UNIT-I

⇒ **Alphabets:-** An alphabet is a finite, non empty set of symbols. The symbols are called the letters of the alphabet.

Example 1: $S = \{0, 1\}$, the binary alphabet.

Example 2: $S = \{a, b, c, ----, z\}$, the set of all lower case letters.

Example 3: $S = \{0, 1, a, b, c\}$

Power of Alphabet:- Power of alphabet is the set of all strings of ~~lat~~ length K over alphabet S. we use the notation S^K.

Example:- let $S = \{0, 1\}$

$S^1 = \{0, 1\}$, set of all strings of length 1.

$S^2 = \{00, 01, 10, 11\}$, set of all strings of length 2.

$S^3 = \{000, 001, 011, 100, 101, 110, 111\}$, set of all strings of length 3.

Note: $S^0 = \{\epsilon\}$, regardless of what alphabet S is, ϵ is the string having length zero.

⇒ **Strings:-** A string over an alphabet S is a finite sequence of symbols from alphabet S.

Example 1:- Binary alphabet $(S) = \{0, 1\}$

strings are 0111100, 111, 000, ----, etc.

Example 2:- $S = \{a, b, c, ----, z\}$

aa, bb, abacb, acxxy, pqs, mnop, are strings of this alphabet.

length of a string:- The length of the string w is the number of symbols composing the string. It is denoted by $|w|$.

Example 1: $w = aabb$, then $|w| = 4$

Example 2: $w = 010110$, then $|w| = 6$

Empty String:- The empty string is the string with zero symbols and zero length. It is denoted by 'ϵ'.

$|\epsilon| = 0$

Note 1:- The set of all strings over an alphabet Σ is conventionally denoted by Σ^*.

Example :- $\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, ----\}$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots$

Note 2:- The set of all non-empty strings over an alphabet Σ is denoted by Σ^+.

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots$

$\Sigma^+ = \Sigma^* - \{\epsilon\}$

Note 3: $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Prefix and suffix of a string:

A Prefix of a string is any number of leading symbols of that string.

A suffix is any number of trailing symbols.

Example 1: string abc

Prefix : ϵ, a, ab, abc

Suffix : ϵ, c, bc, abc

operation on strings:

concatenation :- concatenation of two strings is writing the first followed by the second string with no intervening space.

Example 1:- ~~text~~ let x, y be two strings over Σ^* concatenation of x, y results new string $z = xy$.

Example 2 :- let $x = ab$, $y = pq$ find xy, yx

$xy = abpq$, $yx = pqab$

Note : The empty string ϵ is the identity for the concatenation operator. That is,

$\epsilon w = w\epsilon = w$, for each string w.

⇒ Languages :- A language L over the alphabet Σ is a subset of Σ^*. That is, a language is a set of strings over the given alphabet.

Note 1: $L \subseteq \Sigma^*$, L is language over Σ.

Note 2: A language can be empty $L = \phi$.

Note 3: Empty language is not equal to $L = \{\epsilon\}$

Example 1: let $\Sigma = \{a, b\}$ then $\Sigma^* = \{\epsilon, a, b, aa, \ldots\}$

The set $L = \{a^n b^n ; n \geq 0\}$ is also a language on Σ.

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

Operations on languages :-

1. Concatenation :-

The concatenation AB of languages A and B is defined by $AB = \{uv \mid u \in A, v \in B\}$

Ex 1: let $\Sigma = \{0, 1\}$, A, B are languages over S

$A = \{00, 11\}$, $B = \{0, 1\}$

$AB = \{000, 001, 110, 111\}$

$BA = \{000, 011, 100, 111\}$

Example 2: let $\Sigma = \{a, b\}$

$A = \{ab, aa\}$, $B = \{bb, ba\}$

$AB = \{abbb, abba, aabb, aaba\}$

$BA = \{bbab, bbaa, baab, baaa\}$

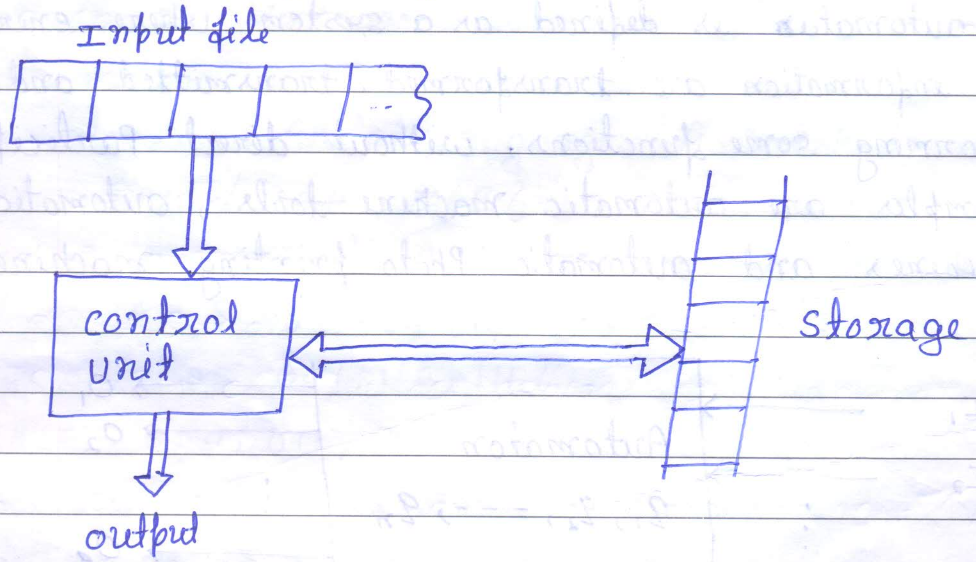
Note 1: Note that $AB \neq BA$.

2. closure operation :-

(i) Kleen closure :- This operation defines on a set S a derived set S^*, having as members, the empty string and all strings formed by concatenating a finite number of strings in S or alternatively.

The automaton can Produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet. The automaton can read and change the contents of the storage cells. Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner.

The following figure shows a schematic representation of a general automaton.



The automaton can be of two types, acceptors and transducers.
Acceptors give the results in the form of accepted or not accepted.
Transducers are the one which produce outputs.

Grammar :-

Each Language has a grammar and Each Grammar generate a Language.

A Grammar is $G = (V, T, P, S)$ where

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

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

(iii) S is a special variable called the start symbol.

