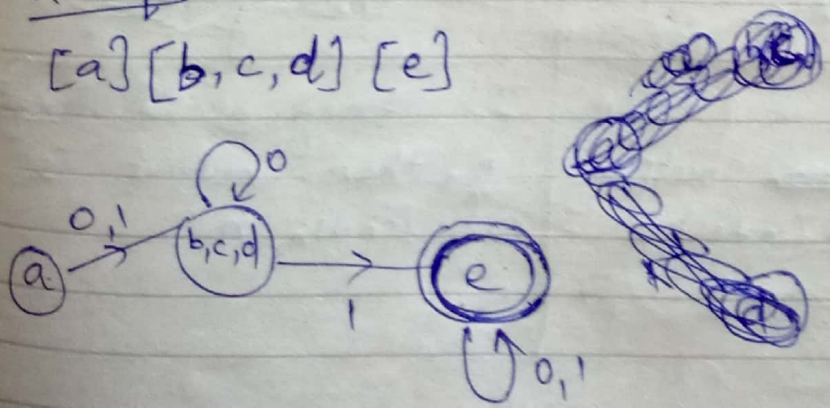
[a,b,c,d] [e]

1-eqv

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

2-eqv

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



* Mealy and Moore does not contain any final state

→ Finite Automata with o/p

$Z(t) = \lambda\{q(t), x(t)\}$ ——→ Mealy M/c

or

$Z(t) = \lambda(q(t))$ ——→ 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 $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Δ : set of output alphabet

λ : output function $\lambda : Q \to \Delta$

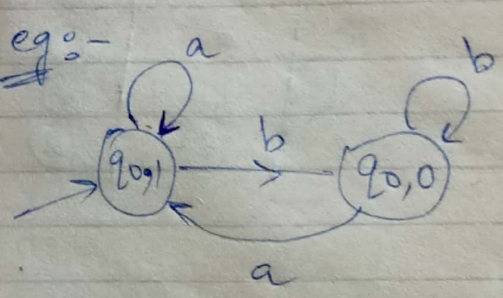
$\delta : Q \times \Sigma \to Q$

Mealy Machine

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

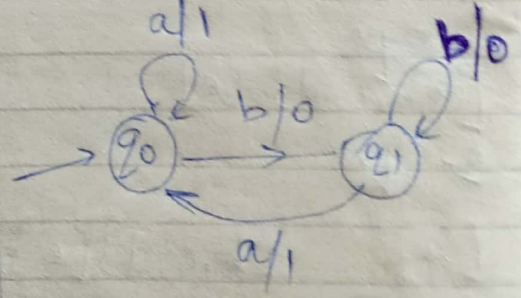
$\lambda : Q \times \Sigma \to \Delta$, $\delta : Q \times \Sigma \to Q$

eg:-



Moore M/c.

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

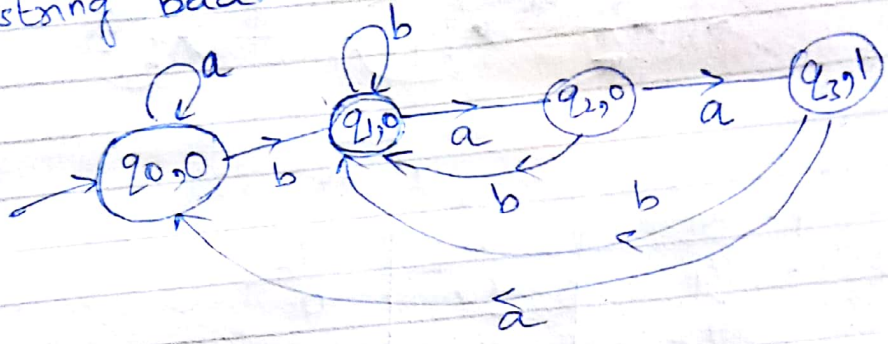


Mealy M/c

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

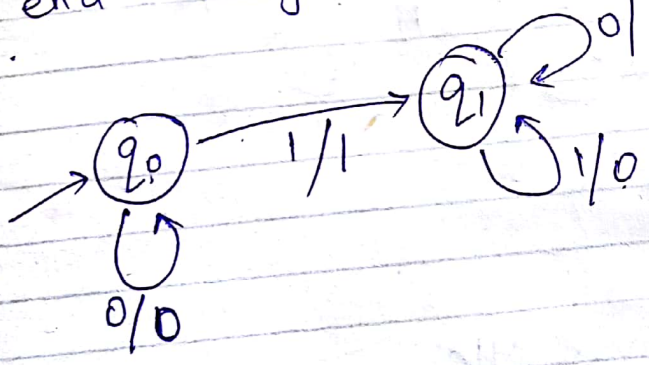
Q. Design a Moore M/c that takes set of all strings over {a,b} and counts the no. of occurrences of substring baa.

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.

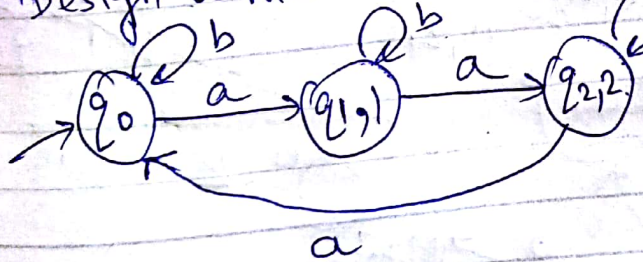
Ans.



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.

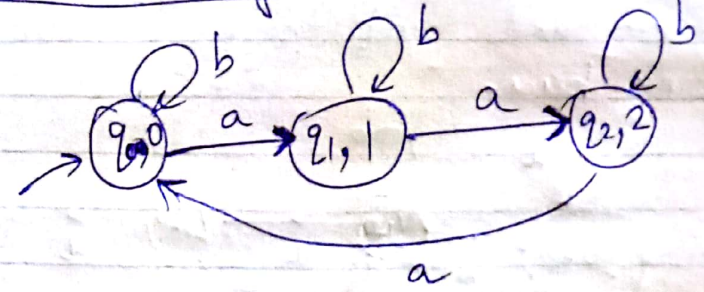
* Both Melay & Moore are equivalent i.e. they can be converted to each other.

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



→ Moore to Mealy M/c

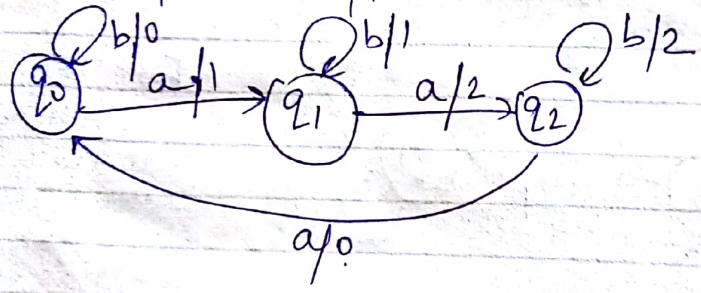
eg:-



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

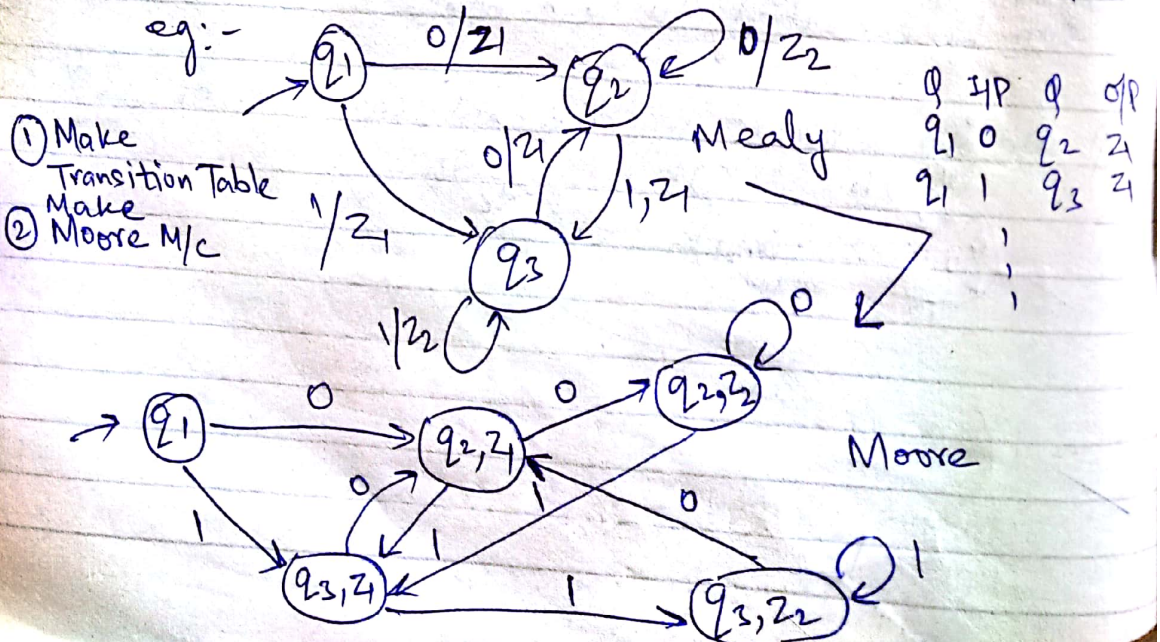
Step 2:- Write o/p with i/p.



→ Mealy to Moore M/c

If mealy contains M states & N outputs, then in worst case Moore contains MXN states

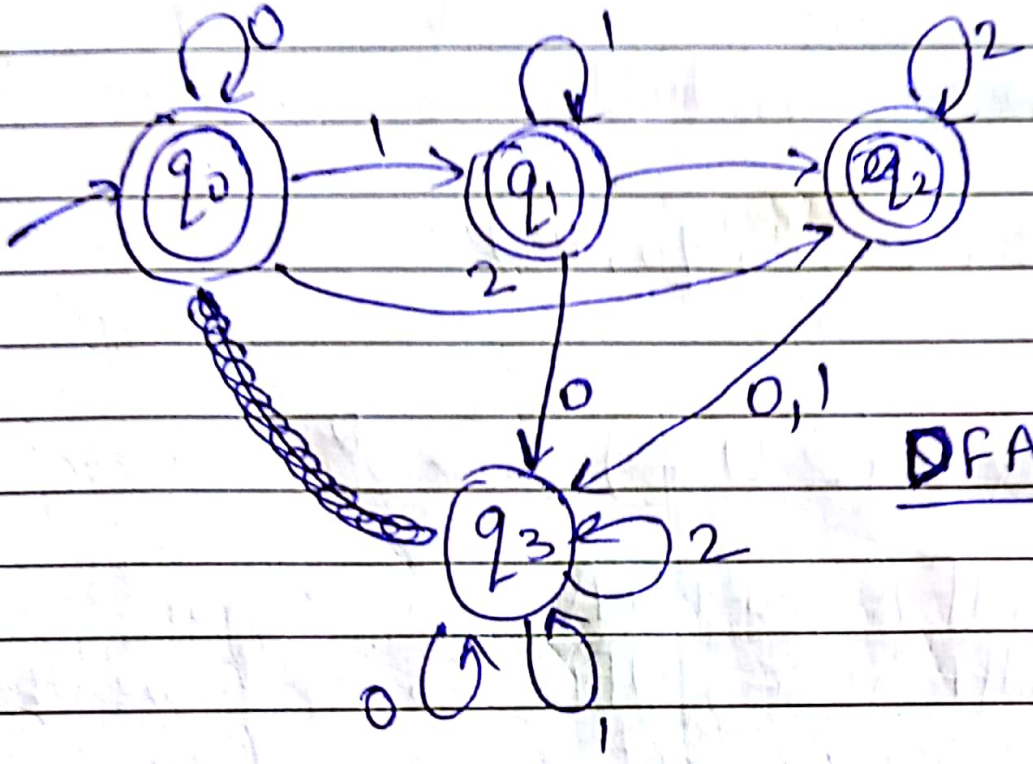
eg:-



① Make Transition Table
② Make Moore M/c

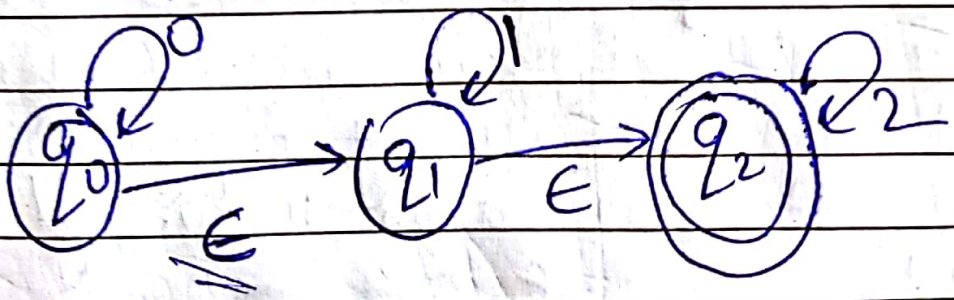
Epsilon NFA (ϵ-NFA)

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



DFA

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



ϵ-NFA

All NFA, DFA, ϵ-NFA are equivalent.

- ϵ-NFA is five tuple

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

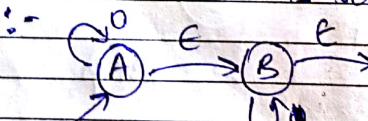
$$\delta: Q \times (\Sigma \cup \epsilon) \longrightarrow 2^Q$$

→ ϵ-NFA to NFA Conversion

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

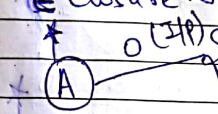
ϵ-Closure of any state is set of all states that can be reached from that state using ϵ only. Current state is also included in set

eg:-



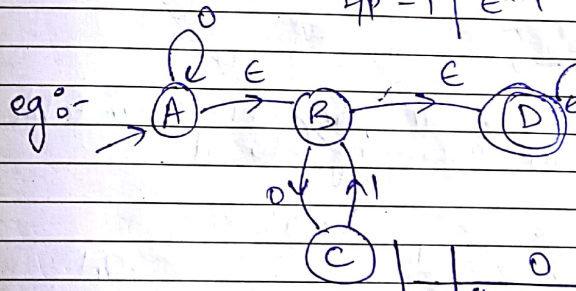
ϵ-Closure of A = {A, B, C}

To convert, make table of NFA, remove all ϵ edges and create new automata according to table. To remove ϵ edge, use ϵ Closure.

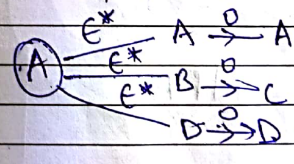


	A
I/P - 0	ϵ* 0 ϵ*
I/P - 1	ϵ* 1 ϵ*

eg:-



$A^{\epsilon^*} = \{A, B, D\}$



	0	1
A	{A, B, C, D}	D.
B	{C, D}	D
C	ϕ	{B, D}
D	D	D.

$A \xrightarrow{\epsilon^*} A \xrightarrow{1} \emptyset \xrightarrow{\epsilon^*} \emptyset$

$A \xrightarrow{\epsilon^*} B \xrightarrow{1} \emptyset \xrightarrow{\epsilon^*} \emptyset$

$A \xrightarrow{\epsilon^*} D \xrightarrow{1} D \xrightarrow{\epsilon^*} D$

$\epsilon^{\epsilon^*} = \{C\}$

$C \xrightarrow{\epsilon^*} C \xrightarrow{1} B$

$\epsilon^* \swarrow \quad \searrow \epsilon^*$

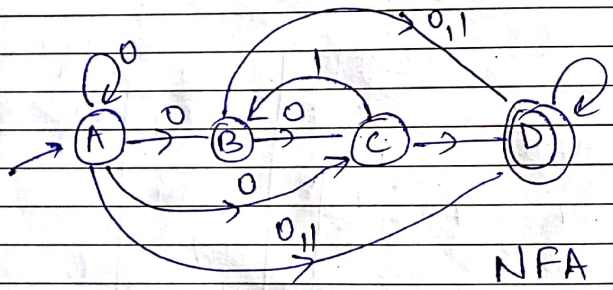
$B \quad D$

$B^{\epsilon^*} = \{B, D\}$

$B \xrightarrow{\epsilon^*} B \xrightarrow{0} C \xrightarrow{\epsilon^*} C$

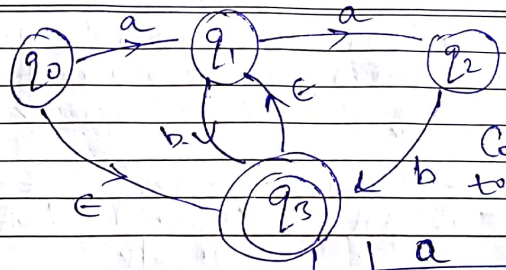
$\xrightarrow{\epsilon^*} D \xrightarrow{0} D \xrightarrow{\epsilon^*} D$

Now, after completing transition table draw NFA according to it



NFA

Q.



