Page No.
Date

Theory of Computation

Unit-1

* **Finite Automata : (FA)**

~~Finite fm~~ A finite Automata can be represented by a 5-tuple. $(Q, \Sigma, \delta, q_0, F)$ where

(i) Q = A finite ~~known~~ non-empty set of states.

(ii) Σ = Finite non-empty set of input symbols

(iii) $\delta : Q \times \Sigma \to Q$ δ is a state transition f^n.

(iv) $q_0 \in Q$ is called the initial state.

(v) $F \subseteq Q$ is the set of final states.

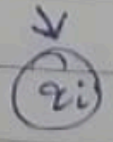
* Create a finite automata that accepts all strings ending in 'ab' over $\{a, b\}$.

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

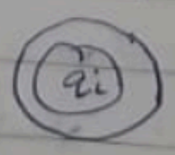
$\Sigma^* = \{a, b\}^*$ = all possible combinations (a, aa, ---- b, bb, bbb, ab, aab, ---)

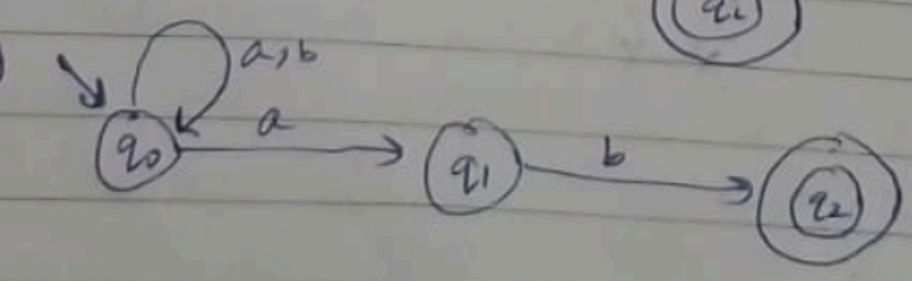
Diagramatically :

1) State 

2) Initial state 

3) $\xrightarrow{a \in \Sigma}$ transition

4) Final State 



Page No.
Date / /

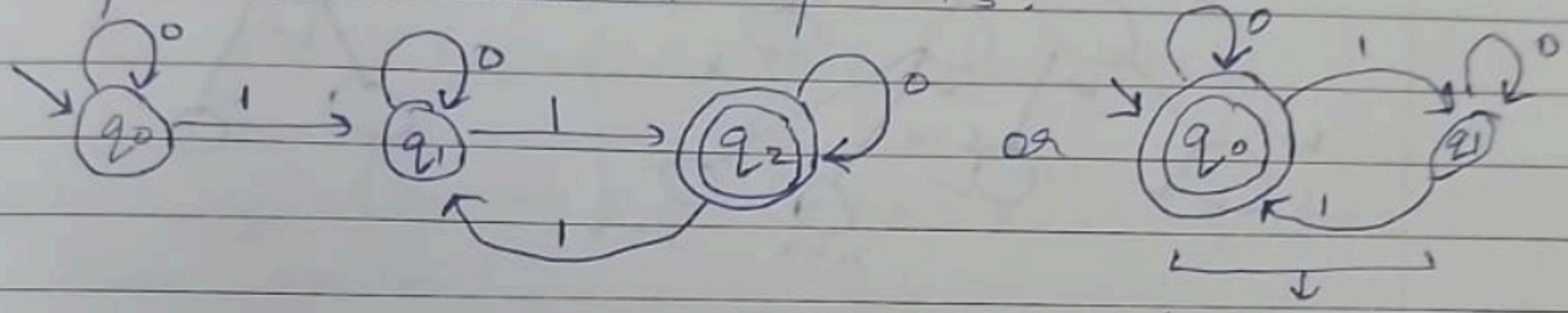
eg: a a b

$q_0 \xrightarrow{a} q_1$ X

$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \notin F$ ×

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \in F$ ✓

Q: Construct a finite automata over $\{0,1\}$ that accepts even number of 1's.

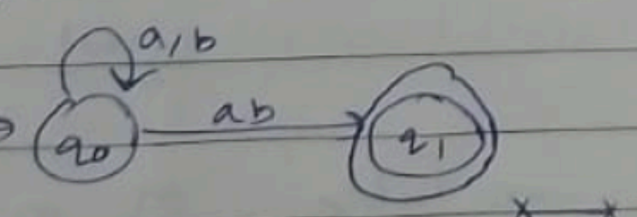


* Transition System: A transition system is defined by a 5 tuple $(Q, \Sigma, \delta, q_0, F)$

here $q_0 \subseteq Q$, $\delta: Q \times \Sigma^* \longrightarrow Q$

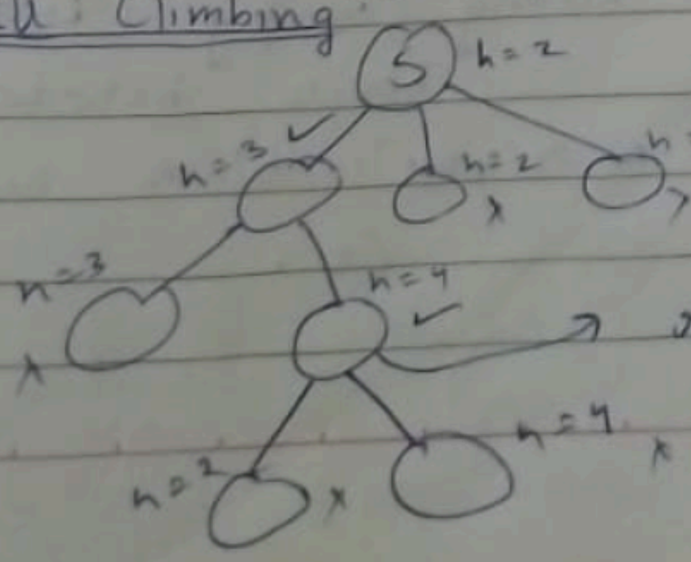
↳ All possible set of strings in Σ.

Q, Σ, F are same as in FA.

eg: 

AI

* Hill Climbing: → for all puzzles



⇒ move strictly in the direction of inc/dec $h(n)$.

→ return this ⇒ Not complete

⇒ Space complexity = const.

↳ best feature.

time comp = $O(n)$

b.w.

eg W = no. of trails for getting K success

W = { K, K+1, ... ∞ }

⇒ W follows -ve binomial dist. fn.