(iv) P is a finite set whose elements are $\alpha \rightarrow \beta$ Productions where α and β are strings on $V \cup T$.
α has at least one symbol from V.
Elements of P are called Production rules.

Chomsky classification :- The four classes of Languages are often called the chomsky hierarchy.

1. Type 3 or Regular Grammar
2. Type 2 or Context Free Grammar
3. Type 1 or Context sensitive Grammar.
4. Type 0 or unrestricted or phase structured Grammar.

1. **Regular Grammar :-** A regular Grammar is one that is either Right linear or left linear.

(i) Right linear Grammar: A Grammar $G = (V, T, P, S)$ is said to be right linear if all Productions are of the form

$$A \rightarrow wB \quad \text{or} \quad A \rightarrow w$$

where $A, B \in V$ and $w \in T^*$

Example :- The Grammar $G = (\{S\}, \{a, b\}, P, S)$ where given as

$$S \rightarrow abS / a$$ is Right linear.

(ii) left-linear Grammar :- A Grammar $G = (V, T, P, S)$ is said to be left linear if all Productions are of the form

$$A \rightarrow Bw \quad \text{or} \quad A \rightarrow w$$

where $A, B \in V$ and $w \in T^*$

Example: The Grammar $G = (\{S, S_1, S_2\}, \{a, b\}, P, S)$ with Productions
$S \rightarrow S_1 ab$
$S_1 \rightarrow S_1 ab / S_2$
$S_2 \rightarrow a$ is left linear.

2. context free Grammar:- A context Free Grammar is denoted as $G = (V, T, P, S)$ where
(i) V is finite set of variables.
(ii) T is finite set of terminals.
V and T are disjoint.
(iii) P is a finite set of Productions. Each Productions is of the form $A \rightarrow \alpha$ where A is a variable.
α is a string of symbols from $(V \cup T)^*$. S is a special variable called the start variable (Symbol).

Example:- The Grammer $G = (V, T, P, S)$ where $V = \{S, A, B\}$, $T = \{a, b\}$ and P consists:-
$S \rightarrow aB$, $S \rightarrow bA$, $A \rightarrow a$, $A \rightarrow aS$, $A \rightarrow bAA$, $B \rightarrow b$, $B \rightarrow bS$, $B \rightarrow aBB$ is a context Free Grammar.

3. Context sensitive Grammar:- A Grammar $G = (V, T, P, S)$ is said to be context sensitive Grammar if the Production are of the form

$$\boxed{\alpha \rightarrow \beta}$$

where α is any string of terminals and non terminals with at least one non terminal.
β is any string of terminals and non terminal with $\beta \neq \epsilon$ with exception for starting symbols where $S \rightarrow \epsilon$ is Possible.
The restriction on the Productions is $|\beta| \geq |\alpha|$.
so the term context sensitive originates Productions of the form $\alpha_1 A \alpha_2 \longrightarrow \alpha_1 \beta \alpha_2 \quad (\beta \neq \epsilon)$
where α_1 is known as left context of A. and α_2 is

known as Right context of A.

<u>Example :-</u> $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ where P is defined as $S \to SB$, $SB \to SC$, $C \to a$.

4. Type 0 Grammar or unrestricted or phase structured Grammar.

A Grammar $G = (V, T, P, S)$ is said to be type 0 Grammar if its Productions are of the form

$$\boxed{\alpha \to \beta}$$

where (i) α is a string of symbols with atleast one non terminal and $\alpha \neq \epsilon$.

(ii) β is a string of terminals and nonterminal including ϵ, i.e., $\beta \in (V \cup T)^*$.

<u>Example:-</u> $G = (\{A, B, C\}, \{a, b, c\}, P, S)$. P is defined as $A \to AB$, $AB \to BC$, $B \to a$.

=> Automatic machine (has two Type)

- Machine without output (Finite Automata) [I/P String → Machine Program → Give decision that I/P string is valid or not]
- Machine with output (Melay & Moore Machine) [I/P String → Machine Program → O/p string]

1. <u>Finite Automata :-</u> Finite Automata is a Logical Machine that contains finite no. of states and which is used to check the validity of tokens. ~~(string)~~

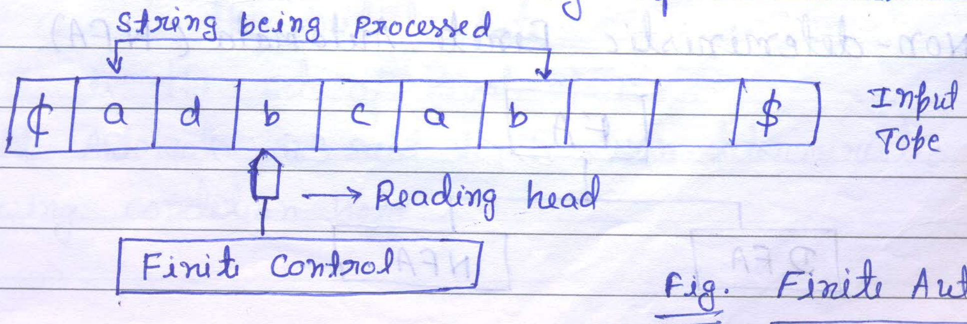


Fig. Finite Automata

Finite automata has Input Tape, Reading head & Finite contorol.

(i) <u>Input Tape:-</u> It is used for storing the input symbols. Input Tape is divided into squares where each square contain single symbol.

(ii) <u>Reading Head:-</u> It is used for reading the input from input Tape. The head Examines only one square at a time and can move one square either to the left or to the Right.

(iii) <u>Finite control:-</u> The finite control decides the next state on receiving particular input from input tape.

<u>Formal Definition of Finite Automata:-</u>

A Finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

(i) Q is a finite non-empty set of states.

(ii) Σ is a finite non-empty set of inputs.

(iii) δ is a transition function which describe the change of states during the transition.

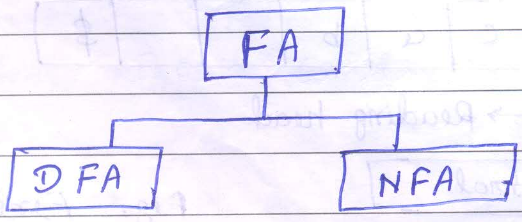
(iv) q_0 is the initial state. $(q_0 \in Q)$

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

⇒ Finite Automata has two types :- (FA)

(i) Deterministic Finite Automata (DFA)

(ii) Non-deteministic Finite Automata (NFA).