Convert to NFA

	a	b
q_0	$\{q_1, q_2\}$	$\{q_1, q_3\}$
q_1	q_2	$\{q_2, q_3\}$
q_2	$\emptyset$	$\{q_1, q_3\}$
q_3		

$q_0^{\epsilon^*} = \{q_0, q_3, q_1\}$

$q_0 \xrightarrow{\epsilon} q_0 \xrightarrow{a} q_1 \xrightarrow{\epsilon} q_1$

$q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon} q_2$

$q_0 \xrightarrow{\epsilon} q_3 \xrightarrow{a} \emptyset \xrightarrow{\epsilon} \emptyset$

$q_0 \xrightarrow{\epsilon} q_0 \xrightarrow{b} \emptyset \xrightarrow{\epsilon} \emptyset$

$q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{b} q_3 \xrightarrow{\epsilon} q_1$

$\epsilon \quad q_3 \xrightarrow{b} \emptyset \quad \epsilon \quad q_3$

$\epsilon \quad \emptyset$

$q_2^{\epsilon^*} = \{q_2\}$

$q_2 \xrightarrow{\epsilon} q_2 \xrightarrow{a} \emptyset \xrightarrow{\epsilon} \emptyset$

$q_2 \xrightarrow{\epsilon} q_2 \xrightarrow{b} q_3 \xrightarrow{\epsilon} q_3 \rightarrow q_1$

$q_1^{\epsilon^*} = \{q_1\}$

$q_1 \xrightarrow{\epsilon} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon} q_2$

$q_1 \xrightarrow{\epsilon} q_1 \xrightarrow{b} q_3 \xrightarrow{\epsilon} q_3 \rightarrow q_1$

NOTE -

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

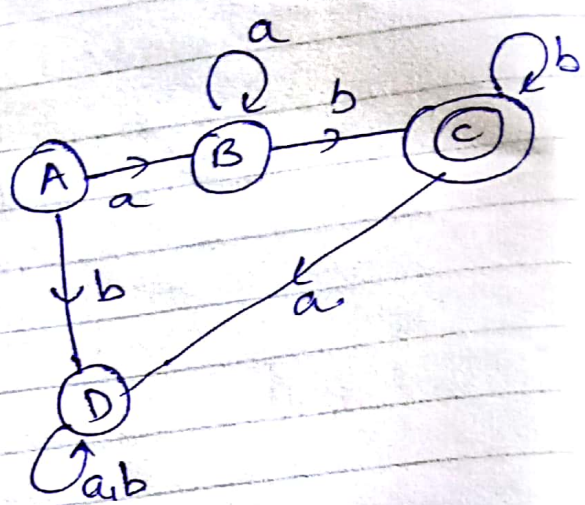
↓

States can be reached = ∈-Closure of state → States reached on I/p → ∈-Closure of state

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

→ Family of Languages.

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

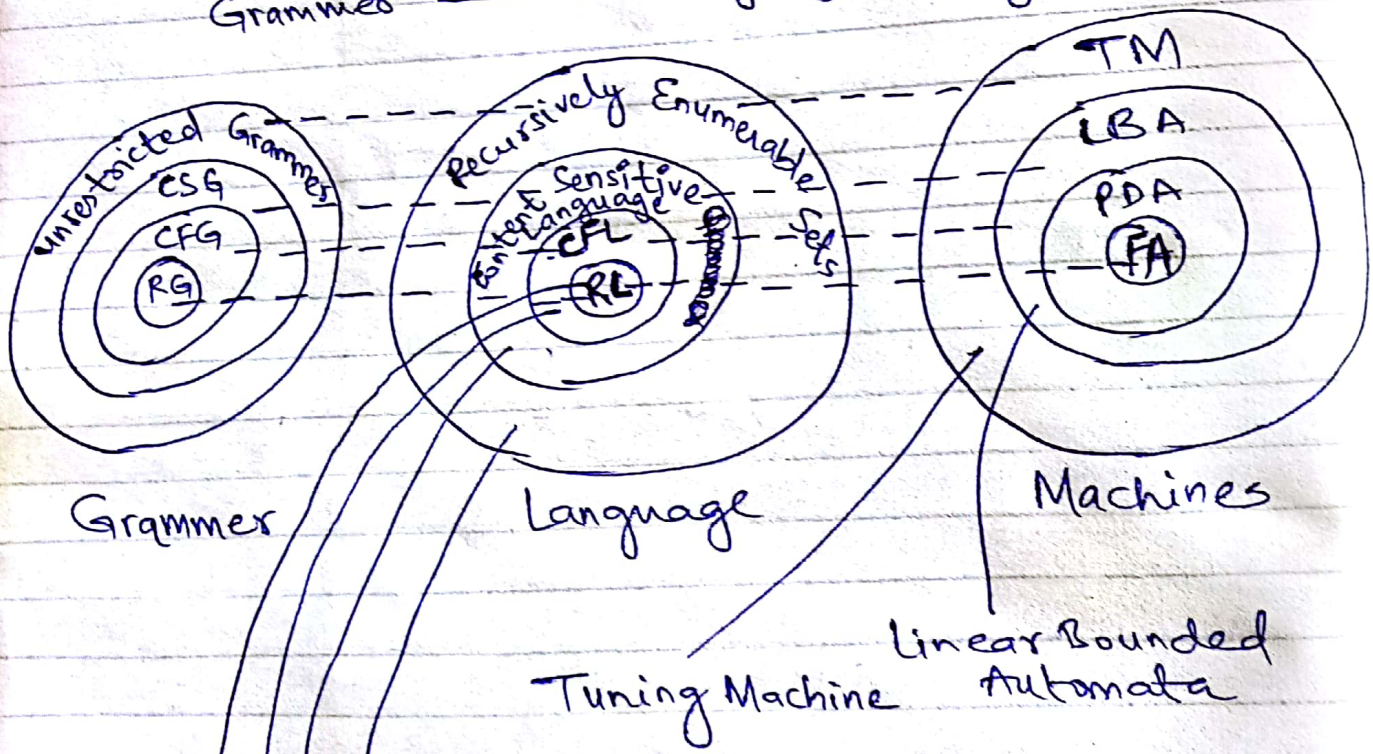


This automata is an acceptor as it accepts all strings like $a^n b^n$ but it is not rejecting $a^n b^m$.

F.A + memory = Push Down Automata

* Languages accepted by finite automata are regular languages.

Grammer generates Language accepted by Machines



→ eg:- a^p, p → prime

→ eg:- $a^n b^n c^n / n \geq 1$

→ eg:- $a^n b^n / n \geq 1$

→ eg:- $a^n / n \geq 1$

→ Regular ~~Language~~ Expression (RE)

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

(a) ϕ, ϵ and $a \in \Sigma$ are ~~regular~~ primitive R.E
(b) If r_1 and r_2 are two RE then $r_1 + r_2$, $r_1 r_2$ and R_1^* are also R.Es.
(c) A string is a RE if and only if it can be drived from the premetive REs by a finite no. of the application of the rules in (b)

eg:- $(a+b+c) \cdot (d+\phi)$ ⟶ It is a R.E.
($a+b+c$: RE, $d+\phi$: RE)

eg:- $(a+b+)$ ⟶ It is not a R.E.
($a+b+$: RE.)
↳ As r_1+ is not a R.E, r_2. is not a regular expression but r^* is a R.E.

Q. Design R.E for given Language

		R.E
(1)	$\{\epsilon, 0, 00, 000, 0000, \dots\}$	⟶ 0^*
(2)	$\{1, 11, 111, 1111, \dots\}$	⟶ $1 \cdot 1^*$

If R_1 & R_2 are RE then

1. $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
2. $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$
3. $L((R_1)) = L(R_1)$
4. $L(R_1^*) = [L(R_1)]^*$

* 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

1. $R = L(a^* \cdot (a+b))$
   ↳ {a, aa, aaa, aaaa, ---, b, ab, aab, aaab, aaaab}
2. $R = (a+b)^* (a+bb)$
3. $R = (aa)^* (bb)^* b$

Ans

1. $L(a^*) \cdot L(a+b)$
   ⇒ $[L(a)]^* (a, b)$
   ⇒ $(a^*) \cdot (a, b)$
   → $\{\in, a, aa, aaa, ---\} \cdot (a, b)$
   ⇒ {a, aa, aaa, ---, b, ab, aab, ---}

(2) $\{\epsilon, (a+b), (a+b)^2, \cdots\} \cdot \{a, bb\}$

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

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

(3) It can be seen that R is a RE with even no. of a's & odd no. of b's.

So,

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

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

$\Sigma = \{0, 1\}$

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

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

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

③ $L = \{$at least two length$\}$

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

④ $L = \{$atmost two length$\}$

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

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

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

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

$\rightarrow b^* a b^* a b^*$

(7) $L = \{$ atleast 2 a's $\}$

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

(8) $L = \{$ at most two a's $\}$

$\Rightarrow \quad b^* + b^* a b^* + b^* a b^* a b^*$

or

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

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

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

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

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

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

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

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

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

(13) $L = \{$ set of all strings in which no two a's and b's come together $\}$

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

$(ba)^*(b+\epsilon)$

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

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

$\Rightarrow (ab)^*a + (ab)^* + (ba)^*$

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

OR

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

(14) $L = \{a^n b^n c^p \mid n, n, p \geq 1\}$

$\Rightarrow a \cdot a^* b \cdot b^* c \cdot c^*$

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

$\Rightarrow a \cdot a^* b (aa)^* bb$

(16) $L = \{w \mid |w| = 3k_1 + 5k_2, k_1 \geq 0, k_2 \geq 0\}$

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

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

→ 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 a unique solution given by $R = QP^*$

Proof :-

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

$$R = Q + RP \quad \text{—— (1)}$$

$$= Q + (QP^*)P$$

$$= Q(\epsilon + P^*P)$$

$$= QP^*$$

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)

$$= Q + RP$$

$$= Q + (Q + RP)P$$

$$= Q + QP + RPP$$

$$= Q + QP + RP^2$$

$$= Q + QP + \dots + QP^i + RP^{i+1}$$

$$= Q(\epsilon + P + P^2 + \dots + P^i) + RP^{i+1} \quad \text{—— (2)}$$

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 $Q(\epsilon + P + P^2 + \dots + P^i) + RP^{i+1}$ as P does not contain ε, so $w \notin R.P^{i+1}$ and thus

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

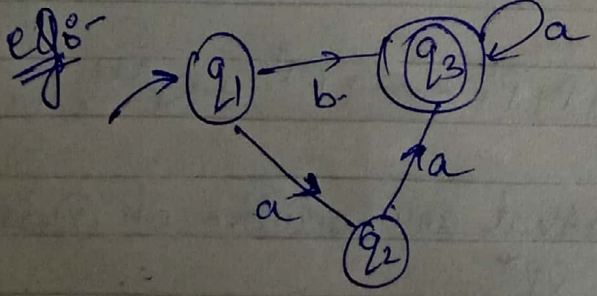
Consider, Now, $w \in QP^*$, then w is in the set QP^k for some $k \geq 0$ and hence in $Q(\epsilon + P + P^2 + \cdots + P^k)$, so w is in RHS of (2).
Therefore $w \in R$ of (2)
Thus R and QP^* represents the same set.
As,
$w \in R$, then QP^*
$w \in QP^*$, then R.

→ FA to RE

Rules :-

1. The diagram does not have ϵ moves
2. It has only one initial state
3. Its vertices are $q_1 \cdots q_n$
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.

eg:-



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

$q_2 = \epsilon \cdot a = a$

$q_3 = \epsilon \cdot b + a \cdot a \cdot + q_3 \cdot a$

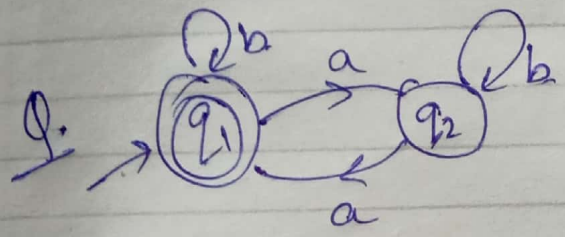
$= b + aa + q_3 a$

By Ardens Theorem, $R = Q + RP$

So, $R = q_3$, $Q = b + aa$, $P = a$

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

Q.



$q_1 = \epsilon + q_1 \cdot b + q_2 \cdot a$ — ①

$q_2 = q_1 \cdot a + q_2 \cdot b$ — ②

~~Using Ardens Theorem on q_1,~~

~~$q_1 = (\epsilon + q_2 \cdot a) \cdot b^*$ — ③~~

Using Ardens Theorem on q_2,

$q_2 = (q_1 \cdot a) \cdot b^*$ — ④

~~Put ④ in ③~~

~~$q_1 = b^* + ab^* q_2$~~ $q_2 \cdot ab^*$

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

~~Using Ardens Theorem on q_1.~~

~~$\therefore q_1 = b^* (ab^* ab^*)^*$~~

Put ④ in ①

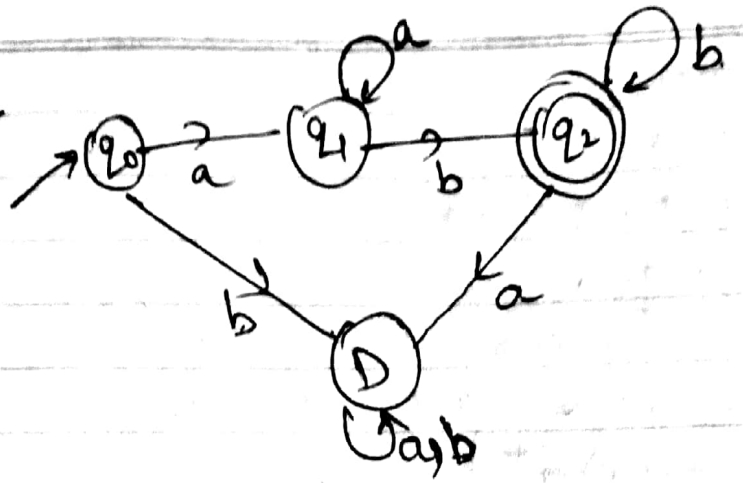
$q_1 = \epsilon + q_1 \cdot b + (q_1 \cdot a) \cdot b^* a$

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

Using Ardens Theorem,

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

Q.



→ $q_0 = \epsilon$.

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

$= a + q_1 \cdot a$

→ $q_1 = a \cdot a^*$

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

$= aa^*b + q_2 b$

→ $q_2 = (aa^*b)b^*$

Regular Expression to FA

- **Kleene's Thm**

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

$$L = T(M)$$

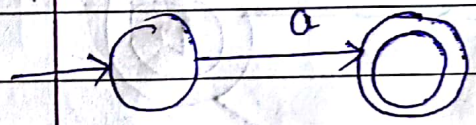
↳ NFA or FA

$R = (a+b)$ ⇒ characters = 3

Proof: Using Induction.

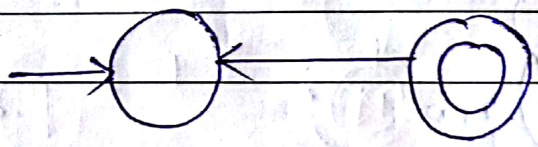
$n = 1$ $R = \epsilon, \phi$ or a.

a:



ϵ:

ϕ:

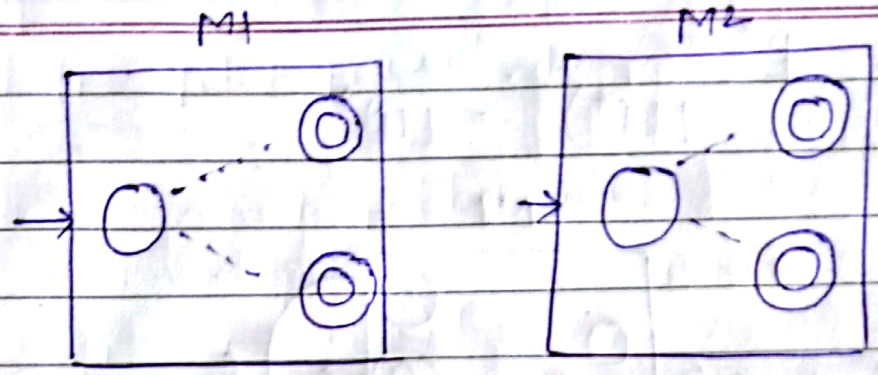


Regular expressions possible of R of length less than equal to 'n'.