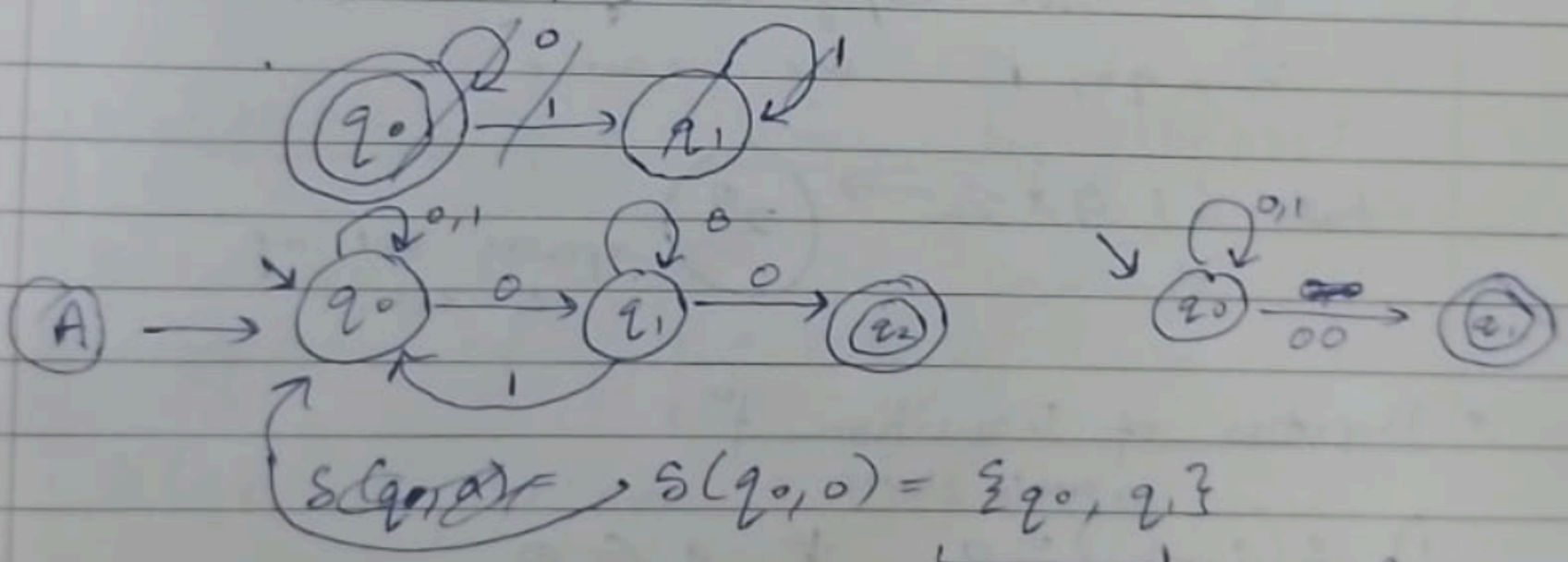
* Geometric is a memoriless property.

Theory of Computation

Q. Construct an automata that recognizes all the binary numbers divisible by 4.
↳ last 2 digits = 0

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

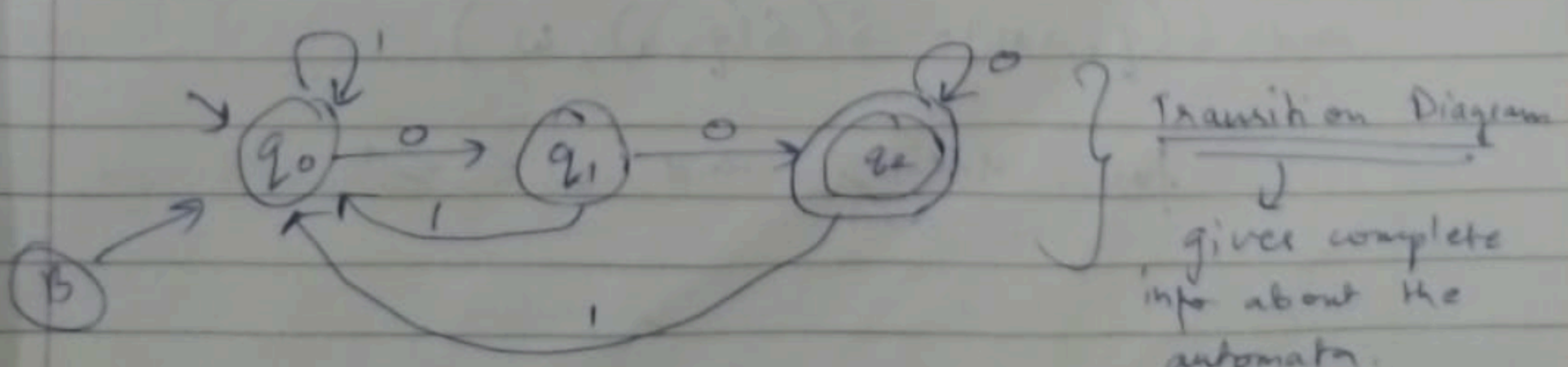
$\Sigma = \{0, 1\}$



↳ not DFA (NDFA)

NDFA : possibility of multiple states (Deterministic finite automata)

DFA : only one state possible.



Transition Diagram
↓
gives complete info about the automata.

Page No.
Date

Transition table :

	Next State	
	0	1
→ q_0	q_0, q_1	q_0
q_1	q_1, q_2	q_0
(q_2)	ϕ	ϕ

for (A)

* **Acceptibility of a string by DFA M :**

A string $w \in \Sigma^*$ is said to be accepted by $M = (Q, \Sigma, \delta, q_0, F)$ if

$$\delta(q_0, w) = q \in F$$

* **NDFA $(Q, \Sigma, \delta, q_0, F)$:**

Q = non-empty set of states

Σ, q_0, F are same

but $\delta : Q \times \Sigma \rightarrow (2^Q)$ → power set of Q-set.

* **Properties of Transition fn :**

1) $\delta(q, \lambda) = q \quad \forall \; q \in Q$ (empty string)

2) $\delta(q, wa) = \delta(\delta(q, w), a)$

and $\delta(q, aw) = \delta(\delta(q, a), w)$

for $a \in \Sigma$ and $w \in \Sigma^*$

$\delta(q_0, 1011) = \delta(\delta(q_0, 1), 011)$

Q: Give the entire sequence of states for the input string 110101 for the automata below.

State	0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$\delta(q_0, 110101)$
$= \delta(\delta(q_0, 1), 10101)$
$= \delta(\delta(q_1, 1), 0101)$
$= \delta(\delta(q_0, 0), 101)$
$= \delta(\delta(q_2, 1), 01)$
$= \delta(\delta(q_3, 0), 1) = \delta(\delta(q_1, 1), \Lambda)$
$= \delta(q_0, \Lambda)$
$= q_0$

* Acceptibility of a string by NDFAM:

A string $w \in \Sigma^*$ is said to be accepted by $m = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, w)$ contains a state from F.

Q: Prove that for any transition f^n δ and 2 strings x and y.

