Automata :- Automata theory is the study of abstract computing devices or machines.

Basically automaton = machine & Automata = machines.

OR

Automata is a Abstract models for different aspects of computation.

OR

The study of mathematical properties of abstract computing machines.

Basic Mathematics -

1] Sets :- Set is a collection of things called element or members.

ex. 1. {5} - a set containing only one element 5.
2. {1, 2, 3, ...} is a set of natural numbers.

A set can also be described using the notation.

$A = \{x \mid x \text{ is an odd integer}, x > 1\}$.

(A is a set of all x such that x is an odd integer and x is greater than 1).

Equality of sets :- Two sets A & B are equal iff A & B contains the same elements or both are empty. If the two sets A and B are equal then we write $A = B$.

Empty set :- A set contain no element is called empty set. (ϕ)

subset:- a set A is a subset of a set B, and we write $A \subseteq B$, iff every elements of A is an element of B.

Proper subset.- If $A \subseteq B$ and $A \neq B$, then A is called a proper subset of B and it is written as $A \subset B$.

The Power set- The power set of a set A, denoted as P(A), is a set of all subset of A.

ex if $A = \{a, b, c\}$ then
$P(A) = \{\phi, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{abc\}\}$

Operations on sets:-

Complement of a set:- Complement of a set A is represented as A'. A' is a set of every element that is not in A.

Union of two sets:- Union of two sets A and B is represented as $A \cup B$. Union of two sets A and B is the set of elements in A or in B.

Intersection of two sets:- $(A \cap B)$ is the set of elem which belong to both A and B.

set difference:- $(A - B)$ is a set of those elements of A which are not in B.

$A - B = \{x \mid x \in A \text{ and } x \notin B\}$

Page No

1. $A \cup B = B \cup A$ ..... Commutative laws
$A \cap B = B \cap A$

2. $A \cup (B \cup C) = (A \cup B) \cup C$ Associative laws.
$A \cap (B \cap C) = (A \cap B) \cap C$

3. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ Distributive laws.
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

4. $A \cup A = A$ Idempotent laws
$A \cap A = A$

5. $A \cup (A \cap B) = A$ Absorptive laws.
$A \cap (A \cup B) = A$

6. $(A \cup B)' = A' \cap B'$ De Morgan laws.
$(A \cap B)' = A' \cup B'$

Relations:-

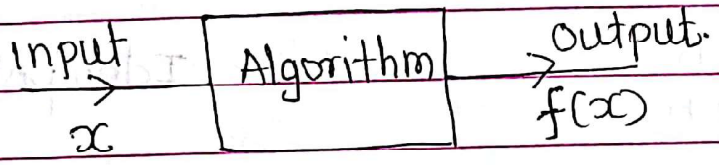
classmate

Theory of computation/program/algorithm

Theory of Computation:- computation refers to writting a program & execute it on a machine or computer.

program:- program express algorithms.

Algorithm:- is a recipe for carrying out input to output transformation.



$f : D \longrightarrow R$

D :- represent Domain of all input.

R : Range of output.

every algorithm computes a function.

Basic goal:- to identify the class of functions which admit algorithms to compute them.

For function f There is an algorithm to compute f, and rest of function they do not admit any algorithm to compute them.

Set membership problem:- There is a set S and we have given any a(input), to decide whether $a \in S$ or not.

function f is mapping of domain to some range.

$f : D \rightarrow R$

There is very natural set which is associated with any function. that set is called graph(f)

$graph(f) = \{ \underbrace{(a, b)}_{\text{tuple.}} \mid f(a) = b \}$

suppose we show that there is no algorithm to solve the set membership problem for the set graph(f), then we can conclude that then there no algorithm to compute f.

our sets are going to be sets of finite strings

Symbols :- ex 0, 1, a, b

Alphabet :- An alphabet is a finite (Non empty) set of symbols.

ex :- $\Sigma\{0, 1\}$ - the binary alphabets.

$\Sigma\{A, B, \ldots Z\}$ - the set of uppercase letters.

$\Sigma\{0, 1, \ldots, 9\}$ - decimal alphabets.

The symbol Σ (sigma) is used for alphabet.

$\Sigma\{0, 1, 2\}$ - the ternary alphabet.

String :- It is a finite sequence of symbols over an alphabet.

ex - $\Sigma\{0, 1\}$ then string is 010, 11010, etc.

suppose Σ is an alphabet then Σ^* denotes the set of all finite strings over Σ

Page No

ex $\Sigma = \{0,1\}$ then Σ^* = is the set of all finite binary strings.
finite means in terms of length of individual string Σ^* is an infinite set. but the length of individual string is finite.

Formal Language- A formal language L over the alphabet Σ is a subset of Σ^*.

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

$L = \{0, 1, 10, 11, 110, 1101010\} \subseteq \Sigma^*$

$L1 = \{x \in \{0,1\}^* \mid x$ has even no of 0's & even no of 1's$\}$

$L1 \subset \{0,1\}^*$

L proper subset.

We shall be concerned with the set membership problem of formal language

We have given a language L then given a string x & we have to decide whether that string x is member of language L

$x \in L$

basically string is nothing but input to any algorithm & output is nothing but the solution to set membership problem. To solve this set membership problem we are having models of computation (Automata) of various kinds.

~~Models of computation~~ -

1. ~~Finite state Automata~~ -

The empty string :- The empty string is the string with zero occurance of symbols. denoted by ϵ, is a string chosen from any alphabet eg.

Length of string → No of symbols in the string.
eg $x = 11010$, then $|x| = 5$

Powers of an alphabet :- We can express the set of all strings of certain length from that alphabet by using an exponential notation.
Σ^k - set of strings of length k, each of which chose from Σ ex $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$
$\Sigma^0 = \{\epsilon\}$, $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$ etc.
$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$ set of nonempty strings.

Concatenation of strings :-
If x & y be strings. xy denotes concatenation of x & y i.e. string formed by making copy of x followed by string y.
i.e $x = a_1 a_2 \ldots a_n$ & $y = b_1 b_2 \ldots b_n$ then
$xy = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_n$.

Finite State Automata (FSA)
OR
Machines

set membership problem for language L is given x, to decide if $x \in L$.

To decide set membership problem we define machine or grammer & so on.

Let us understand FSA using example.

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

$L = \{ x \in \{0,1\}^* \mid x \text{ has even no of } 0\text{'s} \}$

$01101101010 \in L$, $1111 \in L$, $01010 \notin L$

Way to do:-

start scanning the string from its left end, go over the string one symbol at a time, to be ready with the ans as soon as the string is entirely scan.

0 1 1 0 1 1 0 1 0

↑

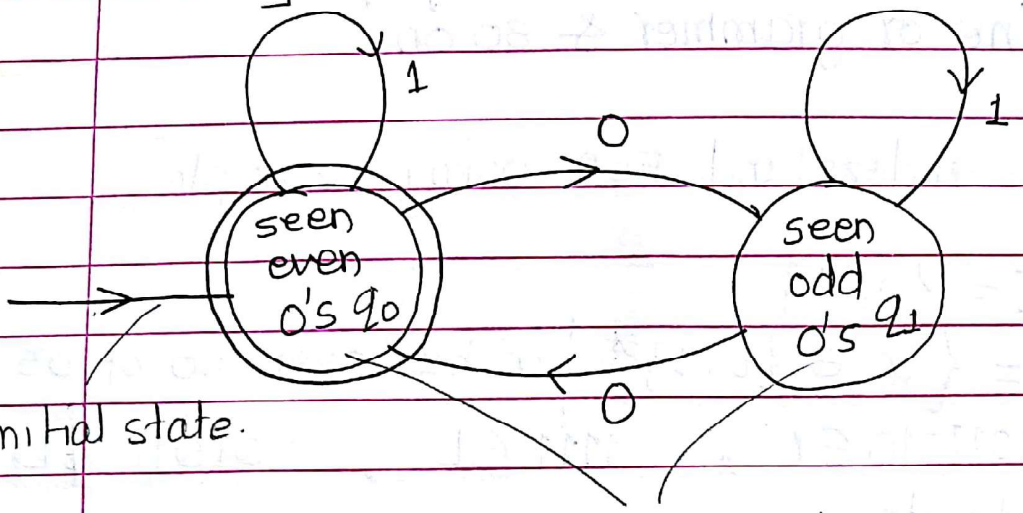
Initial state
of mind is even)
(0 no of zero)

so before reading any symbol of string initially the state of mind is even no of zero (initially zero no of 0's are there). As i read first symbol i.e 0 the state of mind change to odd, for next symbol state of mind remains same (i/p is 1). & so on.. so after reading the

the last symbol, whatever is the state of mind that decide whether given string is in L or not

Above same thing can be represented by diagram as follow.

As my mind has two state i will draw two states.



Components

1. states -(Q) set of finite states.
2. Alphabets (Σ) - finite set of symbols.
3. δ - state transition function.

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

δ tells us if i am in state Q & a symbol came Σ. then which is the state i will go to.

δ is a mapping function whose domain is set of state & symbol and range is set of states.

Q \ Σ	0	1
→ q_0	q_1	q_0
q_1	q_0	q_1

state transition table

Page No

4. Initial state (q_0) - Initial state of a machine (before scanning any symbol). This state is unique.

5. set of final states (F) - There are one or many final / accepting state - $F \subseteq Q$

Definition of FSA :- FSA M is 5 tuples

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

Q - is a finite set of states.

Σ - finite set of symbol.

δ is a state transition function.

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

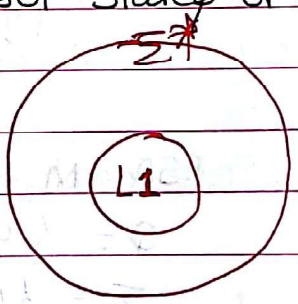
q_0 - Initial state $q_0 \in Q$ (unique).

F - set of final states $F \subseteq Q$.

ex 2. $L1 = \{ x \in \{0,1\}^* \mid x$ has even no of 0's and has even no of 1's $\}$

→ Firstly, try to find all possibilities of states of mind i.e.

1. even 0's even 1's
2. even 0's odd 1's
3. odd 0's odd 1's
4. odd 0's even 1's



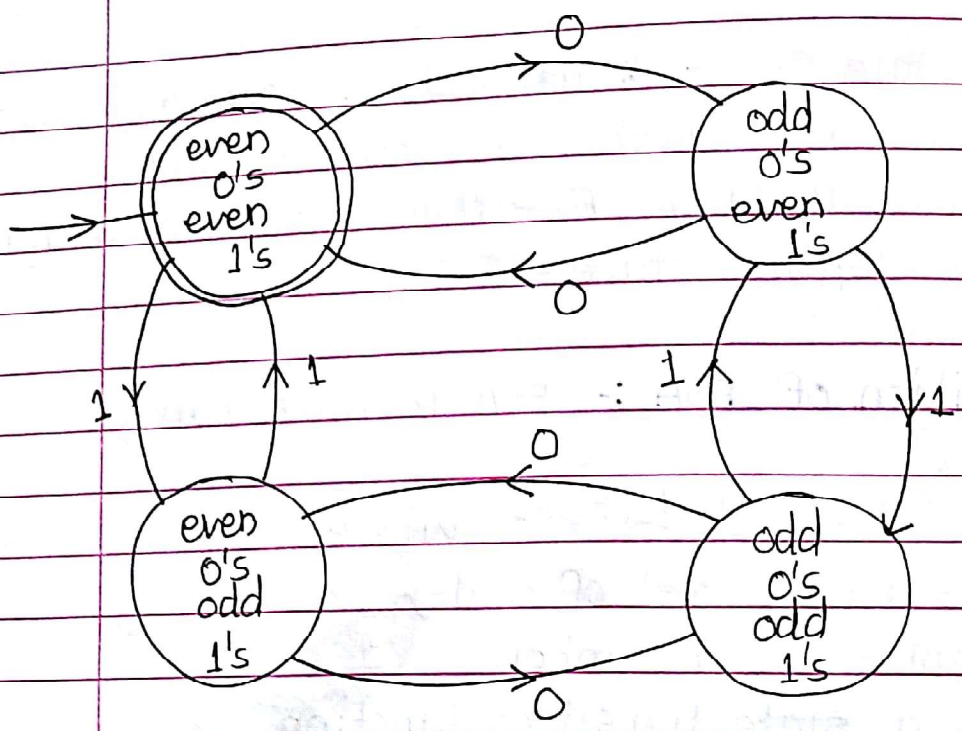
$\delta : Q \times \Sigma \rightarrow Q$

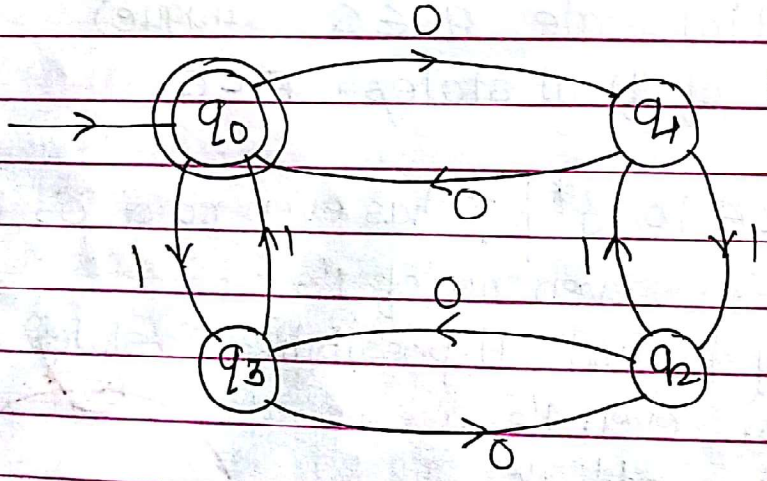
$\delta : Q \times \Sigma^* \rightarrow Q$

hat δ is the function which tells the machine (FSA) will be after scanning the string x. starting from the state q

suppose $L2 = \{00, 1100\}$ Que: is L2 accepted by DFA M

Ans is No because $L2 \subseteq L(M)$ & required is $L2 = L(M)$.





q_0 q_1 q_2 q_1 q_0 q_3
0 1 1 0 1

FSA M for L is

$Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 = q_0$

$F = \{q_0\}$

δ

Q \ Σ	0	1
→ q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_1	q_3
q_3	q_2	q_0

Page No

Let us consider FSA $M = (Q, \Sigma, \delta, q_0, F)$

$L(M) = \{ x \in \Sigma^* \mid \hat{\delta}(q, x) \in F \}$

$L(M)$ - represent language accepted by FSA M

Deterministic Finite state Automata (DFA)

Convention - 1. a, b, $a_1, \ldots a_n$ etc used as elements of Σ

2. $x, y, z, x_1 \ldots$ are strings over Σ

The language accepted by M, called $L(M)$ is defined as

$L(M) = \{ x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in F \}$

Definition of DFA -

A DFA can be described in 5 tuples i.e

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

Q = Finite set of states.

Σ = Finite set of input symbols.

δ = transition function, which takes an argument a state & an input symbol & return a state

q_0 = start state $q_0 \in Q$.

F = A set of final or accepting states, $F \subseteq Q$.

for DFA $\delta : Q \times \Sigma \rightarrow Q$.

In DFA from each state on each input symbols there is exactly one transition.

Regular Language:- A language L_1 is called regular if there is a DFA M such that $L(M) = L_1$

Page No

Every DFA associated with unique Language.

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

$x_1, x_2 \in \Sigma^*$ then

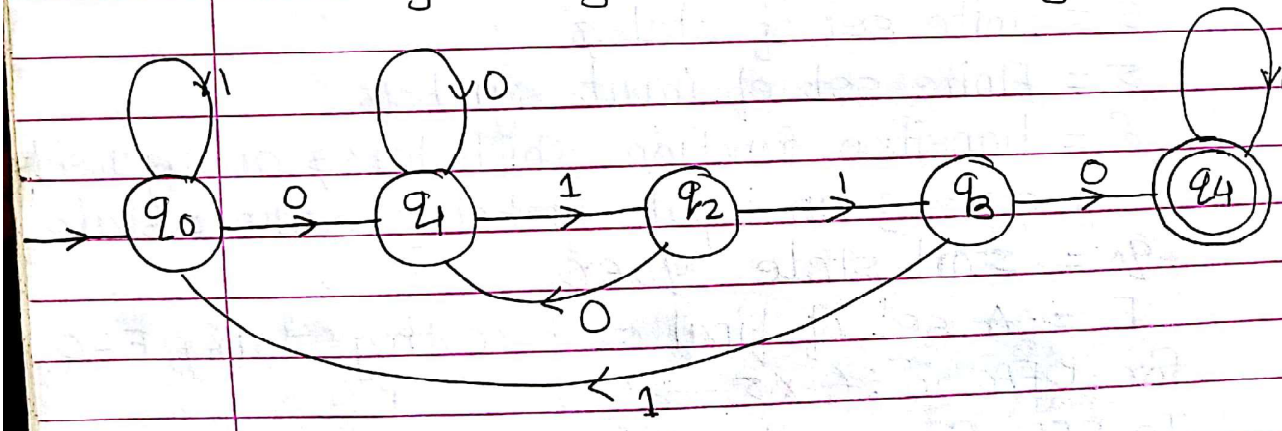
$\hat{\delta}(q, x_1) = \hat{\delta}(q, x_2)$

$\Rightarrow \hat{\delta}(q, x_1 y) = \hat{\delta}(q, x_2 y) \quad \forall y \in \Sigma^*$

ex 3. $L = \{ x \in \{0,1\}^* \mid 0110 \text{ is a substring of } x \}$

Ans - To prove L is regular we need to design DFA

How to design DFA - consider yourself as a Machine (DFA), and you have given a large sequence of input, at a time you are scanning one symbol and keeping some finite amount of information in your head & after scanning last symbol you should be ready with answer.



$q_0\ q_0\ q_0\ q_1\ q_2\ q_3\ q_4\ q_4\ \boxed{q_4} \in F$

$1\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \in L$

$q_0\ q_0\ q_0\ q_1\ q_2\ q_1\ \boxed{q_2} \notin F$

$1\ 1\ 0\ 1\ 0\ 1 \notin L$

Page No

If $L \subseteq \Sigma^*$ then the complement of L is denoted by

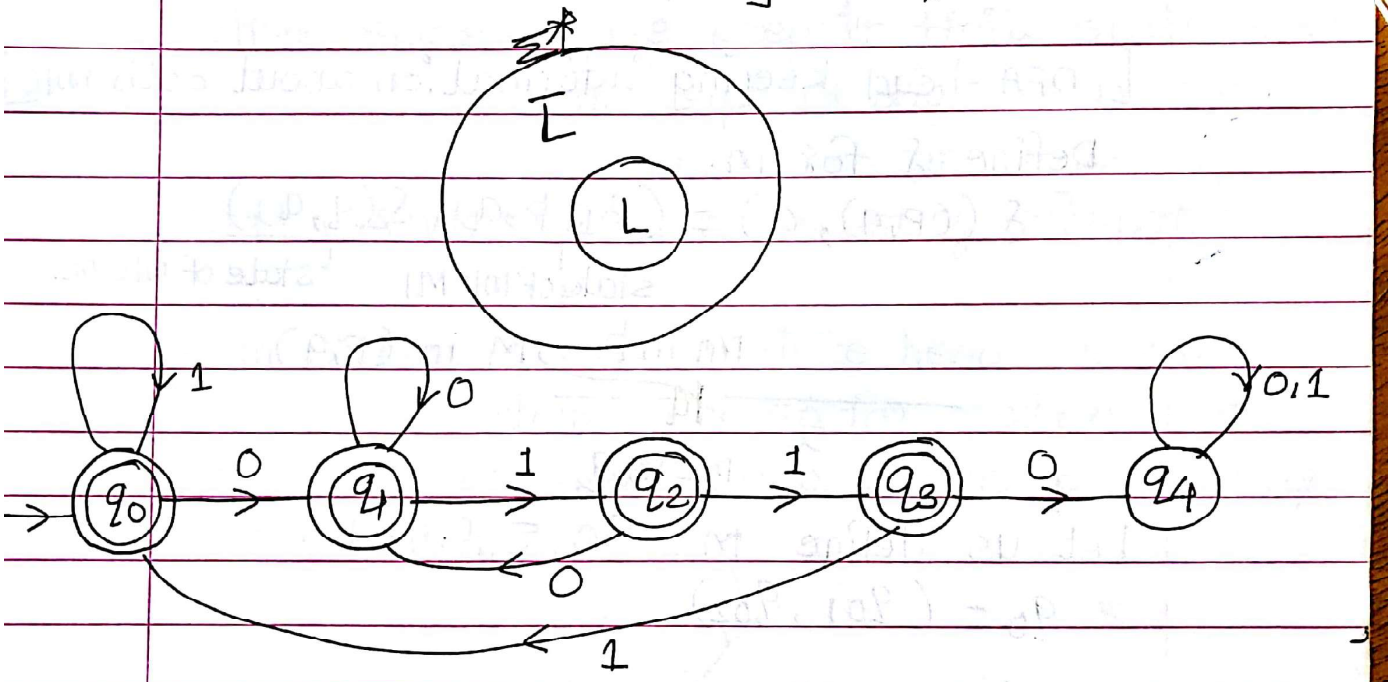
$\bar{L} = \{x \in \Sigma^* \mid x \notin L\}$

$= \Sigma^* - L$

if L is regular then $\bar{L}$ is also regular.

considering example 3

$\overline{L(M)} = \{x \in \{0,1\}^* \mid 0110$ does not occur as a substring in $x\}$



If M accepts L then M_1 obtained from M by switching/Interchanging the set of final states with set of nonfinal state, will accept $\bar{L}$

If $L_1, L_2 \subseteq \Sigma^*$ then $L = L_1 \cup L_2$

suppose, L_1 and L_2 are regular then L is also regular.

since L1 and L2 are regular, There are two DFAs say M1 and M2, accepting these two languages respectively.

$M1 = (Q1, \Sigma, \delta1, q01, F1)$

$M2 = (Q2, \Sigma, \delta2, q02, F2)$

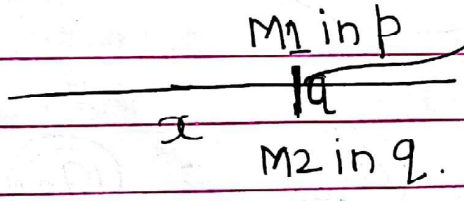
M has $Q = Q1 \times Q2$ as its set of states.

$\therefore Q = \{(P,q) \mid P \in Q1, q \in Q2\}$

[DFA head keeping information about both m/c]

Define δ for M.