(i) Deterministic Finite Automata (DFA) :- A Deterministic Finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

(i) Q is a Finite non empty set of states.
(ii) Σ is a Finite non empty set of inputs.
(iii) δ is a direct transition Function which maps $Q \times \Sigma$ into Q.

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

(iv) $q_0 \in Q$ is the initial state.
(v) $F \subseteq Q$ is the set of final states.

Finite Automata is a logical machine that contains finite number of states and which is used to check the validity of tokens. ~~string~~

A finite Automata is said to be deterministic if the following condition hold:-

(i) only one Transition is possible for single I/P symbol.
(ii) Transitions are available for all input symbol on each state.

(ii) Non-Deterministic Finite Automata (NFA) :- Non-deterministic Finite automata can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

(i) Q is a Finite non empty set of states.
(ii) Σ is a Finite non empty set of inputs.
(iii) δ is a direct transition Function which maps $Q \times \Sigma$ into 2^Q.

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

(iv) $q_0 \in Q$ is the initial state.
(v) $F \subseteq Q$ is the set of final states.

A finite Automata is said to be Non-deterministic if the following condition hold:-

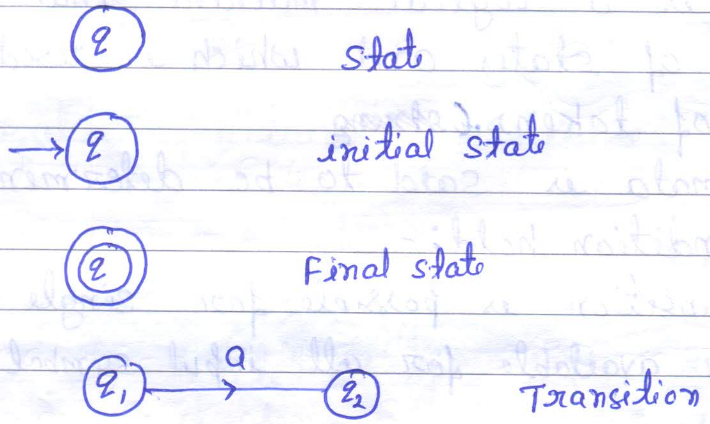
(i) More than one Transition is Possible for single I/p symbol.

(ii) It is not compulsory that Transitions are available for all input symbol on each state.

It ~~is used~~ ~~to~~ ~~also~~ check the validity of token in non-deterministic behaviour.

<u>Note:-</u> So Real Machine is implemented by DFA.

⇒ <u>Transition diagram:-</u>

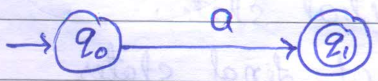


$\delta : q_1 \times a \rightarrow q_2$

$\delta(q_1, a) = q_2$

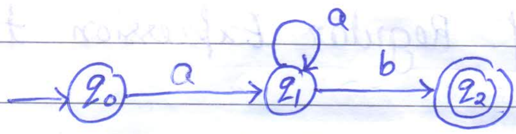
⇒ <u>Regular Expression:-</u> Regular Expression is a Pattern or string which is accepted by a Finite Automata. Regular Expression is used to represent token.

<u>Example 1:-</u>



Regular Expression (R.E.) = a

Example 2:-



Regular Expression (R.E.) = $a a^* b$.

⇒ **Regular set:-** A Regular set is a set of acceptable values or strings of any Regular Expression.

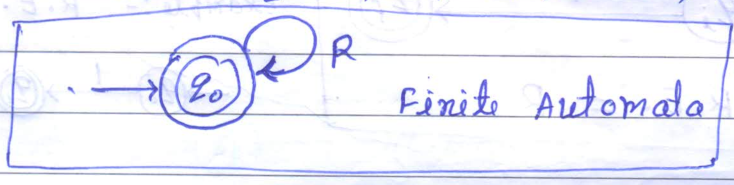
Example:-

if R.E. = aa^*b then

Regular set = { ab, aab, aaab, aaaab, ----- }

⇒ **Kleen closure (*) :-** let R is a Regular Expression (RE), the kleen closure represent for R is R^*

where $R^* = \{ \epsilon, R, RR, RRR, RRRR, ----- \}$



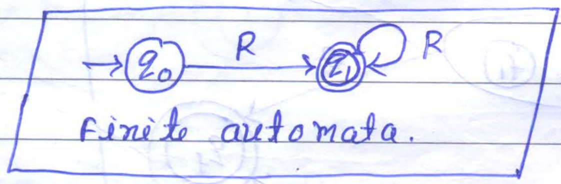
R.E. = R^*

Regular set of R^* = { ε, R, RR, RRR, ----- }

~~Here~~ ~~Regular~~ Kleen closure of any Regular Expression is also called the Regular Expression. so R^* is Regular Expression.

⇒ **Positive closure (+) :-** let R is a Regular Expression, the Positive closure R is represented by R^+ where

$R^+ = \{ R, RR, RRR, RRRR, ---- \}$



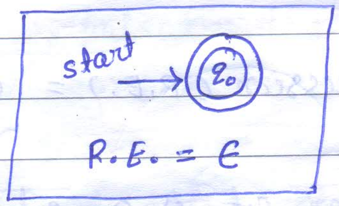
Positive closure of any Regular Expression is called Regular Expression. so R^+ is Regular Expression.