$\delta(q, xy) = \delta(\delta(q, x), y)$

by PMI on $|y|$: Base $|y| = 1$

math induction

$|y| = n$

$|y| = n+1 \qquad y = y_1 a \quad , |y_1| = n$

$\delta(q, xy) = \delta(q, xy_1 a)$

$= \delta(\delta(q, xy_1), a)$

$= \delta(\delta(\delta(q, x), y_1), a)$

Stotastic Process

$$X = \begin{cases} 1 & p \\ 0 & 1-p \end{cases}$$

$y = \{0, 1, 2, \ldots n\}$: no. of success on n tr

$z = \{1, 2, 3, \ldots n\}$: no. of trials to get first success.

$w = \{k, k+1, \ldots \infty\}$: no. of trials to get k success

$w = z_1 + z_2 + \ldots + z_k$

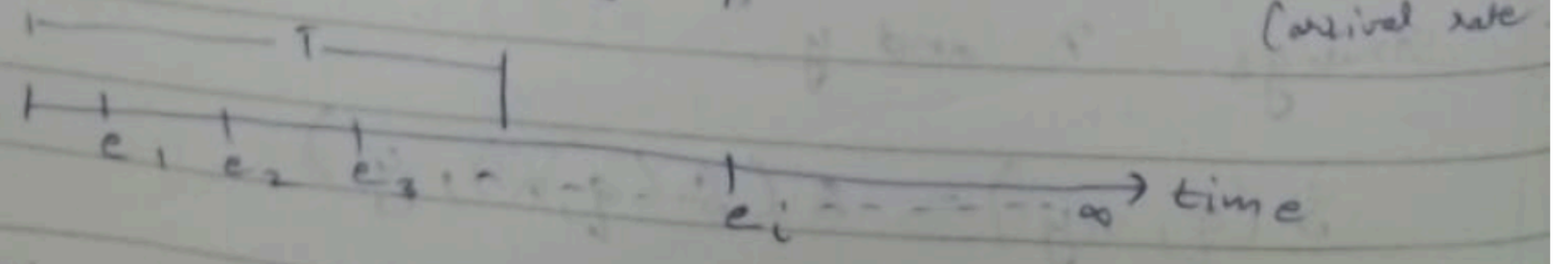
* Bernoulli trial : $x_1 \; x_2 \; x_3 \; \ldots \; x_i \; \ldots \; \infty$

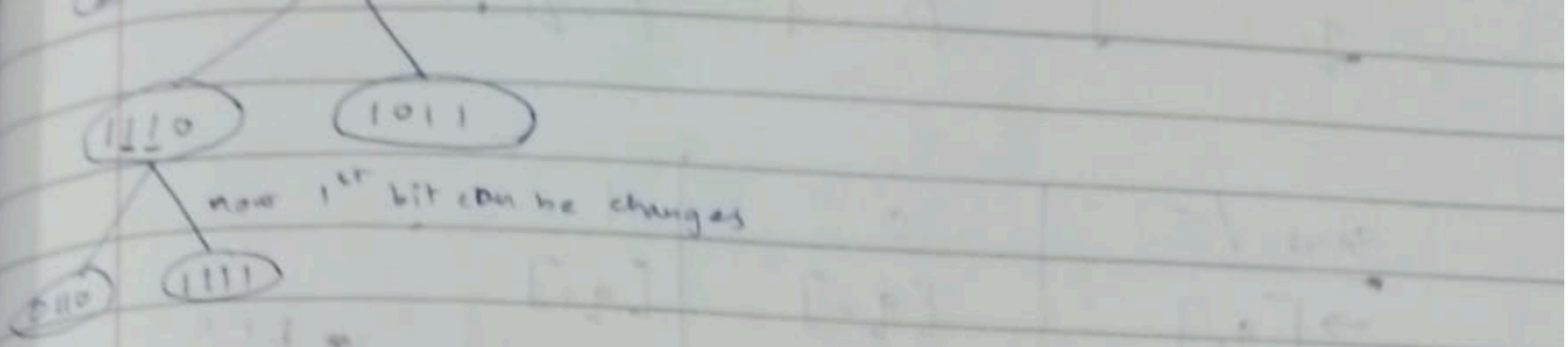
$1/0 \quad 1/0 \quad 1/0 \quad \ldots \quad 1/0 \quad \ldots \quad 1/0$

$P(x=k) = {}^{n}C_k \, p^k (1-p)^{n-k}$

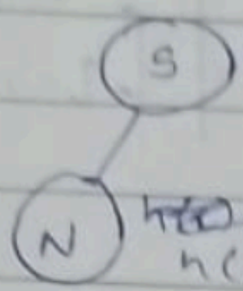
* Poisson Process :

no. of events occuring per unit time $= \lambda$ (arrival rate)





* STOCHASTIC HILL CLIMBING : (only goes one state



prob. threshhold = 2.

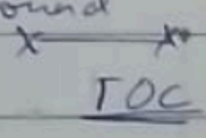
$\Delta E = h(N) - h(c) = 120 - 107 = 13$

use sigmoid : $\frac{1}{1 + e^{-\Delta E/T}}$ to calc. prob.

fix T at start (say T = 10)

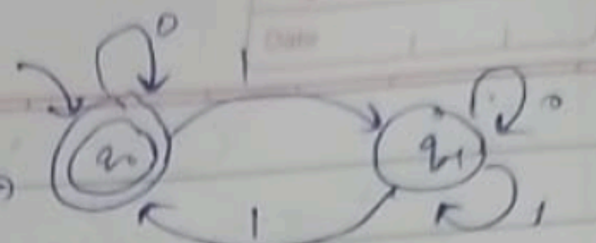
⇒ change T iteration by iteration (start with high T go to low T)
simulated Annealing

* Study Branch n Bound



* Conversion of NDFA ⟶ DFA

q. Construct a DFA equivalent to
$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$ where



S is defined as:

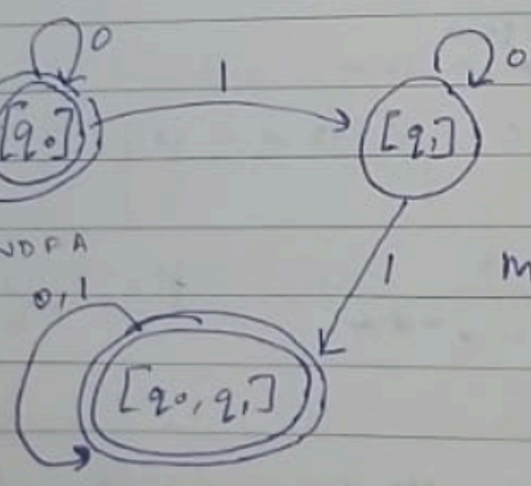
State / Σ	0	1
→ (q0)	q0	q1
q1	q1	q0, q1

→ NDFA

State / Σ	0	1
→ [q0]	[q0]	[q1]
[q1]	[q1]	[q0, q1]
[q0, q1]	[q0, q1]	[q0, q1]