$\delta((P,q), a) = (\underbrace{\delta_1(P,a)}_{\text{state of m/c M1}}, \underbrace{\delta_2(q,a)}_{\text{state of m/c M2}})$



Let us define $M = (Q, \Sigma, \delta, q0, F)$

$q0 = (q01, q02)$

We have maintain through the definition of δ. The invariant that

$\forall x, \hat{\delta}((q01, q02), x) = (P1, P2)$

this will be the case iff

$\hat{\delta}_1(q01, x) = P1$ and $\hat{\delta}_2(q02, x) = P2$

Machine M - set of final states

$F = \{(P,q) \mid P \in F1 \underline{\text{or}} \ q \in F2\}$

AND → $L1 \cap L2$

$L(M) = L1 \cup L2$

Page No

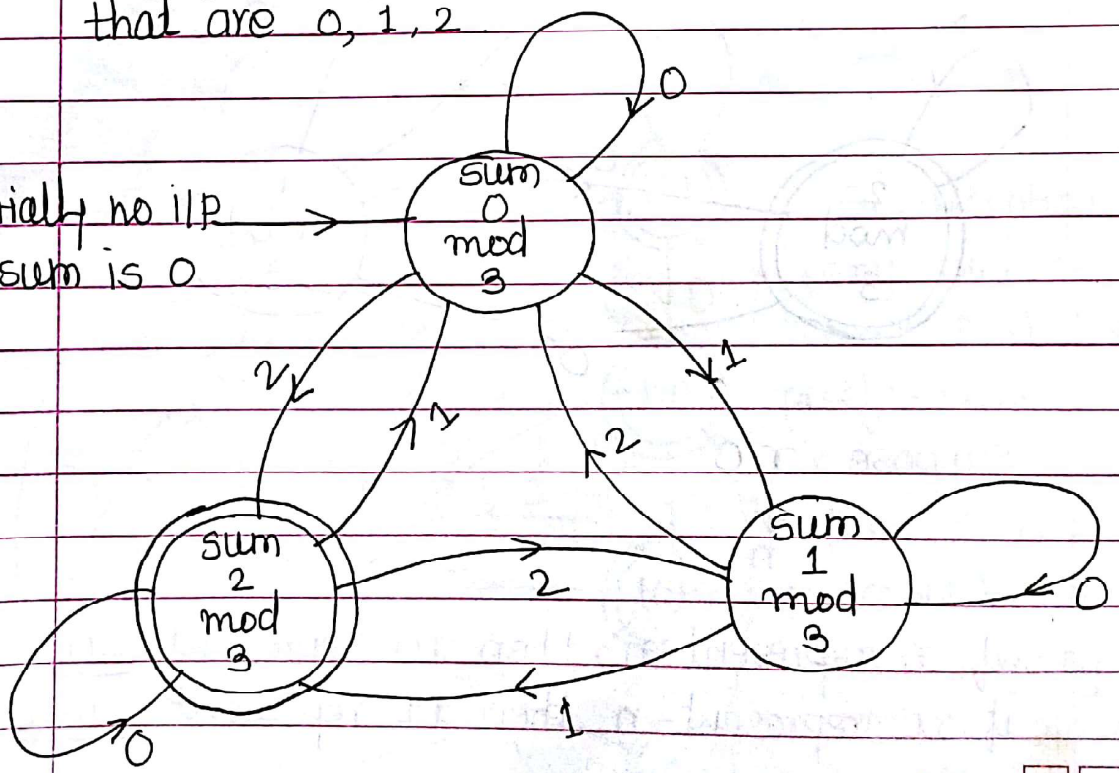
L1 & L2 are regular then show L1 ∩ L2 is also regular. → refer previous proof.

ex 4. ✓ $L1 = \{x \in \{0,1,2\}^* \mid$ The sum of digits in x is 2 mod 3$\}$

- 2101012 — sum = 07 mod 3 = $(1^{mod3}) \notin L1$
- 12101012 — sum = 08 mod 3 = 02 (mod 3) $\in L1$ ✓

Wrong strategy – We keep in finite state head the sum of digit seen so far. ✗ wrong (sum can be arbitrary large & head can hold only finite amount of information)

Right strategy – keep in finite head the sum of digit seen so far modulo 3. so at any given time there are only three possibilities that are 0, 1, 2



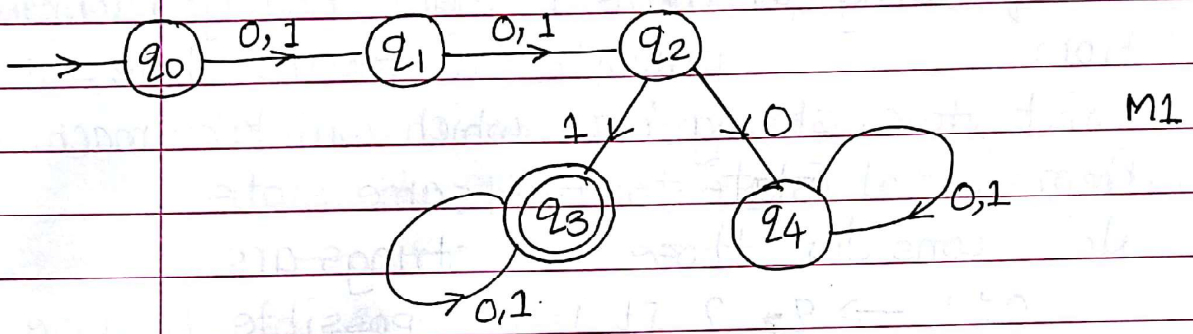
Page No

ex 6. $L1 = \{ x \in \{0,1\}^* \mid$ The Third bit from the left end is 1 $\}$

$L2 = \{ x \in \{0,1\}^* \mid$ The Third bit of x from its right end is one $\}$

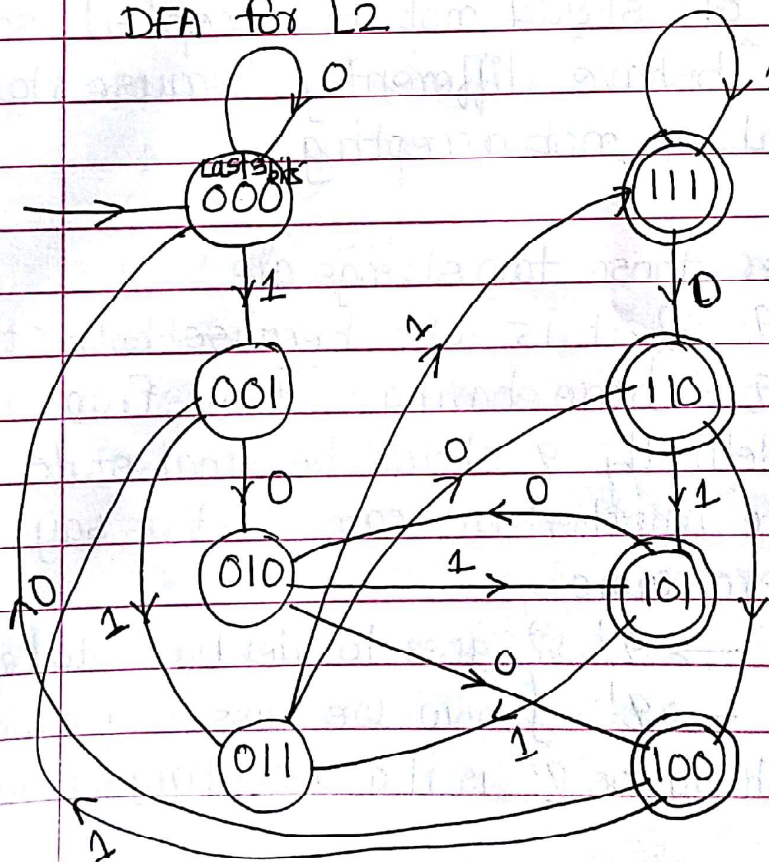
→ 1000100 $\notin$ L1 (3rd bit from left) 1000100 $\in$ L2

DFA for L1



M1

DFA for L2



for accepting L2 any DFA needs at least 8 states.

proof:- suppose M accept L2 & has less than 8 states consider running M with following 8 string 000, 001, 010, 011, 100, 101, 110, 111

String states

→ 000 → at the end M reached at q_1

001 → q_2

010 ⋮

011

101

110

111

100

there are total no of states are seven or less no of string we have is eight. now you remember from _____ principle, it means there will exist two string here which will take machine from initial state to the same state.

Now consider those two strings are

011 → q
100 → q
} It is not possible because 100 should be accepted where as 011 should not be accepted. so m/c cant behave differently. because state is either final or not accepting.

now consider those two strings are.

101 → q
111 → q
} it is ok because both strings are having 3rd bit from right is 1. and definetly q should be final state now imagine another bit came Let us say another zero came

$101^{20}0$ → q'
$111^{20}0$ → q'
} goes to identical state now we have a problem now what should be q' is it a accepting or non accepting

so what ever we have contradit initially that is wrong hence L2 required at least 8 states.

Time complexity of algorithm which implement DFA is $O(n)$ where n = Length of input.

ex 8 show that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Ans: $000111 \in L$

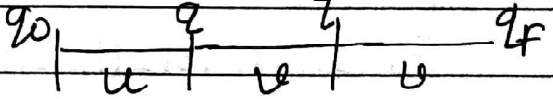
$001 \notin L$ Proof method - Proof by contradictio

Assume L is accepted by a DFA. Let that DFA have some k states.

Imagine there is very long string which is in L now consider the sequence of states M goes throw.

q_0 q_{i1} q_{i2} q_{i3}

x x x x x ..... x x x x



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

$\hat{\delta}(q_0, u) = q.$

$\hat{\delta}(q, v) = q$

$\hat{\delta}(q, w) = q_f \in F$

string uvw take m/c from q_0 to final state.

$\therefore uvw \in L$

what about $uvvw \in L$

$\vdots$

$uv^i w \in L \quad i \geq 0$

Page No

Pumping Lemma for regular languages:-

Let L is regular then there is a constant k (depends only on the language L) such that for all $x \in L$ & $|x| \geq k$,

There exist u,v,w satisfying (string u, v, & w.)

1] $uvw = x$

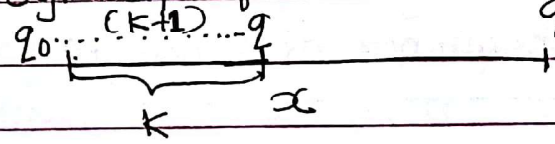
2] $|uv| \leq k$, $|v| > 0$ (The length of v is non zero)

3] For all i, $i \geq 0$, $uv^i w \in L$

Proof:- since L is regular. There is DFA M to accept L

Let k be the number of states in M.

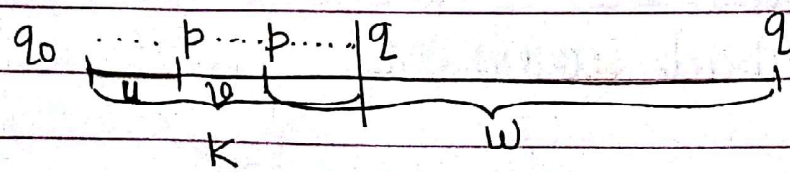
Let x is in L & $|x| \geq k$, Now consider the first k symbols of x



$\hat{\delta}(q_0, x) = q_F \in F$

For an n length string, the length of sequence of state the DFA goes through is (n+1)

∴ by pigeon-hole principle there will be at least one state which will occur more than once in this sequence on first k symbol on string x.



$\hat{\delta}(q_0, x) = q_f \in F$

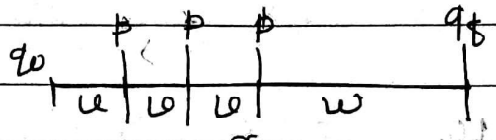
$\hat{\delta}(q_0, u) = p$

u = is a string of x which took M into state p which occure one more

$\hat{\delta}(p, v) = p$
$\hat{\delta}(p, w) = q_f$

Now let us determine the behaviour of M for the string uvvw

$\hat{\delta}(q_0, uvvw) = q_f$



$\hat{\delta}(\hat{\delta}(\hat{\delta}(\hat{\delta}(q_0, u), v), v), w) = q_f$

$\Rightarrow uvvw \in L$ → if i put any no. of copies of v still that string is in L.

for $\hat{\delta}(q_0, uw) = q_f$

if value of $i = 0$

uv^0w, $v^0 = \epsilon$ ∴

$uw \in L$

q_0 p q_f

ex 9. $L = \{x \in \{0,1\}^* \mid 0^n 1^n, n \geq 0\}$ is not regular.

Proof → Let L be regular. Let K be the pumping lemma constant for L. Consider the string $x = 0^K 1^K$.

From pumping lemma, there exist u, v, w satisfying the three condition of lemma.

$$\underbrace{000\ldots0}_{K}\underbrace{111\ldots1}_{K}$$

condition 1 & 2 imply that v consist only of zeros because uv it self consist of only zero.

considering condition 3. where $i = 0$

i.e $uw \in L$ ..... condition 3

In uw no of zero is strictly less than the no of 1's so by definition of L $uw \notin L$

Page No □□□

∴ so we have a contradition so what ever assumption we had made is wrong.

∴ L is not regular.

ex 10. $L = \{x \in \{1\}^* \mid x = 1^{n^2}, n \geq 1\}$ is not regular

Proof $n=1$ $1 \in L$ *The length of string must be perfect square

$n=2$ $1^4 = 1111 \in L$

$n=3$ $1^9 = 111111111 \in L$

$11111 \notin L$ (1^5)

---

suppose L is regular. Let K be the pumping lemma constant for L.

Consider $1^{K^2} \in L$

$1^{K^2} = uvw$ (u, v, w)

$|uv| \leq K$, $|v| \leq K$, $|v| > 0$ v is a Non empty string whose length is minimum 1 & maximum K

$uvvw \in L$ — what we can say about length of uvvw

$|uvvw| \leq K^2 + K$

$1^m \in L$, $K^2 < m > (K+1)^2$

$1^{K^2} \xrightarrow{\text{next}} 1^{(K+1)^2} = K^2 + 2K + 1$ *The length of uvvw is K^2 & maximum length of v is K

∴ $K^2 + K \leq K^2 + 2K + 1$ ∴ $|uvvw| \leq K^2 + K$.

so we have a contradition so what ever assumption we had made is wrong.

∴ L is not regular.

Is the converse of pumping lemma true?

Converse:- If L satisfies the condition of pumping lemma then L is regular.

→ Converse is false.

Proof is by conter example :- L

$$L = \{a^i b^j c^k \mid i,j,k \geq 0 \text{ if } i=1 \text{ then } j=k\}$$

abbbccc $\in L$	suppose $k=3$, $x \in L$, $\lvert x\rvert \geq 3$.
aaabccc $\in L$	if x has exactly 2 a's

so aabbb.... bcc.. .. c

Let $u = \epsilon$, $v = aa$, w = rest of x.

otherwise, choose $u = \epsilon$, v as the first symbol and rest as w.

so for $uvvw \in L$.

$aaaaw \in L$.

$uw \in L$.

∴ It satisfy all condition of lemma so it is regular, but actually this language L is not regular

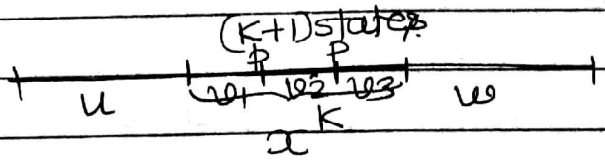
Generalized Pumping Lemma for regular Language:-

Let L be the regular language then there is a constant k (depending on L) such that for all $x \in L$, $\lvert x\rvert \geq k$ and $x = uvw$, where $\lvert v\rvert = k$ We have that there exist v_1, v_2, v_3 such that $v = v_1 v_2 v_3$, and for all $i \geq 0$ the string

$\lvert v_2\rvert > 0$ $uv_1v_2^i v_3 w \in L$

classmate

Suppose L is regular then there is DFA M to accept L. Let M have K states.



ex 11. Prove that

$L = \{ a^i b^j c^k \mid i,j,k \geq 0 \text{ if } i=1 \text{ then } j=k \}$

is not regular (Use:- Generalized pumping lemma)

→ Let L be regular, Let K be the constant of Lemma

Now consider string

$ab^K c^K \in L$ (because $i=1$ the $j=K$)

break up string in u, v, w.

$u = a \quad v = b^K \quad w = c^K$

Now: (length is between 1 to K)

$uv_1v_2v_3w$

a — b (all b) — c (all c)

on pumping we will get the string. (No of b get increase case)

$ab^l c^K$ where $l > K$

so lemma says $ab^l c^K \in L$ but it contradict the definition of Language.

∴ L is not regular.

General structure to prove Language is not regular using pumpping lemma.

step 1: consider given L is regular.

step 2: Given any K, we choose an xe, now given any u, v, w satisfying constrients.

step 3: We find $i \geq 0$ such that $uv^i w \notin L$

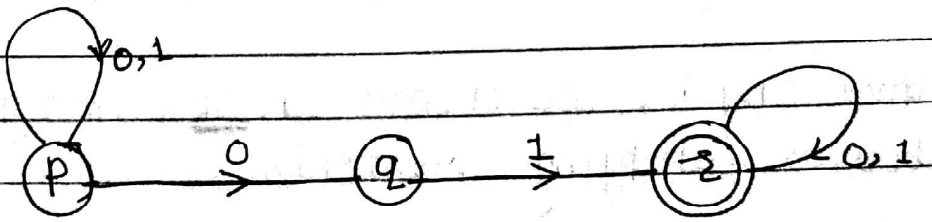
Nondeterministic Finite Automata (NFA)

- NFA is generalization of Finite Automata.
- NFA also will not accept anything other than regular Languages.

Definition of NFA:-

NFA can be described in 5 tuples
$M = (Q, \Sigma, \delta, q_0, F)$ where.
Q is the finite set of states of M.
Σ is the finite set of symbols
δ : is a transition function.
q_0 - is the initial state of M.
F - set of final or accepting states, $F \subseteq Q$.
$\delta : Q \times \Sigma = P(Q)$ (Power set of Q)
set of all subsets of Q.
ex $P(\{1,2\}) = \{\phi, \{1\}, \{2\}, \{1,2\}\}$
$\delta(q,a) = R \subseteq Q$, In NFA from each state on each input symbol, there can be zero, one or more transitions

ex 01.



$L(M) = \{ x \in \{0,1\}^* \mid x \text{ contain substring } 01 \}$

$\delta(P,0) = \{P, q\}$ $\quad \delta(q,0) = \phi$
$\delta(P,1) = \{P\}$ $\quad \delta(q,1) = \{2\}$
$\delta(2,0) = \{2\}$ $\quad \delta(2,1) = \{2\}$

Page No

Language accepted by NFA :- Let consider NFA M then L(M) is the Language accepted by M informally defined as.

$$L(M) = \{ x \in \Sigma^* \mid x \text{ can take } M \text{ from } q_0 \text{ to one of its final states} \}$$

for input x we are going to find all possible states that NFA can go to and if one of those states if included in final states then we are saying that the string x is accepted by NFA M.

$$\therefore \hat{\delta} : Q \times \Sigma^* \longrightarrow P(Q)$$

$\hat{\delta}(P, x) = R \subseteq Q$ then definition of $\hat{\delta}$ will insure that R is the set of all states M can go to on string x from state P

$\hat{\delta}(P, \epsilon) \longrightarrow \{P\}$ — base case.

We need to define $\hat{\delta}(P, xa)$ using the value for $\hat{\delta}(P, x)$

Let $\hat{\delta}(P, x) = R \subseteq Q$

$$\hat{\delta}(P, xa) = \bigcup_{r \in R} \delta(r, a)$$

Formally

$$L(M) = \{ x \in \Sigma^* \mid \hat{\delta}(q_0, x) \cap F \neq \phi \}$$

Page No

* If L is accepted by DFA, then L is accepted also by an NFA.
- Reason :- because every DFA is also an NFA.

There is no language which is accepted by NFA and Not by DFA.

Conversion of NFA M to equivalent DFA M'

NFA M to DFA M' -

Idea is → the states of DFA M', on reading x, will remember in its states, the set of states NFA M could have been on reading x.

Let

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

$M' = (Q', \Sigma, \delta', q_0', F')$ → is a set of power set of P(Q)

Q' - The set of states of DFA to be the P(Q)

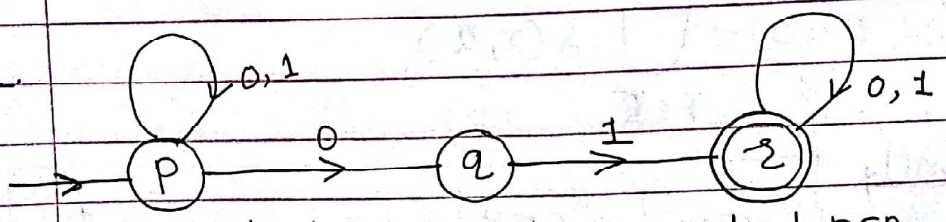
$\delta'(R, a) = R'$ $\quad R \subseteq Q$.

where $R' = \bigcup_{r \in R} \delta(r, a)$

$q_0' = \{q_0\}$

$F' = \{R \subseteq Q \mid R \cap F \neq \phi\}$

ex 02.



convert above NFA to equivalent DFA.