R.E. = R^+

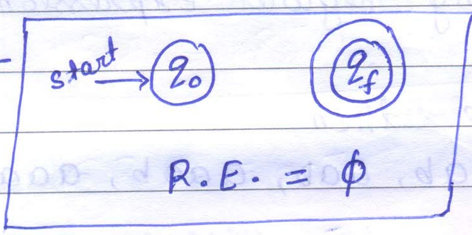
Regular set of R^+ = { R, RR, RRR, RRRR, ---- }

⇒ Conversion of Regular Expression to Finite Automata. (ε–NFA)

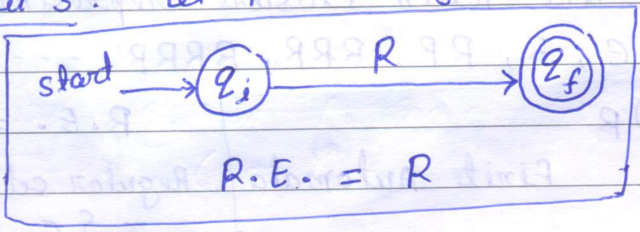
Format 1 :-



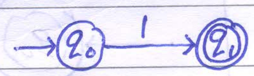
Format 2 :-



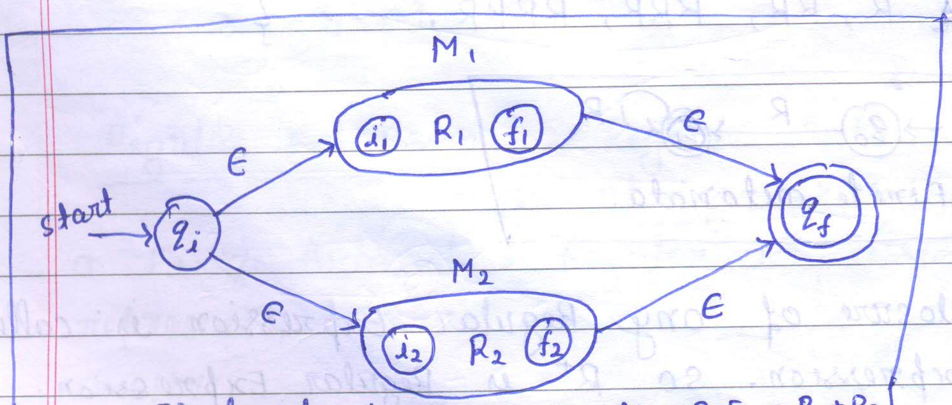
Format 3 :- let R is a Regular Expression (R.E.) the Finite Automata for R is :-



Example :- R.E. = 1



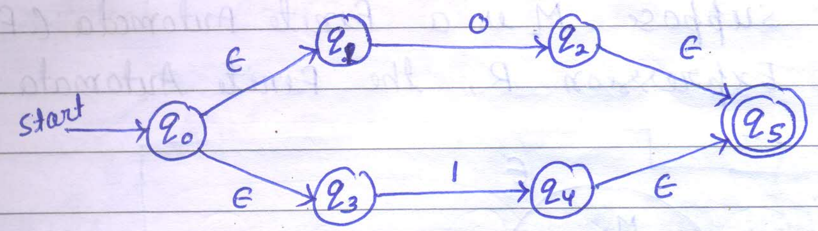
Format 4 :- Suppose M_1 & M_2 are two FA's for Regular Expression R_1 & R_2 respectively. The FA for $R_1 + R_2$ (OR operation) is :-



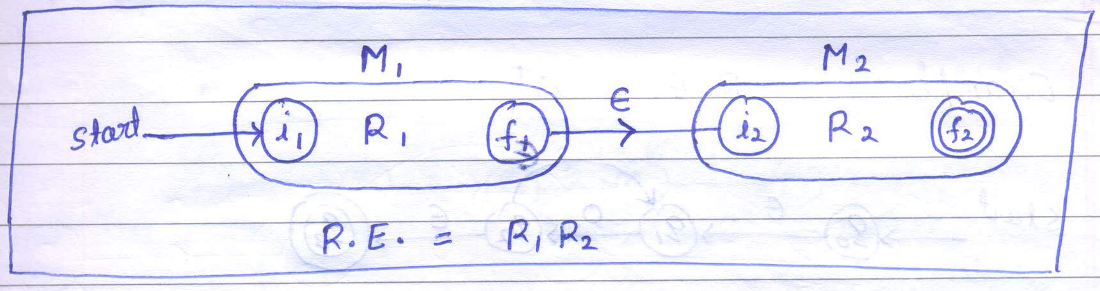
Finite automata for $R_1 + R_2$ then R.E. = $R_1 + R_2$

Example :-

R.E. = 0+1

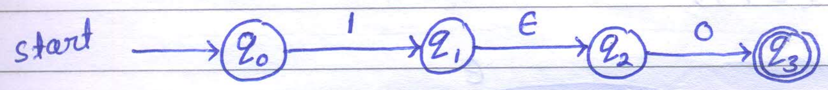


Format 5 :- suppose M_1 & M_2 are two FA's (Finite Automata's) for Regular Expressions R_1 & R_2 respectively. The Finite Automata (FA) for R_1R_2 is :-