→ DFA

combination of q0, q1



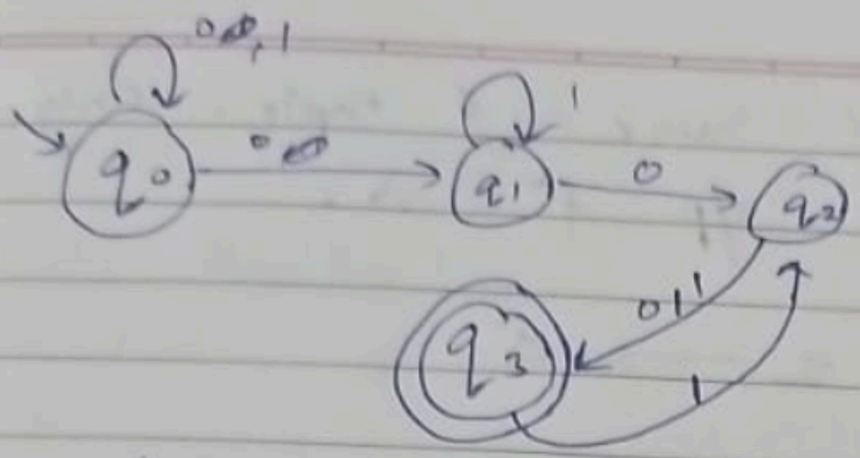
as $q_0 \xrightarrow{w} q_0$ in NDFA and DFA

$M' = (\{[q_0], [q_1], [q_0, q_1]\}, \Sigma = \{0, 1\}, S', [q_0], F)$

note: final state in NDFA is also a final state in DFA.

Q1

State / Σ	0	1
→ q0	q0, q1	q0
q1	q2	q1
q2	q3	q3
(q3)	X ϕ	q2



State / Σ	0	1
→ [q0]	[q0, q1]	[q0]
[q0, q1]	[q0, q1, q2]	[q0, q1]
[q0, q1, q2]	[q0, q1, q2, q3]	[q0, q1, q3]
[q0, q1, q2]	[q0, q1, q2]	[q0, q1, q2]
[q0, q1, q2, q3]	[q0, q1, q2, q3]	[q0, q1, q2, q3]

↳ final state as q_3 occurs in them

* **Mealy and Moore Machine :**

⇒ **o/p f^n :** An o/p f^n $z(t)$ in general is a f^n of the present state $q(t)$ and the present i/p symbol $x(t)$ i.e.

$$\boxed{z(t) = \lambda(q(t), x(t))}$$

⇒ If $z(t)$ depends only on the present state then $z(t) = \lambda(q(t))$

* **Moore Machine :** It is defined as a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where Q = set of non-empty finite states

Σ = set of i/p symbols, Δ = set of o/p symbols

$\delta : Q \times \Sigma \to Q$, λ = o/p f^n, q_0 is initial state

↳ next state transition state.

$\lambda : Q \to \Delta$

* Mealy Machine : Same 6 tuple. Only diff. is the o/p fⁿ λ.

$$\lambda : Q \times \Sigma \longrightarrow \Delta$$

Eg.

Next State

Present State	a=0	a=1	o/p
→ q_0	q_3	q_1	0
q_1	q_2	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

↳ Moore Machine

Present State	a=0	o/p	a=1	o/p
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

↳ mealy machine

i/p : 0111 : (for moore machine)

$q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$

o/p string = 00010

↳ length = 1 + length of i/p string

i/p : 0011 : (for mealy machine)

$q_1 \xrightarrow{0} q_3 \xrightarrow{0} q_2 \xrightarrow{1} q_4 \xrightarrow{1} q_3$

o/p string = 0100

↳ length of o/p = length of i/p.

$c_1 = \underbrace{2\ 3\ 5\ 3}_{r_1}\ \underbrace{5\ 1\ 1\ 1\ 1}_{r_2}$

$C_2 = \underbrace{6\ 2\ 6\ 2}_{r_2}\ \underbrace{3\ 1\ 2\ 2\ 1}_{r_1}$

} using single pt. crossover

$c_1 = 2\ 4\ 7\ 5\ 9\ 1\ 3\ 6\ 8$ } path seq.

TOC

* Thm: For every NDFA, ∃ a DFA which simulates the behaviour of NDFA. Alternatively if L is the set accepted by an NDFA, then ∃ a DFA that accepts L

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NDFA accepting L. We ~~construct~~ construct a DFA M' as:

$M' = (Q', \Sigma, \delta', q_0', F')$ where:

(i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, \ldots q_i]$ where $q_1, q_2, \ldots q_i \in Q$)

(ii) $q_0' = [q_0]$

(iii) F' is the set of all subsets of Q containing an element of F.

Before defining δ', we look at the construction of Q', q_0' and F. M is initially at q_0. But on application of an i/p symbol, say 'a', M can reach any of the states $\delta(q_0, a)$.

To describe M, just after the i/p symbol 'a',

we require all possible states that M can reach on application of 'a'.

So M' has to remember all these possible states at any instant of time. i.e., the states of M' are defined as subsets of Q

As M starts with initial state q_0, q_0' is defined as $[q_0]$.

A string 'w' belongs to T(M) [set of all i/p strings accepted by M], if a final state is one of the possible states that M reaches on processing 'w'. So, a final state in M' (i.e., an element of F') is any subset of Q containing some final state of M.

Define δ':

(i) $\delta'([q_1, q_2, \ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \ldots \cup \delta(q_i, a)$

Equivalently,

$\delta'([q_1, q_2, \ldots, q_i], a) = [p_1, p_2, \ldots, p_j]$

if

$\delta(\{q_1, q_2, \ldots q_i\}, a) = \{p_1, p_2, \ldots, p_j\}$

Before, proving $L = T(m')$; we prove that
$\delta'(q_0', x) = [q_1, \ldots, q_i]$

Before proving $L = T(m')$; we prove that
$\delta'(q_0', x) = [q_1, q_2, \ldots q_i]$

iff $\delta(q_0, x) = \{q_1, q_2, \ldots q_i\}$ $\forall$ $x \in \Sigma^*$

We prove this result by induction on $|x|$

We first prove $\delta'(q_0', x) = [q_1, q_2, \ldots q_i]$ iff
$\delta(q_0, x) = \{q_1, q_2, \ldots q_i\}$ — (A)

Λ = empty string.

When $|x| = 0$
$\delta(q_0, \Lambda) = \{q_0\}$ and by def^n of δ'.