Equivalent DFA M' :

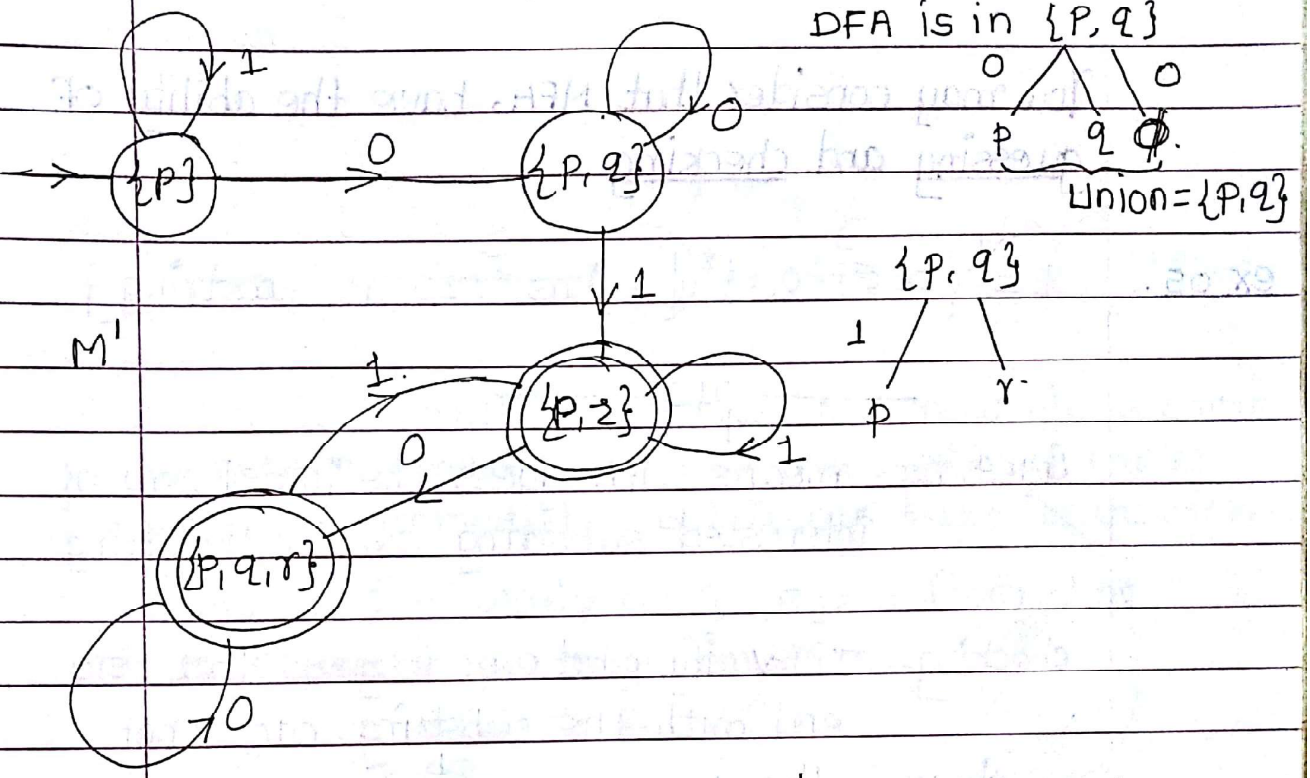
so from above defⁿ

Initial state for DFA is p & we should always start with Initial state.

classmate

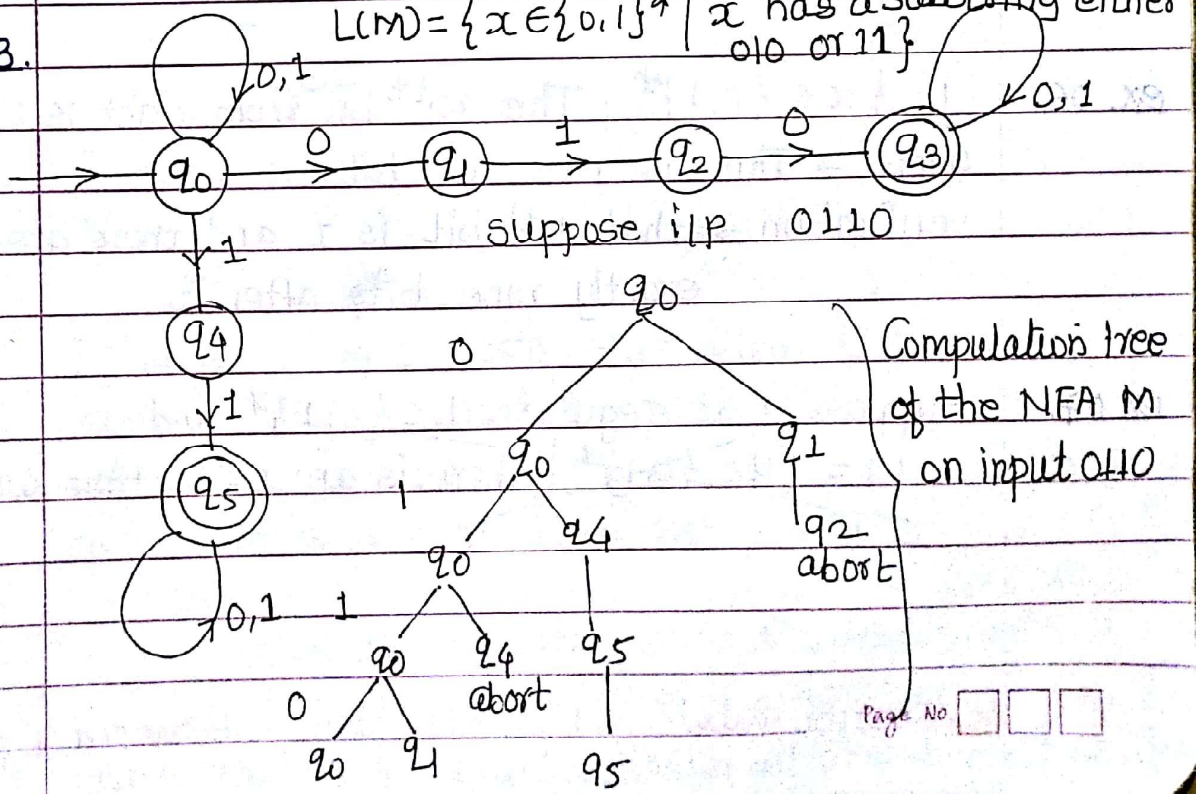
	0	1
→ {P}	{P,q}	{P}
{P,q}	{P,q}	{P,r}
{P,r}	{P,q,r}	{P,r}
{P,q,r}	{P,q,r}	{P,r}

- so we found p another state which is reachable from p. so now consider that state q



ex 03.

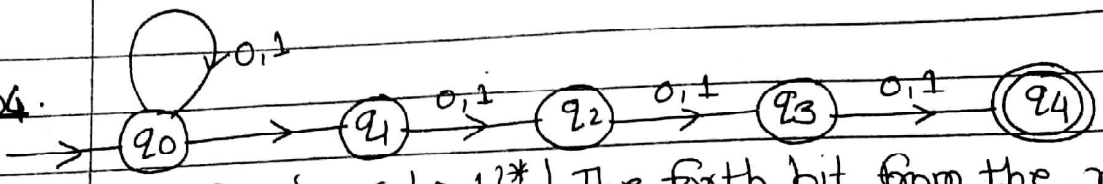
L(M) = {x ∈ {0,1}* | x has a substring either 010 or 11}



Computation tree of the NFA M on input 0110

Page No

ex 04.



$L(M) = \{ x \in \{0,1\}^* \mid$ The forth bit from the right end is 1 $\}$

You may consider that NFA, have the ability of guessing and checking

ex 05. $L = \{ x \in \{0,1\}^* \mid x$ has 010 as substring $\}$

——— 010 ———

Guessing - means NFA guess the initial zero of interested substring come in the string.

checking → It varify what ever is guess that zero end with the substring 010 or not.
where is the first zero of sub string 010

ex 06. $L = \{ x \in \{0,1\}^* \mid$ The 10^{th} bit from right is 1 $\}$

Guess → This is the 10^{th} bit.

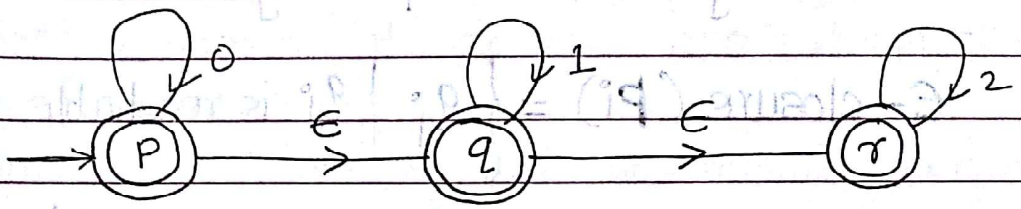
verification → that 10^{th} bit is 1 and there are exactly nine bits after it.

ex 07. ~~suppose L is regular, $L \subseteq \{0,1\}^*$ and $L1 = \{ y \in \{0,1\}^* \mid$ There is an $x \in L$, from which is~~

NFA with ϵ-transition :-

- It is a generalization of NFA.
- It allow transition not only on symbols but also on ϵ (epsilon)

ex



consider initially system is in p. now i/p 2 came. now m/c have choses to not use that input symbol immediatly, but it can take transition over ϵ & it would be q, again from that q it take another ϵ to go to state r

$\therefore \epsilon\epsilon 2 \Rightarrow 2$

i/p = 2

p r

ϵϵ2

suppose i have a string

a1 a2 ..... an

I have a m/c NFA that allow ϵ transition above string is equal to padding n number of ϵ before the first symbol of string or in between two symbol or after the last symbol.

$\therefore \epsilon\epsilon a1\epsilon\epsilon\epsilon a2 \ldots \epsilon\epsilon\epsilon an\epsilon\epsilon\epsilon\epsilon$

so when i want to find the states through which NFA can go I should consider all possible ϵ transitio

$00112 \in L$

$211 \notin L$

$L(M) = \{ 0^i 1^j 2^k \mid i,j,k \geq 0 \}$

A Language L is accepted by an NFA with ϵ-transition if and only if L is accepted by a DFA.

ϵ- closure $(p_i) = \{ q_j \mid q_j$ is reachable from p_i using 0 or more ϵ transition $\}$

ϵ- closure $(p) = \{ p, q, r \}$

↓

We can reach to p with zero ϵ- but we are there only.

ϵ- closure $(q) = \{ q, r \}$.

ϵ- closure $(r) = \{ r \}$.

Suppose M is NFA with ϵ transition.

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

$\delta : Q \times \Sigma \cup \{\epsilon\} \longrightarrow P(Q)$

$\hat{\delta} : Q \times \Sigma^* \longrightarrow P(Q)$

$\hat{\delta}(p, x)$ = The set of all states M can be in starting from p an Input x }

$\hat{\delta}(p, \epsilon) = \epsilon$- closure $(p) = \{ p, q, r \}$

Page No

ϵ-closure(P) , $P \subseteq Q$. (P: subset of Q.)

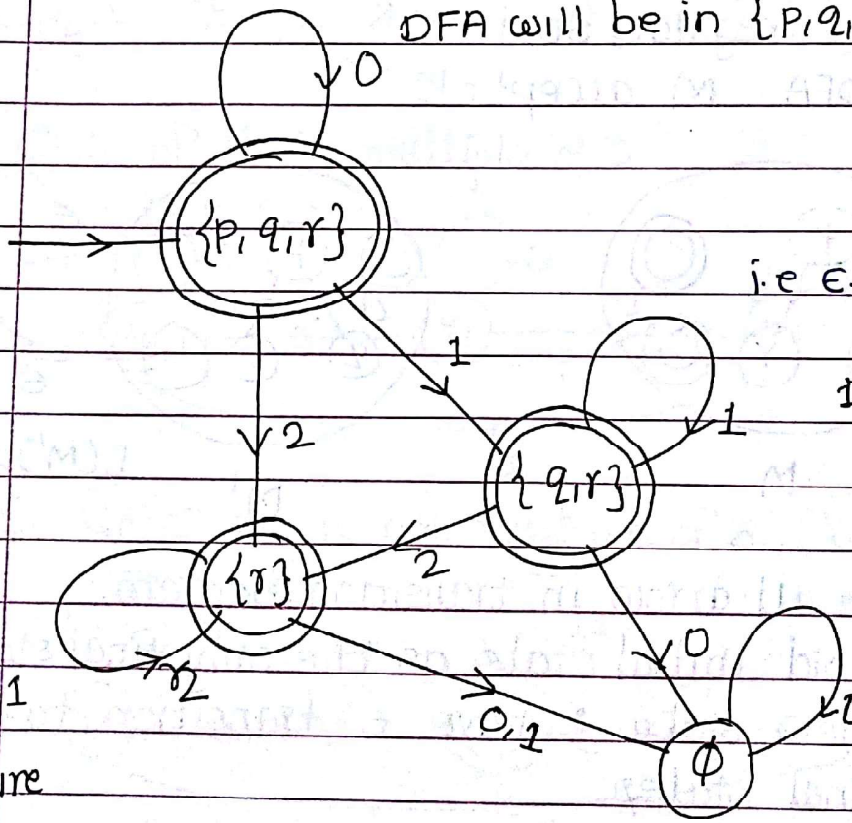
$$= \bigcup_{q_i \in P} \epsilon\text{-closure}(q_i)$$

Conversion of NFA with ϵ to equivalent DFA :-

In general DFA is having states that are subset of states of (NFA with ϵ) M.

There are three states so initially DFA can be in either p, q or r (without seeing any I/P).

DFA will be in {p, q, r} states.



{p,q,r}
0
{p, ϕ, ϕ}
then consider all ϵ
i.e ϵ-closure(p) = {p,q,r}
{p,q,r}
1
ϕ q ϕ
take closure of q
i.e {q,r}

{p,q}
1
q ϕ
ϵ closure
{q,r}

{p,q,r}
2
ϕ ϕ r
ϵ-closure(r)
i.e {r}.

{q,r}
0
ϕ ϕ

Reversal of the string :-

Suppose given string is x then reversal of x is denoted as x^R

$x = 011$ then $x^R = 110$

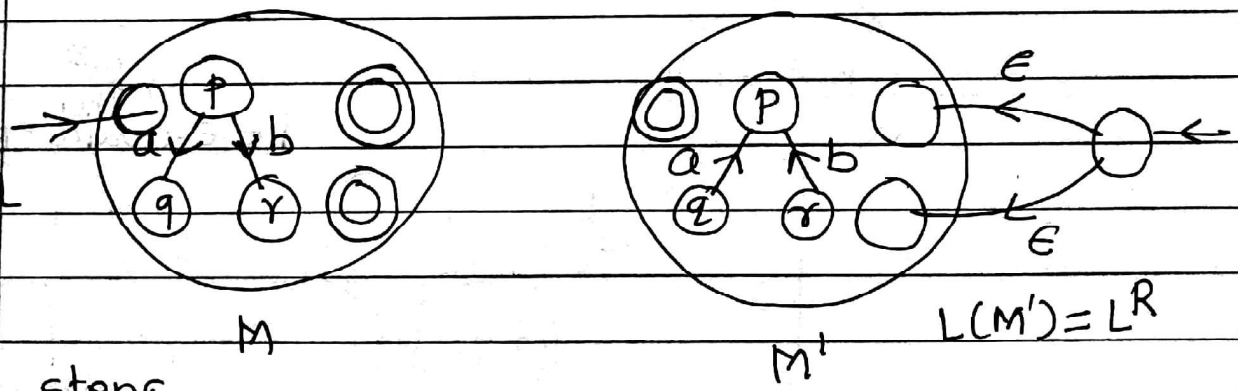
Reversal of Language -

L is language, $L \subset \Sigma^*$ then L^R is revesal of L.

$$L^R = \{ x \in \Sigma^* \mid x^R \in L \}$$

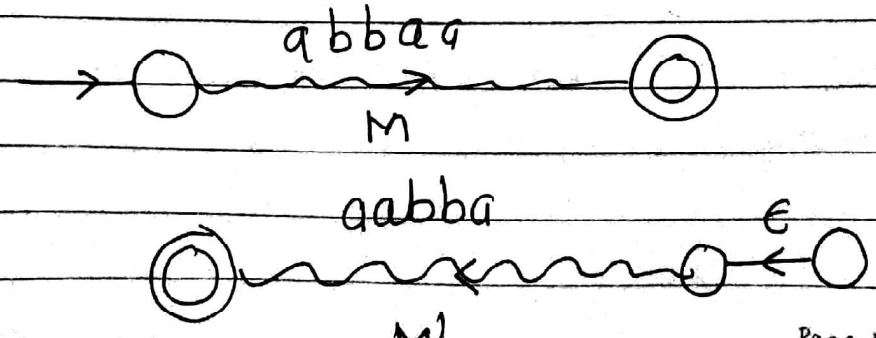
if L is regular, so is L^R

Proof - Let DFA M accept L



steps

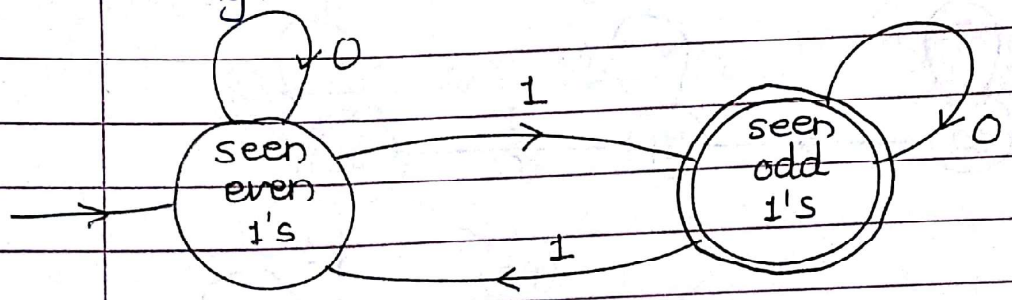
1] Reverse all arrow in transition diagram.
2] Mark old initial state as the only final state
3] Add new state & have ϵ transition to the -old final states.



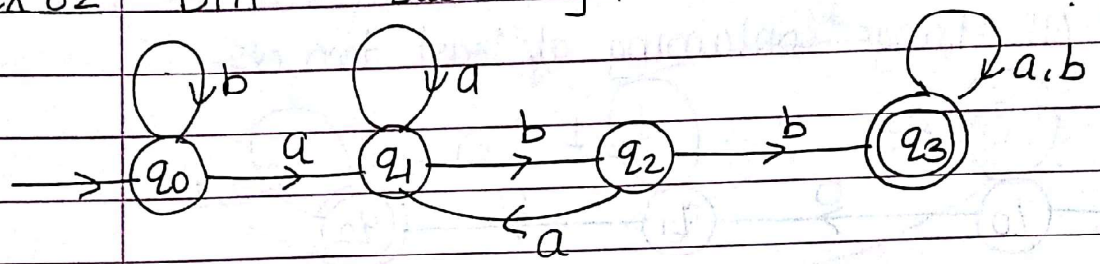
classmate

Examples

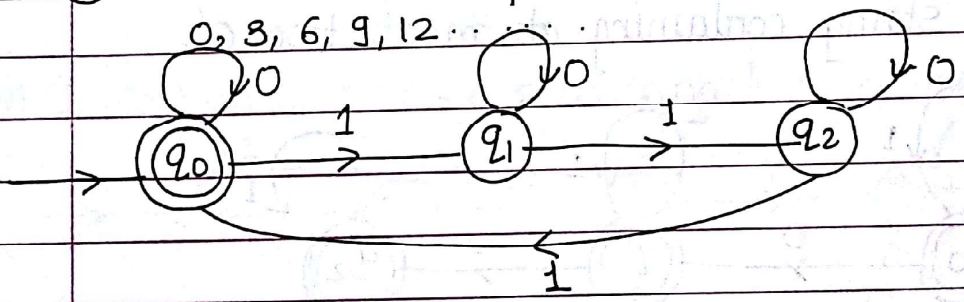
ex 01 Design DFA for $L = \{x \in \{0,1\}^* \mid x \text{ has odd no. of } 1's\}$



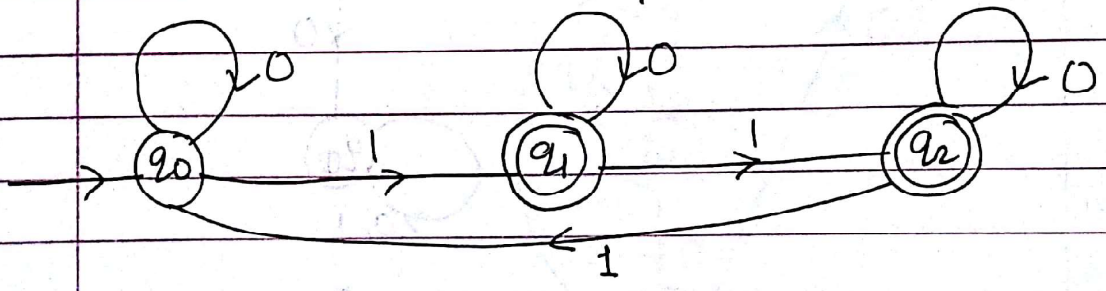
ex 02 DFA → substring abb.



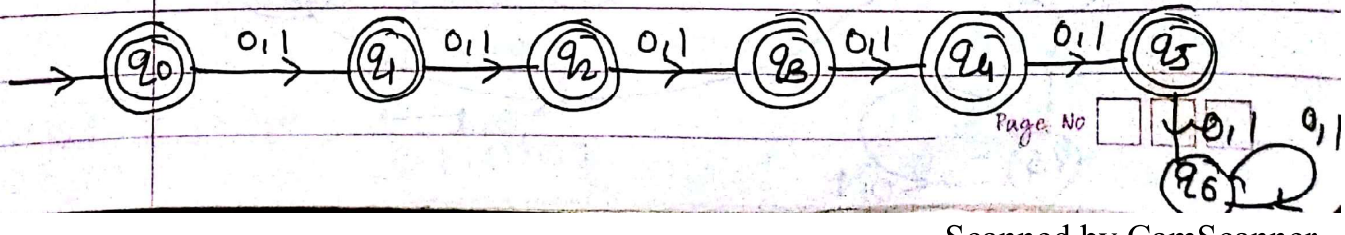
ex 03 (a) No of 1's is multiple of 3.



(b) No of 1's is not multiple of 3. (complements)

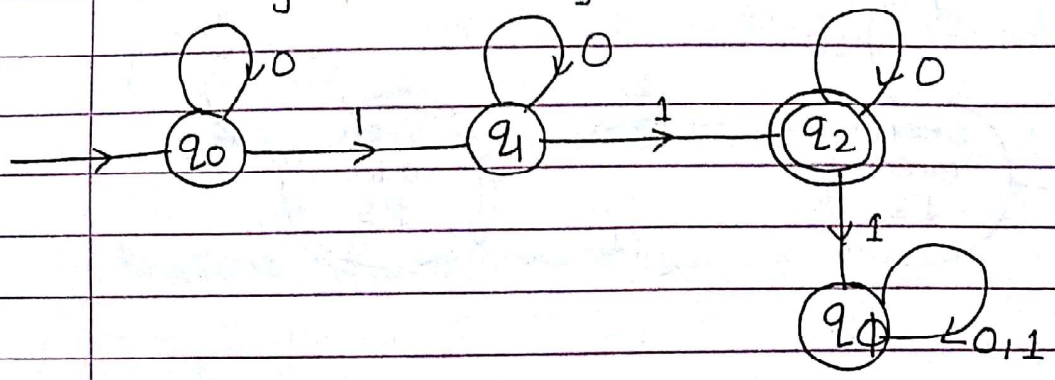


ex 04 Draw DFA for the following language over $\{0,1\}$.
i] All strings of length at most five.