$R = P + Q \rightarrow M2$

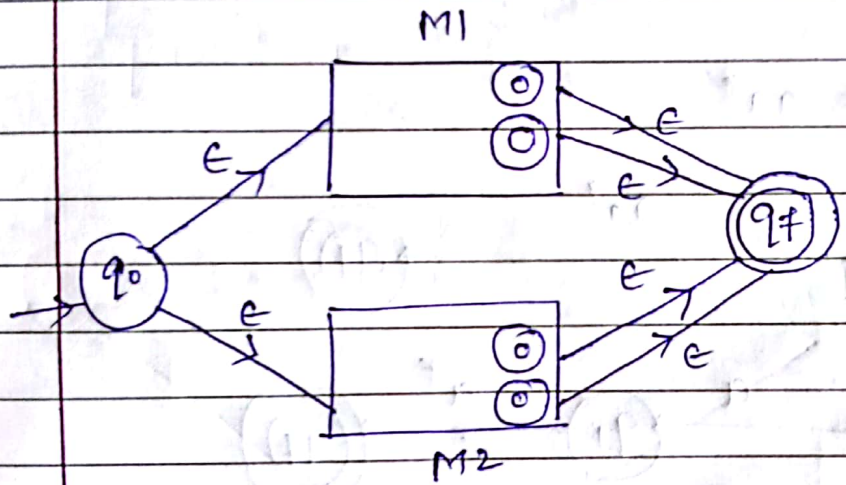
$R = PQ \rightarrow M1$

$R = P^* \rightarrow M1$

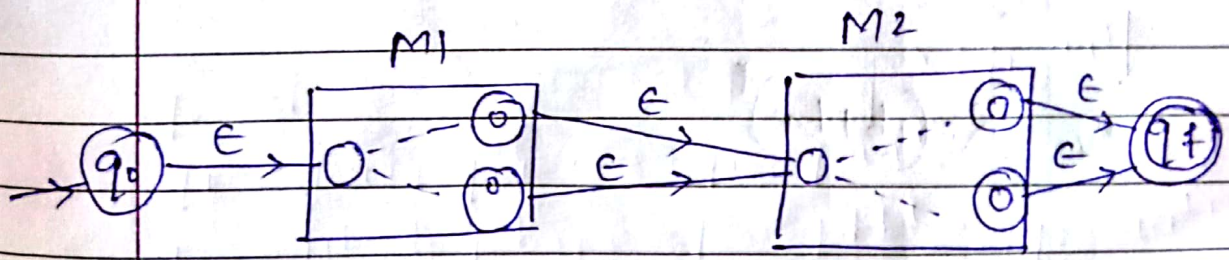


i). $L(R) = L(P+Q)$

$= \boxed{L(P) \cup L(Q)}$



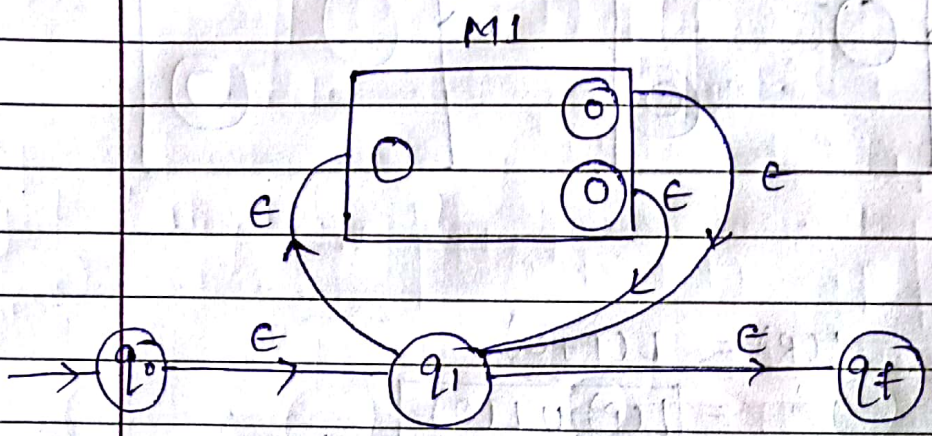
ii). $L(R) = L(PQ) = L(P) \cdot L(Q)$



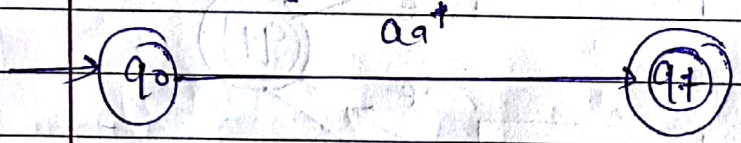
Page No.
Date

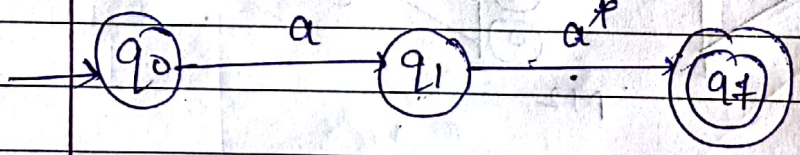
3). P^*

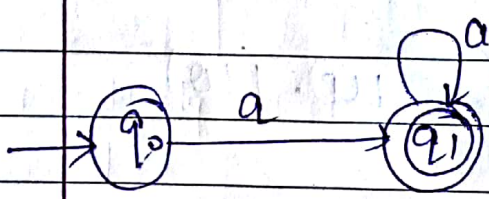
$L(P^*) = L(P)^*$



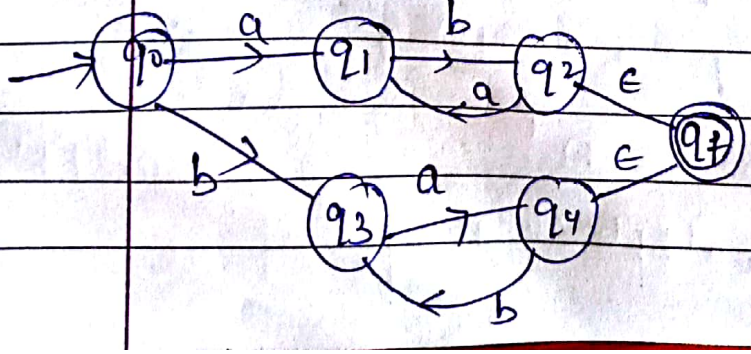
Q. $R = aa^*$







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

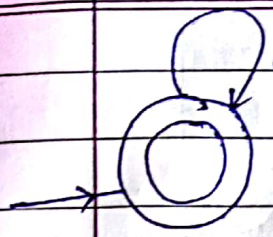


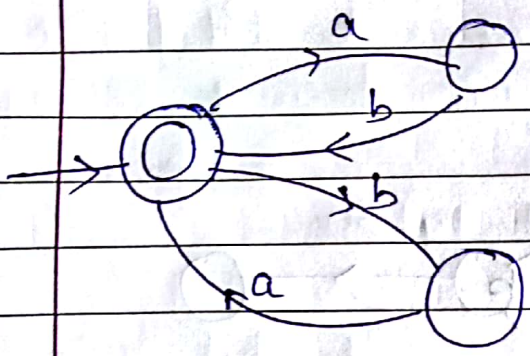
IMP $(a+b)^+$ mean all combo accept ∅.

i.e. $n \geq 1$ but $n \neq 0$

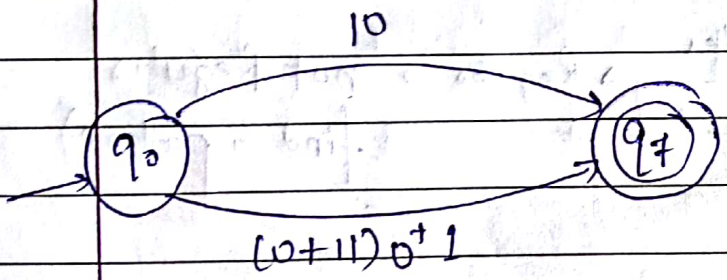
Page No.

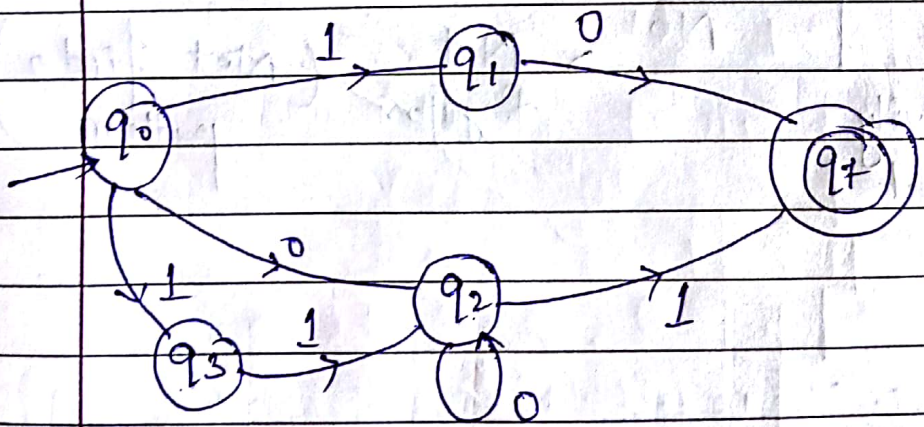
Date





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



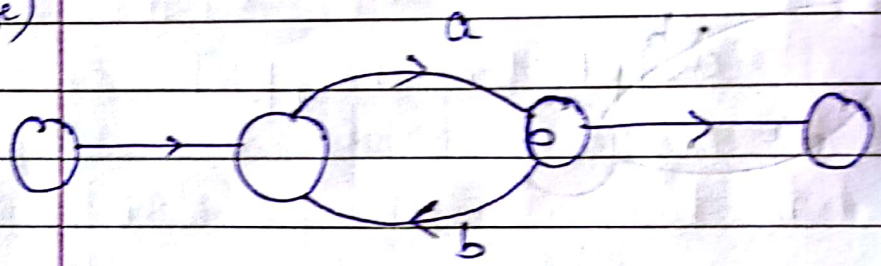


Pumping Lemma Theorem

Language

- Finite
- Infinite

Eg, L = {ab, abab ---}
(infinite)



Loop of ab

- Pumping lemma Test
  - Yes → Regular or not Regular (find a pattern)
  - No → Not Regular (Not find a pattern)
- negativity test

Page No.
Date

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.

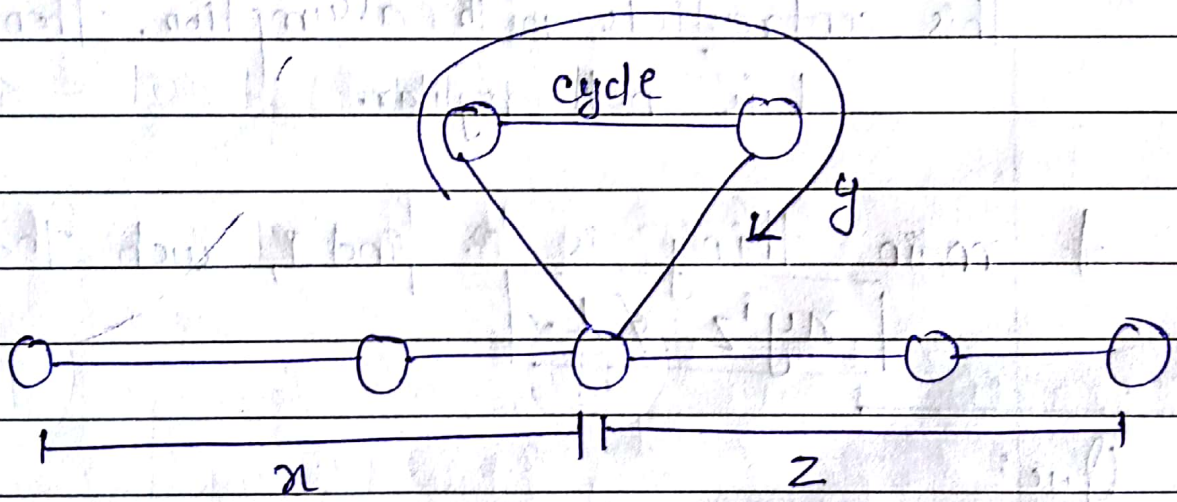
Let, $w \in L$ and $|w| \geq m$.

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

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

or, $|y| > 0$

or, $|y| \geq 1$



$|xy| \leq n$

Questions

Page No.
Date

Application of Pumping lemma

step1: Assume that L is regular. Let 'n' be the no. of states in corresponding FA.

step2: choose a string w such that $|w| \geq n$ using pumping lemma to write, $w = xyz$ with,

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

step3: find a suitable i, such that, $xy^iz \notin L$. This contradicts with assumption. Hence, L is not regular.

* main thing is to find i, such that, $\boxed{xy^iz \notin L.}$

Que:

1). $a^n \mid n \geq 1$ (R)
2). $a^n b^m \mid n, m > 1$ (R)
3). $a^n b^n \mid n < 10^{10^{10}}$ (R)
4). $a^n b^n \mid n \geq 1$ (NR)
5). $ww^R \mid |w| = 2, \Sigma = \{a, b\}$ (R)
6). $ww^R \mid w \in (a,b)^*$ (NR)
7). $ww \mid w \in (a,b)^*$ (NR)
8). $a^n b^m c^t \mid n, m, t \geq 1$ (R)
9). $a^i b^{2j} \mid i = j \geq 1$ (R)

Page No.
Date

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

11).
(a) $a^n \mid n$ is even (R)
(b) $a^n \mid n$ is odd (R)
(c) $a^n \mid n$ is prime (NR)
(d) $a^{n^2} \mid n \geq 1$ (NR)
(e) $a^{2^n} \mid n \geq 1$ (NR)

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

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

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

15). $L = \{w \mid n_a(w) = n_b(w)$ (NR)
$\rightarrow n_a(w) > n_b(w)$ —]
$\rightarrow n_a(w) < n_b(w)$
}

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

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

18). $wxw^R \mid w \in (0,1)^+$ & $|x| = 5$
↓
(NR)
∵ $|x|$ is fixed to 5.
So, if length exceeds, we'll need to check.

IMP
if pattern is not there or, pattern is in gp.
=
if A.P. is there, then pattern is there
=

x
(001111001)
w w^R
IMP
we can make it as, starting & ending symbol are same.
∵ $|x|$ is not fixed, so for all string.
$|w| = 1$ & rest will be taken under x

Page No.
Date

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). ~~$w = a^x$, where x is prime & $x \geq n$~~
Acc.
$w = a^p$, where, p is prime & $p \geq n$

acc. to Pumping lemma,
$w = xyz \quad |xy| \leq n$
$|y| > 0$

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

$xyz = w = a^k a^m a^\theta$ → z
x ← → y
$k + m + \theta = p$

$xy^i z \neq a^m$ (2) $(y^i)^m$

$|xy^i z| = |xyz| + |y^{i-1}|$
$= p + (i-1)m$
for $i = p+1$
$= p + (p+1-1)m$
$= p + pm = p(1+m)$ → product of two no's can't be a prime

one prime is there

Page No.
Date

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

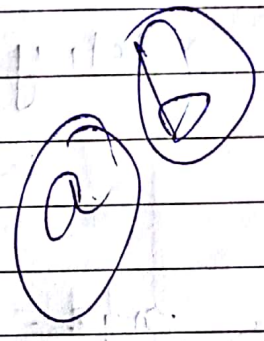
1). assume 'L' is regular.
let 'n' be the no. of states.

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

acc. to pumping lemma

$\omega = xyz$, $|xy| \leq n$
$|y| > 0$

let
$y = (ab)^m$, $m \geq 1$ & $m \leq n$



$a^i b^i$

$xyz = \omega = (ab)^k (ab)^m (ab)^0$

$[(k+m+0) = i]$ $\forall i$

$|xy^p z| = |xyz| + |y^{p-1}|$

$= i + (p-1)m$

~~for~~ $i \neq p+1$

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

$= k + pm =$

Put $p = i+1$

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

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

i.e. (b)

Page No.
Date

Here, for m = odd.
result will be odd.
∴ Not regular.

---

*Correct proof

Let, $w = a^n b^n$

$|w| = 2n > n$

$w = xyz$

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

Let, $y = a^k$, $1 \leq k < n$

Let, $x = a^q$, $0 \leq q \leq n$

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

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

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

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

$n+k \neq n$, $(k \geq 1)$ ✓

Page No.
Date

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

Let, $w = a^n b$

1). Let, L is regular.

$ww = a^n b a^n b$

Let, $y = a^k$, $k \geq 1$ & $k \leq n$

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

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

$x \quad y \quad z$

$|xyz| = 2n+2$

$|xy^i z| = |xyz| + |y^{i-1}|$

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

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

$= 2n+2+(nk)$

$= 2n+nk+2$

$= n(2+k)+2$

~~Put i=0~~

~~$xy^0 z$~~

Put $i = 2$

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

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

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

Page No.
Date

L1 and L2 are two Regular languages, then,

L1 ∪ L2, L1 ∩ L2, L1.L2, $\overline{L1}$, L1*, L1/L2 (Right Quotient)

Proof simply by property using R.E.

are also regular.

→ By making finite automata.
→ By generating regular expression.
→ By " regular grammar.

I) UNION (L1 ∪ L2)

Set of start

$L1 \rightarrow M1 = \{ S_1, \Sigma, \delta_{01}, \delta_1, F1 \}$

$L2 \rightarrow M2 = \{ S_2, \Sigma, \delta_{02}, \delta_2, F_2 \}$

$L(M1) = L1$, $L(M2) = L2$

We can define an,

F.A.,

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

and,

$\delta = (S_1 \cup S_2) \times \Sigma \rightarrow P(S_1 \cup S_2)$

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

$S = S_1 \cup S_2$

$F = F_1 \cup F_2$

Page No.
Date

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

for all $S \in S_1 \cup S_2$ and for all $a \in \Sigma$.

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

II) CONCATENATION $(L_1 . L_2)$

$L_1 \rightarrow M_1 = (S_1, \Sigma, S_{01}, \delta_1, F_1)$

$L_2 \rightarrow M_2 = (S_2, \Sigma, S_{02}, \delta_2, F_2)$

$S_1 \cap S_2 \neq \phi$

$M = (S, \Sigma, S_0, \delta, F)$ where, $S = S_1 \cup S_2$

$S_0 = \{S_{01}\}$ if $\epsilon \notin L_1$ ← $S_0 = \{S_{01}\}$ if $\epsilon \notin L_1$

or, $\{S_{02}\}$ if $\epsilon \notin L_1$

$F = \{F_2$ if $\epsilon \notin L_2$

$F_1 \cup F_2$ if $\epsilon \in L_2\}$

and,

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

$$\delta(S,a) = \begin{cases} \delta_1(S,a) & \text{if } S \in S_1 . \\ \delta_1(S,a), \delta_2(S_2,a) & \text{if } S \in F_1 \\ \delta_2(S,a) & \text{if } S \in F_2 \end{cases}$$

Page No.
Date

III) COMPLEMENT ($\bar{L}$)

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

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

IV) INTERSECTION ($L_1 \cap L_2$)

→ pair of states too

(first below automata & second automata)

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

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

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

we define a DFA :-

$M = (S, \Sigma, \delta, S_0, F)$ such that,

$S = S_1 \times S_2$

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

↙ Pair

$F = F_1 \times F_2$

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

if defined by,

$\delta((q,p), a) = (\delta_1(q,a), \delta_2(p,a)) = (r,s)$

Page No.
Date

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

$\delta((q,p), a) = (r,s)$

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

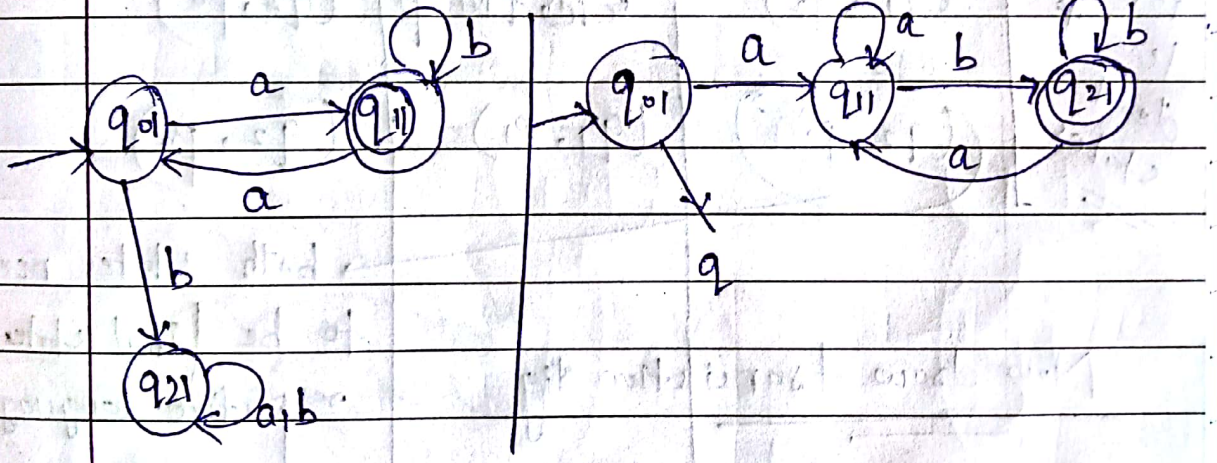
---

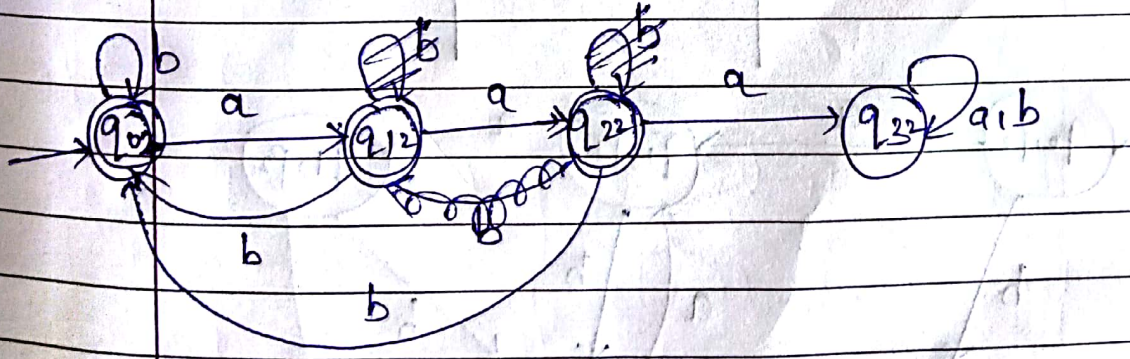
to 7

Q. Intersection

$\Sigma(a,b)$, L_1 is the language that begins with a & end with b.