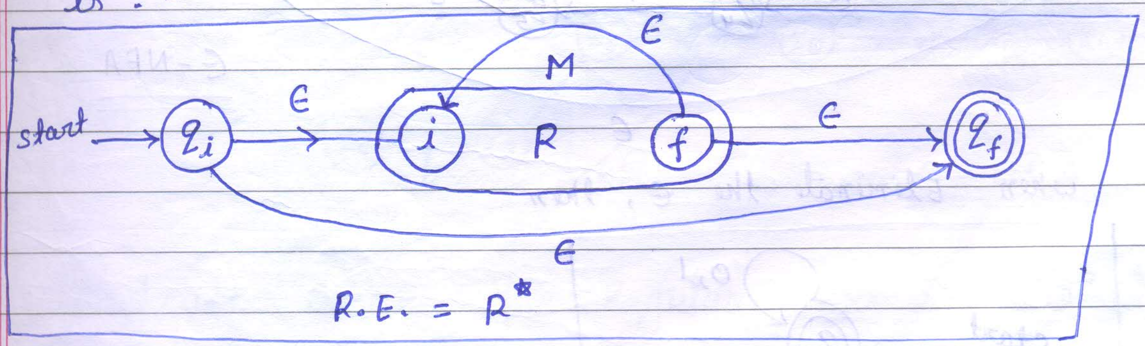


Example :-

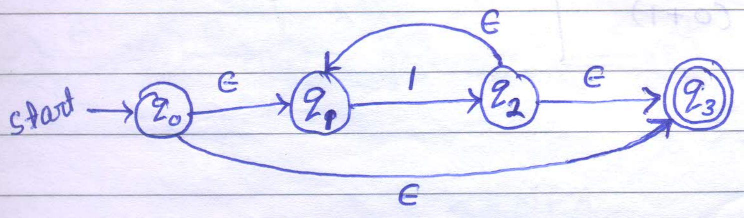
R.E. = 10



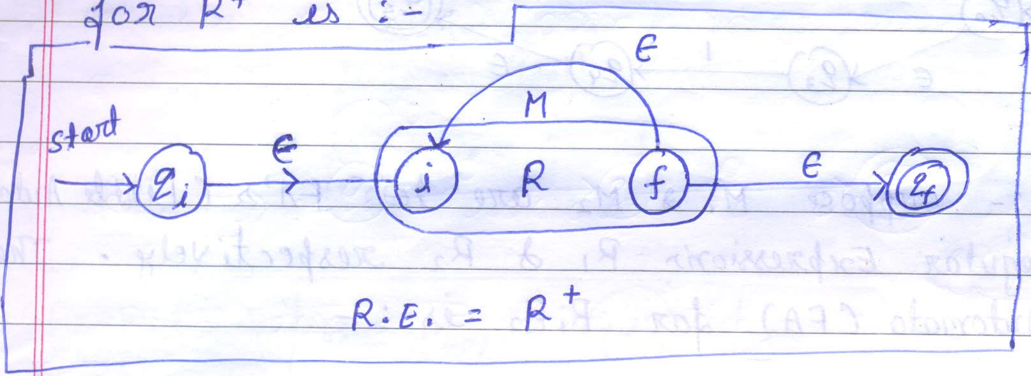
Format 6 :- suppose M is a Finite Automata for Regular Expression (R.E.) R, the Finite Automata (FA) for R^* is :-



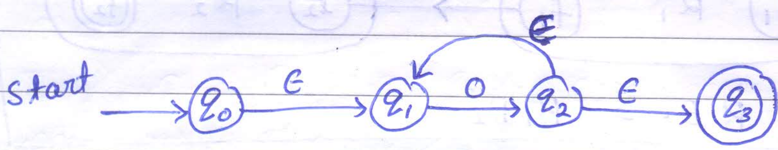
Example: R.E. = 1^*



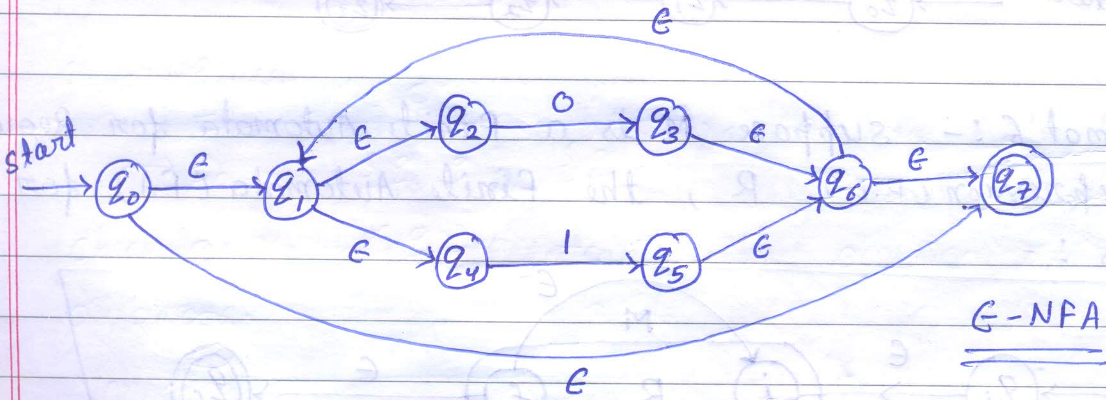
Format 7 :- Suppose M is a Finite Automata (FA) for Regular Expression R, the Finite Automata (FA) for R^+ is :-



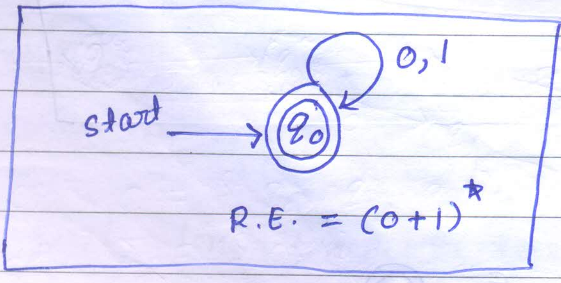
Example :- R.E. = 0^+



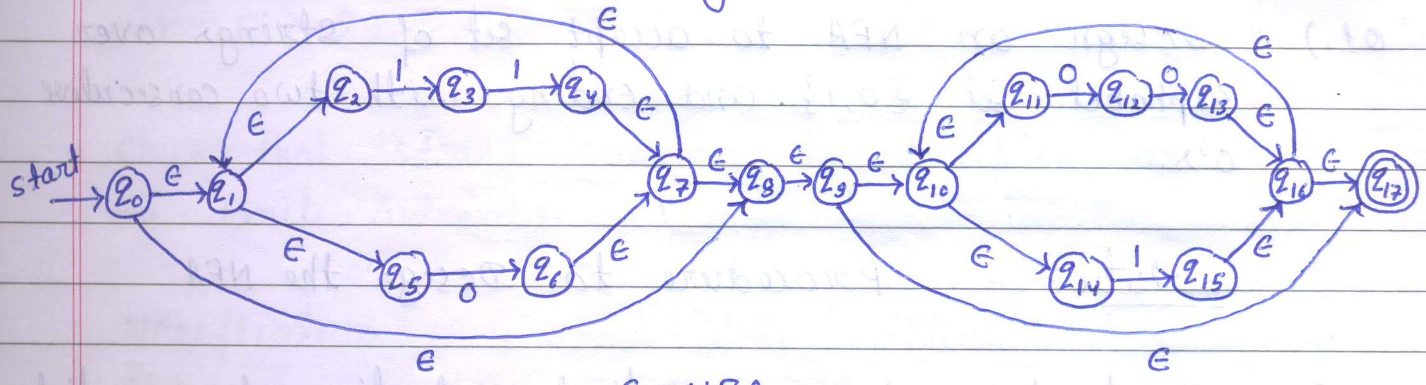
Q1) Construct Finite Automata (ε-NFA) for ~~following~~ Regular Expression $(0+1)^*$