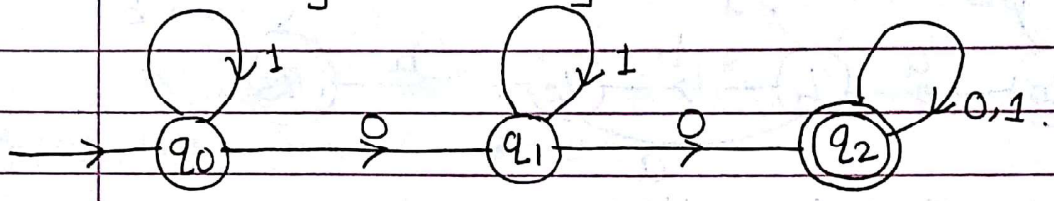


Page No

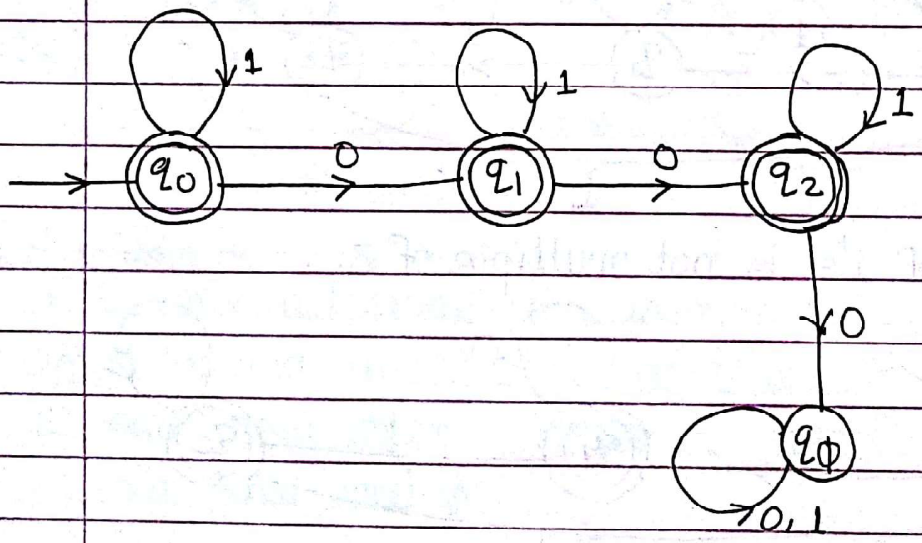
ii] All strings with exactly two 1's.



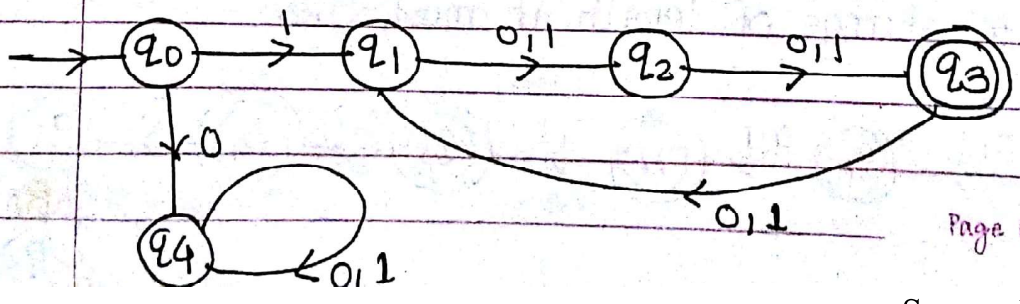
iii] All strings containing at least two 0's.



iv] All string containing at most two 0's

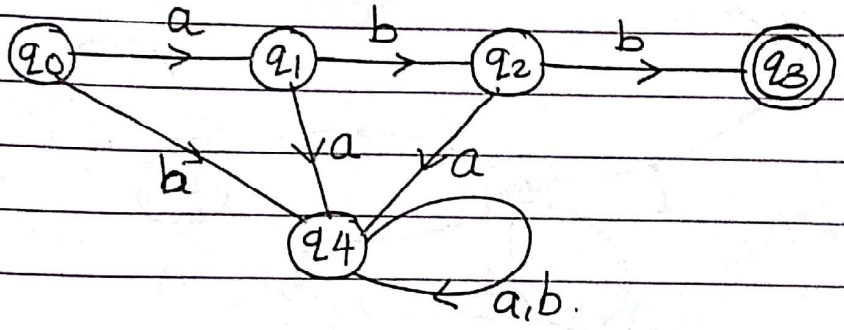


v] All strings starting with 1 and length is divisible by 3

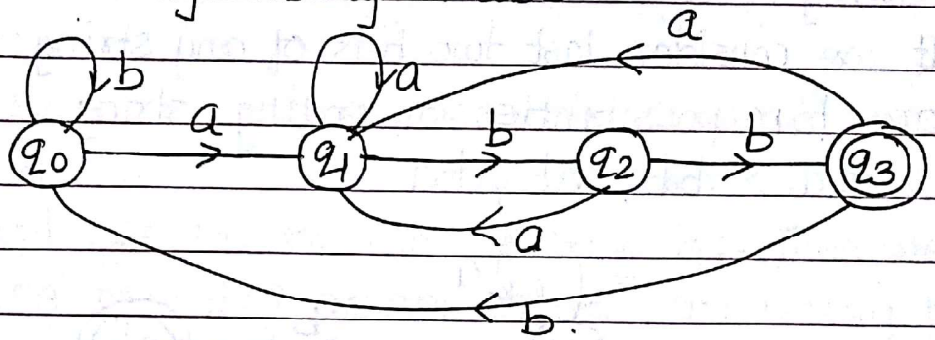


Page No

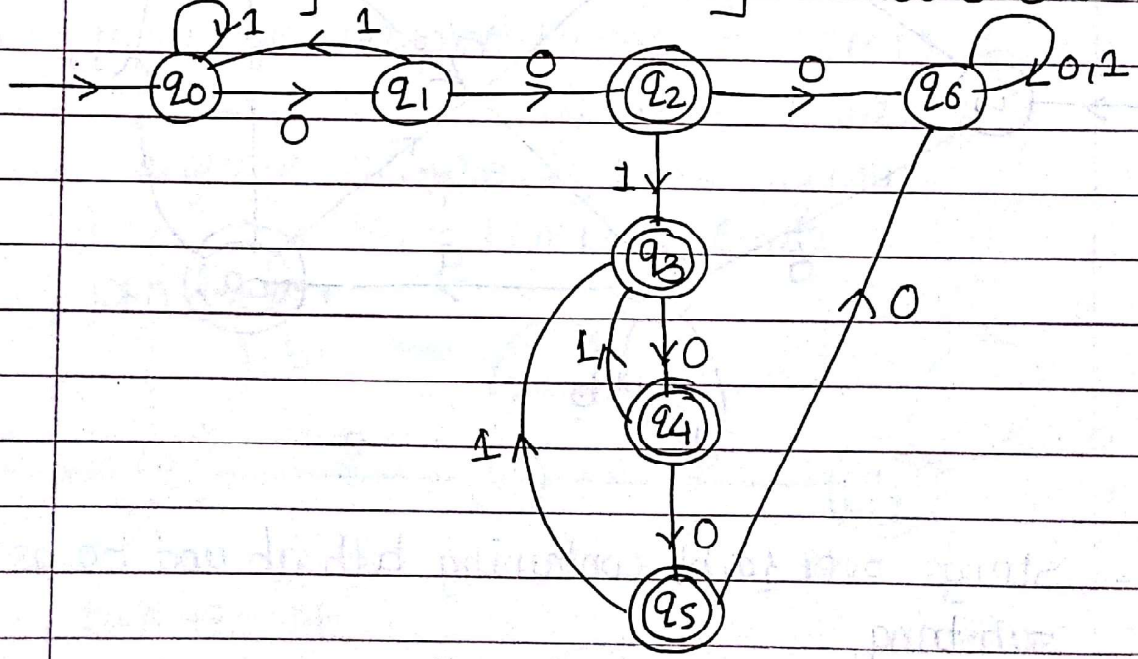
vi] All strings starting with abb.



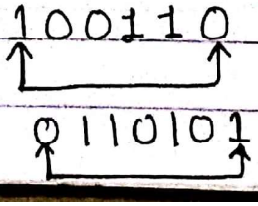
vii] All strings ending in abb.



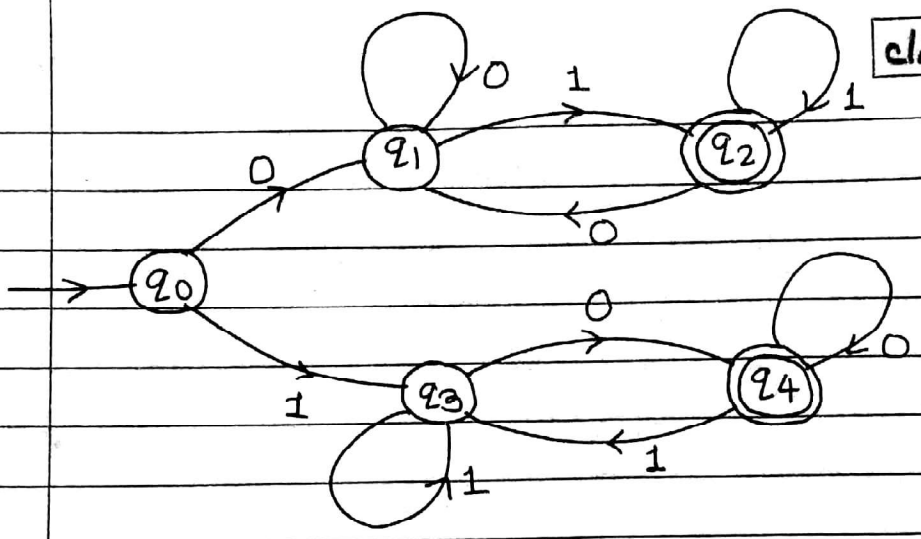
viii] containing 00 as a substring but not 000 as a substrin



ex 05 L containing string in which leftmost symbol differ from rightmost symbol, ($\Sigma = \{0,1\}$)



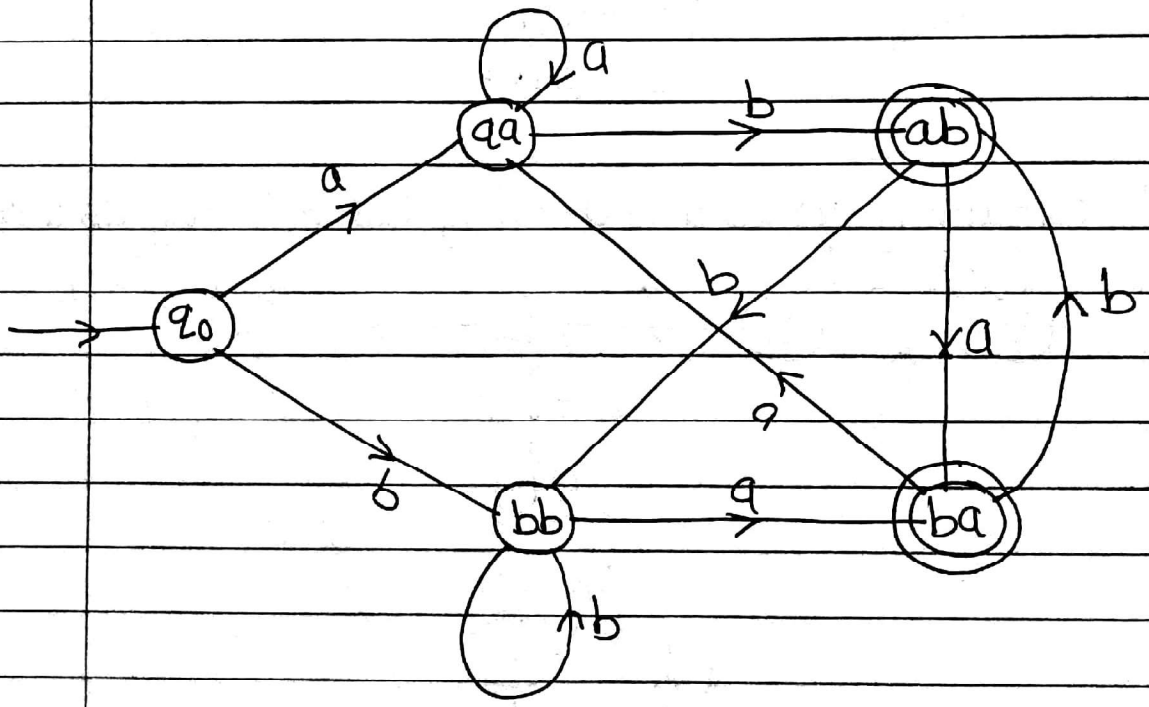
Page No



ex 06 String over {a, b} ending in either ab or ba.

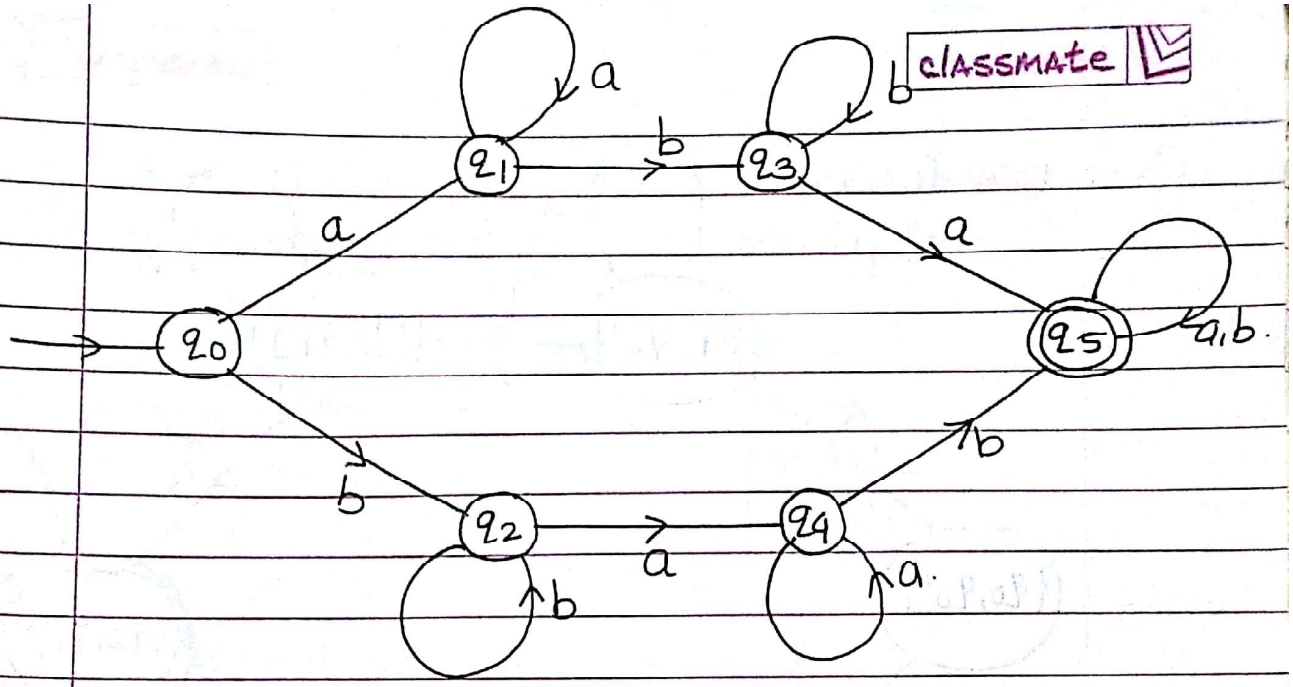
If we consider last two bits of any string; then there are four possibilities of ending string.

aa, bb, ab, ba.



ex 07 - strings over {a, b} containing both ab and ba as a substring.

i.e abα OR bab



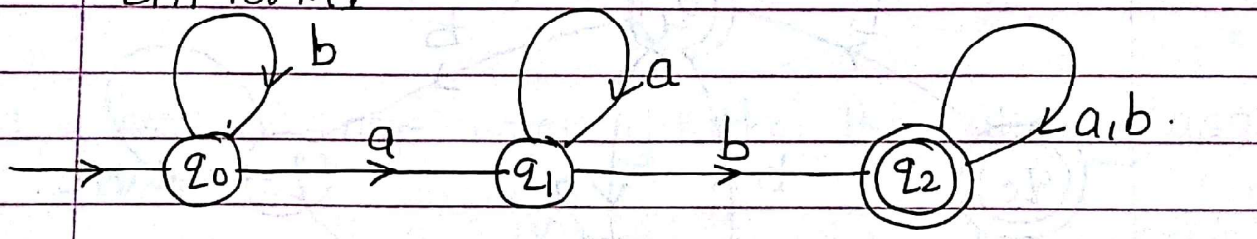
second way to solve above problem -

We can construct DFA M1 for the substring ab. and DFA M2 for the substring ba. Then we can construct DFA M which accept ~~string~~ containing both ba & ab as a substring.

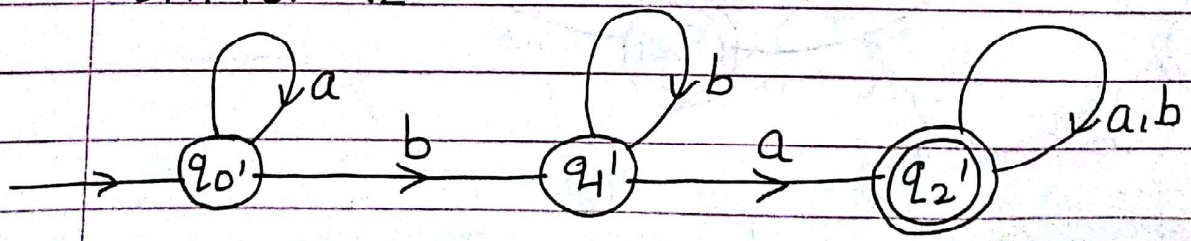
Language accepted by M1 is L(M1) ∩
Language accepted by M2 is L(M2) &
Language accepted by M is L(M)

$$L(M) = L(M1) \cap L(M2)$$

DFA for M1

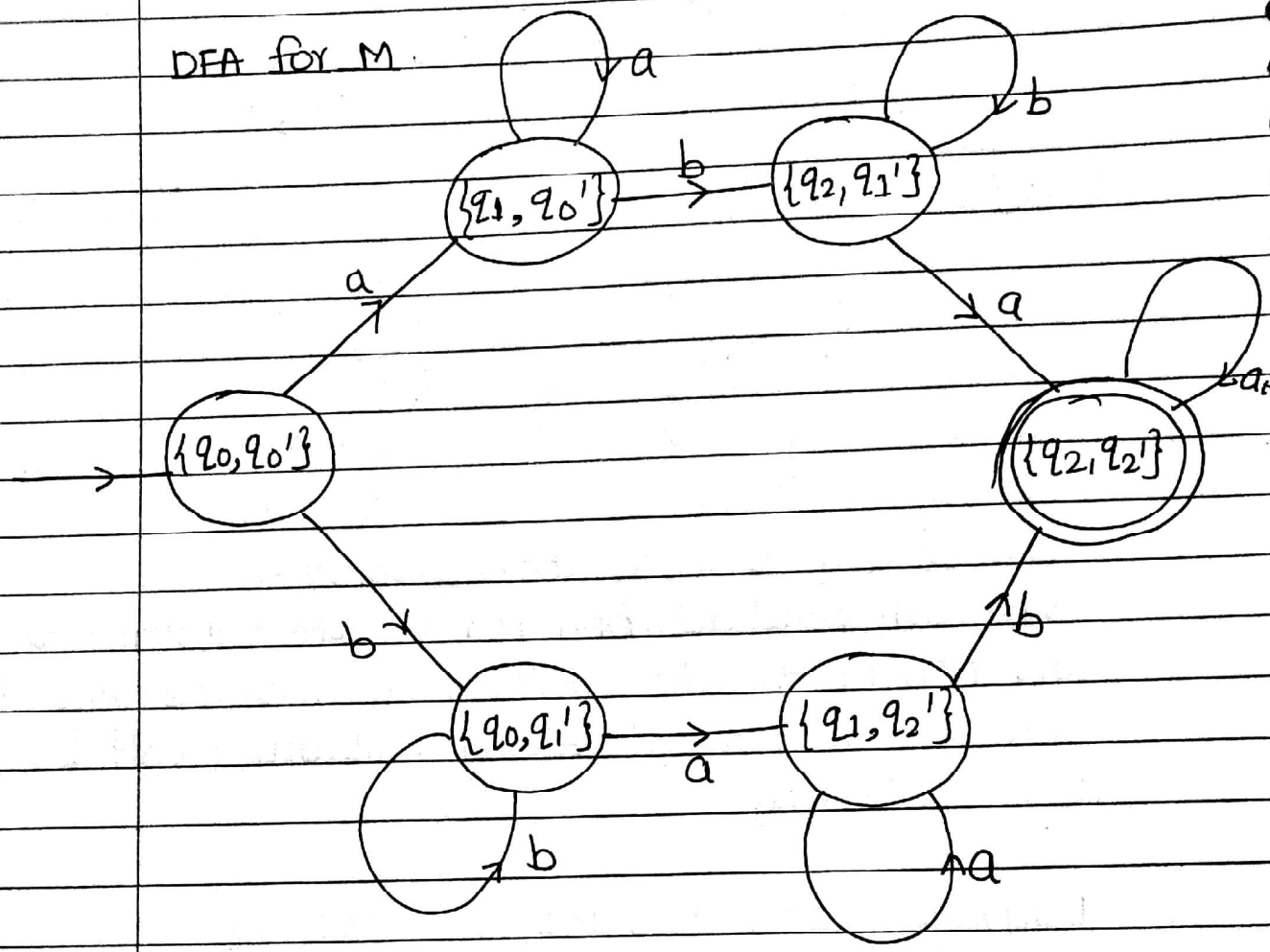


DFA for M2

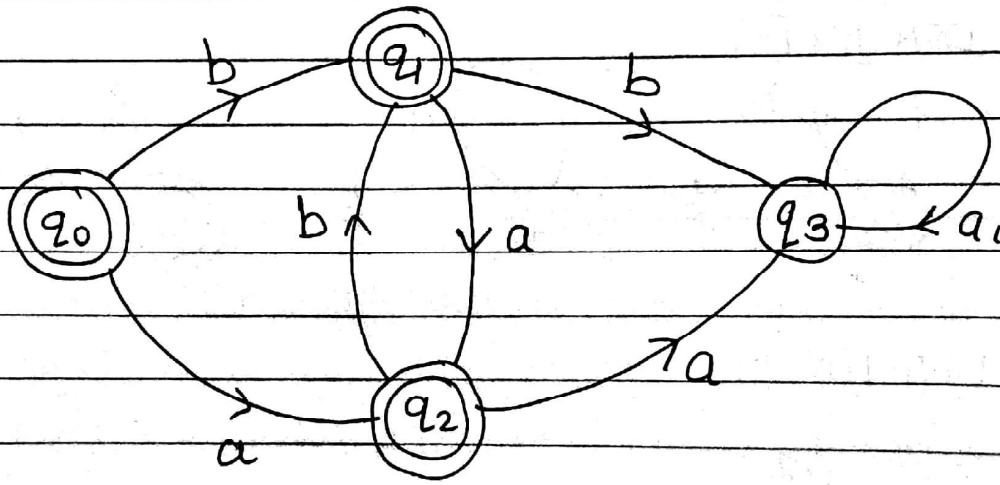


Page No

DFA for M.