L_2 → triple a is not a substring.





Page No.
Date

Transition table of

(H)

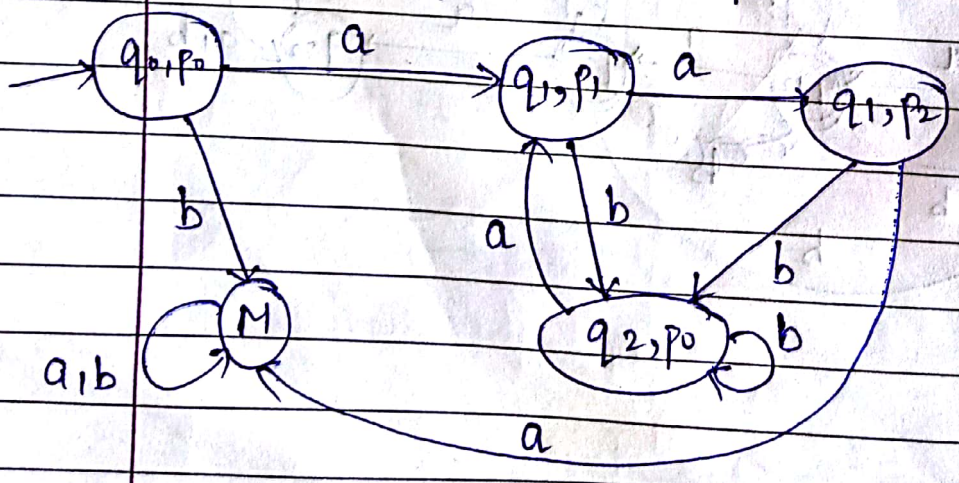
	a	b
(q_{01}, q_{02})	(q_{11}, q_{12})	(q_{21}, q_{02})
(q_{11}, q_{12})	(q_{01}, q_{22})	$(q_{21},$

IMP
if any one leads to dead state, whole lead to dead states

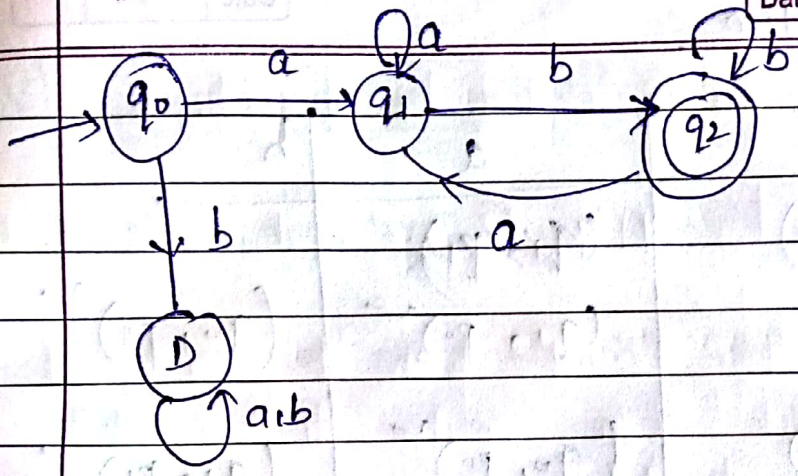
Pairs	a	b
(q_0, p_0)	(q_1, p_1)	ϕ
(q_1, p_1)	(q_1, p_2)	(q_2, p_0)
(q_1, p_2)	ϕ	(q_2, p_0)
(q_2, p_0)	(q_1, p_1)	(q_2, p_0)

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

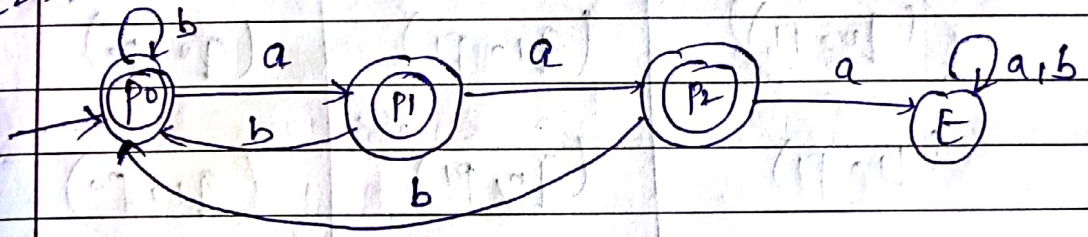
Now draw transition dig



L1:

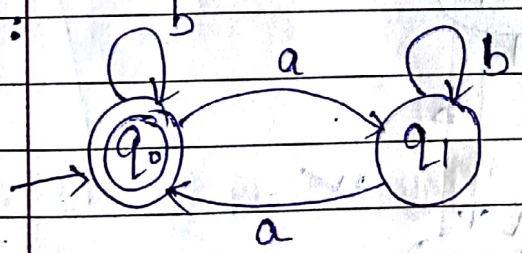


L2:

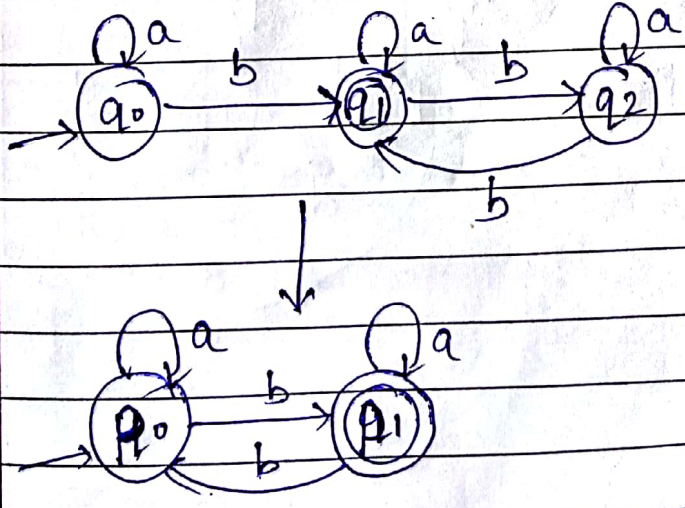


Q. Draw for, no. of a's - even (Intersection)
no. of b's - odd

L1:



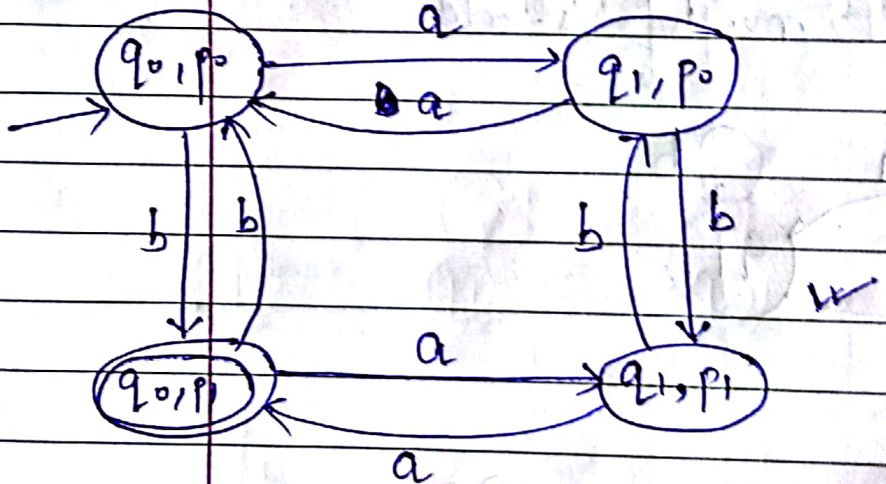
L2:



Page No.
Date

Transition

	a	b
→ (q_0, p_0)	(q_1, p_1)	
	(q_1, p_0) ✓	(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) ✓



Page No.
Date

V) RIGHT QUOTIENT. (L1/L2)

$L1/L2 = \{x : xy \in L1, \text{ for some } y \in L2\}$

IMP.

From L1, we take all those strings whose suffix is in L2 after removing their suffix part.

① $L1 = \{Fish, dog, Carrot\}$

$L2 = \{rot\}$ then,

$L1/L2 = \{Car\}$

② $L1 = \{Carrot, Parrot, rot\}$

$L2 = \{rot\}$

$L1/L2 = \{Car, Par, \in\}$

$L3 = \{rot, cheese\}$

$L1/L3 = \{Car, Par, \in\}$

③ $L1 = \{xab, Jab\}$, $L2 = \{b, ab\}$ then,

$L1/L2 = \{Xa, Ja, X, J\}$

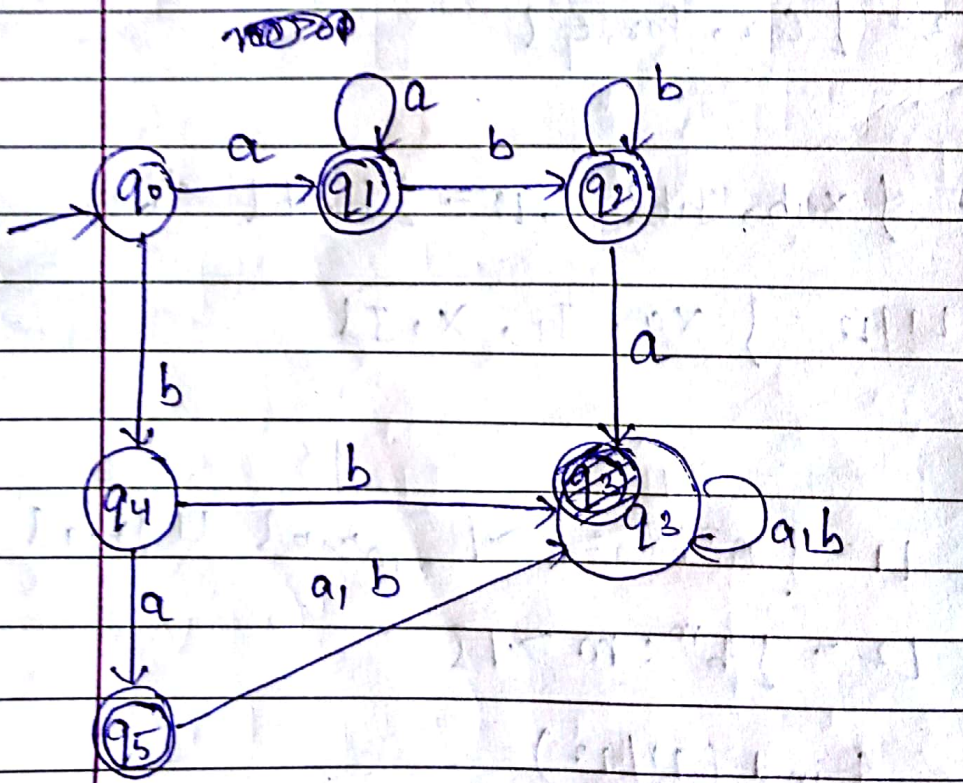
Q. If, $L1 = \{a^n b^m : n \geq 1, m \geq 0\} \cup \{b, a\}$

$L2 = \{b^m : m \geq 1\}$

Find $(L1/L2)$

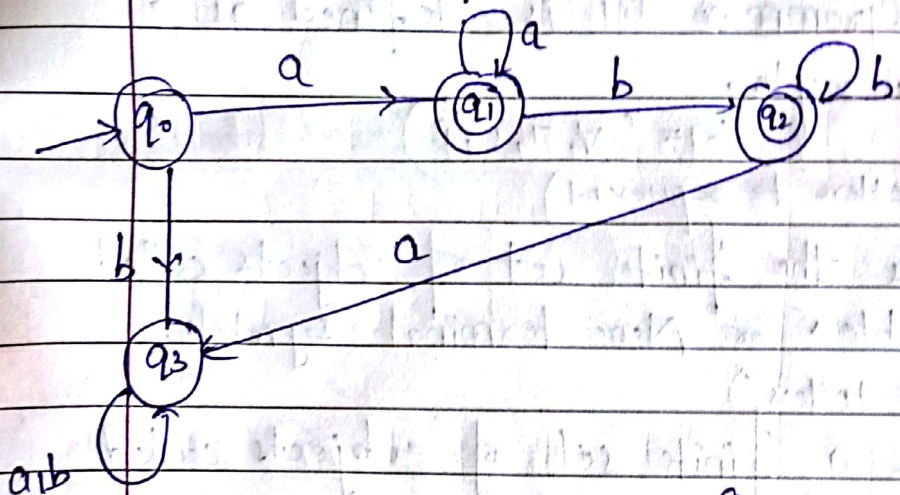
1. Draw finite automata of L1.
2. Try to apply L2 on each state of L1.
3. If, there exist a path to any final state then make that state as a final state.
4). Repeate (3) for all states.
5). At last, all final states received are final states and others are non-final.

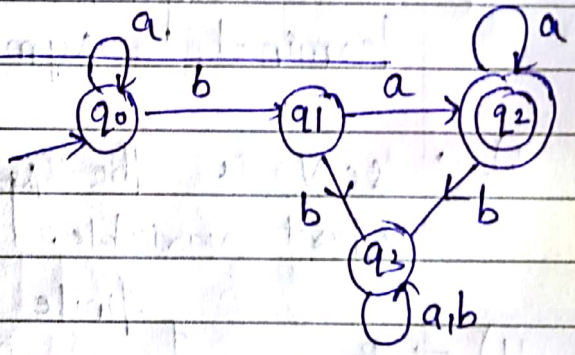
Q. $\{a^n b^m : n \geq 1, m \geq 0\} \cup \{ba\}$



Proof is given → Peter Links

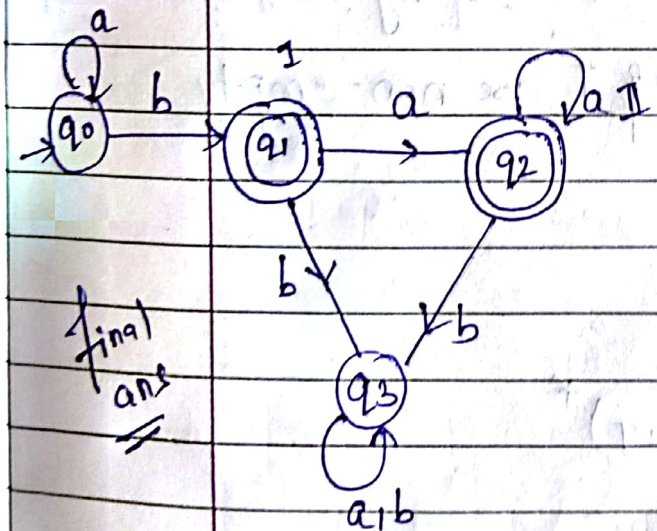
Page No.
Date





Q. $L1 = L(a^*baa^+)$

$L2 = L(ab^*)$



final ans =

$L(M_0) \cap L2 = \phi$

$L(M1) \cap L2 = \{a\} \neq \phi$

$L(M_2) \cap L2 = \{a\} \neq \phi$

$L(M3) \cap L2 = \phi$

$a^*b + a^*baa^+$ ← to reach final state I; → to reach final state II

$= a^*b(\epsilon + aa^+)$

$= a^*baa^+$

Page No.
Date

GRAMMAR :-

→ Called generators of the language.

→ A Grammar G is defined as a quadruple,

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

G (Capital letters to represent)

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

(small letters)

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, a how string is generated

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

$x \rightarrow y$

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

$(V \cup T)^+$

(not epsilon)

* T is the superset of Σ.

Page No.
Date

Eg. $G = \{\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\}\}$

Derivation from a grammar

$w = uxv$

$x \to y$ is applicable to this string.

$z = uyv$

it can be replaced by LHS in derivation.

$w \Rightarrow z$

$w_1 \to w_2 \Rightarrow \cdots \Rightarrow w_n$ ⟶ by applying multiple derivations.

or, $w_1 \overset{*}{\Rightarrow} w_n$

$L(G) = \{ w \in T^* \mid S \overset{*}{\Rightarrow} w \}$

from another string.

If, $w \in L(G)$ then the sequence, $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w$ is a derivation or of the sentence w.

↳ final string

↓ intermediary forms

≡ sentential form.

Page No.
Date

Eg: $G = \{S\}, \{a, b\}, S, P\}$ where,

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

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb = a^n b^n$ $(n \geq 0)$

aSb, aaSbb — Sentential form.

$a^n b^n$ — Sentence.