$\delta'(q_0', \Lambda) = q_0' = [q_0]$

So, result result holds for $|x| = 0$. Thus, there is a basis of induction. Assume (A) holds for all strings y with length 'n'. Let x be a string of length $(m+1)$ we can write x as ya, where $|y| = m$ and $a \in \Sigma$. Let
$\delta(q_0, y) = \{p_1, p_2, \ldots, p_j\}$ and
$\delta(q_0, ya) = \{r_1, r_2, \ldots r_k\}$

As $|y| \geq m$; by induction hypothesis, we have
$\delta'(q_0', y) = [p_1, p_2, \ldots p_j]$
Also
$\{r_1, r_2, \ldots, r_k\} = \delta(q_0, ya) = \delta(\delta(q_0, y), a)$
$= \delta(\{p_1, p_2, \ldots p_k\}, a)$

By defⁿ of δ':

$$\delta'([p_1, p_2, \ldots, p_j], a) = [r_1, r_2, \ldots, r_k]$$

Hence,

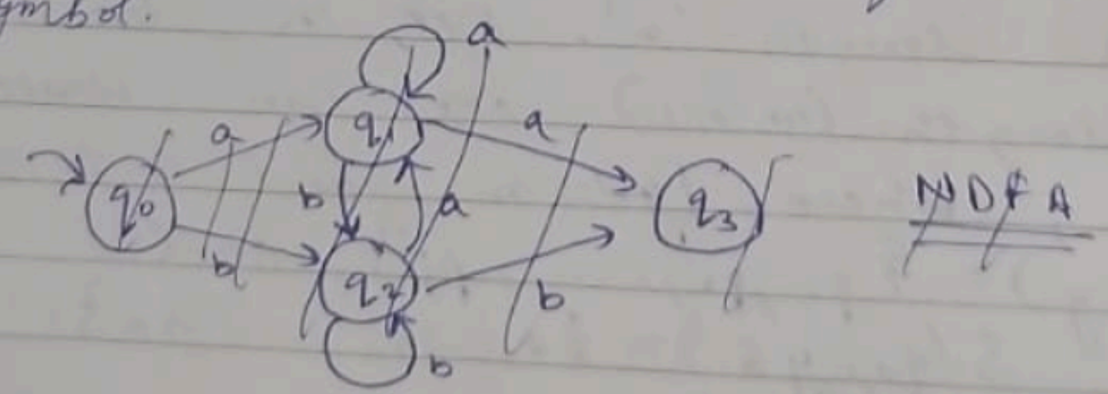
$$\delta'(q_0', ya) = \delta'(\delta'(q_0', y), a)$$
$$= \delta'([p_1, p_2, \ldots p_j], a)$$
$$= [r_1, r_2, \ldots, r_k]$$

Hence (A) holds for $\forall\, x \in \Sigma^*$
similarly, the converse part can be proved.
Now, $x \in T(M)$ iff $\delta(q_0, x)$ contains a state of F. $\delta(q_0, x)$ contains a state of F iff. $\delta'(q_0', x)$ is in F'.

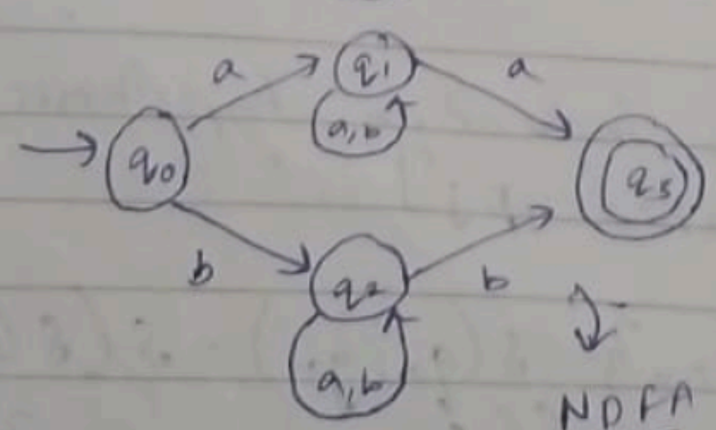
Hence, $x \in T(M)$ iff $x \in T(M')$

This proves that DFA M' accepts L.

Q. Construct a DFA that accepts all strings over $\{a, b\}$, beginning and ending in the same symbol.



NDFA

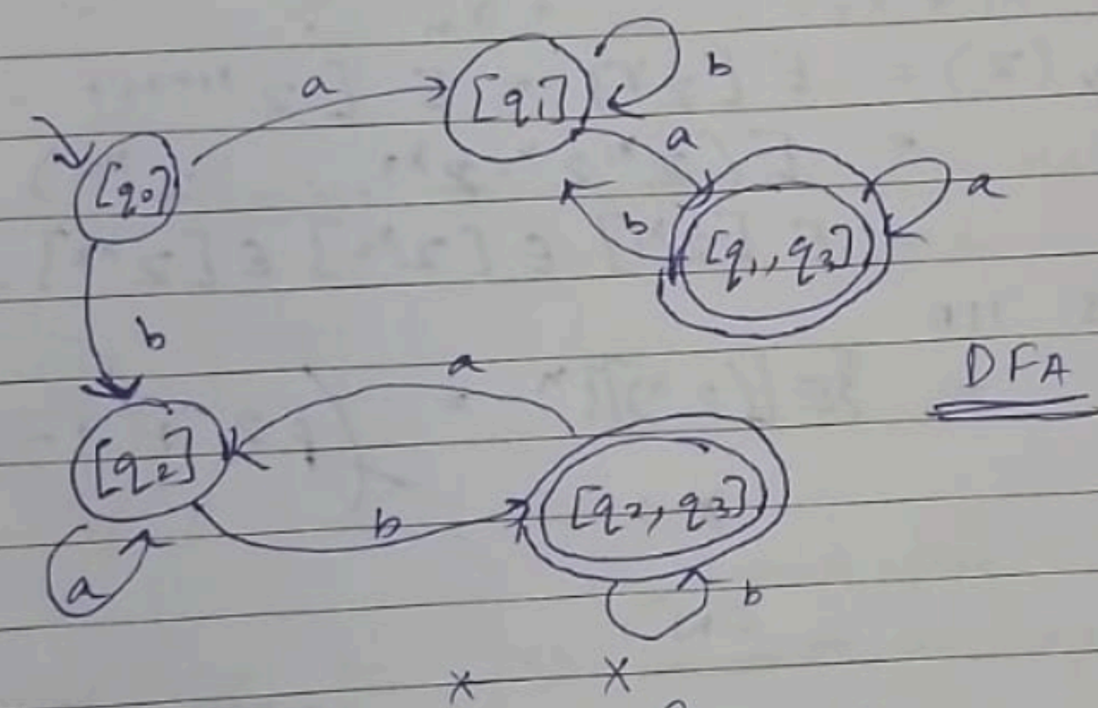


NDFA

$$T(M) = L$$
$$= \{a(a+b)^*a,\ b(a+b)^*b\}$$

State	a	b
→ q_0	q_1	q_2
q_1	q_1, q_3	q_1
q_2	q_2	q_2, q_3
q_3	ϕ	ϕ

State	a	b
→ $[q_0]$	$[q_1]$	$[q_2]$
$[q_1]$	$[q_1, q_3]$	$[q_1]$
$[q_2]$	$[q_2]$	$[q_2, q_3]$
$[q_1, q_3]$	$[q_1, q_3]$	$[q_1]$
$[q_2, q_3]$	$[q_2]$	$[q_2, q_3]$



DFA

Stochastic Processes

* **Moment Generating F^n and Prob. Gen. F^n:**

Let X be a discrete random variable.