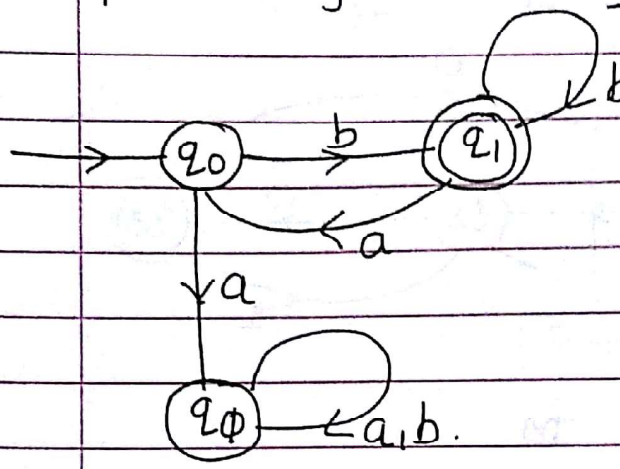


ex 08: $\Sigma = \{a, b\}$ containing neither aa nor bb as a substr
$a \in L$, $b \in$, $\varepsilon \in L$, $ab \in L$, $abab \in L$, $baba \in$



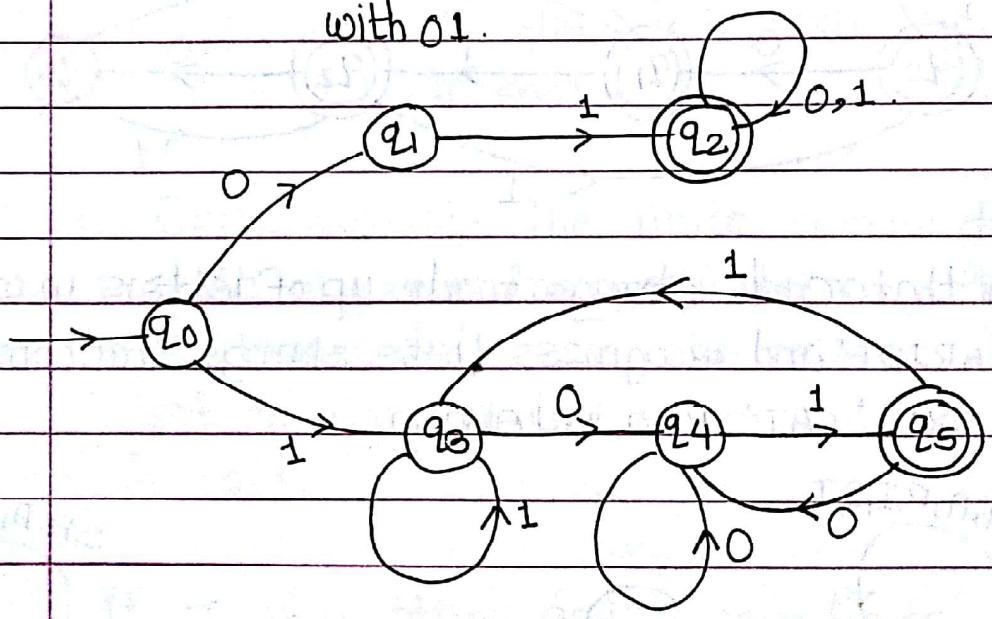
Page No

ex 09 $\Sigma = \{0,1\}$ such that each a in x is immediately preceded & immediately followed by "b"

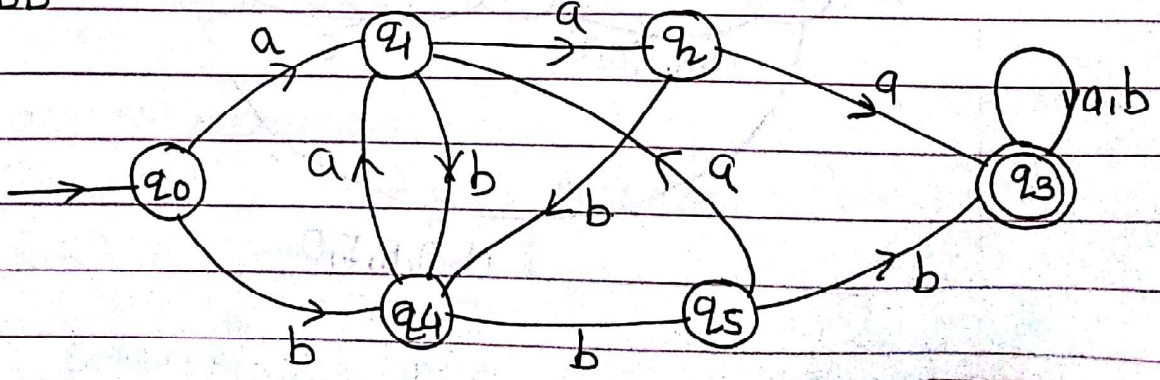


$b \in L$
$bb \in L$
$bbab \in L$
$bba \notin L$
⋮

ex 10 - $\Sigma\{0,1\}$ - such that strings either begin or end (or both) with 01.



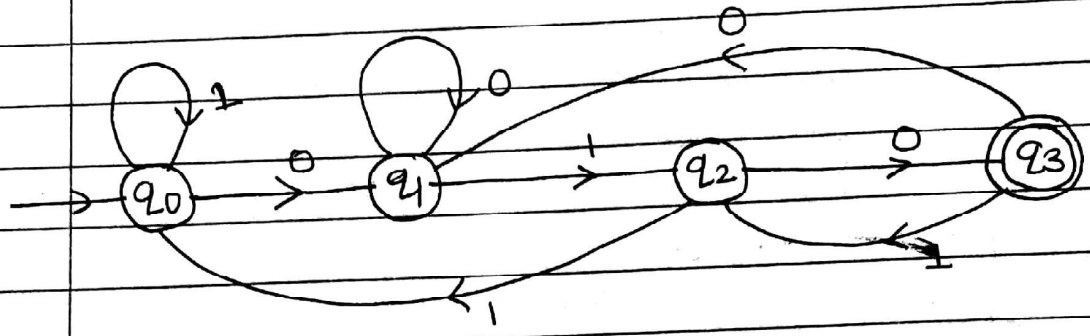
ex 11. sets of strings containing either the substring 'aaa' or 'bbb'



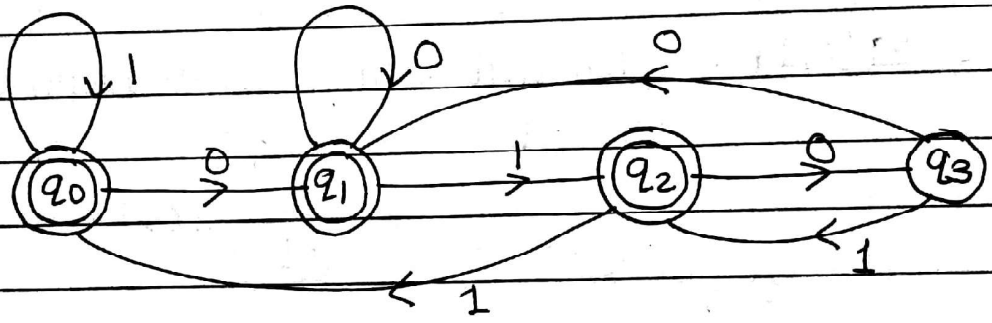
Page No

ex 12. $\Sigma = \{0,1\}$ not ending in 010

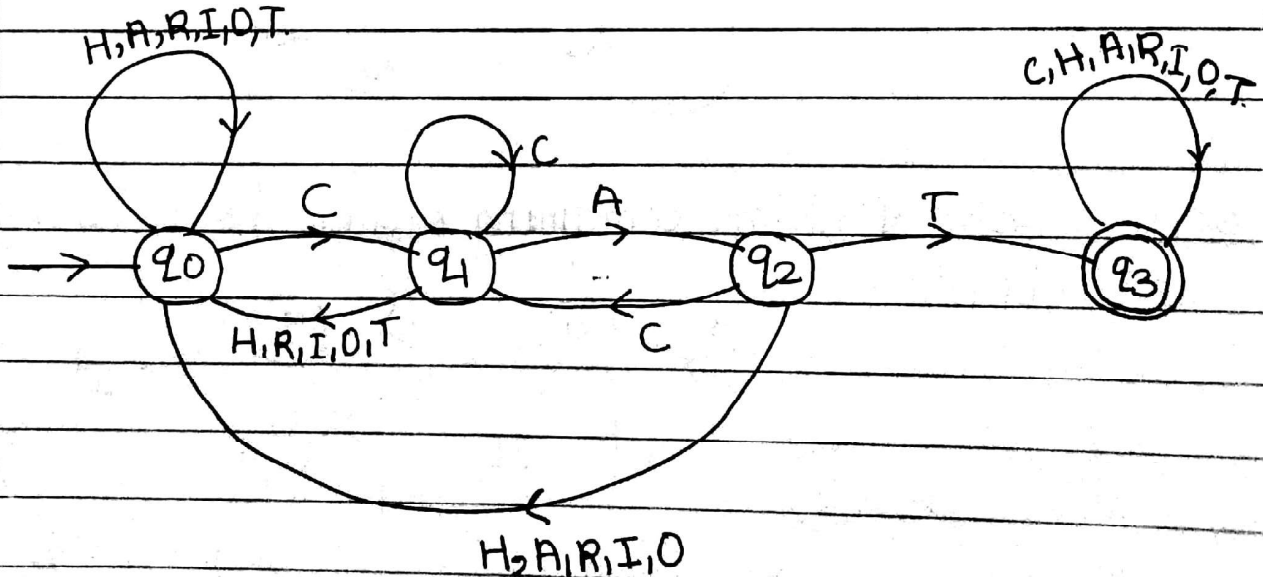
→ DFA M for accepting strings ending in 010.



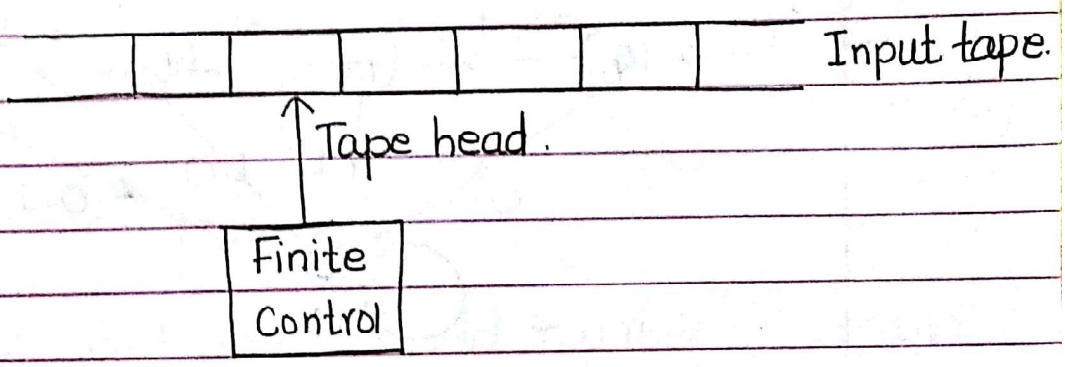
Complement of DFA M.



ex 13. DFA that reads strings made up of letters in word 'CHARIOT' and recognizes these strings that contain the word 'CAT' as a substring.



Finite Automata Model: The finite automata can be represented as:



The finite automata can be represented using

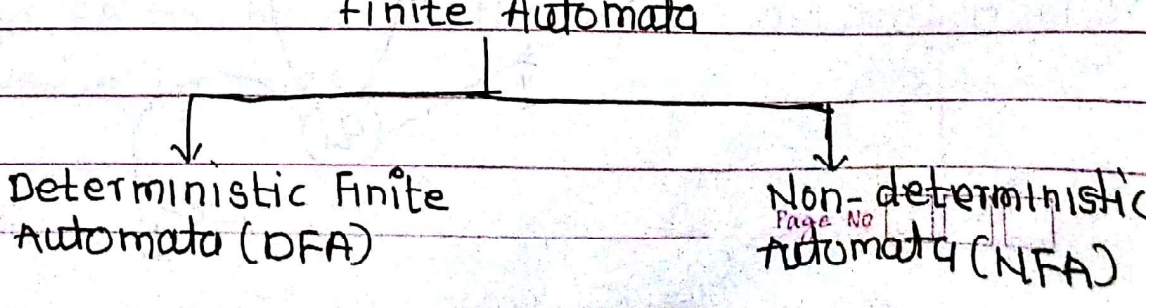
i) Input tape - It is a linear tape having some number of cells. Each input symbol is placed in each cell.

ii) Finite control - The Finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right and at a time only one input symbol is read.

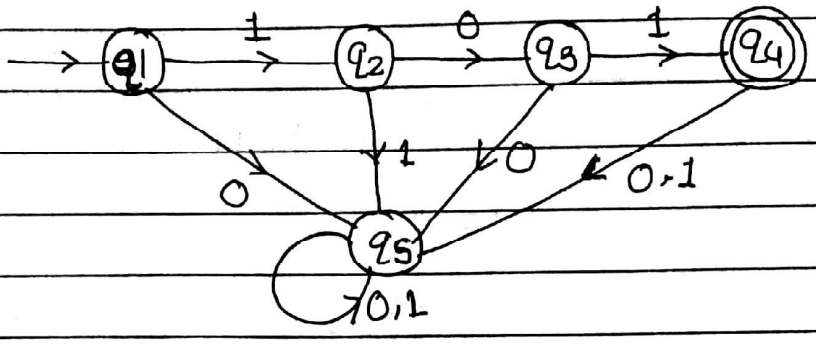
If x is a string and M is a finite automata, then x is accepted by the FA.

iff $\hat{\delta}(q_0, x) = q \in F$

Types of Automata



ex 14 Design a FA which accepts the only input 101 over the input set $\Sigma = \{0,1\}$

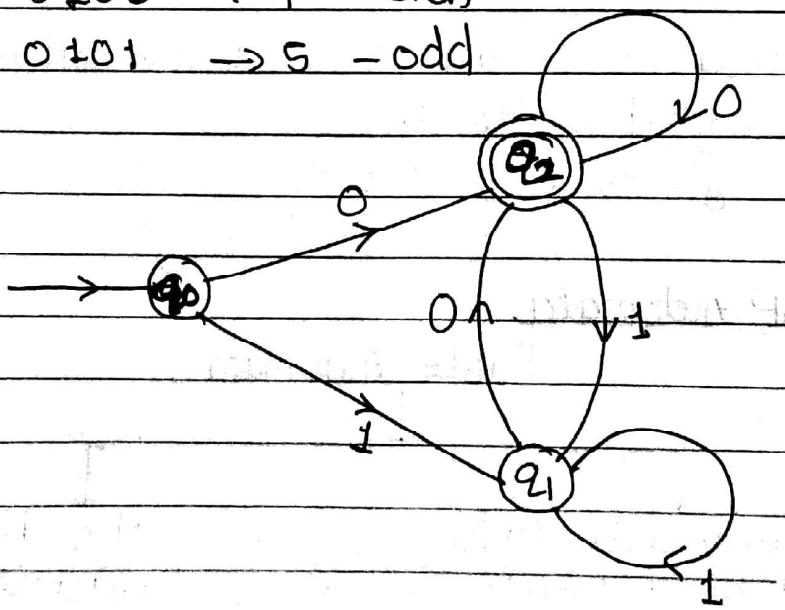


ex 15. Design a FA which checks whether the given binary number is even

→ The binary number is made up of 0's and 1's when any binary number ends with 0 it is alway even and when a binary number ends with 1 it is always odd.

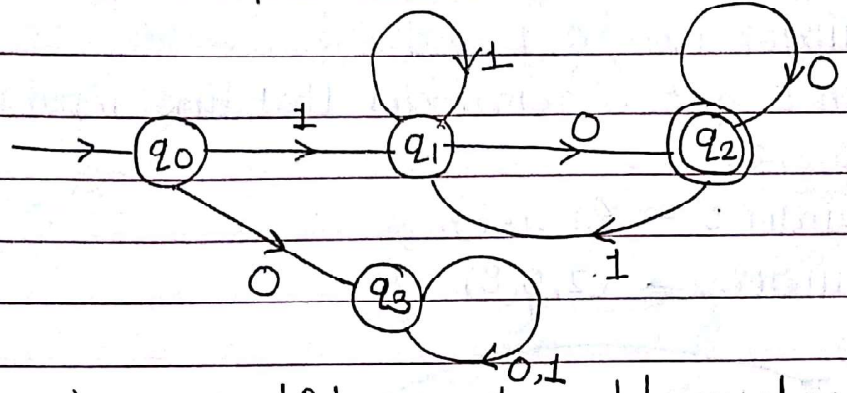
ex

0000 → 0 → even
0001 → 1 → odd
0010 → 2 – even
0011 → 3 – odd
0100 → 4 – even
0101 → 5 – odd

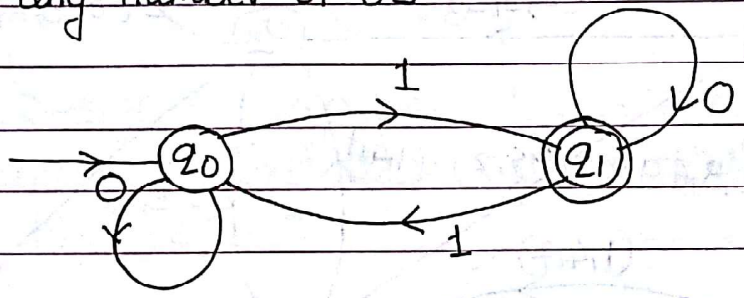


Page No

ex 16 Design FA which accepts only those strings which start with 1 and ends with 0.

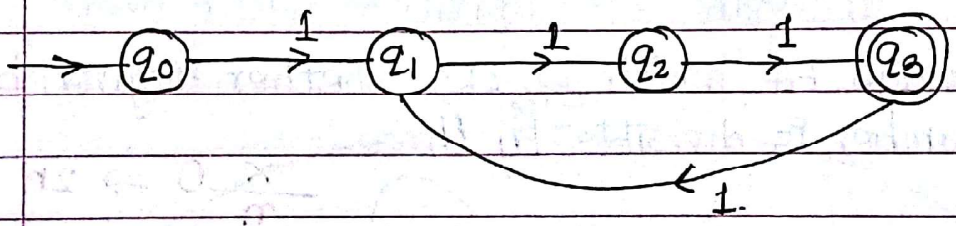


ex 17 Design FA which accepts odd number of 1's and any number of 0's



ex 18. Design FA which checks whether the given unary number is divisible by 3.

→ The unary number is made up of ones. The number 3 can be written in unary form as 111.



ex 19 Design FA to check whether given decimal number is divisible by three.

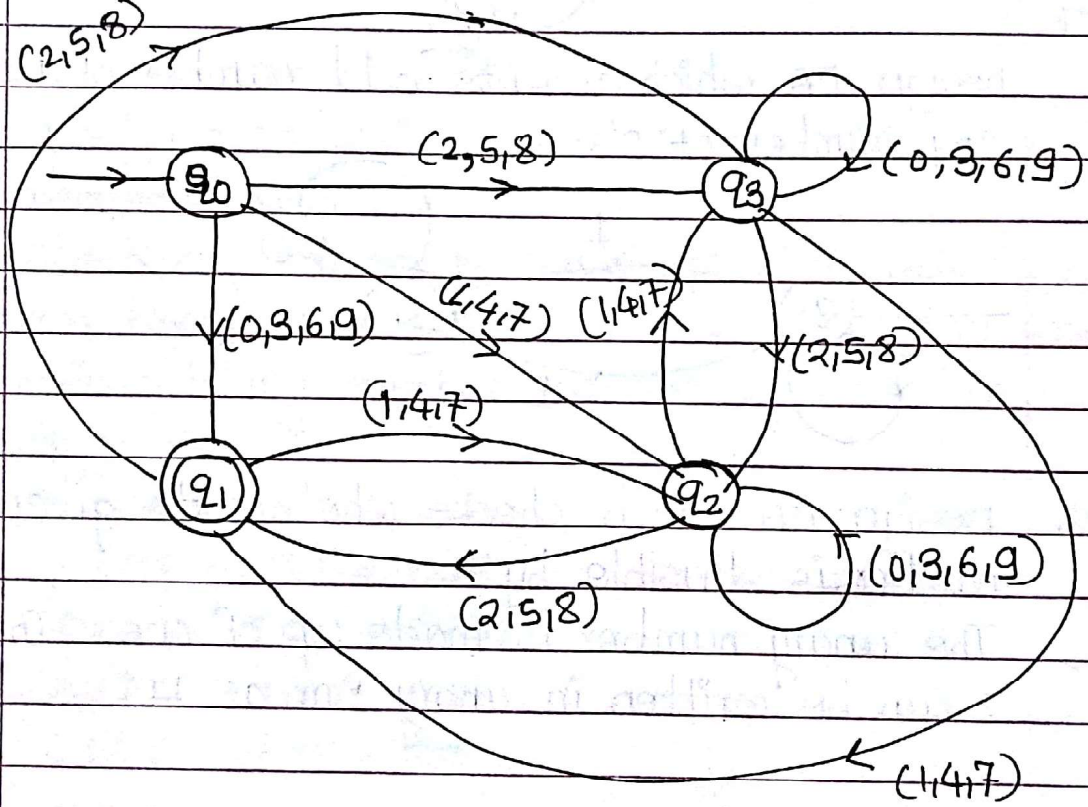
→ $\Sigma\{0,1,2,\ldots,9\}$

Number divisible by three has three chances of remainder i.e 0, 1 and 2.