II. How we convert Grammar to a Language

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

$P = \{ S \to aAb \mid \epsilon$
$A \to aAb \mid \epsilon \}$

$S \Rightarrow aAb \Rightarrow aaAbb \Rightarrow aaaAbbb \dots a^n b^n$ $(n \geq 0)$

Q.2) $G = \{ \{S\}, \{a\}, S, \{S \to SS\} \}$

$S \Rightarrow SS \to SSS \to SSSSS \dots$

$SS \to SSS \to \dots$

$(\lambda)^n ; n \geq 1$

Language is ϕ, ∵ no terminal symbol

Page No.
Date

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

$P = \{S \to aCa,\ C \to (aCa) \mid b\}$

~~$S \Rightarrow aCa \Rightarrow aaCaa \Rightarrow aa$~~

~~$\Rightarrow aacbaa$~~

$S \Rightarrow aCa \Rightarrow aaCaa$

$\Rightarrow abaa \quad aabaa \quad (a^n b a^n) \mid n \geq 1$

Q.4. $S \to aSB$

$S \to aB$

$B \to b$

$S \to aSB \Rightarrow aaSBB$

$aaBB \Rightarrow aabb \quad a^n b^n \mid n \geq 1$

Q.5. $S \to aSb \mid ab$

$S \Rightarrow aSb \Rightarrow aaSbb$

$\to ab \quad a^n b^n \mid n \geq 1$

Page No.
Date

GENERATE GRAMMAR FOR A LANGUAGE

Q1. (All strings of length 2)

~~S → AB~~
~~A~~

S → aa
S → ab
S → ba
S → bb

S → aa | ab | ba | bb.

$\left[\begin{array}{l} S \to AA \\ A \to a \mid b \end{array}\right]$

{ all strings of 2 len
(a+b)(a+b)
A

Q2. $L = \{ a^n \mid n \geq 0 \}$

$\boxed{S \to aS \mid \epsilon}$

* building block for *, $S \to SS \mid \epsilon$ S^*

(3). $L = (a+b)^*$

~~S → ABS | ϵ~~
~~A → a | b~~
~~B → a | b~~

S → aS | bS | ϵ

Page No.
Date

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

$S \to AAB$
$A \to a|b$
$B \to aB|bB|\epsilon$

$S \to ACB|\epsilon$
$A \to a|b$
$B \to a|b$
$C \to ACB|\epsilon$

S-ACB

CACB CEE

(5). $L = \{$ at most 2 length $\}$

$S \to AB|A|B|\epsilon$
$A \to a|b$
$B \to a|b$

$S \to AA$
$A \to a|b|\epsilon$

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

~~$S \to aSb$~~

$S \to aACBb|\epsilon$
$A \to a|b|\epsilon$
$B \to a|b|\epsilon$
$C \to aACBb|\epsilon$

$S \to aBb$
$B \to aB|bB|\epsilon$

$a(a+b)^*b$

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

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

$S \to aBa|bBb|a|b|\epsilon$
$B \to aB|bB|\epsilon$

even length odd length palindrome

Page No.
Date

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

✓ $S \to aSa \mid bSb \mid a \mid b \mid \epsilon$

$S \to aSa \mid bSb \mid \epsilon$ → even length, $S \to aSa \mid bSb \mid a \mid b$ → odd length

2. $L = \{$ Even length Strings $\}$

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

$S \to AS \mid \epsilon$	✓
$A \to BB$	
$B \to a \mid b$	

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

$S \to AB$

$A \to aA \mid a$

$B \to bB \mid b$

4. ~~$a^n b^n c^m \mid n, m \geq 1$~~

$S \to AC$

✓ ~~$A \to aA \mid a$~~ $A \to aAbb \mid ab$

~~$B \to$~~ $C \to cC \mid c$

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

~~$A \to aCb$~~

~~$C \to aCb \mid c$~~

$S \to aSb \mid aAb$

$A \to cA \mid c$

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

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

$S \rightarrow AB$ ✓

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

$S \rightarrow AB$
$A \rightarrow aA \mid a$
$B \rightarrow bbB \mid bb$

$S \rightarrow aSbb \mid abb$ ✓

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

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

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

$A \rightarrow aAb \mid ab$
$S \rightarrow aSc \mid aAc$ ✓

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.

$\alpha \rightarrow \beta$

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

and $\beta \in (V \cup T)^*$

Eg: $aAb \rightarrow bB$

$aA \rightarrow \epsilon$ or $CB \rightarrow DB$

$Bc \rightarrow acB$

- Type 1.

$aAb \rightarrow \beta$

left context

right context of A

Page No.
Date

context free / ϵ $A \epsilon \rightarrow \beta$.

context sensitive / $aAb \rightarrow \beta$
→ only changed to β when A is preceded by 'a' & 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.

$AB \rightarrow AbBc$
$S \rightarrow aSBc$
$S \rightarrow abc$
$cB \rightarrow Bc$
$bB \rightarrow bb$

ϵ cannot be there on RHS. of any production

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

If language is accepting ϵ, then follow this.

• Type 2 :-

$A \rightarrow \alpha$

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

and, $A \in V$

$A \rightarrow e$

$A \rightarrow BeD$

can have many variables on RHS but can have only one variable on LHS that too w/o context.

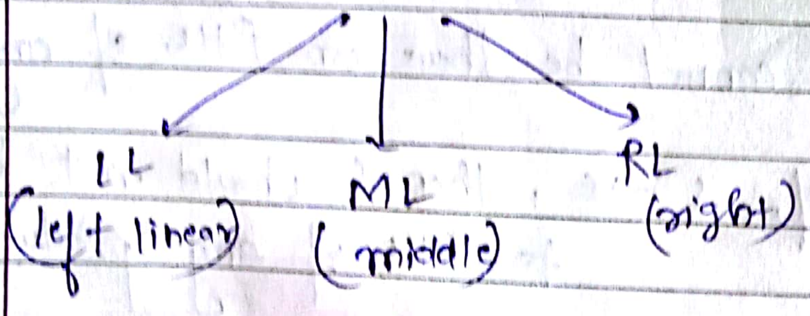
(context is in terms of small a/b)

LINEAR GRAMMAR.

any grammar in which, there exist,

exactly one variable on LHS

& atmost one variable on RHS.



$S \rightarrow aSb$ (ML)

$A \rightarrow aB|b$ (RL)

$S \rightarrow Ba|b$ (LL)

Page No.
Date

Linear Grammar = LL + ML + RL

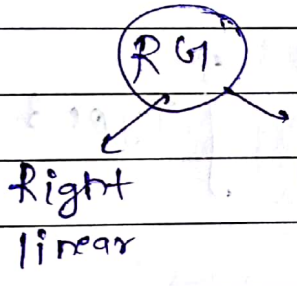
any combination

• Type 3: Regular Grammar

RG = LL + RL

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

R.G. ⊂ L.G.

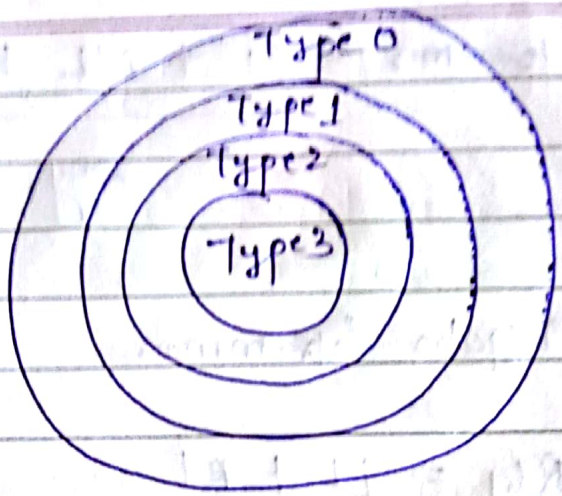


V → T* | T*V V → T* | VT*

Eg. A → xB | x (R.G.L.) where A, B ∈ V

A → Bx | x (L.L) x ∈ Σ* or T*

{ LHS is context free
RHS has only one variable.



[Type 3 ⊂ Type 2 ⊂ Type 1 ⊂ Type 0]

⇒ Highest form of the grammar?

Eg. $S \to Aa$, $A \to c \mid Ba$, $B \to abc$] Type 3

Eg. $S \to ASB \mid d \Rightarrow S \to A\,SB \mid d$ (Type 2)

$A \to aA \to$ Type 3

Combined Type 2

Conversion Of FA to RG

Page No.
Date

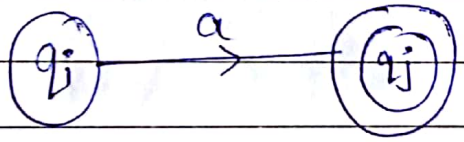
We can convert an automata to a grammar

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

where, P is defined by the following rules.

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

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

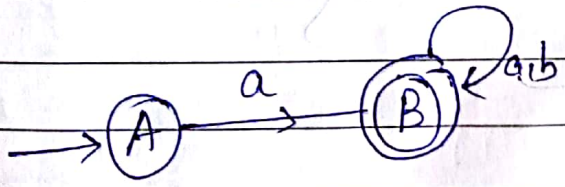
** 

final state

$A_i \rightarrow aA_j$ & $A_i \rightarrow a$

or

$A_i \rightarrow aA_j$ & $A_j \rightarrow \epsilon$

Eg. 

$A \rightarrow aB$
$B \rightarrow aB \mid bB \mid \epsilon$
or
$B \rightarrow a \mid b$

Right linear

$A \rightarrow Ba$
$B \rightarrow Ba \mid Bb \mid \epsilon$

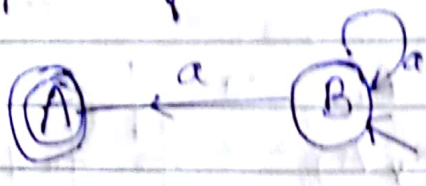
Page No.
Date

Reverse

Automata (L) → L^R → (R.L.G) → (L)

Right linear grammar

Q. To find left linear grammar



$B \rightarrow aA$

$B \rightarrow aB \mid bB$

$A \rightarrow \epsilon$

Now reverse,

$B \rightarrow Aa$

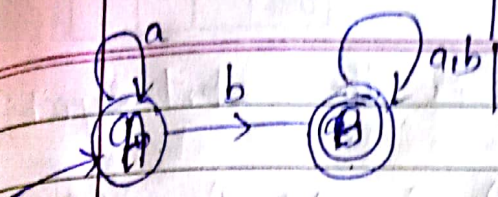
$B \rightarrow Ba \mid Bb$

$A \rightarrow \epsilon$

$$\begin{cases} B \rightarrow Aa \mid Ba \mid Bb \\ A \rightarrow \epsilon \end{cases}$$

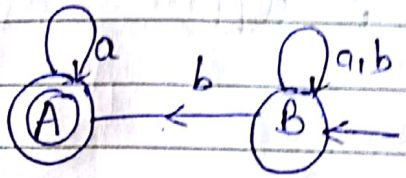
Q. R.G. for,

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



S → A. Starting state
if multiple starting state, choose any one start production from it.

$A \rightarrow aA \mid bB$
$B \rightarrow aB \mid bB \mid \epsilon$ } Right linear.



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

Reverse

$A \rightarrow Aa$
$B \rightarrow Ab \mid Ba \mid Bb$

Regular Grammar to Automata.

(1). Each production $A_i \rightarrow a A_j$ indicates a transition from q_i to q_j with label a.

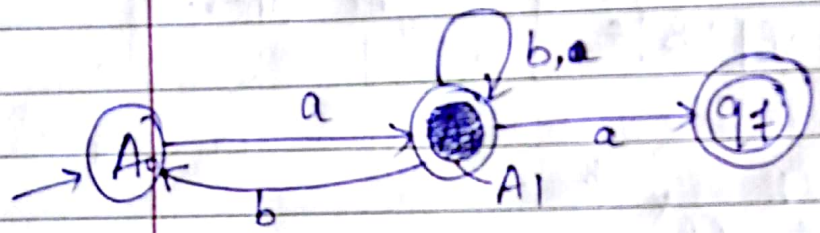
(2). Each production $A_k \rightarrow a$ induces a transition from q_k to q_f with label a where, $q_f \in F$.

(3). Make a new state from A_k to q_f as final with label a.

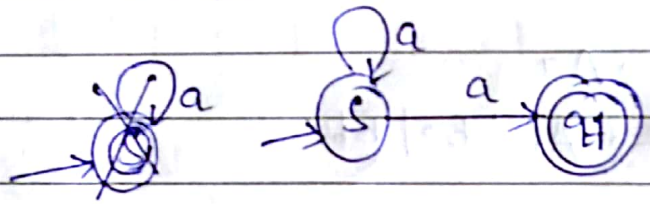
Page No.
Date

Q.1. P is given by. A_0 is the start symbol.

$A_0 \rightarrow aA_1$, $A_1 \rightarrow bA_1$, $A_1 \rightarrow a$, $A_1 \rightarrow bA_0$



Q2. $S \rightarrow as/a$



IMP

* Automata from left linear grammar.

1). Reverse it to get $(R.L.G_1)$

2). Make automata for $(R.L.G_1)$ i.e. automata for L^R.

3). Now, Reverse automata.
i.e. Automata of $(\bar{L})$.

Context Free Grammar or Type-2 Grammar

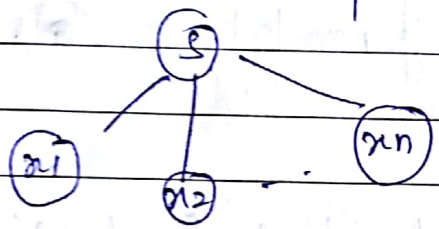
Page No.
Date

Derivation Tree or Parse Tree

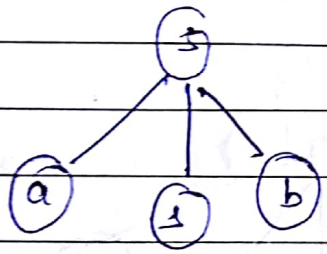
1. Root vertex (starting symbol)
2. Vertex (variables - capital letters)
3. leaves (terminal symbols) or ε
4. If $S \to x_1 x_2 x_3 \dots x_n$ is production rule the the
5. Derivation Tree or parse tree,

$A \to x$
where, $A \in V$ and
$x \in (V \cup T)^*$

Eg. $S \to aSa$
$S \to bSb$
$S \to \epsilon$



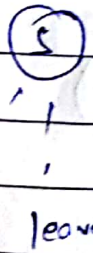
$S \to aSb$



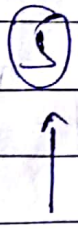
Tree construction

Top Down

(Start from top / move towards the leaves)



Bottom up leaves.
(leaves to root)



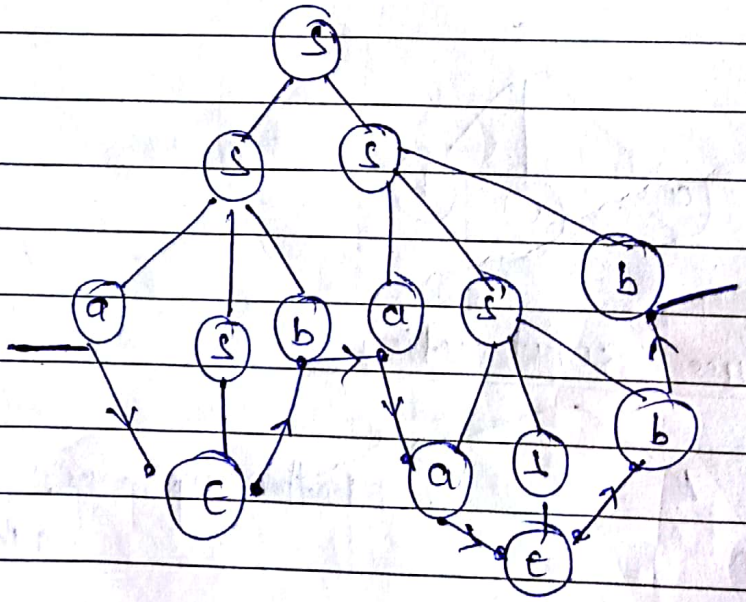
→ Tree is not for grammar.
⇒ It is drawn for a particular string.