$X = 1, 2, \ldots \infty$

prob. gen $f^n \longrightarrow G_X(z)$ → random variable

$$G_X(z) = \sum_{i=0}^{\infty} P(X=i)\, z^i$$

↳ coeff of z^i gives prob. of $(X = i)$

moment gen. $f^n \longrightarrow M_X(t) = E[e^{tX}]$ → random var.; dummy variable

$$= \int_{-\infty}^{\infty} e^{xt} f(x)\, dx$$

$\frac{dM(t)}{dt}$ gives 1st moment $= \int_{-\infty}^{\infty} x e^{xt} f(x)\, dx$

Page No.
Date

TOC

* Mealy and Moore Machine :

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

$\lambda : Q \times \Sigma \to \Delta$ (mealy)

$\lambda : Q \to \Delta$ (moore)

Mealy $\longleftrightarrow$ Moore (Equivalent?)

⇒ Mealy and Moore are equivalent if they recognise the same set of i/p o/p behaviours, i.e. they produce the same o/p sequence for a given i/p sequence (with a one symbol delay)

* Procedure for Transforming Mealy to Moore Machine :

Present State	Input a=0	o/p	a=1	o/p
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

$q_1 \to 1 \to q_1$

$q_2 \to 0, 1 \to q_{20}$ and q_{21}

$q_3 \to 0 \to q_3$

$q_4 \to 0, 1 \to q_{40}$ and q_{41}

State	a=0	o/p	a=1	o/p
→ q_1	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

↓

Equivalent Moore Machine

State	a=0	a=1	o/p
→ q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_{41}	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

i/p 0110

o/p :

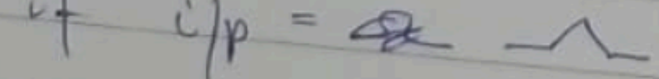
$q_1 \xrightarrow{0} q_3 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_1$

∴ Mealy : 0101

$q_1 \xrightarrow{0} q_3 \xrightarrow{1} q_1 \xrightarrow{1} q_{20} \xrightarrow{0} q_1$

Moore : 10101

↳ same as Mealy

if i/p =  then Mealy not equivalent to Moore

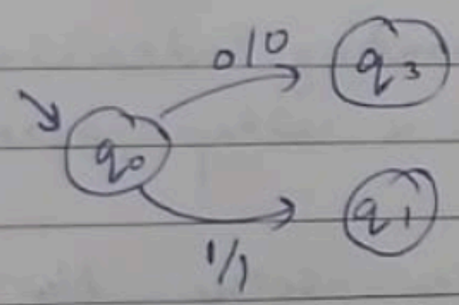
* Procedure for converting Moore to Mealy Machine:

Present State	a = 0	a = 1	o/p
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

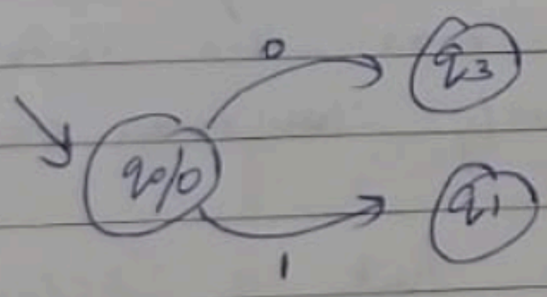
moore

Present state	a = 0	o/p	a = 1	o/p
q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

mealy



for Mealy



for Moore

Q. Given FA = $(Q, \Sigma, \delta, q_0, F)$ → (Mealy / Moore)

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

w ∈ q ∈ F accept

∈ q ∉ F reject

Moore: $\lambda : Q \rightarrow \Delta$

$q \in F, \lambda(q) = 1$.

$q \notin F \rightarrow \lambda(q) = 0$

Toc

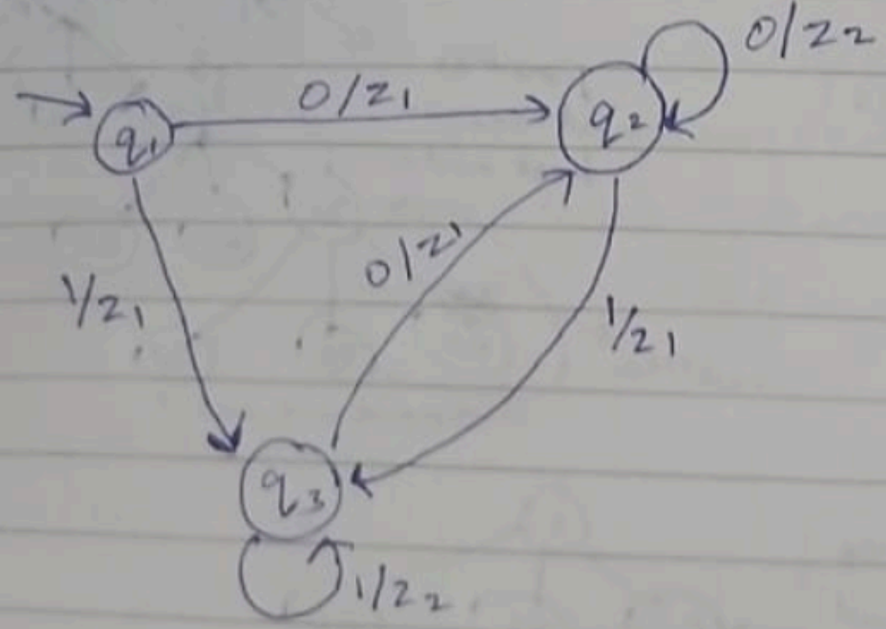
5 Convert Moore's to Mealy machine

Present state	Next State		o/p
	a = 0	a = 1	
~~q~~			
q_1	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

Ans

Present State	a = 0	o/p	a = 1	o/p
q_1	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

Q. Construct Moore machine equivalent to Mealy machine below.



Ans

state	a = 0	o/p	a = 1	o/p
→ q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_2	q_3	z_1
q_3	q_2	z_1	q_3	z_2

$q_0 \rightarrow q_{20}, q_{21}$

$q_3 \rightarrow q_{30}, q_{31}$

$q_1 \rightarrow q_{10}, q_{11}$

$a \equiv b$ Equivalent i.e. reflexive, symmetric and transitive.

	a = 0	a = 1	O/P
→ q_{11}	q_{21}	q_{31}	z_1
→ q_{12}	q_{41}	q_{31}	z_2
q_{21}	q_{22}	q_{31}	z_1
q_{22}	q_{22}	q_{31}	z_2
q_{31}	q_{21}	q_{32}	z_1
q_{32}	q_{21}	q_{32}	z_2

* **Minimization of Automata** : (will come in Exam)

Equivalent States : 2 states q_1 and q_2 are said to be equivalent if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are either final states or both are non-final states $\forall\ x \in \Sigma^*$

K- Equivalent states : 2 states q_1 and q_2 are said to be 'k' equivalent ($k \geq 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final or both are non-final states for all strings in Σ^* of length k or less.