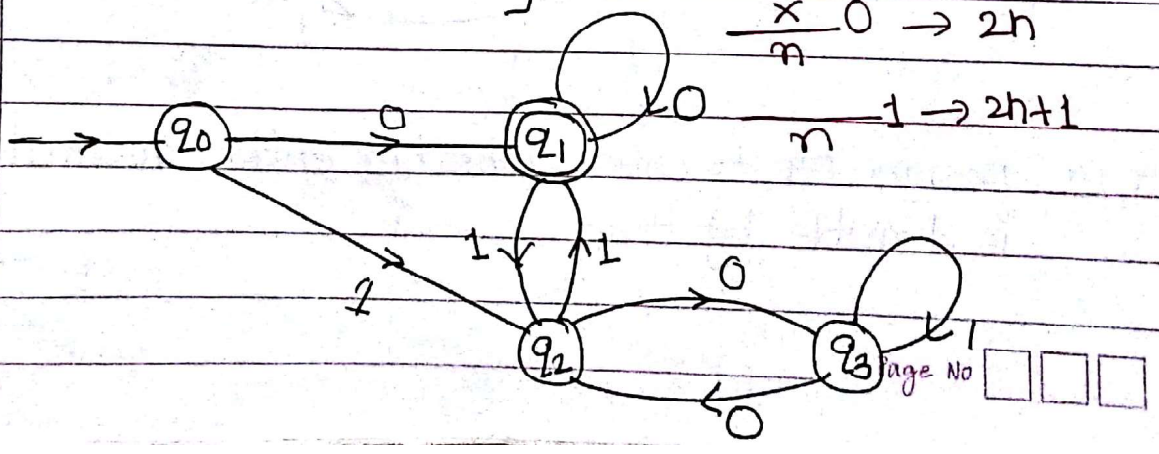
When I get 0 remainder that time input will be $\{0,3,6,9\}$

remainder 1 → $\{1,4,7\}$

remainder 2 → $(2,5,8)$

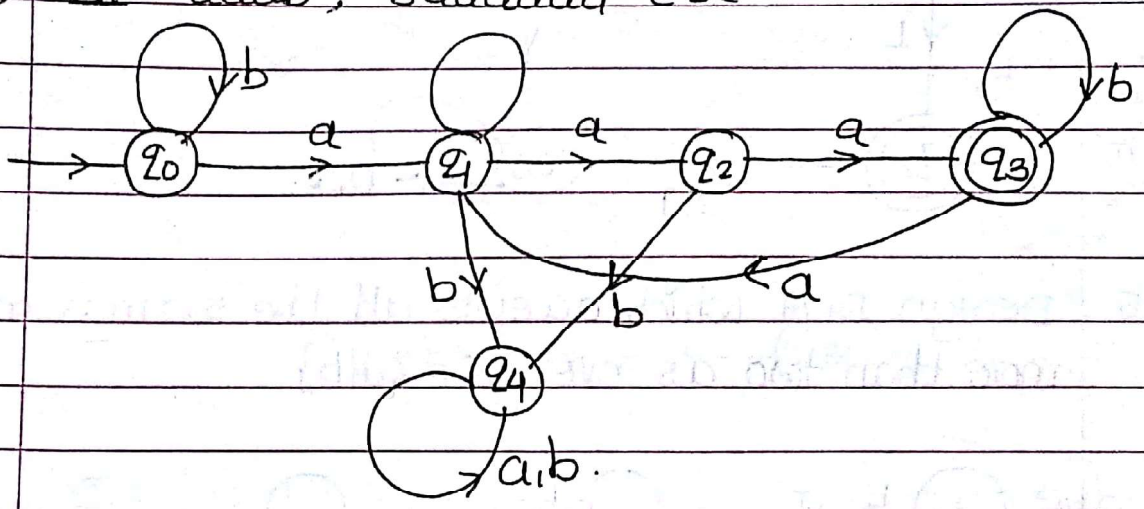


ex 20 Design FA which checks whether a given binary number is divisible by three.

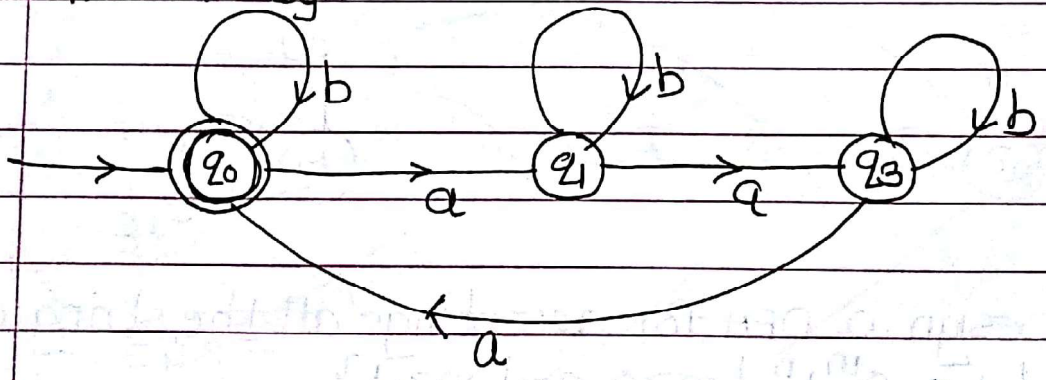


ex 21. Design FA to accept L, where L = {string in which a always appears tripplet} over the set {0,1}

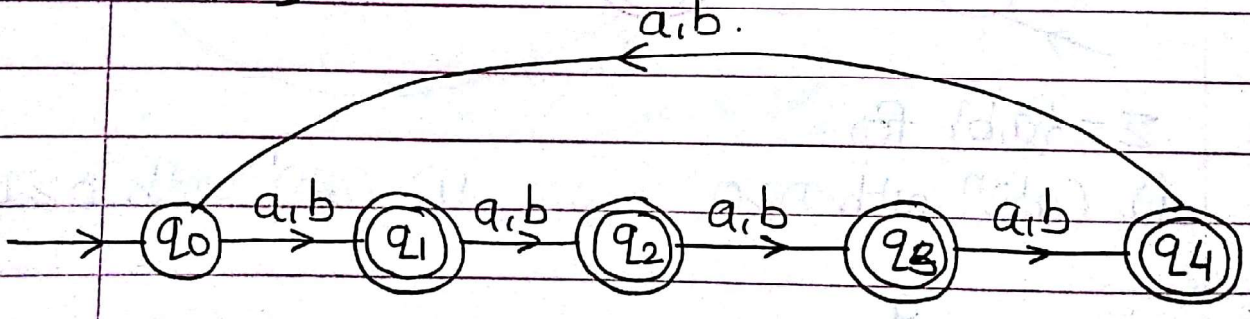
→ ex aaab, baaaaaq etc.



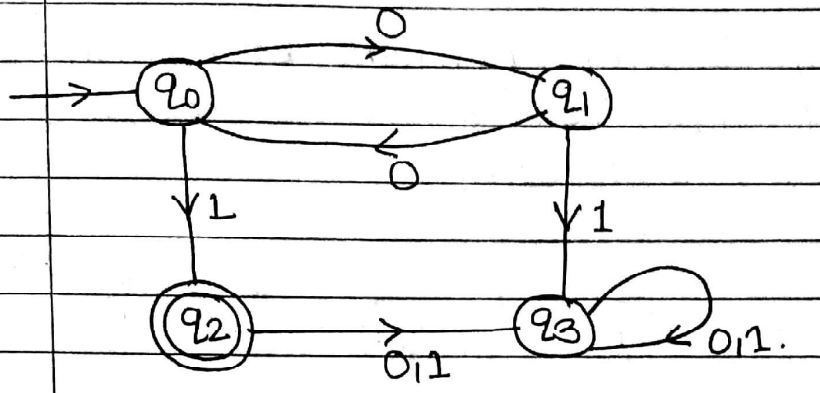
ex 22. Design FA to accept L where all the strings in L are such that total number of a's in them are divisible by 3.



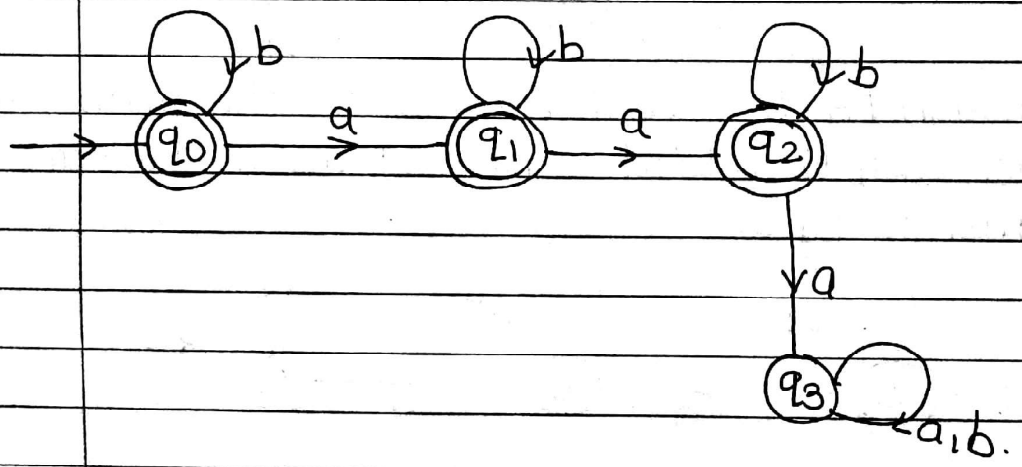
ex 23 Write a DFA to accept the language L = {L : |w| mod 5 ≠ 0}



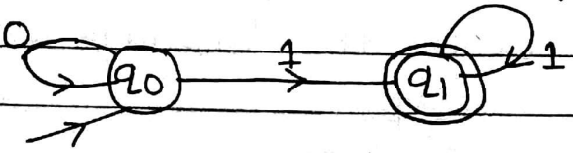
ex 24 - Design a DFA which accepts strings with even numb of 0's followed by single 1 over Σ = {0,1}



ex 25. Design DFA which accepts all the strings not having more than two a's over $\Sigma = \{a, b\}$.

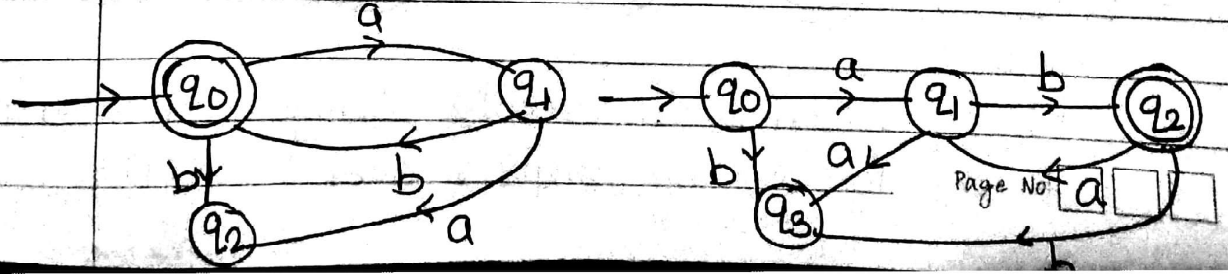


ex 26. Design a DFA for accepting all the string of $\{L = 0^m 1^n \mid m \geq 0 \text{ and } n \geq 1\}$

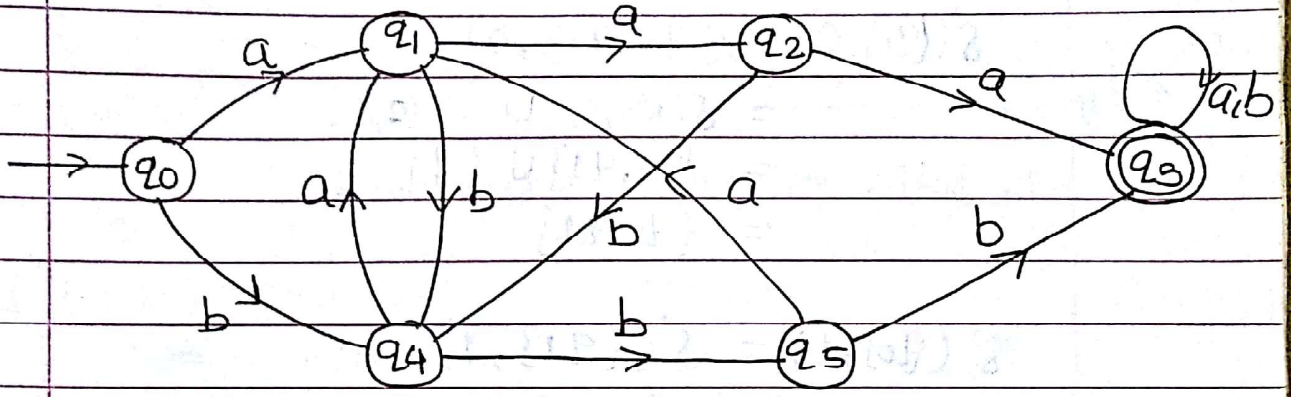


ex 27. $\Sigma = \{a, b\}$ for.

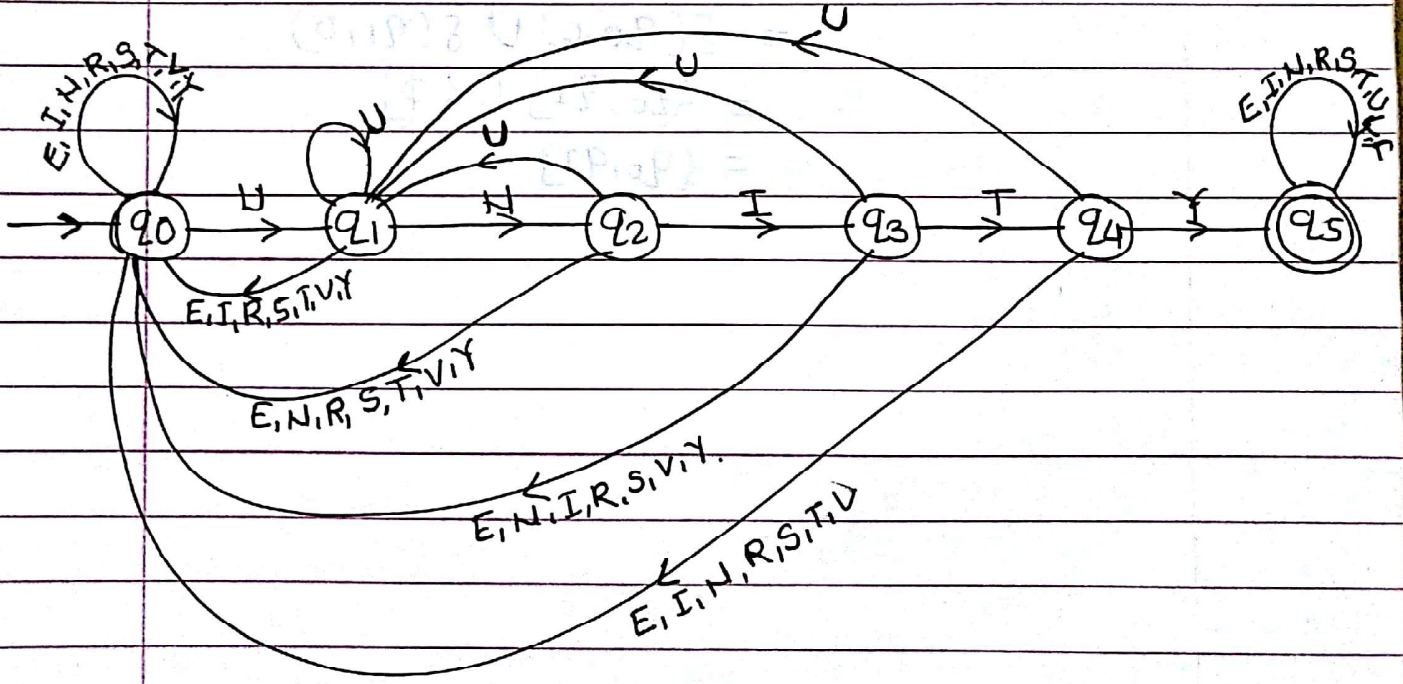
i) $(ab)^n$ with $n \geq 0$ ii) $(ab)^n$ with $n \geq 1$



ex28. construct a DFA for set of strings containing either the substring 'aaa' or 'bbb'

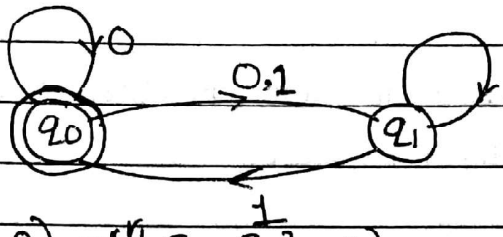


ex29. $\Sigma = \{U, N, I, V, E, R, S, I, T, Y\}$ x contain substring UNITY.



Page No

ex 3D.



convert NFA to equivalent DFA.

$\delta'([q_0], 0) = \delta(\{q_0, q_1\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}.$

$\delta'([q_0], 1) = \delta'(\{q_1\}, 1)$

$= \delta(q_1, 1)$

$= q_1.$

$\delta'[\{q_0, q_1\}, 0) = \delta'(\{q_0, q_1\}, 0)$

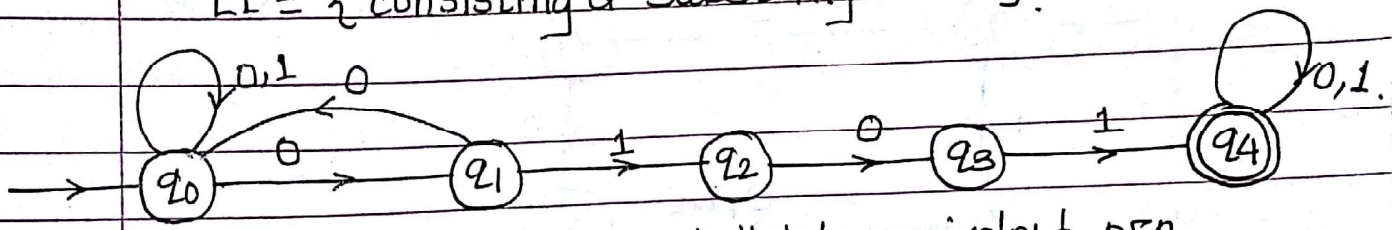
$= \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

ex 31. construct a NFA for the language.

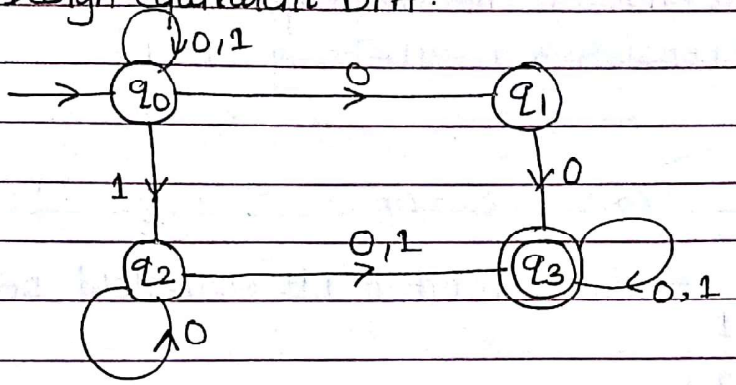
L1 = { consisting a substring 0101 }.



& also convert it into equivalent DFA.

	0	1
{q0}	{q0,q1}	{q0}
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1,q3}	{q0}
{q0,q1,q3}	{q0,q1}	{q0,q2,q4}
{q0,q2,q4}	{q0,q1,q3,q4}	{q0,q4}
{q0,q1,q3,q4}	{q0,q1,q4}	{q0,q2,q4}

ex 32. Design equivalent DFA.



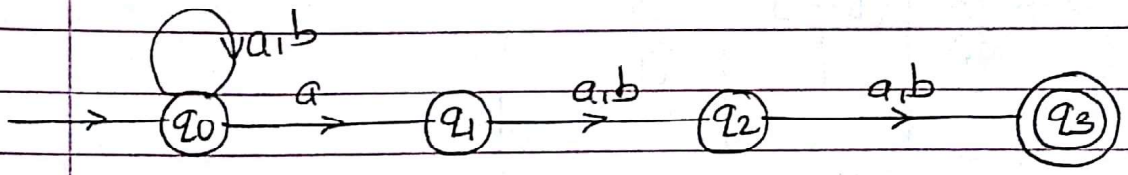
Convert NFA to equivalent DFA

1] Start with the initial state of NFA, Find transition of initial state on Σ.

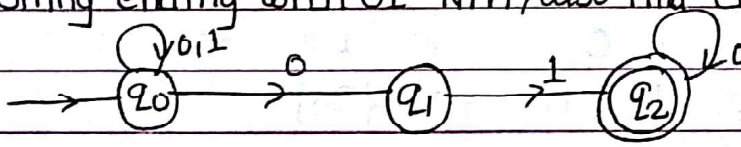
2] We got two new output state in step 1, if that state is not repeated, consider it as a state of DFA & continue to find transition over Σ.

3] so for all new state follow step 2.

ex 33. construct a NFA for a language L which accept all the strings in which the third symbol from right end is always a, over $\Sigma = \{a, b\}$. also find equivalent DFA.



ex 34. String ending with 01 NFA, also find equivalent DFA.



Page No

ex 35. convert given NFA to equivalent DFA.

Q	Q	1
→ q_0	$\{q_0, q_1\}$	$\{q_1\}$
(q_1)	ϕ	$\{q_0, q_1\}$

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_1\}$
{(q_1)}	ϕ	$\{q_0, q_1\}$
($\{q_0, q_1\}$)	$\{q_0, q_1\}$	$\{q_0, q_1\}$

Transition table for equivalent DFA.

ex 36.

	0	1	
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$	
q_1	$\{q_2\}$	$\{q_1\}$	to equivalent DFA
q_2	$\{q_3\}$	$\{q_3\}$	
(q_3)	ϕ	$\{q_2\}$	

	0	1
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
($\{q_1\}$)	$\{q_2\}$	$\{q_1\}$
$\{q_2\}$	$\{q_3\}$	$\{q_3\}$
$\{q_3\}$	ϕ	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$
($\{q_0, q_1, q_2\}$)	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_3\}$
($\{q_0, q_1, q_3\}$)	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$

classmate

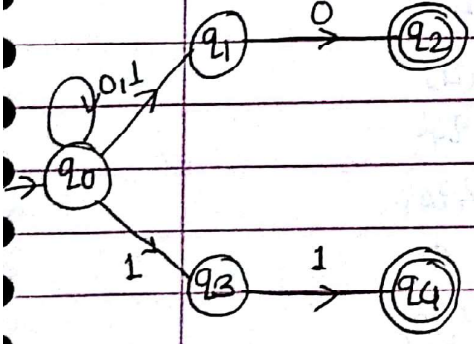
ex 37.

Q \ Σ	0	1
→P	{P,q}	P
q	r	r
r	s	—
(s)	s	s

→ P	{P,q}	P
q	{r}	r
r	{s}	—
(s)	{s}	s
~{P,q}	{P,q,r}	{P,r}
{P,q,r}	{P,q,r,s}	{P,r}
{P,r}	{P,q,s}	P
{P,q,r,s}	{P,q,r,s}	{P,r,s}
{P,q,s}	{P,q,r,s}	{P,r,s}
{P,r,s}	{P,q,s}	{P,s}
{P,s}	{P,q,s}	{P,s}

No need to consider because the state q,r,s has not been rechable from any state or Initial state

ex 38.