- **Yeild 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. $G = \{ \{S\}, \{a, b\}, S, P = S \to SS \mid aSb \mid \epsilon \}$

$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abaaSbb \Rightarrow abaabb$



Yeild :-
abaabb (ignore ϵ)

Left most & right most derivation

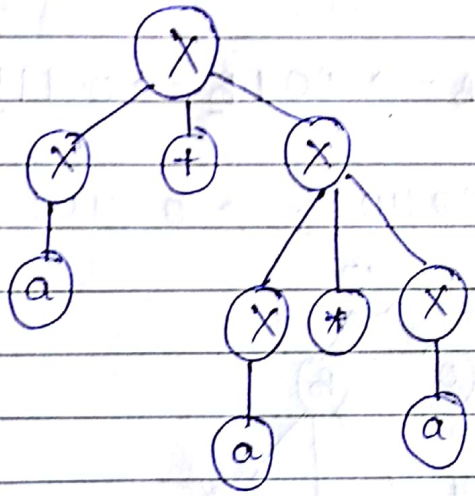
$\Sigma = \{a\}$

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)

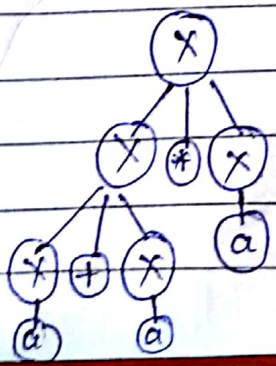


Rightmost

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

or,

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



Page No.
Date

Q. Consider the grammar

$S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

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

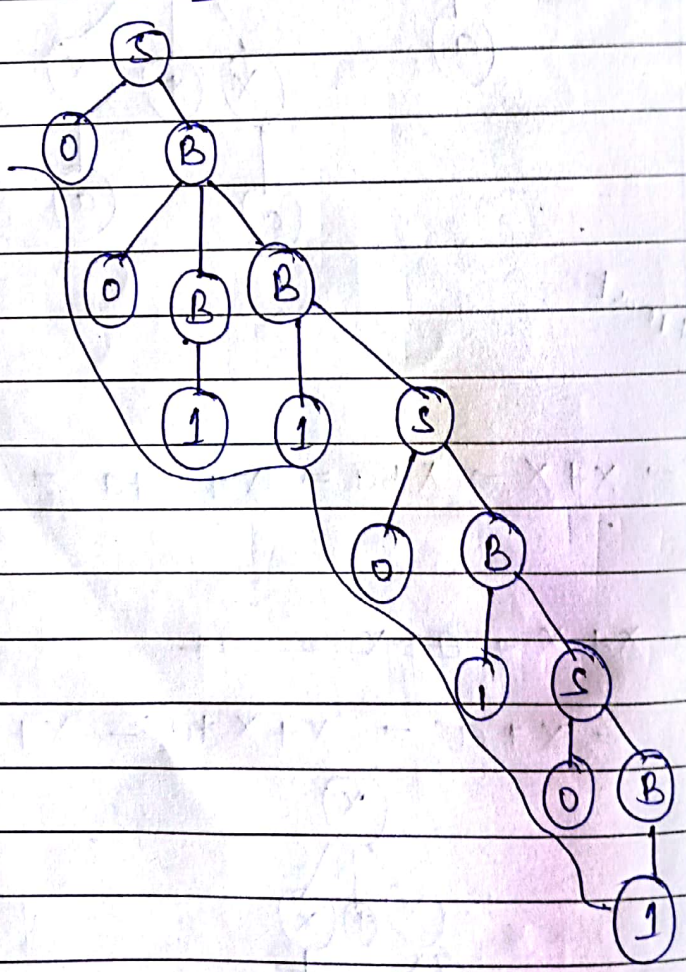
Find left most & right most derivation

String = "00110101"

left most

$S \rightarrow 0B \rightarrow 00BB \rightarrow 001B \rightarrow 0011S \rightarrow 00110B$

$\rightarrow 001101S \rightarrow 0011010B \rightarrow 00110101$



00 1 0 1 0 1

Page No.
Date

right most

S → 0B → 00BA → 00B1 → 00

Page No.
Date

- **Ambiguity in CFG**

$\omega \in L(G)$ — derivation

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

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

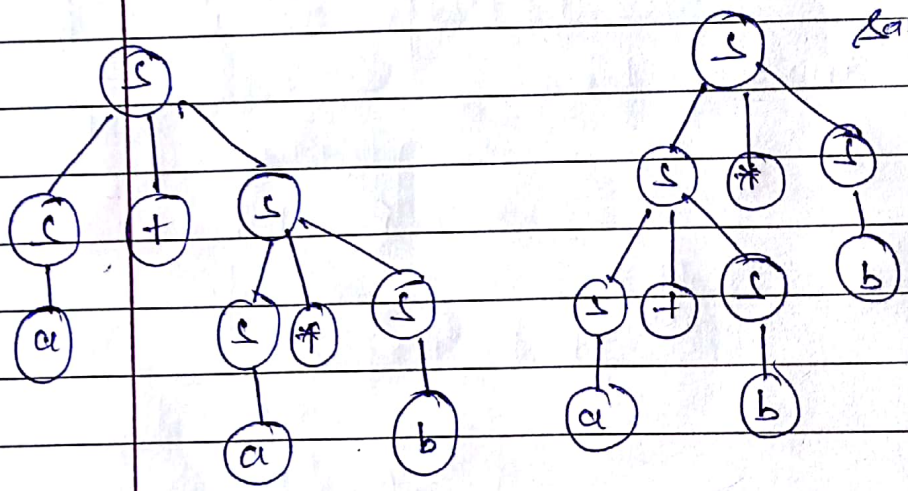
Two left most derivations

$a+a*b$

$S \Rightarrow S+S \Rightarrow a+S \Rightarrow a+S*S \Rightarrow a+a*S \Rightarrow a+a*b$

$S \Rightarrow S*S \Rightarrow S+S*S \Rightarrow a+S*S \Rightarrow a+a*S \Rightarrow a+a*b$

Same string.



- To check ambiguity → use two left most
  or
  two right most

Page No.
Date

Q. S → SbS | a , show that G is ambiguous

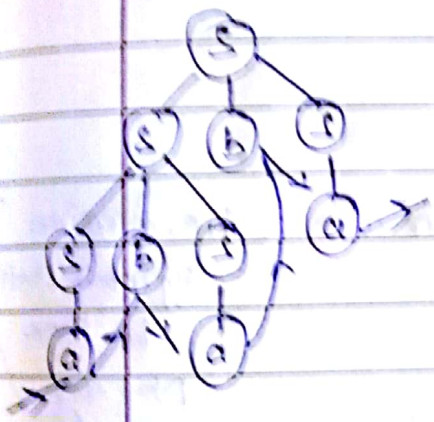
S → SbS → ~~abaaba~~

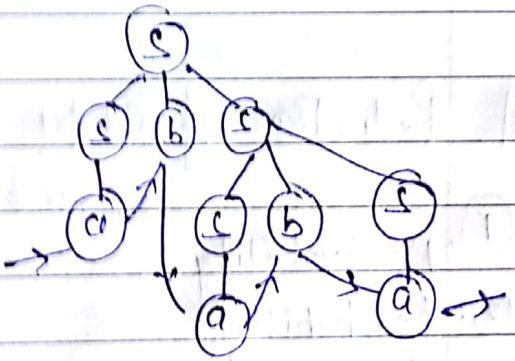
SbSbS → abSbS → ababS → ababa

S → SbS → abS → abSbS → ababS

→ ababa

(same strings)





Simplification Of CFG

Page No.
Date

1. Reduction of CFG
2. Removal of unit productions
3. Removal of Null productions.

unit production eg:-

$A \to B$
$B \to C$
$C \to a$
} unnecessary
→ only
$A \to a$

null productions

$A \to \epsilon$
$B \to \epsilon$

Reduction of CFG

→ what is our aim (i.e. terminal symbols should be there Not only variable to variable)

1. Derivability
2. Reachability
   (if ~~terminal~~ is reachable from start
   ↓
   that state

Eg. $S \to Aa$
$A \to a$
$B \to CD$
$C \to D$
} Here
x } are
x } not
reachable
from S.

Page No.
Date

Q. $P = S \to Ac \mid B$
$A \to a$
$C \to c \mid Bc$
$E \to aA \mid e$

$T = \{a, c, e\}$

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

check derivability

$W_1 = \{a, A, C, E\}$ — set that generate terminal symbol.

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

all those variables whose RHS is in $\{W_1 \cup T\}$

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

$W_3 = W_2 \cup \phi$ → stop.

⇒ remove all production of B.

$P =$ $S \to Ac$
$A \to a$
$C \to c$
$E \to aA \mid e$

check reachability

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

$V_2 = \{A, S, C\}$

↓

$V_3 = \{S, A, C, a\}$

↓

Here, C & E are not reachable.

Reduced Grammar =

$$\begin{bmatrix} S \to Ac \\ A \to a \end{bmatrix}$$

Q. $S \to ABC \mid BaB$

$A \to aA \mid BaC \mid aaa$

$B \to bBb \mid a$

$C \to \epsilon A \mid AC$

$T = \{a, b\}$

$W_1 = \{A, B\}$

$W_2 = \{\{A, B\}, S\}$

$S \to BaB$

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

$B \to bBb \mid a$

Reachability

$\{S\} \to \{S, B\}$ (a)

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

($S \to BaB$, $B \to bBb \mid a$)

Page No.
Date

Q. $S \rightarrow aAa$
$A \rightarrow Sb \mid bCC \mid DaA$
$C \rightarrow abb \mid DD$
$E \rightarrow aC$
$D \rightarrow aDA$

$T = \{a, b\}$

$\pi_1 = \{abb\ C\}$

$\pi_2 = \{C, A, E\}$

$\pi_3 = \{S, C, A, E\}$

$S \rightarrow aAa$
$A \rightarrow Sb \mid bCC$
$C \rightarrow abb$
$E \rightarrow aC$

$\{S\} \rightarrow \{S, aA\} \rightarrow \{S, aA, bC\}$

$\rightarrow \{S, aA, bC\}$

E not reachable.

P = $S \rightarrow aAa$
$A \rightarrow Sb \mid bCC$
$C \rightarrow abb$

Page No.
Date

Elimination of unit Productions

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

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

1. Not Increasing the length
2. Not introducing the terminal symbol to the length.

Rules: To remove $A \to B$, add

1. Production, $A \to X$ to P where, $B \to X$ occurs in the grammar $[x \in (T \cup \epsilon)]$ (ε → epsilon.)
2. Delete $A \to B$ from G
3. Repeat Step 2 untill all unit Productions are removed.

Q.

$S \to XY$
$X \to a$
$Y \to Z \mid b$
$Z \to M$
$M \to N$
$N \to a$

$M \to a$
$S \to XY$
$X \to a$
$Y \to M \mid b$

$S \to XY$
$X \to a$
$Y \to a \mid b$

Page No.
Date

ELIMINATION OF ∈-PRODUCTIONS.

(1). find all null productions [A → ∈]

(2). find all nullable variables, which are moving towards ∈ i.e. directly or by applying various derivations i.e.

X → ∈

or, X ⇒ () ⇒ () ⇒ () ⇒ ∈

We can't remove all ∈ moves.

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

Q. S → aSb | aAb

A → ∈

Nullable set = {A, S}

S → aSb | aAb | ab
↳ A has been removed.
to remove A → ∈.

∵ A is not reachable.

S → aSb | ab

Page No.
Date

Q. $S \rightarrow AbaC$
$A \rightarrow BC$
$B \rightarrow b \mid \epsilon$
$C \rightarrow D \mid E$
$D \rightarrow d$

Nullable set = { A, B, C }

$S \rightarrow AbaC \mid baC \mid Aba \mid ba$
$D \rightarrow d$ ✓ , $A \rightarrow BC \mid B \mid C$
$C \rightarrow D$ ✓
$B \rightarrow b$ ✓

~~$S \rightarrow Abad \mid bad \mid Aba \mid ba$~~
~~$A \rightarrow$~~

Q. $S \rightarrow ASA \mid aB \mid b$
$A \rightarrow B$
$B \rightarrow b \mid \epsilon$

nullable = { A, B, S }

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

↓ now can be reduced.

NORMALS FORMS OF CFG

Page No.
Date

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

CNF

- Length of RHS
- and natur of symbols on the RHS.

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

$A \to a$ or $A \to BC$ and $S \to \epsilon$ if $\epsilon \in L(G)$. When ϵ is in $L(G)$, we assume that S, does not appear on the RHS of any production.
Here, $(A, B, C) \in V$ & $a \in T$.

Eg. ①. $S \to AB \mid \epsilon$

$A \to a$

$B \to b$

②. $S \to As \mid AAs$ — ③

$A \to Ia \mid aa$ — ②

✗ Not chomsky

If $\epsilon \notin L(G)$ then for CNF all functions are of

$A \to a$ or, $A \to BC$

How to Convert CFG to CNF

(i) If start symbol occur on RHS of some production, create a new start symbol S' and a new production

S' → S [imp in case of language ε]

* unit production
* remove all null productions

[after that, we'll apply it in CFG]

(ii). Remove all null productions.

(iii). Remove all unit productions.

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

(v). If RHS of any production is in the form $A \to aB$, where a is the terminal & $A, B \in V$, then the production is replaced by,

$A \to XB$

$X \to a$

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

$A \to aB$.

Q. $S \to asb$

$S \to ab$

we can ignore for this S should not come to RHS

$A \to a$

$B \to b$

$S \to ACB$

$C \to AB$

$S \to AX \mid AB$ ✓ (CNF)

$X \to CB$.

$A \to a$

$B \to b$

Q. $S' \to S$

$S \to asb$

$S \to ab$

$S' \to asb \mid ab$

$S \to asb \mid ab$

$A \to a$

$B \to b$

$S' \to ASB \mid AB$

$S \to ASB \mid AB$

$S' \to AX \mid AB$ ✓

$X \to SB$

$S \to AX \mid AB$

$A \to a$

$B \to b$

Page No.
Date

Q. $S \to ASA|aB$
$A \to B|S$
$B \to b|\epsilon$

$\{B\}$
A

$S \to ASA|SA|AS|aB|a|S$
$A \to B|S$
$B \to b$

$S \to ASA|SA|AS|aB|a|S$
$A \to b|S$, $B \to b$

$S \to ASA|SA|AS|aB|a|S$
$A \to b$
$A \to S$, $B \to b$

$S \to ASA|SA|AS|aB|a|S$
$A \to ASA|SA|AS|aB|a|S$
$A \to b$, $B \to b$

$S \to AX|SA|AS|YB|a$ ✓
$A \to AX|SA|AS|YB|a$ ✓
$A \to b$
$X \to SA$
$Y \to a$
$B \to b$

GREIBACH NORMAL FORM (GNF)

Page No.
Date

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

$A \to a\alpha$,

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

No restriction on length.

(terminal = three variables then any)

$S \to aAB|\epsilon$
$A \to bC$
$B \to b$, $C \to \alpha$ is in GNF

Context free to GNF :-

1. If start symbol S occurs on some RHS create a new start symbol S' and add a new production.
   $S' \to S$ (in ϵ case)
2. Remove null productions.
3. Remove unit productions.
4. Remove all direct & indirect left recursion.
5. Do Proper Substitutions of productions to convert it into the proper form of GNF.

Page No.
Date

1. $S \to AB$
$A \to aA \mid bB \mid b$
$B \to b$

$S \to aAB$
$S \to bBB$
$S \to bB$
$A \to aA \mid bB \mid b$
$B \to b$

2. $S \to aAbB \mid aa$

$A \to a$
$B \to b$

$S \to aBSB \mid aA$
$A \to a$
$B \to b$
..

3. $S \to AB$
$A \to BS \mid a$
$B \to SA \mid b$
} CNF form

* If & type is not CNF convert it to CNF

Step1: Rename the variables in order $A_1, \dots, A_n$ starting symbol is the first.

$A_1 \to A_2 A_3$
$A_2 \to A_3 A_1 \mid a$
$A_3 \to A_1 A_2 \mid b$

Step2: Check for,
$A_i \to A_j \beta$ (such that $i \geq j$)

$A_3 \to A_2 A_3 A_2 \mid b$

$A_3 \to A_3 A_1 A_3 A_2 \mid b \mid aA_3A_2$ (Substitute A_2 completely)

** left recursion

formula: to convert left recursion to right recursion.

If $A \to A\alpha \mid \beta$ then α, convert it to,

$A \to \alpha A$ ~ right recursion. (it is in GNF)

$A' \to \alpha A' \mid \alpha$
$A \to \beta A' \mid \beta.$

$A_3 \to A_3A_1A_3A_2 \mid aA_3A_2 \mid b$
(α = $A_3A_1A_3A_2$, β = $aA_3A_2 \mid b$)

$A' \to A_1A_3A_2A' \mid A_1A_3A_2$
$A_3 \to aA_3A_2A' \mid bA' \mid aA_3A_2 \mid b$

A_1, A_2 & A' are not in GNF

Page No. / Date

Ques

$A_2 \rightarrow aA_3A_2A'A_1 \mid bA'A_1 \mid aA_3A_2A_1 \mid bA_1 \mid a$

$A_1 \rightarrow aA_3A_2A'A_1A_3 \mid bA'A_1A_3 \mid aA_3A_2A_1A_3 \mid bA_1A_3 \mid aA_3$

$A' \rightarrow aA_3A_2A'A_1A_3A_3A_2A' \mid bA'A_1A_3A_3A_2A' \mid aA_3A_2A_1A_3A_3A_2A' \mid bA_1A_3A_3A_2A' \mid aA_3A_3A_2A'$ (same) with last A_3A_2.

Q. $S \rightarrow AA \mid a$

$A \rightarrow SS \mid b$

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow A_1A_1 \mid b$

(α = A_1, β = b)

$A' \rightarrow A_1A' \mid A_1$

$A_1 \rightarrow bA' \mid b$

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow bA' \mid b$

$A' \rightarrow A_1A' \mid A_1$

Page No. / Date

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow A_1A_1 \mid b$

$A_2 \rightarrow A_2A_2A_1 \mid b \mid aA_1$

(α = A_2A_1, β = $b \mid aA_1$)

$A' \rightarrow A_2A_1A' \mid A_2A_1$

$A_2 \rightarrow bA' \mid aA_1A' \mid b \mid aA_1$

$A_1 \rightarrow bA'A_2 \mid aA_1A'A_2 \mid bA_2 \mid aA_1A_2 \mid a$

$A' \rightarrow bA'A_1A' \mid aA_1A'A_1A' \mid bA_1A' \mid aA_1A_1A' \mid bA'A_1 \mid aA_1A'A_1 \mid bA_1 \mid aA_1A_1$

PUMPING LEMMA FOR CFL (context free language)

$A_1 \rightarrow A_2 A_2 | a$

$A_2 \rightarrow A_1 A_1 | b$

$A_2 \rightarrow A_2 A_2 A_1 | b | a A_1$

α β

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

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

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

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

Pumping Lemma for CFL (context free language)

Let 'L' be a context-free language. Then we 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. $|vx| \geq 1$
3. $|vwx| \leq n$

④. $uv^k wx^k y \in L$ for all $k \geq 0$.
Here, n is called the pumping length.

* Prof (by yourself)

[Here, q' loops for v of n.]

Eg. Show that, $L = \{a^p \mid p \text{ is prime}\}$ is not a CFL.

①. Let, $L = L(G1)$ 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,
$z^k = uv^k wx^k y$ for, $k = 0$ as, $k \geq 0$, we get,
$z^0 = uv^0 wx^0 y = uwy \in L$ so,
$|uwy|$ is a prime no. say (q)
Let, $|vx| = r$ so,

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

$\Rightarrow |z^q| = |uwy| + q \times |vx|$

$= q + q \times (r) = q(r+1)$

$\because$ it is prime.