when Eliminate the ε, then

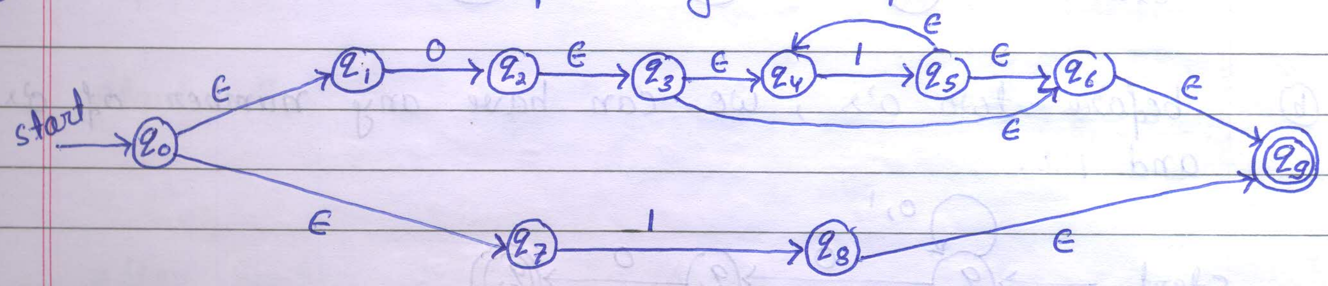


Q2) Construct ε-NFA for Regular Expression $(11+0)^*(00+1)^*$.



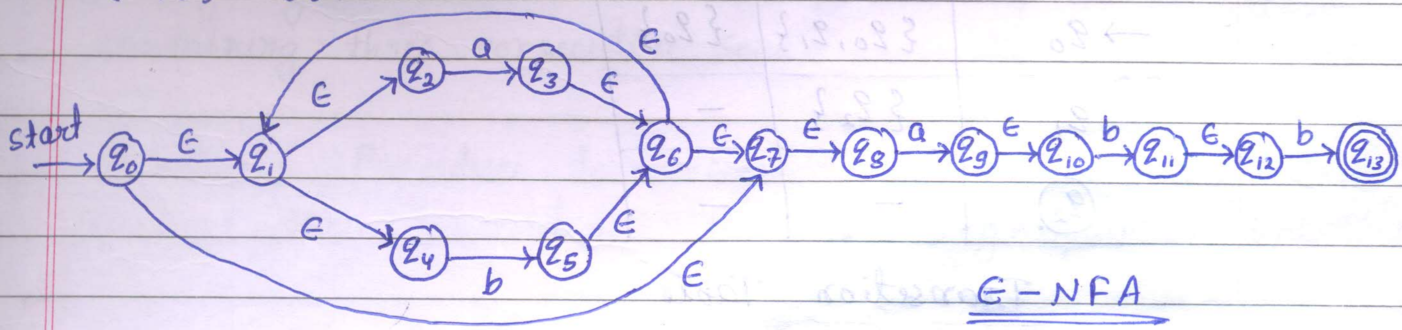
ε-NFA

Q3) Construct ε-NFA for Regular Expression 01^*+1.



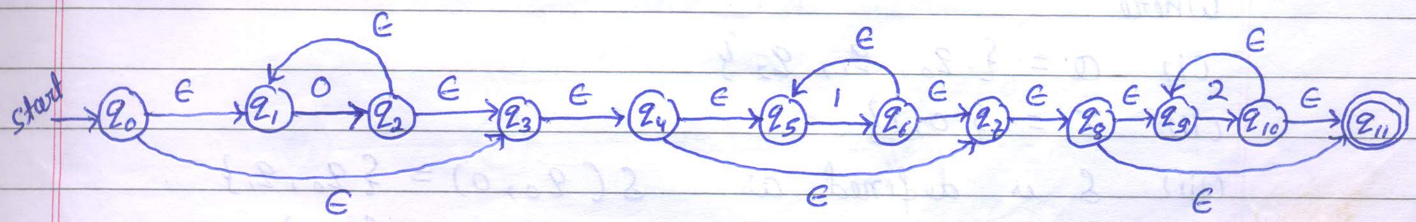
ε-NFA

Q4.) Construct ε-NFA for Regular Expression $(a+b)^*abb$ or $(a/b)^*abb$.



ε-NFA

Q5) Construct ε-NFA for Regular Expression $0^*1^*2^*$.



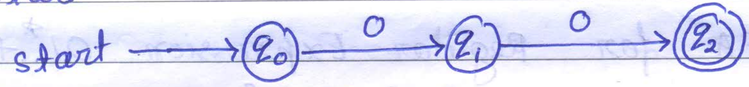
ε-NFA

Design of NFA

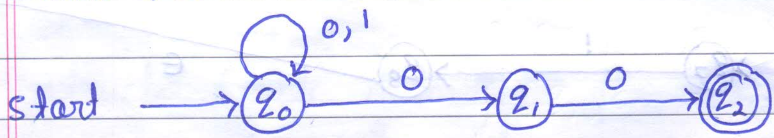
Q1.) Design an NFA to accept set of strings over alphabet set {0,1} and ending with two consecutive 0's.

Solution:- Procedure to Design the NFA

(a) Read two 0's consecutively starting from initial state.



(b) before two 0's, we can have any number of 0's and 1's.



Transition diagram   R.E. = $(0+1)^*00$

state/input	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	—
q_2	—	—

Transition Table

NFA represent by 5 Tupples $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_0\}$

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

(iv) q_0 is the initial state.
(v) $F = \{q_2\}$

Check that string 100100 is accepted or not by Finite Automata.

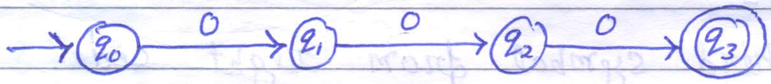
Verification :- Start from initial state. after Reading whole string if we obtain final state then string is accepted. if we obtain other state in all possible alternative then string is not accepted.

Start $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$ ✗
$q_0 \xrightarrow{0} q_1$ ~~dead~~
$q_0 \xrightarrow{0} q_1$ ~~dead~~
$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2$

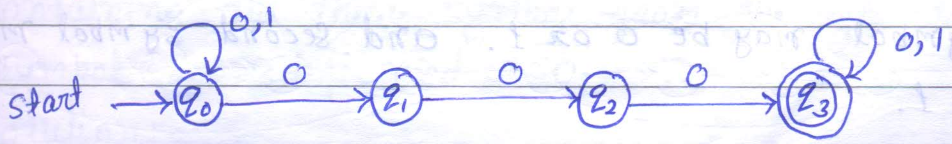
after Reading the whole string, we obtain final state q_2, so it is accepted.