0 eq : $x = \varepsilon$ 1 eq : $x = \{a, b\}$

2 eq : $x = \{aa, ab, ba, bb, a, b, \varepsilon\}$

Q: Construct a minimized automata for the given finite automata

State	0	1
q_1	q_1	q_5
q_2	q_6	q_4
q_3		q_2
q_4		q_6
q_5		q_3

State / Σ	0	1
→ q_0	q_1	q_5
q_1	q_1	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

partition 0-equivalent

↳ $\Pi_0 = \{ \{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\} \}$

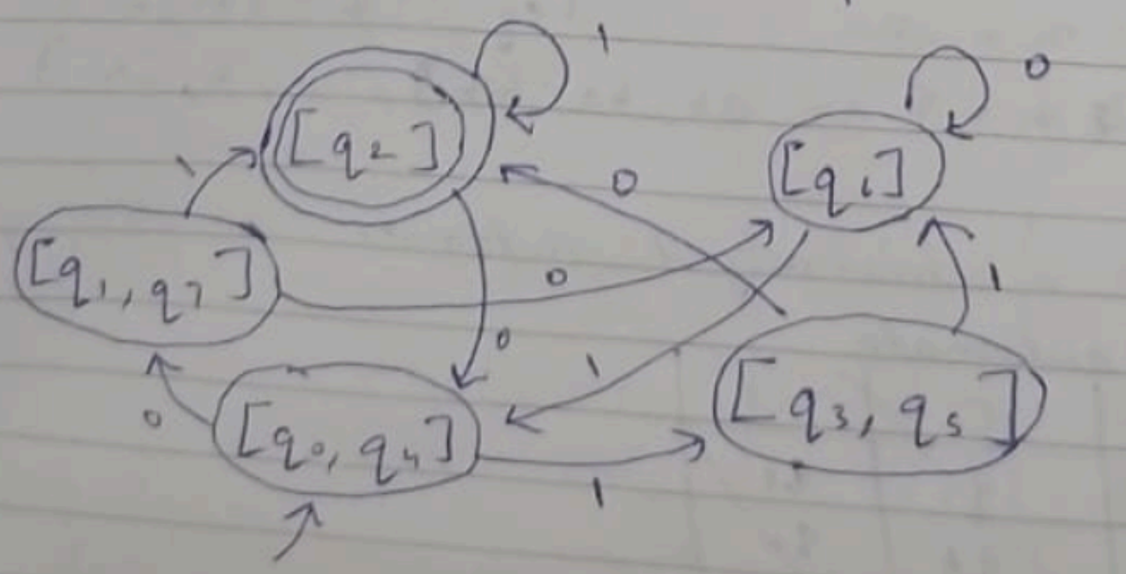
$\Pi_1 = \{ \{q_2\}, \{q_0, q_4, q_6\}, \{q_3, q_5\}, \{q_1, q_7\} \}$

$\Pi_2 = \{ \{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_3, q_5\}, \{q_1, q_7\} \}$

↳ check in Π_1

$\Pi_3 = \{ \{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_3, q_5\}, \{q_1, q_7\} \}$

$\Pi_2 = \Pi_3$ ∴ we stop.



* Construction of Min. Automata:

Step 1 : construction of π_0.

By definition of 0-Equivalence, $\pi_0 = \{Q_1^0, Q_2^0\}$
where Q_1^0 = set of final states
Q_2^0 = set of non-final states

Step 2 : Construction of π_{k+1} from π_k

Let Q_i^k be any subset of π_k. If q_1 and q_2 are in Q_i^k, they are $(k+1)$ equivalent. Find out weather $\delta(q_1, a)$ and $\delta(q_2, a)$ are in the same equivalence class in π_k $\forall\ a \in \Sigma$. If so, q_1 and q_2 $(k+1)$ equivalent. In this way Q_i^k is further divided into $(k+1)$ equivalence classes. Repeat this for every Q_i^k in π_k to get all elements of π_{k+1}.

Step 3 : Construct π_n for $n = 1, 2, 3, \cdots$ until
$\pi_n = \pi_{n+1}$

Step 4 : Construction of min. Automata : For the required min. state automata, the states are the equivalence classes obtained in step 3, i.e., the elements in π_n. The state transition table is obtained by replacing the state q by corresponding equivalence class $[q]$.

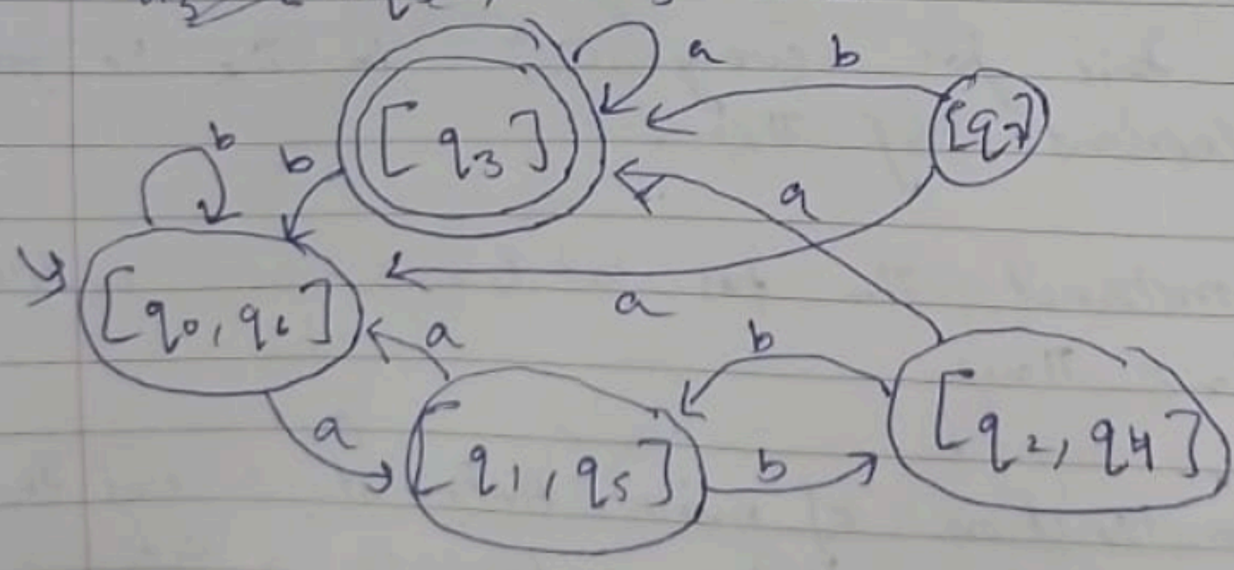
	a	b
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
(q_3)	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