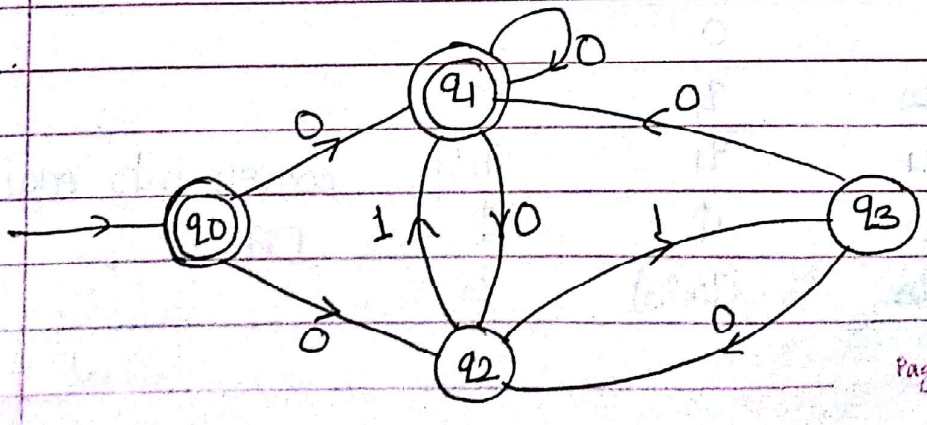


→ q0	{q0,q1}	{q0,q3}
q1	{q2}	ϕ
* q2	ϕ	ϕ
q3	ϕ	{q4}
* q4	ϕ	ϕ
✓ {q0,q1}	{q0,q1,q2}	{q0,q3}
✓ {q0,q3}	{q0,q1}	{q0,q3,q4}
✓* {q0,q1,q2}	{q0,q1,q2}	{q0,q3}
✓* {q0,q3,q4}	{q0,q4}	{q0,q3,q4}

No need to consider

ex 39.

inp



Page No

→	q_0	$\{q_1, q_2\}$	ϕ
*	q_1	$\{q_1, q_2\}$	ϕ
	q_2	ϕ	$\{q_1, q_3\}$
	q_3	$\{q_1, q_2\}$	ϕ
*	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_3\}$
*	$\{q_1, q_3\}$	$\{q_1, q_2\}$	ϕ

ex 40.

		0	1
→	q_0	$\{q_0, q_2\}$	$\{q_4\}$
	q_1	q_3	q_4
	q_2	ϕ	q_4
	q_3	q_3	ϕ
	q_4	q_3	q_3

	0	1
$\{q_0\}$	$\{q_0, q_2\}$	$\{q_4\}$
$\{q_4\}$	$\{q_3\}$	$\{q_4\}$
$\{q_2\}$	ϕ	$\{q_4\}$
$\{q_3\}$	q_3	ϕ
* $\{q_4\}$	$\{q_3\}$	$\{q_3\}$
$\{q_0, q_2\}$	$\{q_0, q_2\}$	$\{q_1, q_4\}$
* $\{q_1, q_4\}$	$\{q_3\}$	$\{q_4, q_3\}$
* $\{q_4, q_3\}$	$\{q_3\}$	$\{q_3\}$

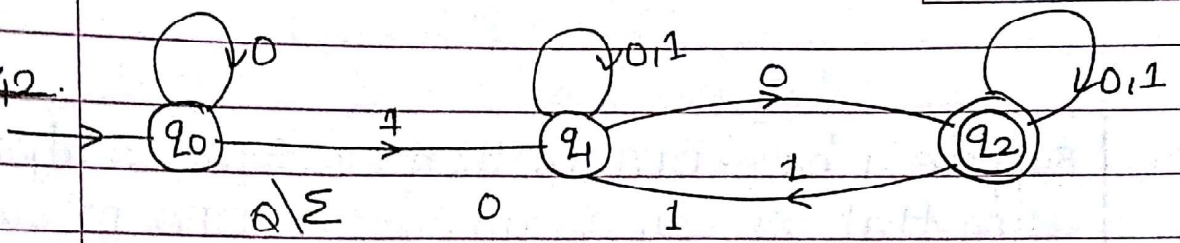
ex 41.

	0	1
q_0	q_1	q_2
q_1	q_1	q_1, q_3
q_2	ϕ	ϕ
q_3	$\{q_0, q_3\}$	q_3

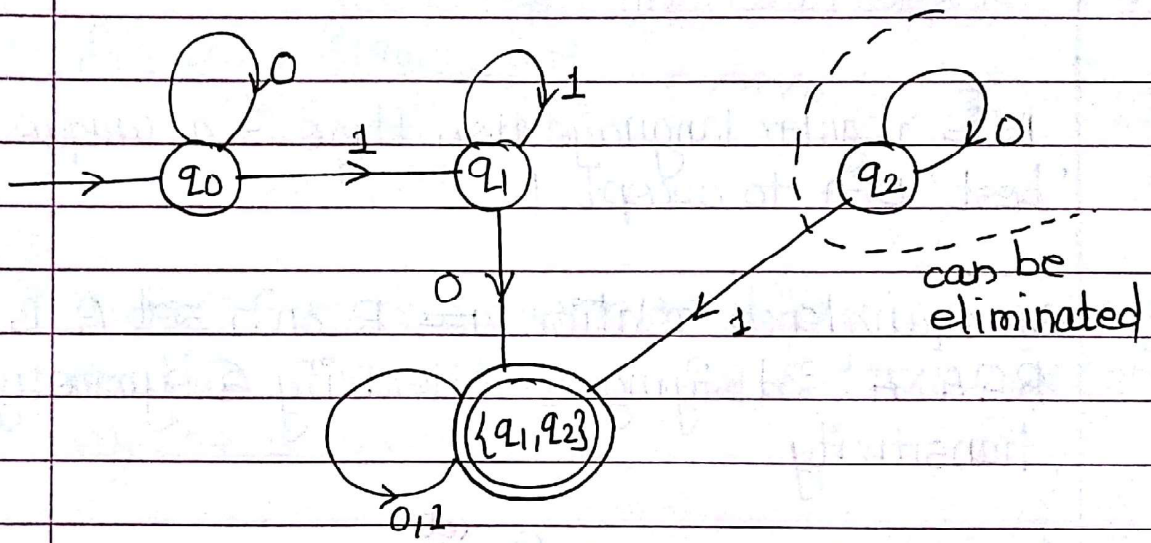
convert into equivalent DFA.

Page No

ex 42.



Q \ Σ	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_1\}$
$\{q_1\}$	$\{q_1, q_2\}$	$\{q_1\}$
$\{q_2\}$	$\{q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$



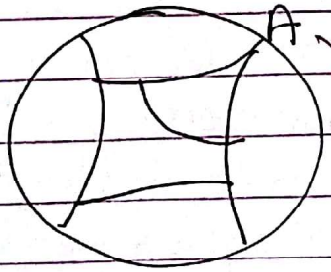
Myhill-Nerode Theorem → For any Regular language L, there is a unique 'best' DFA to accept L

suppose I have DFA M then we have an algorithm using that we can obtain another DFA M' such that these two DFA are equivalent ∴ L(M)=L(M')
For DFA M', there can be no other DFA that contain less no. of states and it accept the same language.

L is regular language then there is a unique 'best' DFA to accept L.

An equivalence relation ~~on~~ R on a set A is R⊆A×A satisfying ① reflexivity ② symmetry & transitivity.

An equivalence relation R on set A create a partition of the set A, and the partition comprises of the equivalence classes of R.



set of student in an Institution.

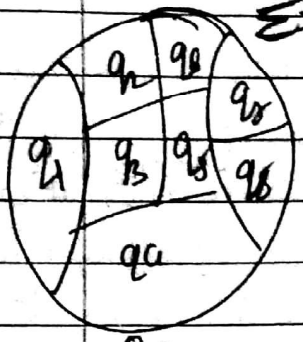
s1 R s2 :
if s1 and s2 both stay in the same hostel.

Def^n 1 :
suppose we have a DFA M with Σ
x R_M y if both x & y take the M from

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

its initial state to the same state

$\therefore \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$



def n2

An equivalence relation R on Σ^* is right-invariant if for all $x, y \in \Sigma^*$

$xRy \Rightarrow (\forall z)\ xzRyz$

R_M is right invariant, suppose $xR_My \Rightarrow$

$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) = p$

$\therefore \hat{\delta}(q_0, xz) = \hat{\delta}(q_0, \hat{\delta}(\hat{\delta}(q_0, x), z)) = \hat{\delta}(q_0, yz)$

$\therefore xz\, R_M\, yz$

def n3

Index of an equivalence relation:- an equivalence relation R is of finite index if the number of induced eq. classes is finite

R1 and R2 are eq. relation on A we say R1 is a refinement of R2 if $xR_1y \Rightarrow xR_2y$

Myhill - Nerode Theorem:-

The following three statements are equivalent: L is regular over Σ

① $L \subseteq \Sigma^*$ is accepted by a finite automaton

② L is the union of some of the equivalence classes of a right invariant eq. relation of finite index.

③ R_L is of finite index, where xR_Ly iff for all z either both xz and yz are in L or neither is in L.

statement A and B are equivalent if $A \Rightarrow B$ and $B \Rightarrow A$

proof → ① $\Rightarrow$ ②

since ①, Let DFA M accept L, $M=(Q,\Sigma,\delta,q_0,F)$

$x R y$:. iff $\hat{\delta}(q_0,x)=\hat{\delta}(q_0,y)$

Minimization of states of DFA

We have a DFA M, we would like to find out a DFA M', such that $L(M)=L(M')$ & M' has the minimum number of state. (Among all DFA that accept that lang)

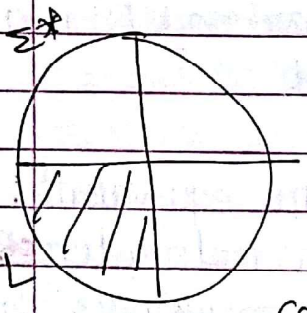
→ L is a language, & M is a DFA to accept it.

equivalence relation → $R_M \Rightarrow x R_M y$ iff both x & y take M from its initial state to the same state. R_M has so many eq. classes |Q| represent eq. classes in R_M & that are one for each state.

$$A_q = \{ x \in \Sigma^* \mid \hat{\delta}(q_0,x)=q \}$$

for the language L, a DFA M_1 has minimum no. of states iff $R_{M_1}=R_L$

:. for all $x,y \in \Sigma^*$, $x R_{M_1} y$ iff $x R_L y$



R_M for M to accept L has to be a refinement of R_L

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

Can we identify $q_1,q_2 \in Q$ s.t in R_M the eq. classes corresponding to q_1,q_2 are both subset of the same eq. class of R_L

(q_1,q_2) is distinguishable iff the eq. classes in R_M q_1 & q_2 fall in different eq. class of R_L

A_{q_1} is the eq. class correspond to q_1 in $R_M = \{x \in \Sigma^* \mid \hat{\delta}(q_0,x)=q_1\}$

A_{q_2} q_2 in $R_M = \{x \in \Sigma^* \mid \hat{\delta}(q_0,x)=q_2\}$

if there $M=(Q,\Sigma,\delta,q_0,F)$, $q_1,q_2 \in Q$ $\therefore (q_1,q_2)$ is distinguishable is a $z \in \Sigma^*$ s.t exactly one of $\delta(q_1,z)$ and $\delta(q_2,z)$ is in F

$\therefore x \in Aq_1$ & $y \in Aq_2$

$x \not R_L y$

suppose there is $z \in \Sigma^*$ s.t exactly one of $\therefore \delta(q_1,z)$ and $\delta(q_2,z)$ is in F. (The other is not in F)

suppose $x \in Aq_1$, $y \in Aq_2$, $xz \in L$, $yz \notin L$ $\Rightarrow x \not R_L y$

if one of q_1, q_2 is in F and other is not then (q_1,q_2) is distinguishable.

$q_1 \in F$ and $q_2 \notin F$.

outline of Algo to find all pairs of distinguishable state

output: set S of all pairs of distinguishable state

Initialization: $S = \{(p,q) \mid$ one of p,q in F & other not$\}$

Iteration :- consider ~~$r_1 \neq r_2$~~ $(r_1,r_2) \notin S$ pair of state

each

We will add (r_1,r_2) in S if for some $a \in \Sigma$, $(\delta(r_1,a), \delta(r_2,a)) \in S$

stop the process if in an iteration no new pair is added to S.

Time complexity - At most $\Theta(n^2)$ iteration.

In each iteration we do at most $\Theta(n^2)$ operations

The algorithm is polynomial time.

correctness -

suppose our algo is not correct then for some instance when the algo ends, there are pairs of distinguishable states which are not in S.

Let S' be the set of distinguishable pairs of states not in S.

Page No

* If one of (q_1, q_2) is in F and the other is not then (q_1, q_2) is distinguishable. (Here we are consider z as a empty string) i.e $q_1 \in F$ and $q_2 \notin F$

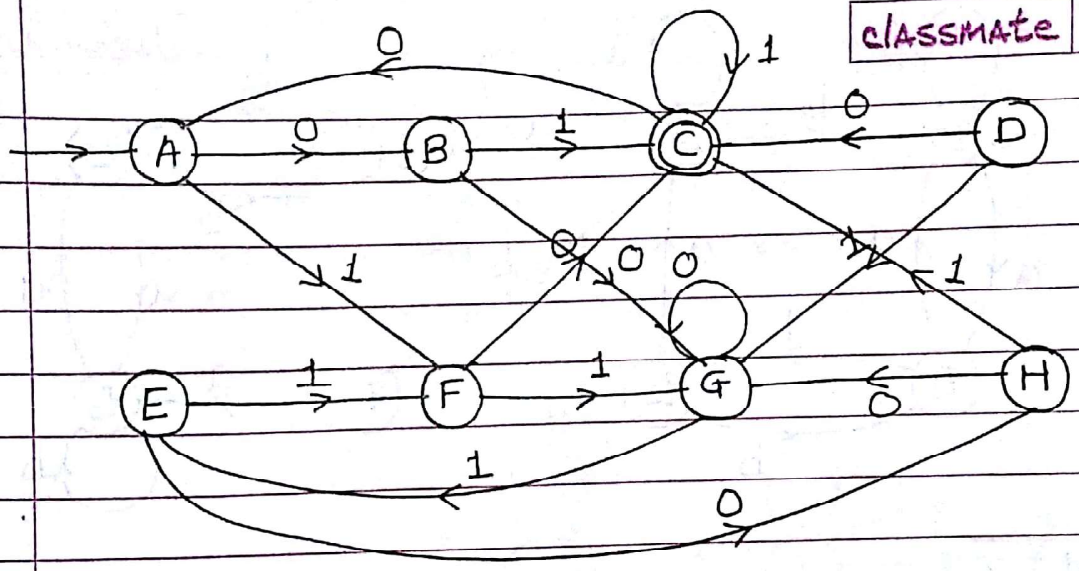
Distinguishable states : (q_1, q_2) is said to be distinguishable only when $(q_1, q_2) \in Q$ and if there is $x \in \Sigma^*$ such that exactly one of $\hat{\delta}(q_1, x)$ and $\hat{\delta}(q_2, x)$ is in final state.

Equivalent states : (q_1, q_2) is said to be equivalent iff $(q_1, q_2) \in Q$ and for all x. $(\hat{\delta}(q_1, x), \hat{\delta}(q_2, x))$ must represent the same state i.e either both state must be accepting states or non-accepting states.

Algorithm :-

step1- start.
step2- select $(p, q) \in Q$.
step3- categories (p, q) into either equivalent or distinguished pairs of states.
step4- select next all possible pairs of states & go to step②
step5- Merge equivalent states of pair & construct Q' for minimize DFA M'.
step6- construct state transition table.
step7- construct DFA M'
step8- stop.

ex.



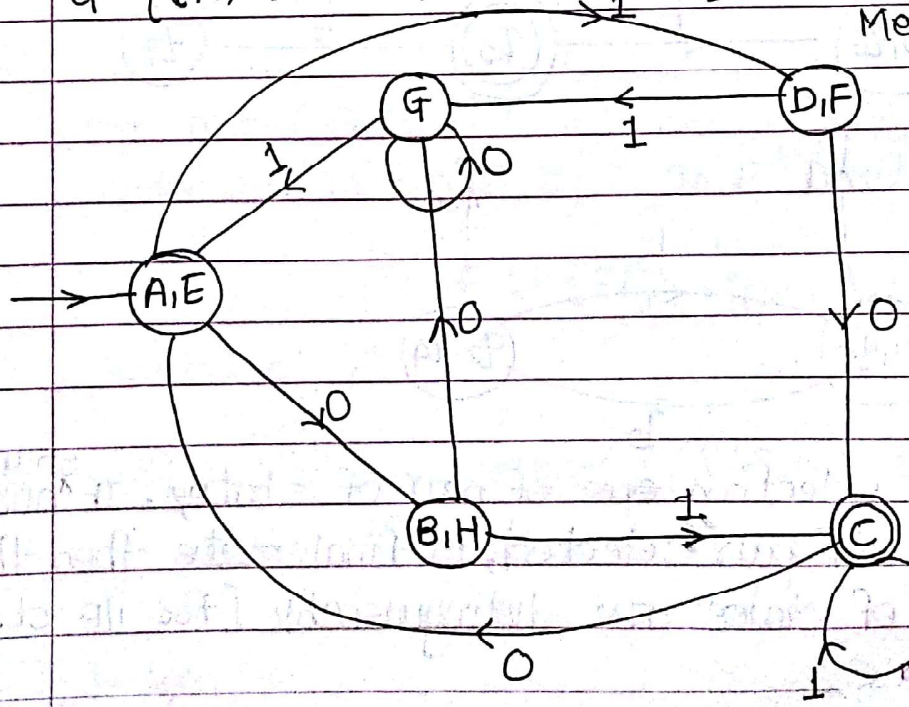
B	X							
C	X	X						
D	X	X	X					
E		X	X	X				
F	X	X	X		X			
G	X	X	X	X	X	X		
H	X		X	X	X	X	X	
	A	B	C	D	E	F	G	

equivalent states (A, E)

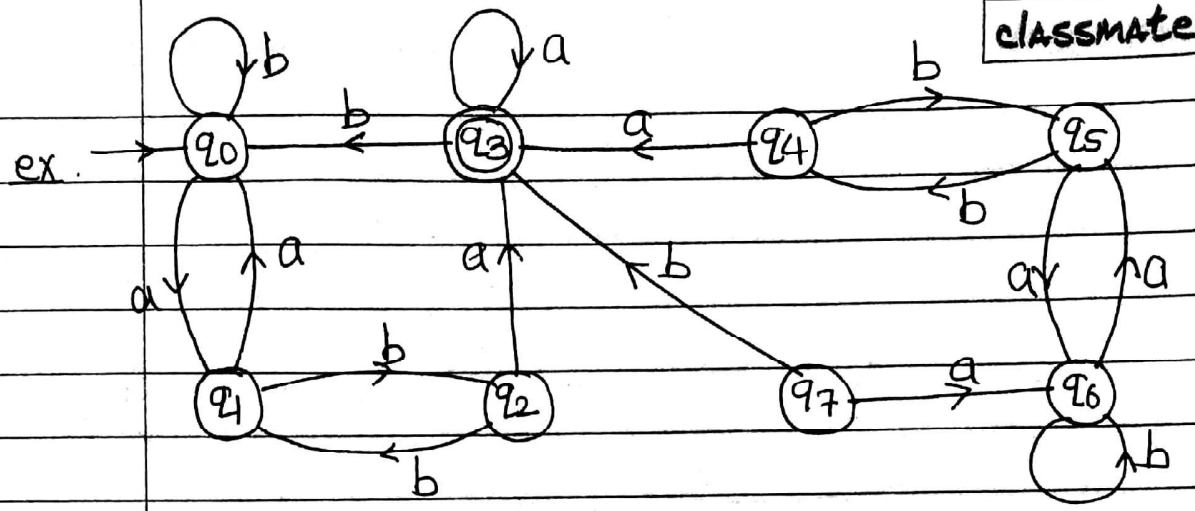
X - represent distinguishable states

$Q = \{(A,E), (B,H), (D,F), C, G\}$ → equivalent states

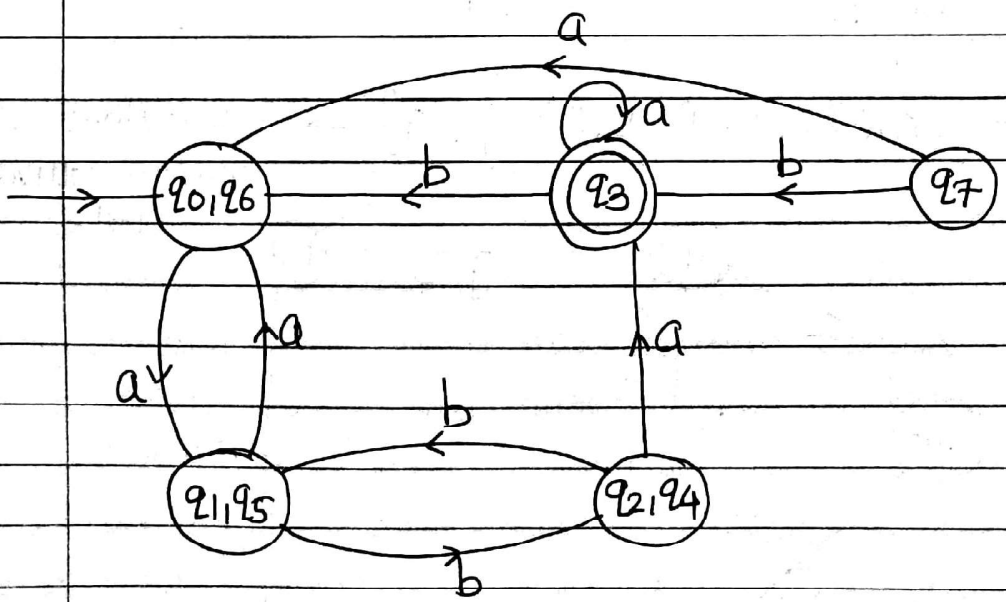
Merge equivant state



Page No



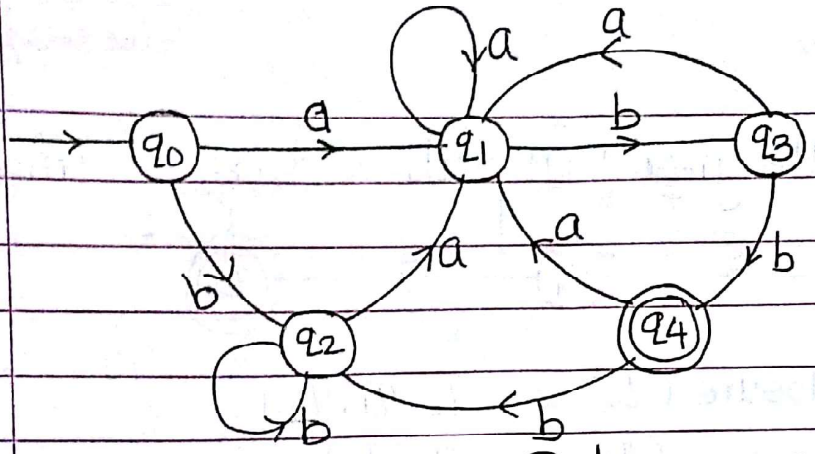
q_1	X						
q_2	X	X					
q_3	X	X	X				
q_4	X	X	·	X			
q_5	X	·	X	X	X		
q_6	·	X	X	X	X	X	
q_7	X	X	X	X	X	X	X
	q_0	q_1	q_2	q_3	q_4	q_5	q_6



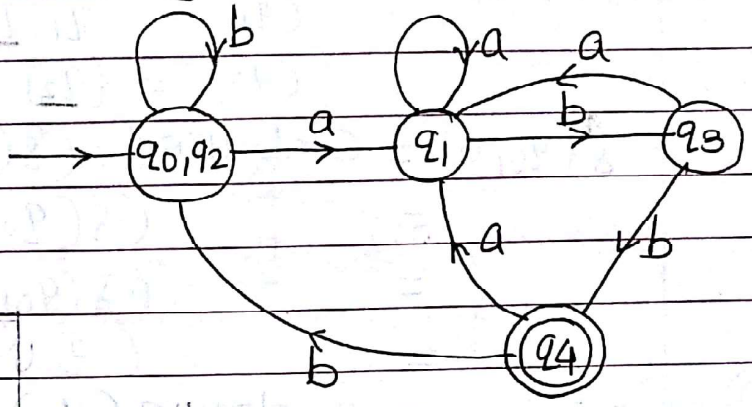
IMP :- After selecting ~~one of~~ pair of states, if only one state of pair selected, is final state then that pair of states are distinguishable [for i/p string $x = \epsilon$]

Page NO

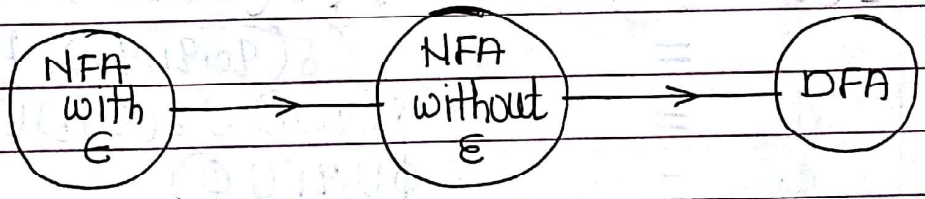
ex



q_1	X			
q_2		X		
q_3	X	X	X	
q_4	X	X	X	X
	q_0	q_1	q_2	q_3



Eliminating ϵ- Transitions :-



Conversion from NFA with ε to NFA without ε

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε transition

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

$F' = F \cup \{q_i\}$ if ε- closure of q_i conta

states of F.