Q2.) Designing an NFA which accepts set of all strings containing three consecutive zero's.

Solution:- Procedure to Design the NFA
(a) Read three consecutive zero's, starting from initial state.



(b) before and after three consecutive zero's, we can have any number of 0's and 1's.



Transition diagram   R.E. = $(0+1)^* 000 (0+1)^*$

state/input	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	—
q_2	$\{q_3\}$	—
q_3	$\{q_3\}$	$\{q_3\}$

Transition Table

NFA Represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as

$$\delta(q_0, 0) = \{q_0, q_1\}$$
$$\delta(q_0, 1) = \{q_0\}$$
$$\delta(q_1, 0) = \{q_2\}$$
$$\delta(q_2, 0) = \{q_3\}$$
$$\delta(q_3, 0) = \{q_3\}$$
$$\delta(q_3, 1) = \{q_3\}$$

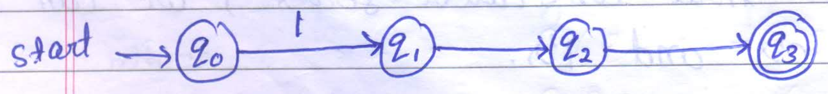
(iv) q_0 is the initial state.

(v) $F = \{q_3\}$

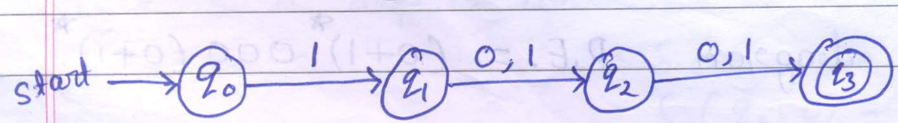
Q3) Design NFA which accepts set of all binary strings containing 3^{rd} symbol from right side is 1.

Solution :-

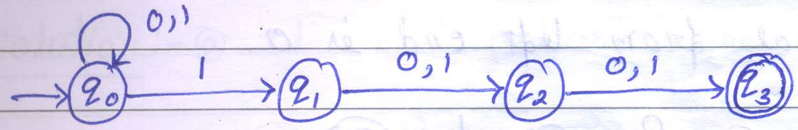
ⓐ Read 1 as Third symbol from right side.



ⓑ First symbol may be 0 or 1. and second symbol may be 0 or 1.



(c) Fourth or upward symbol from right side may be 0 or 1.



Transition diagram

R.E. = $(0+1)^* 1 (0+1)(0+1)$

state/input	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	—	—

Transition Table

NFA Represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as
$\delta(q_0, 0) = \{q_0\}$ $\quad$ $\delta(q_0, 1) = \{q_0, q_1\}$
$\delta(q_1, 0) = \{q_2\}$ $\quad$ $\delta(q_1, 1) = \{q_2\}$
$\delta(q_2, 0) = \{q_3\}$ $\quad$ $\delta(q_2, 1) = \{q_3\}$

(iv) q_0 is the initial state.

(v) $F = \{q_3\}$

Q4) Design an NFA which accept set of all binary strings containing the third symbol from the left end is 1 and second symbol from left end is 0.

Solution:-

(a) Third symbol from the left end is 1.

$$\text{start} \longrightarrow q_0 \longrightarrow q_1 \longrightarrow q_2 \xrightarrow{1} (q_3)$$

(b) second symbol from left end is 0.

$$\text{start} \longrightarrow q_0 \longrightarrow q_1 \xrightarrow{0} q_2 \xrightarrow{1} (q_3)$$

(c) First symbol may be 0 or 1.

$$\text{start} \longrightarrow q_0 \xrightarrow{0,1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} (q_3)$$

(d) Fourth or upward symbol from left may be 0 or 1.

$$\text{start} \longrightarrow q_0 \xrightarrow{0,1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} (q_3) \circlearrowleft 0,1$$

Transition diagram   R.E. = $(0+1)01(0+1)^*$

state/input	0	1
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$
q_1	$\{q_2\}$	–
q_2	$\{q_3\}$	–
(q_3)	$\{q_3\}$	$\{q_3\}$

Transition Table.

NFA represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = \{q_1\}$, $\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \{q_2\}$, $\delta(q_2, 0) = \{q_3\}$

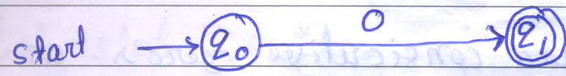
$\delta(q_3, 0) = \{q_3\}$, $\delta(q_3, 1) = \{q_3\}$

(iv) q_0 is the initial state.

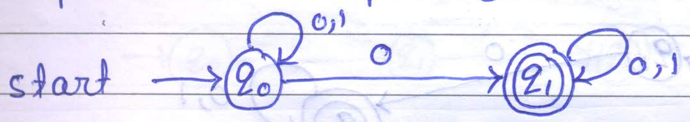
(v) $F = \{q_3\}$.

Q5) construct a NFA accepting all strings that have atleast one 0.

Solution:- (a) It has atleast one 0.



(b) after ~~zero~~ or before zero it has any number of 0's & 1's.



Transition diagram

R.E. = $(0+1)^* 0 (0+1)^*$

state / input	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_1\}$	$\{q_1\}$

Transition Table.

NFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1\}$

(ii) $\Sigma = \{0, 1\}$

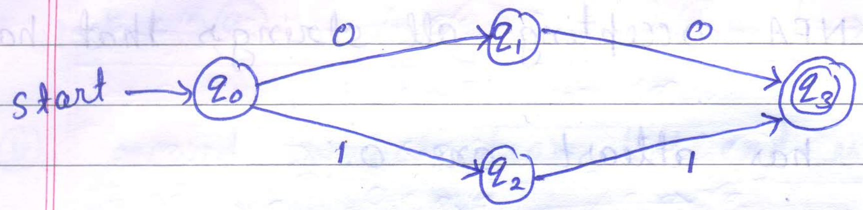
(iii) δ is defined as $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_0\}$
$\delta(q_1, 0) = \{q_1\}$, $\delta(q_1, 1) = \{q_1\}$

(iv) q_0 is the initial state.