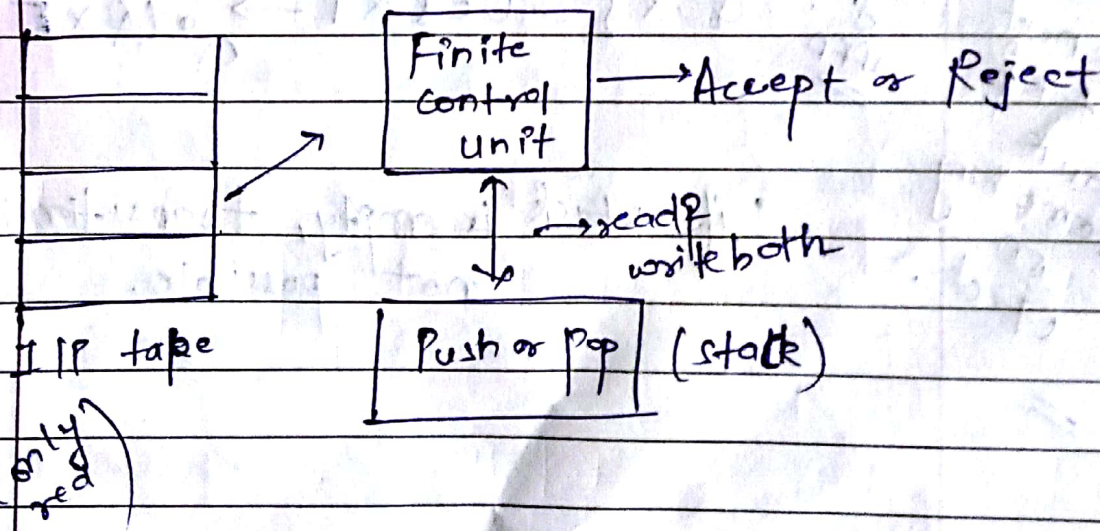
$\therefore q(r+1)$ is not a prime.

Hence, failed.

Push Down Automata (PDA)

PDA = "FA" + Stack

1. An I/P 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
$\Sigma \rightarrow$ i/p symbol
$\delta \rightarrow$ Transistor function
$q_0 \rightarrow$ initial state
$z_0 \rightarrow$ called initial symbol denoted by z_0
$F \rightarrow$ set of final status ($F \subseteq Q$)
$\Gamma \rightarrow$ stack alphabet

or,

{F is a subset of Q.

PDA

Deterministic — Non-Deterministic

DPDA

$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow (Q \times \Gamma^*)$

leads to next state & write something on stack.

NDPDA

$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$

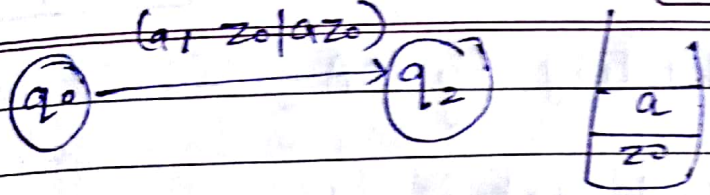
read stack (mandatory)

- if stack is empty, transition is not possible.

Page No.
Date

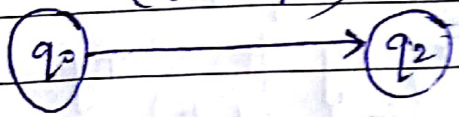
Push → i/p → top → new top

$(q_0, z_0 | a z_0)$



or pop

$(q_0, a z_0 | z_0)$



Instantaneous Description (ID)

$[q, w, s]$ where,

$q \rightarrow$ is the state

$w \rightarrow$ unconsumed i/p (what is left right now)

$s \rightarrow$ Stack contents.

$\delta(q_1, aba, az_0) \Rightarrow (q_2, ba, z_0)$

$\rightarrow (q_3, a, bz_0)$

Turnstile Notation $\Rightarrow$ (we can also replace instead of push or pop.)

$(p, aw, Tb) \vdash (q, w, ab)$

$\vdash^*$

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

$(q_1, aba, az_0) \vdash (q_2, ba, z_0)$

Page No.
Date

Acceptability by PDA

① By Final State :-

$$L(PDA) = \{ w \mid (q_0, w, I) \vdash^* (q, \epsilon, x), \; q \in F \}$$

for any i/p string w.

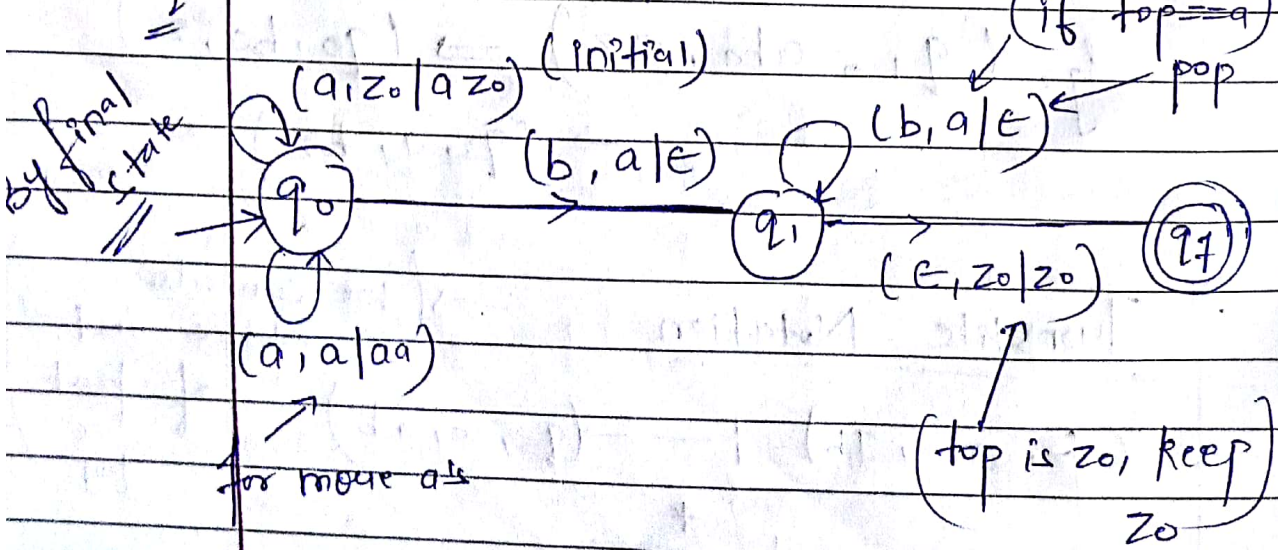
⇒ (q is final state)

② Empty stack :-

$$L(PDA) = \{ w \mid (q_0, w, I) \vdash^* (q, \epsilon, \epsilon), \; q \in Q \}$$

(Here, q may not be a final state.)

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



$\delta(q_0, a, z_0) = (q_0, a, az_0)$

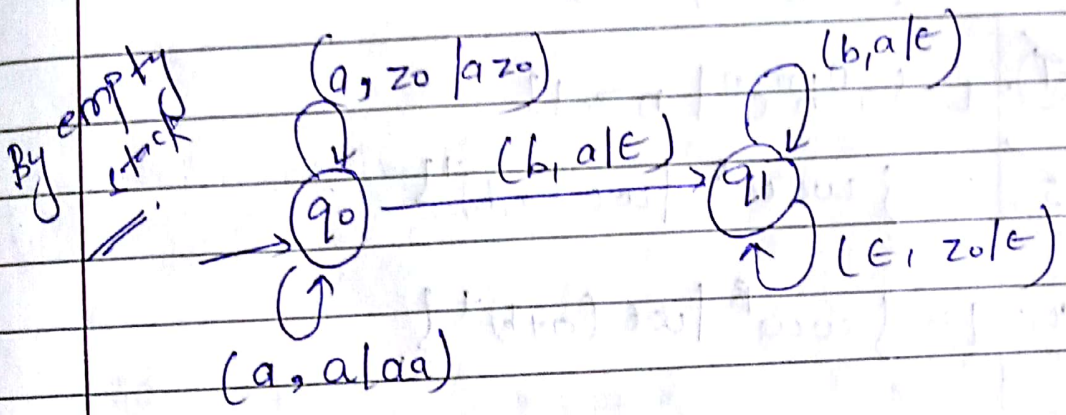
$\delta(q_0, a, a) = (q_0, aa)$

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

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

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

By empty stack



* By default NDFA

Page No.
Date

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

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

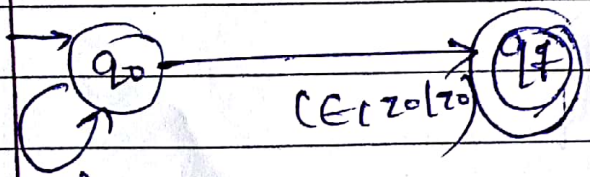
7. $L = \{ a^n b^n c^n \mid n \geq 1 \}$

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

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

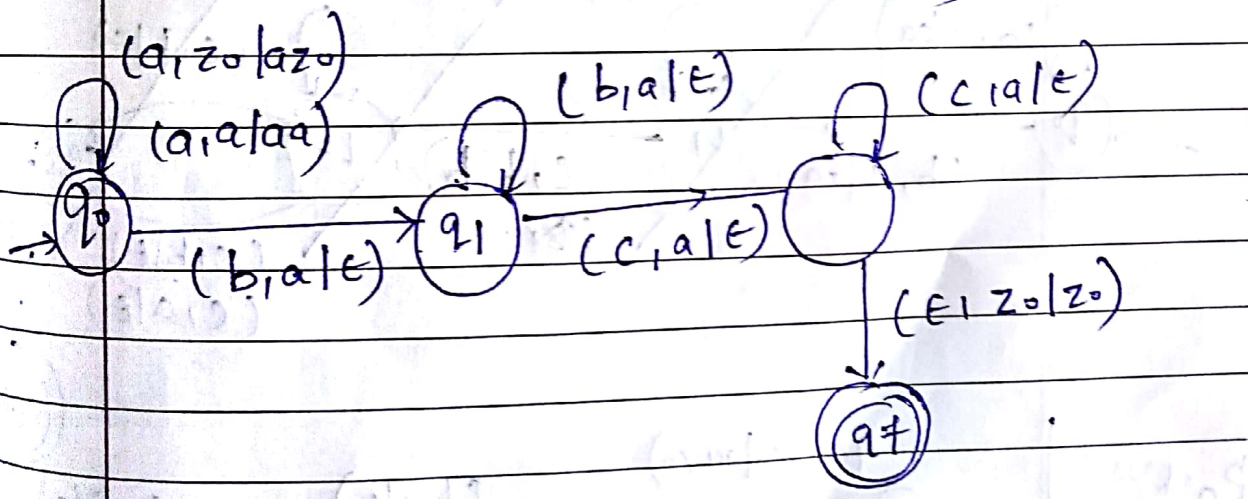
10. $L = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^{2n} \mid n \geq 1 \}$

1.

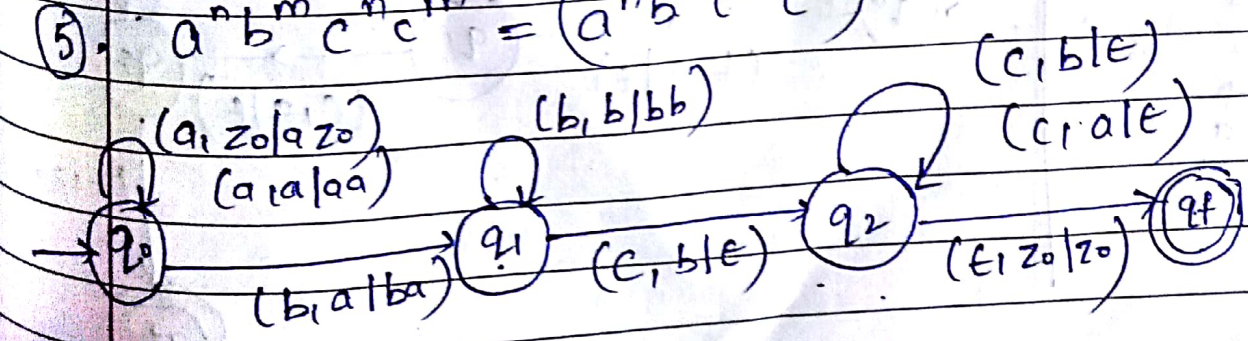


(4). $a^{m+n} b^m c^n$

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

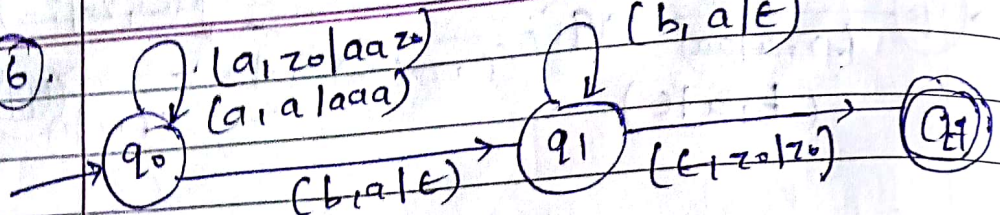


(5). $a^n b^m c^n c^m = (a^n b^m c^m c^n)$

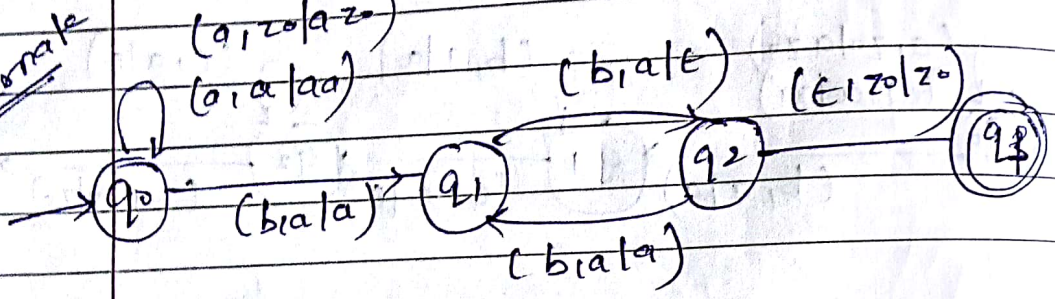


Page No.
Date

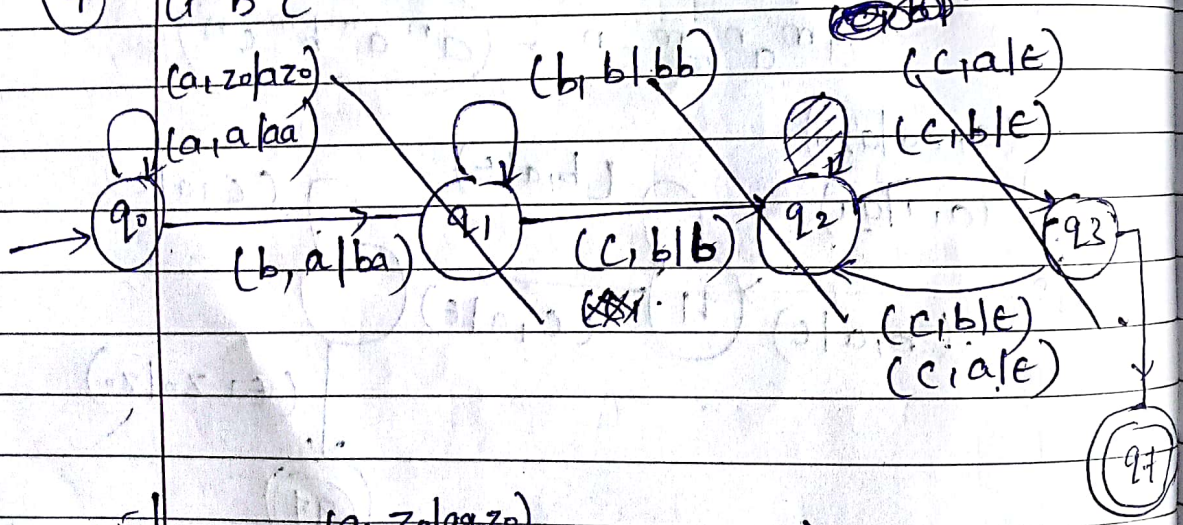
6.



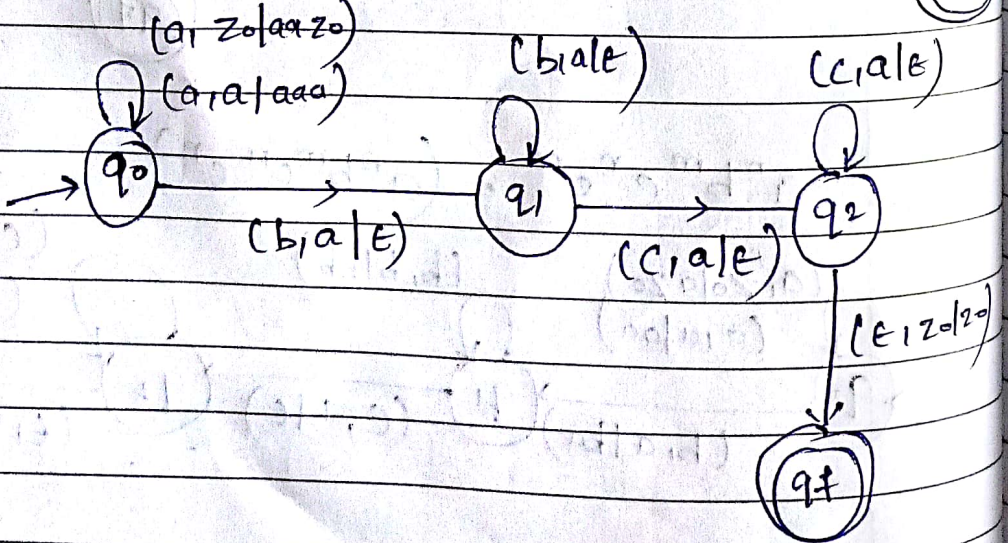
alternate



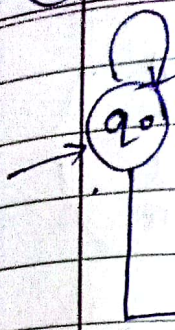
7 $a^n b^n c^n$



$a^n b^m$



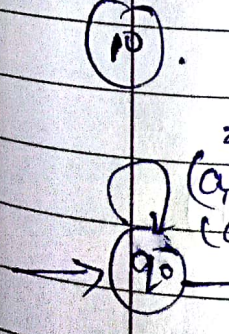
8.