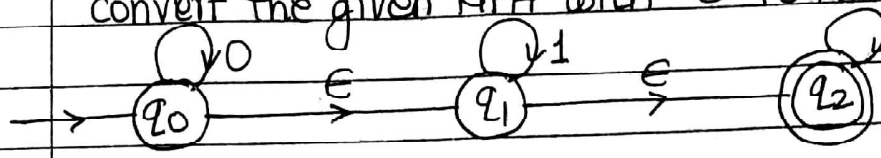
$\delta'(q,a) = \epsilon\text{- closure}(\delta(\hat{\delta}(q,\epsilon), a))$

where $\hat{\delta}(q,\epsilon) = \epsilon\text{- closure}(q)$.

Page No

Technical 2-36

ex- convert the given NFA with ϵ to NFA without ϵ.



ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

$(q_1) = \{q_1, q_2\}$

$(q_2) = \{q_2\}$

$\delta(q_0, 0) = \epsilon$-closure $(\delta(\delta(q_0, \epsilon), 0))$

$= (\delta(q_0, q_1, q_2), 0)$

$= (\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= (q_0 \cup \phi \cup \phi)$

$= \epsilon$-closure (q_0)

$\delta'(q_0, 0) = \{q_0, q_1, q_2\}$.

$\delta'(q_0, 1) = \epsilon$-closure $(\delta(\delta(q_0, \epsilon), 1))$

$= (\delta(q_0, q_1, q_2), 1)$

$= (\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$

$= (\phi \cup q_1 \cup \phi)$

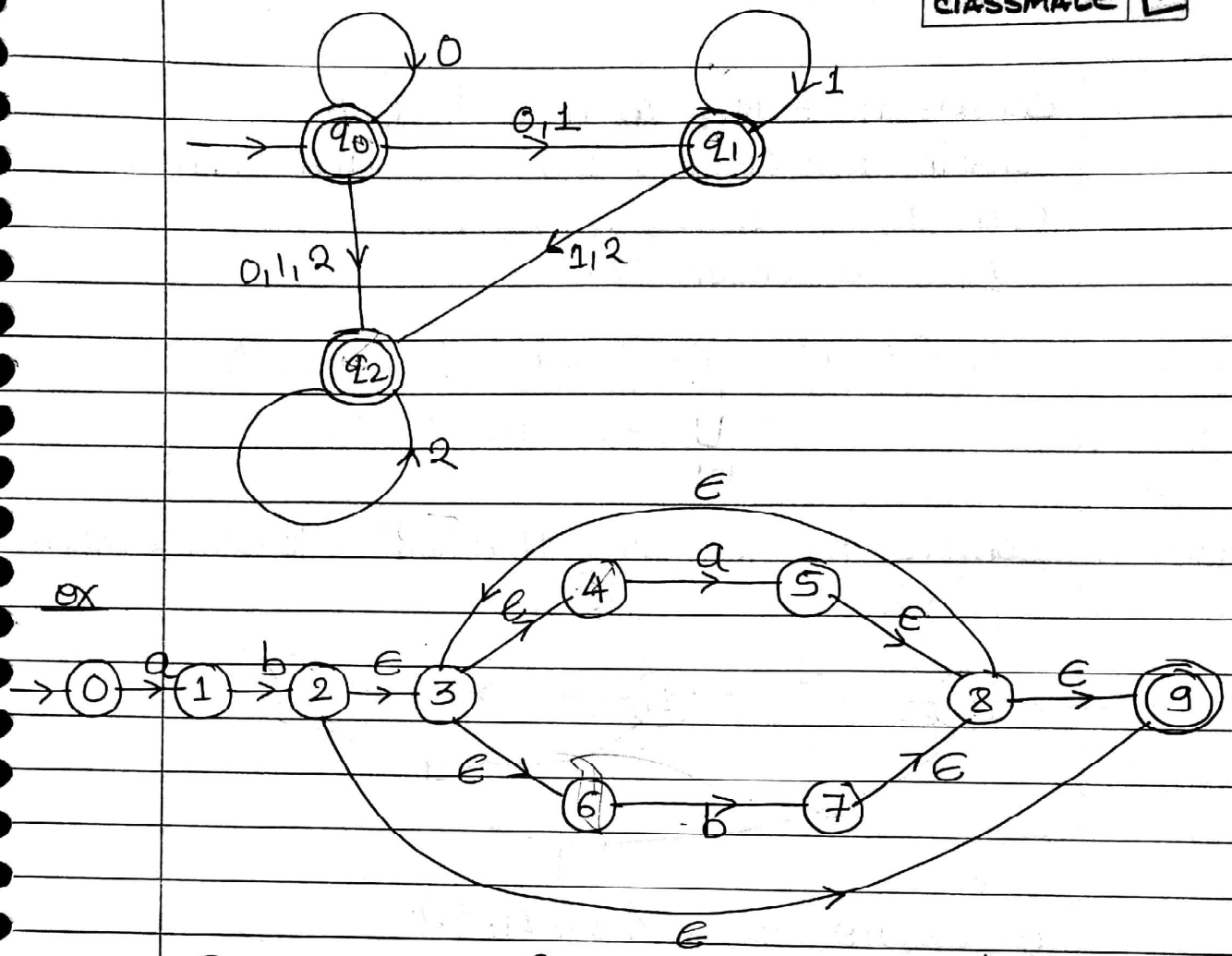
$= \epsilon$-closure (q_1)

$\delta'(q_0, 1) = \{q_1, q_2\}$

$\delta'(q_1, 0) = \phi$	$\delta'(q_0, 2) = \{q_2\}$
$\delta'(q_1, 1) = \{q_1, q_2\}$	$\delta'(q_1, 2) = \{q_2\}$
$\delta'(q_2, 0) = \phi$	$\delta'(q_2, 2) = \{q_2\}$
$\delta'(q_2, 1) = \phi$	

Q / Σ	0	1	2
→ q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

Page No



Q	a	b
{0}	1	ϕ
{1}	ϕ	2,3,4,6,9
{2}	3,4,5,6,8,9	8,4,6,7,8,9
{3}	3,4,5,6,8,9	3,4,6,7,8,9
{4}	8,4,5,6,8,9	ϕ
{5}	3,4,5,6,8,9	3,4,6,7,8,9
{6}	ϕ	3,4,6,7,8,9
{7}	8,4,5,6,8,9	3,4,6,7,8,9
{8}	3,4,5,6,8,9	3,4,6,7,8,9.
{9}	ϕ	ϕ.

Page No

Conversion of NFA with ε to DFA

consider $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA with ε

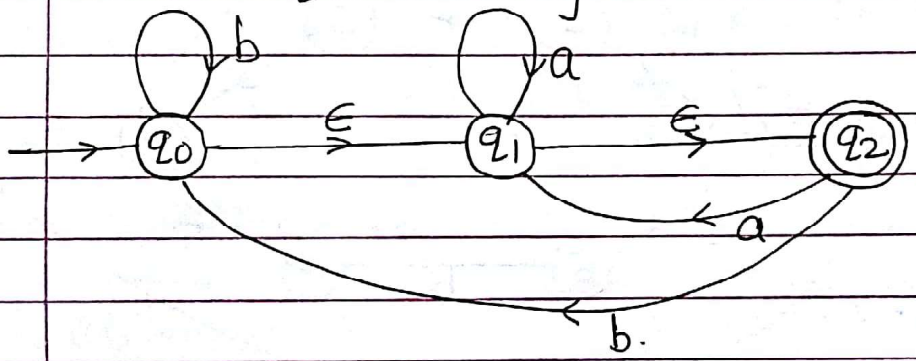
DFA $M_D = (Q_D, \Sigma_D, \delta_D, q_{0D}, F_D)$

q_{0D} = ε- closure (q_0)

$$\delta_D(q_0, a) = \epsilon\text{-closure}(\delta(P_1, a) \cup \delta(P_2, a) \cup \ldots \delta(P_n, a))$$
$$= \bigcup_{i=1}^{n} \epsilon\text{-closure}(P_i, a).$$

Technical 2-76

ex. convert the following NFA with ε to equivalent DFA



ε- closure $(q_0) = \{q_0, q_1, q_2\}$ — A

$(q_1) = \{q_1, q_2\}$

$(q_2) = \{q_2\}$

DFA start with ε- closure of q_0

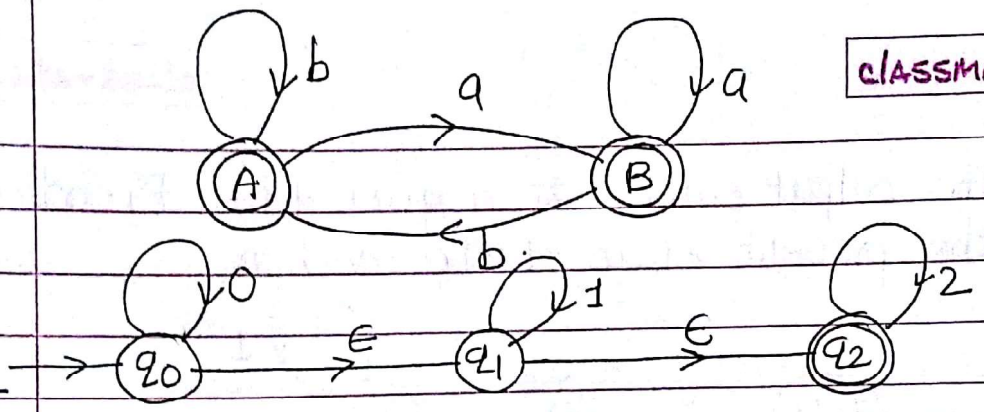
$$\delta'(\underset{A}{q_0}, a) = \epsilon\text{-closure}(\delta(q_0, q_1, q_2), a)$$
$$= (\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$$
$$= (\phi \cup q_1 \cup q_1)$$
$$= \epsilon\text{-closure}(q_1)$$
$$= \{q_1, q_2\} \text{ — new state — B.}$$

$$\delta'(\underset{A}{q_0}, b) = \{q_0, q_1, q_2\}$$

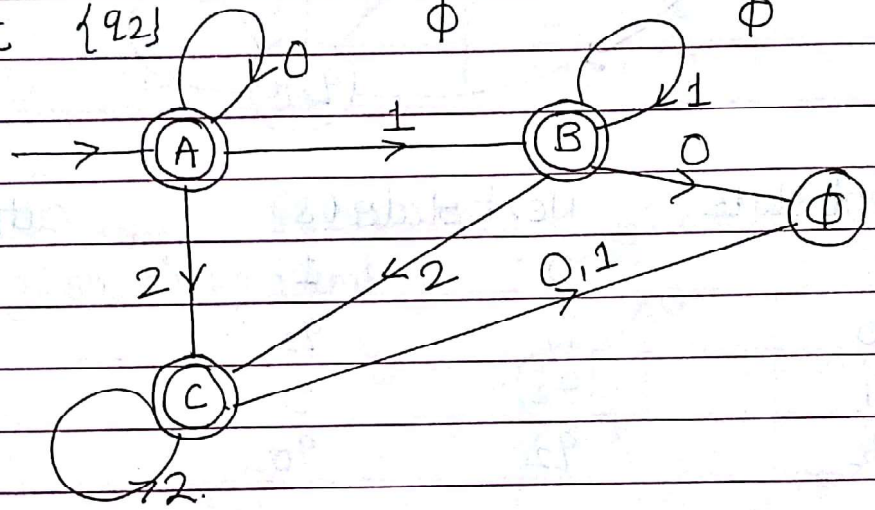
$$\delta'(\{q_1, q_2\}, a) = \{q_1, q_2\}.$$

$$\delta'(\{q_1, q_2\}, b) = \{q_0, q_1, q_2\}$$

Page No



		0	1	2
A	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
B	$\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
C	$\{q_2\}$	ϕ	ϕ	$\{q_2\}$.



FSM with outputs:-

1] Moore Machine :- M is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ Where

Q - is a finite set of states.

Σ - is a finite set of input symbols.

Δ - is an output alphabet.

δ - is an transition function such that

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

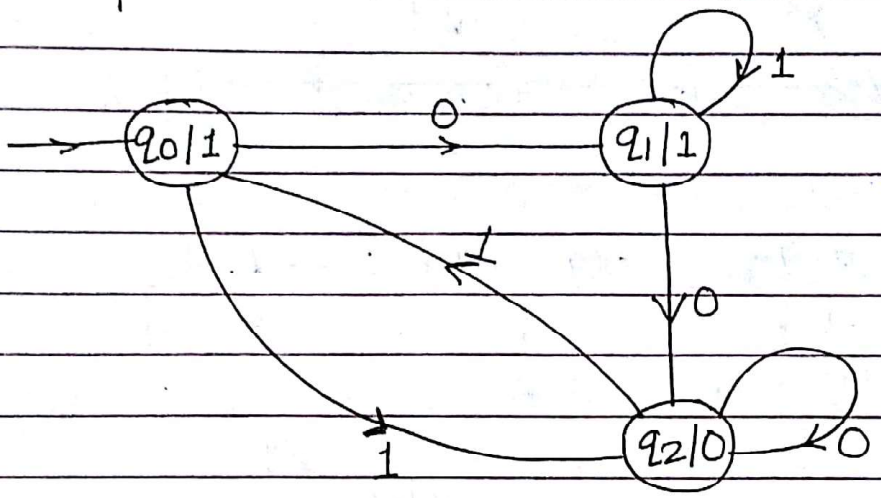
λ - is a output function

$$\lambda : Q \longrightarrow \Delta$$

q_0 - Initial state of Machine.

The output symbol at a given time depends only on the present state of the machine.

ex.



current state	Next state (δ)		output (λ)
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

2] Mealy Machine:

Mealy machine is a machine in which output symbol depends upon the present input symbol and present state of the machine.

Mealy machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

Q - is finite set of states.

Σ - is finite set of input symbols.

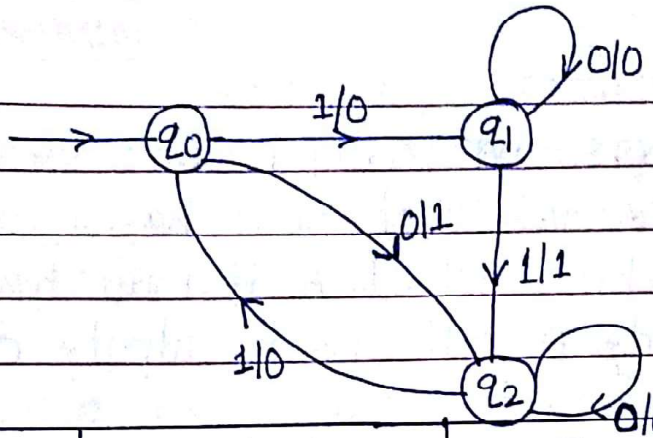
Δ - is an output alphabet.

δ - is state transition function such that $\delta : Q \times \Sigma \rightarrow Q$

λ - is machine function such that $\lambda : Q \times \Sigma \rightarrow \Delta$

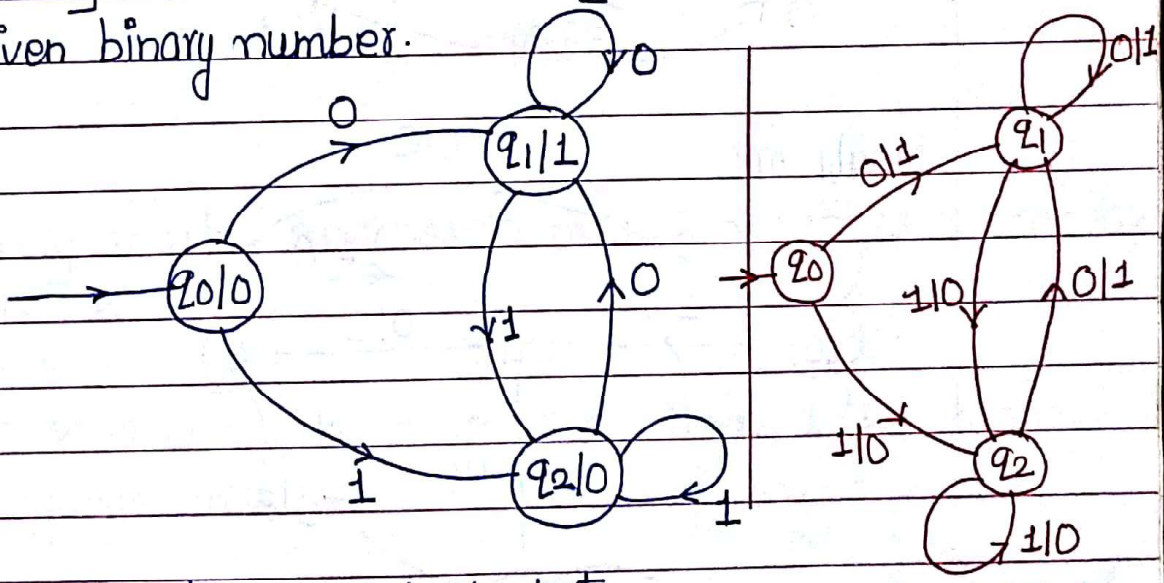
q_0 - is a initial state of m/c.

Page No



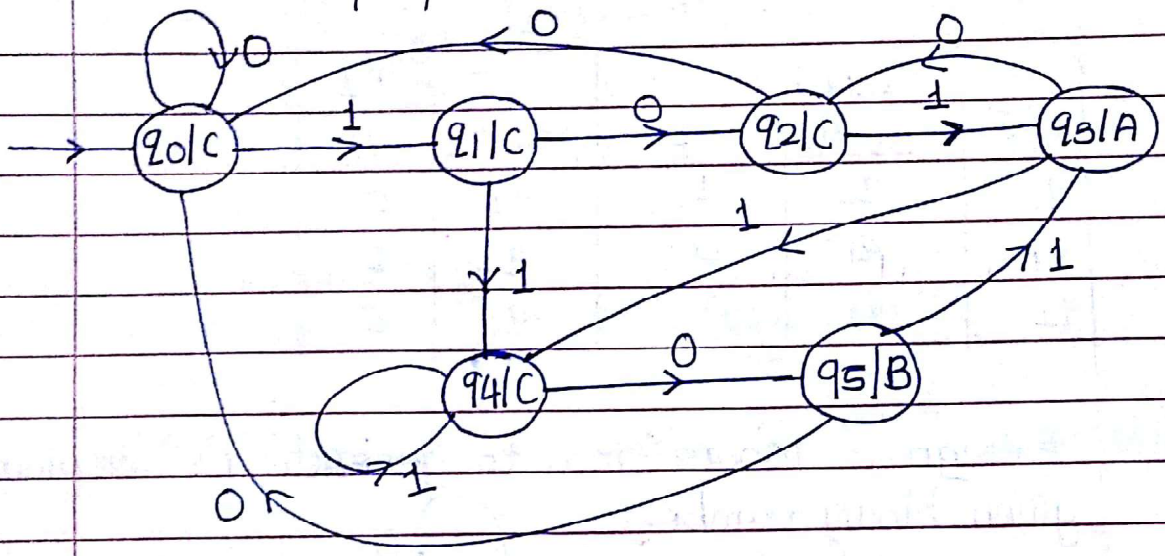
	Input 0		Input 1	
Q./Σ	state	o/p	state	o/p.
q_0	q_2	1	q_1	0
q_1	q_1	0	q_2	1
q_2	q_2	0	q_0	0

ex Design a Moore m/c to generate 1's complement of given binary number.



current state	Next state		output (a)
	0	1	
q_0	q_1	q_2	0
q_1	q_1	q_2	0
q_2	q_1	q_2	0

ex. Design a Moore and Mealy machine for a binary input sequence such that if it has a substring 101 the machine output A. if input has substring 110 it outputs B otherwise it outputs C.



Mealy m|c