$\Pi_0 = \{ \{q_3\}, \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\} \}$

$\Pi_1 = \{ \{q_3\}, \{q_0, q_1, q_5, q_6\}, \{q_2, q_7\}, \{q_4\} \}$

$\Pi_2 = \{ \{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_7\}, \{q_4\} \}$

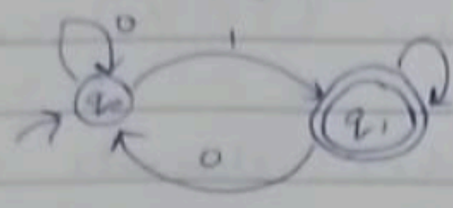
$\Pi_3 = \Pi_2$



Q// Create a automata over {0, 1} which accepts all strings where no. of strings are 3 mod 4 (n % 4 = 3)

Page No.
Date / /

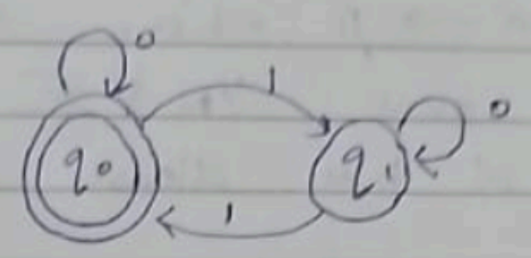
Q. Find the language generated by the following FA

① 

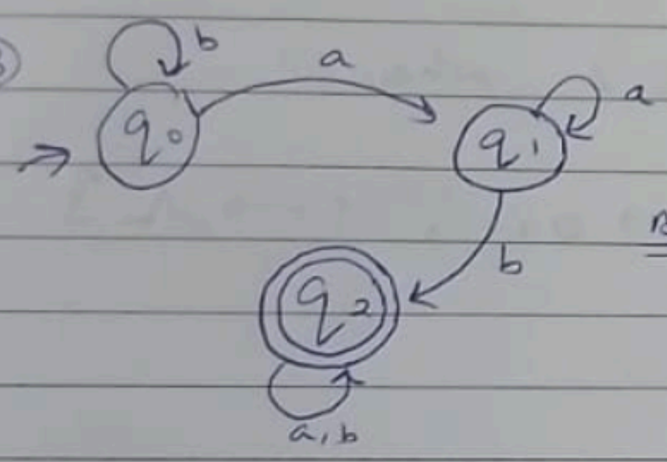
Ans $L = \{ w \in \Sigma^* \mid w \text{ ends in one} \}$

②

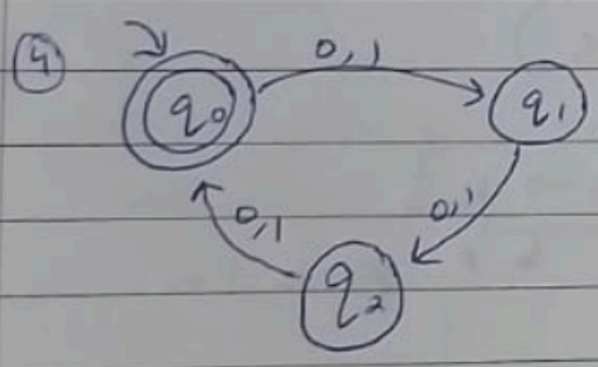
State	0	1
→ q_0	q_0	q_1
q_1	q_1	q_0



Ans $L = \{ w \in \{0,1\}^* \mid w \text{ has even no of ones} \}$

③ 

Ans $L = \{ w \in \{a,b\}^* \mid w \text{ contains substring 'ab'} \}$

④ 

Ans $L = \{ w \in \{0,1\}^* \mid |w| = 3m; m \in IN \cup \{0\} \}$

Unit - 2 (Formal Languages)

* Grammar :

⇒ A grammar of phrase structure grammar is defined using the 4 tuple (V_N, Σ, P, S); where

(i) V_N is a finite non-empty set whose elements are called as variables.

(ii) Σ is a finite non-empty set whose elements are called terminals.

(iii) $V_n \cap \Sigma = \phi$

(iv) S is a special variable $(S \in V_n)$ called as the starting symbol

(v) P is a finite set whose elements are $\alpha \rightarrow \beta$ where α and β are strings over $V_N \cup \Sigma$ $(\alpha, \beta \in V_N \cup \Sigma^*)$
α has atleast one symbol from V_N

⇒ The elements of P are called production rules / productions / Rewritting Rules.

eg. $G_1 = (\{S\}, \{0,1\}, P, S)$ where

$P = \{ \underbrace{S \rightarrow 0S1}_{\alpha \quad \beta}, \; S \rightarrow 01, \; S \rightarrow \wedge \}$

α → S-productions

$G_2 = (\{S, A\}, \{a, b\}, P, S)$

$P = \{ \underbrace{S \rightarrow A}, \underbrace{S \rightarrow a}, \underbrace{A \rightarrow b} \}$

S production; b A-production.

⇒ Production Rules cannot be reversed, i.e.
$S \rightarrow A \not\Rightarrow A \rightarrow S$

⇒ $(V_N \cup \Sigma)^*$ → includes empty string

$(V_N \cup \Sigma)^+ = (V_N \cup \Sigma)^* \setminus \{\wedge\}$

↳ does not include empty string

* One Step Derivation :

⇒ if $\alpha \to \beta$ is a production in grammar G and γ and δ are any 2 strings in $(V_N \cup \Sigma)^*$ then we say $\gamma\alpha\delta$ directly derives $\gamma\beta\delta$ in G, and we write

$$(\gamma\alpha\delta \Rightarrow \gamma\beta\delta)$$

Defⁿ : Reflexive Transitive Closure :

⇒ If α and β are strings on $(V_n \cup \Sigma)$ then we say α derives β ~~if $\alpha \Rightarrow \beta$~~ if

$\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$, here $\overset{*}{\underset{G}{\Rightarrow}}$ represents the reflexive transitive closure of the relation $\underset{G}{\Rightarrow}$ in $(V_N \cup \Sigma)^*$.

Defⁿ ⇒ The lang. generated by a grammar G denoted by $L(G)$ is defined by a set

$$\{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

and elements of the language are called sentences.