(c, b | b
(c, a | a

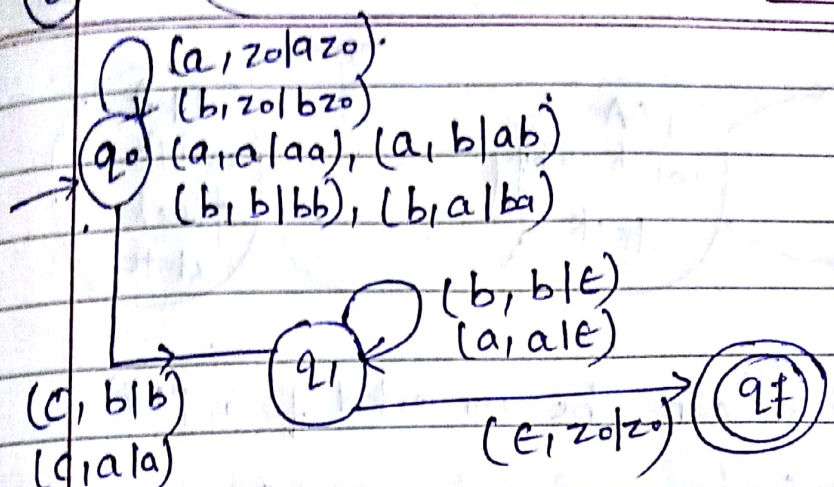
9.

10.

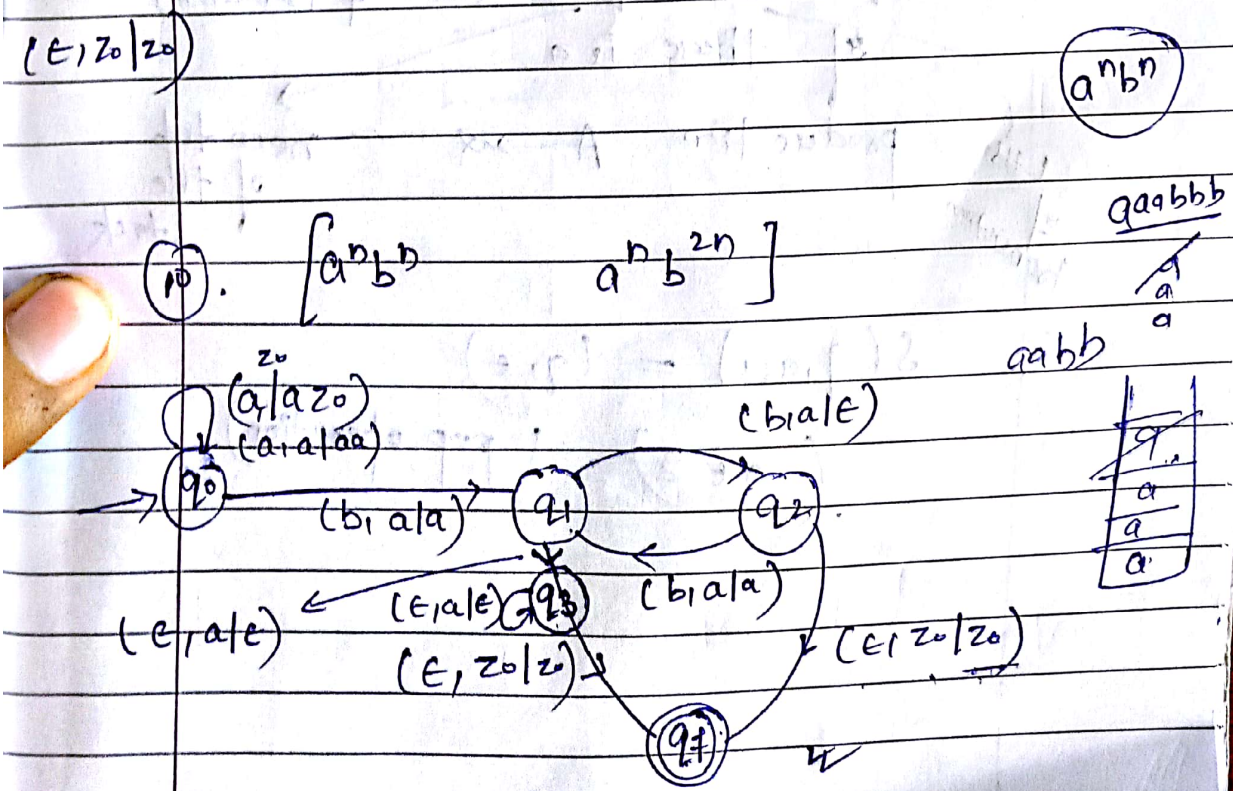


(ε, a | ε

8. (wcw^R)



9. (wcw^R)



10. $[a^n b^n \quad a^n b^{2n}]$

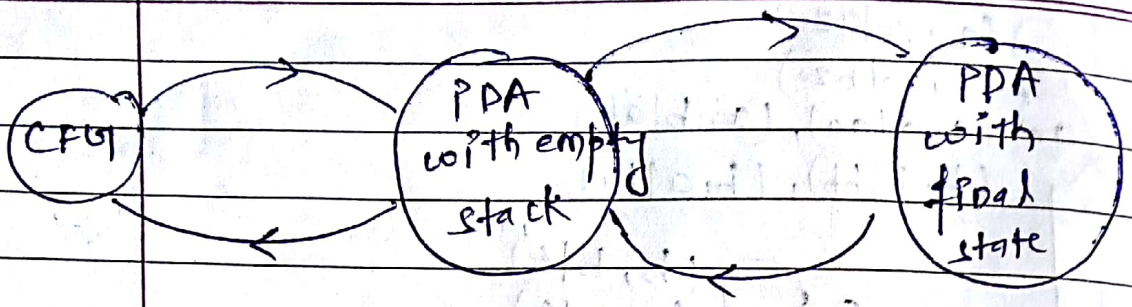
$a^n b^n$

aaabbb

aabb

CFG to PDA Conversion

Page No.
Date



A CFG can be converted to a PDA

$M = \{Q, \Sigma, \Gamma, \delta, q, S, \phi\}$ → Non final states.

- Q → states
- Σ → I/P symbol
- Γ → stack symbol $\in (V \cup T)$
- q → start state
- S → Initial stack variable ($\equiv Z_0$)
- ϕ → No final states

* since converting to a PDA with empty stack. ∴ No final state is req.

(I). $\delta(q, \epsilon, A) = (q, \alpha)$ ← (RHS of transition)

↑ Top of the stack

if there is a production $A \to \alpha$

LHS of transition

new top of the stack

(II). $\delta(q, a, a) = (q, \epsilon)$

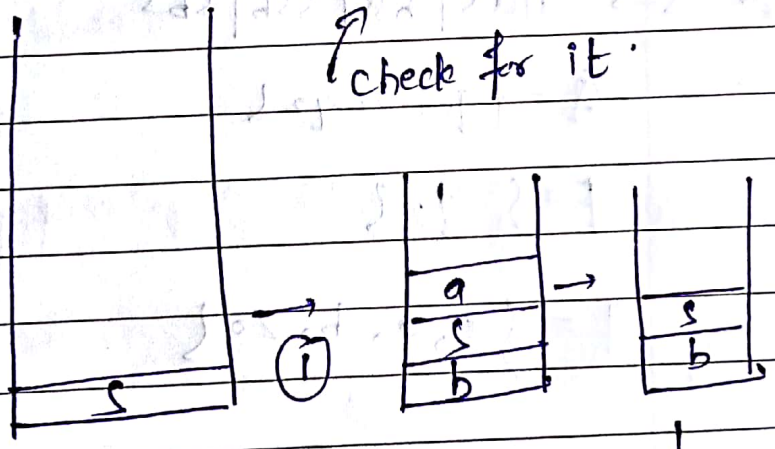
$(a \in \Sigma)$; pop operation

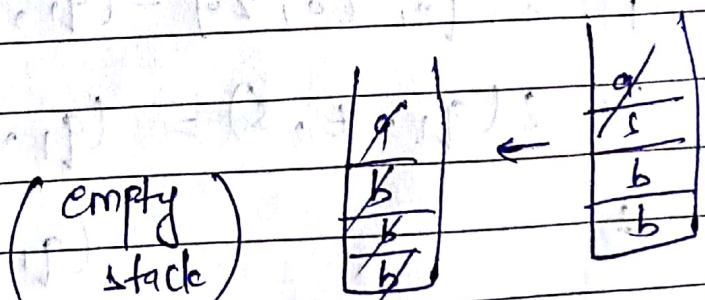
$S \rightarrow aSb$
$S \rightarrow a$

$\Gamma = \{S, a, b\}$

1. $\delta(q_1, \in, S) = (q_1, aSb)$
2. $\delta(q_1, \in, S) = (q_1, a)$ ← write for all variables & terminals.
3. $(q_1, a, a) = (q_1, \in)$
4. $\delta(q_1, b, b) = (q_1, \in)$

(aaabbb) ← check for it.





after consuming whole string.

Page No.
Date

Acceptance by final state

1. $\delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$
2. $\delta(q_1, \epsilon, A) = (q_1, \alpha)$ for all $A \to \alpha$
3. for every $a \in T$, we define,
   $\delta(q_1, a, a) = (q_1, \epsilon)$
4. $\delta(q_1, \epsilon, z_0) = (q_2, z_0)$, $q_2 \in F$

Eg: $S \to a|aS|bSS|SSb|SbS$

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

$F = \{q_2\}$

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

1. $\delta(q_0, \epsilon_0, z_0) = (q_1, Sz_0)$
2. $\delta(q_1, \epsilon, S) = (q_1, a)$
3. (q_1, aS)
4. (q_1, bSS)
   (q_1, SSb)
   (q_1, SbS)

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

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

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$ $q_2 \in F$.

Check (abbaa) =

$(q_0, abbaa, z_0)$

$\vdash$

$(q_1, abbaa, Sz_0)$

$\vdash$

$(q_1, abbaa, SbSz_0)$

$\vdash$

$(q_1, abbaa, abSz_0)$

$\vdash$

$(q_1, \bullet bbaa, \bullet bSz_0)$

$\vdash$

(q_1, baa, Sz_0)

$\vdash$

$(q_1, baa, bSSz_0)$

$\vdash$

(q_1, aa, SSz_0)

$\vdash$

(q_1, aa, aSz_0)

$\vdash$

(q_1, a, Sz_0)

$\vdash$

(q_1, a, az_0)

$\vdash$

$(q_1, \bullet, z_0)$

$\vdash$

(q_2, z_0)

CFG in GNF to PDM.

Page No.
Date

empty stack

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

Here, $\Gamma = V$ only (only variable, no terminal)

and S is on the Top of the stack.

(i) If $A \to a\alpha$ is a production, where, $a \in T$ & $\alpha \in V^*$ then,

(i) $\delta(q, a, A) = (q, \alpha)$

Top of the stack

initially, S is on the top of the stack.

Eg. $S \to aSS$

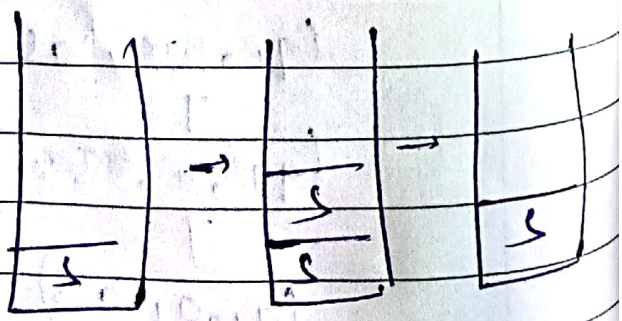
$S \to b$

$\Gamma = \{S\}$

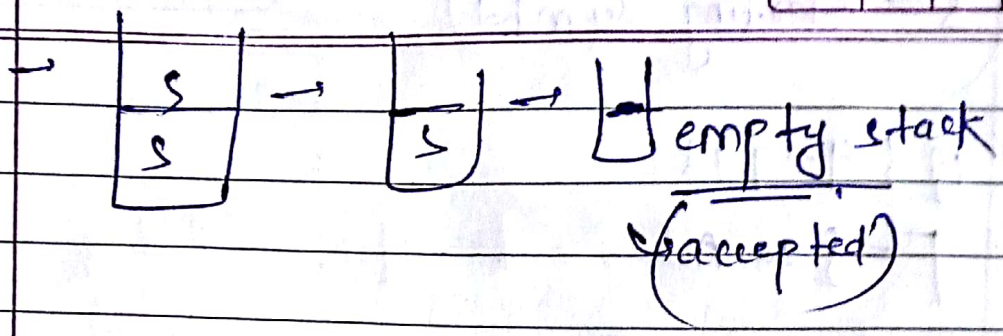
(1). $\delta(q, a, S) = (q, SS)$

(2). $\delta(q, b, S) = (q, \epsilon)$

"abb"



Date



PDA to CFG conversion

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$ be a PDA accepting by empty store,

construct a CFG.

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

$L(G) = L(PDA)$

$\delta(q_1, \epsilon, Z_0) \longrightarrow (q_2, Z_0)$

* to change final state to empty stack form.

(q_1, ϵ)

$(q, p \in Q)$ → states of pushed down automata.

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

$V = S \cup \{ [q, A, p] \}$ → Triplet

states

A → push down symbol

If $|Q| = k$ and $|\Gamma| = m$

then, no. of variables,

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

T = Terminals (⊆ Σ)

S = starting symbol

Page No.
Date

Eg. Q {q0, q1}

Γ = {z0, A}

δ: [q0, z0, q0], [q0, z0, q1], [q0, A, q0], [q0, A, q1], =

S → [q0, z0, p] for each p in Q.

fixed.

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

(1). If, δ(q, x, A) = (p, B1 — Bm)

where x ∈ (ε ∪ Σ)

then we write a production. (should be same)

[q, A, q_{m+1}] → x [p, B1, q2] [q2, B2, q3] [q3, B3, q4] — [qm, Bm, q_{m+1}]

any state — any state — end with q_{m+1}

(2). If, δ(q, x, A) = (p, ε)

[q, A, p] → x

Page No.
Date

M = {{q0, q1}, {0, 1},

{q0, X}, δ, q0, z0, ϕ}

(1). δ(q0, 0, z0) = (q0, X z0)

(2). δ(q0, 0, X) = (q0, XX) } Push

(3). δ(q0, 1, X) = (q1, ε)

(4). δ(q1, 1, X) = (q1, ε)

(5). δ(q1, ε, X) = (q1, ε) } pop

(6). δ(q1, ε, z0) = (q1, ε)

S: [q0, z0, q0], [q0, z0, q1]

[q1, z0, q0],

[q1, z0, q1],

[q0, X, q0]

(1). S → [q0, z0, q0]

(2). S → [q0, z0, q1]

(3). [q0, X, q1] → 1

(4). [q1, X, q1] → 1

(5). [q1, X, q1] → ε

(6). [q1, z0, q1] → ε

Pop

$\delta(q_1, x, A) = (p, \epsilon)$

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

Page No.
Date

[sub for $\delta(q_0, 0, z_0) = [q_0, x z_0]$

1) $[q_0, z_0, q_0] \to 0\,[q_0, x, q_0][q_0, z_0, q_0]$

2) " q_0 " q_1 z_1 " q_0

3) " q_1 " q_1 z_1 " q_1

4) " q_1 " q_0 z_0 " q_1

similarly for $\delta(q_0, 0, x) = (q_0, xx)$

[Number all the triplets and reduce it]

↓ reachability, derivability } etc.

$[q_0, z_0, q_0]$

$[q_0, z_0, q_1]$

$[q_0, x, q_1]$

$[q_1, x, q_1]$

$[q_1, z_0, q_1]$

$[q_0, x, q_0]$

$a_1 = 2 \quad a_2 = 3$

$N = 221$

$M1 = 221/17 = 13$ $\quad m_1 = 17$

$M2 = 221/13 = 17$ $\quad m_2 = 13$

$\text{inv } M_2 = \text{inv}(17, 13)$

$(13, 17)$ ✓

$\gcd(17, 13) = \gcd(r, 0)$

$(13, 17) \quad (0, r)$ $\quad 13/17 = 0$

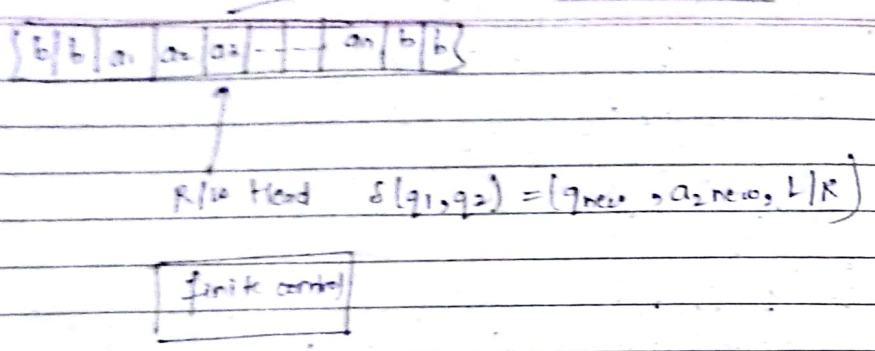
$13/17$

$(13,$

1	3
2	5
3	7
4	11

wrong

TURING M/C



- 7Tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$

1. Q 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 I/P every thing is blank on the tape.
4. Σ is a non-empty set of I/P symbols and is a subset of Γ and $b \notin \Sigma$
5. $\delta(q, x) = (q, y, D)$ $D = L/R$ → left, → Right [move either left or right]
6. $q_0 \in Q$ is the initial state.
7. $F \subseteq Q$ is the set of final states.

$S \to AB$
$A \to A1|0$
$B \to 2B|3$

$0 1^n 2^n 3$

$S \to AB$
$A1B$
$A11B$
$A112B$
$A1123$
01123