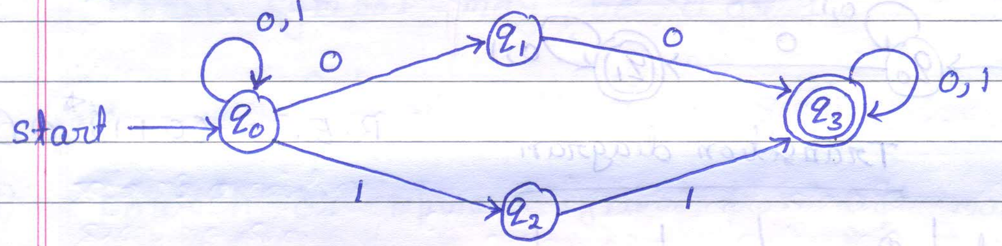
(v) $F = \{q_1\}$.

Q6) Design a NFA to accept the strings with 0's and 1's such that string contains either two consecutive 0's or two consecutive 1's.

Solution:- (a) It reads either two consecutive zero's or two consecutive 1's.



(b) before and after two consicutive zero's or 1's, we have any number of zero's & 1's.



Transition diagram

$$R.E. = (0+1)^* (00+11) (0+1)^*$$

state / input	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_3\}$	—
q_2	$\{$ —	$\{q_3\}$
q_3	$\{q_3\}$	$\{q_3\}$

Transition diagram

NFA represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_0, q_2\}$
$\delta(q_1, 0) = \{q_3\}$, $\delta(q_2, 1) = \{q_3\}$
$\delta(q_3, 0) = \{q_3\}$, $\delta(q_3, 1) = \{q_3\}$

(iv) q_0 is the initial state.

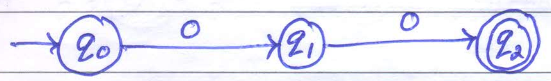
(v) $F = \{q_3\}$.

Design of DFA

Q1) construct a DFA for the strings of 0's and 1's that always ends with substring 00.

Solution:-

Read two 0's consecutively starting from initial state. suppose after reading two 0's it will enter into final state.



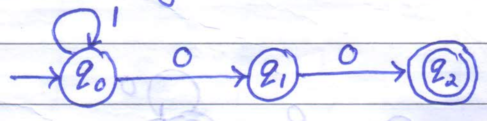
{ compulsory condition }

(i) when requires 00 at end then goto on q_0.
(ii) when one 0 is already available at end and requires only one 0 then goto on q_1.
(iii) when 00 is already available at end then goto on q_2.

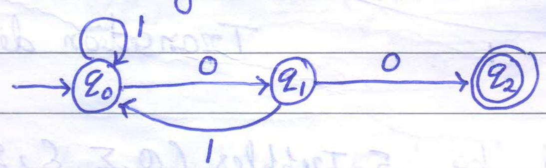
Note: Apply all input symbol on each state in DFA.

⇒ **Method 1:-**

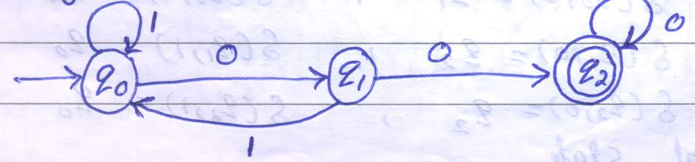
ⓐ if we read 1 on q_0, then requires 00 at end to get final state so go to on q_0.



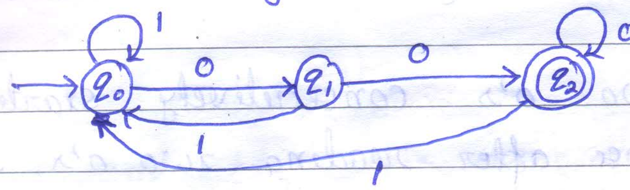
ⓑ if we read 1 on q_1, then requires 00 at end to get final state so go to on q_0.



ⓒ if we read 0 on q_2, then 00 is already available at end so goto on q_2.



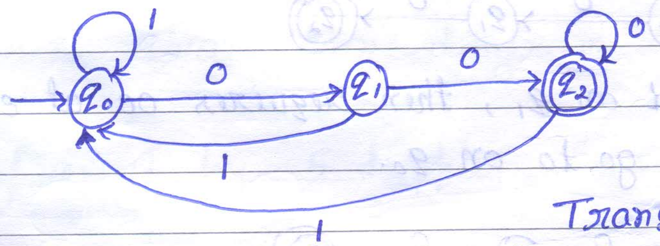
(d) if we read 1 on q_2, then requires 00 at end to get final state so goto on q_0.



Transition diagram

⇒ Method 2 :- (Procedure) Find the Next state :-

string read from starting state to current state	current state	Input on current state	string read from starting state to next state	(logic) Requirement end with 00	Next state
ε	q_0	1	1	00	q_0
0	q_1	1	01	00	q_0
00	q_2	0	000	–	q_2
00	q_2	1	001	00	q_0



Transition diagram

DFA is represented by 5 Tupples $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_0$

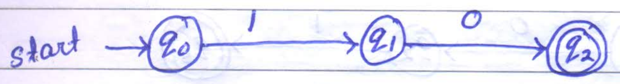
(iv) q_0 is the initial state.

(v) $F = \{q_2\}$.

Q2) construct a DFA for the strings of 0's and 1's end with substring 10.

Solution:-

Read 10 starting from initial state, suppose after reading 10 it will enter into final state.



{compulsory condition}

(i) when requires 10 at end then goto on q_0.

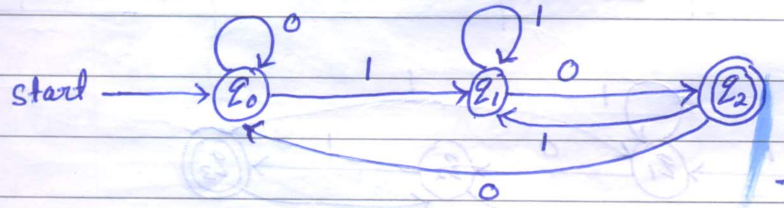
(ii) when 1 is already available and requires only 0 at end then goto on q_1.

(iii) when 10 is already available at end then goto on q_2.

⇒ Find the Next state:-

Note:- Apply all input symbol on each state in DFA.

string read from starting state to current state	current state	Input on current state	string read from starting state to Next state	(Logic) Requirement end with 10	Next state
∈	q_0	0	0	10	q_0
1	q_1	1	11	0	q_1
10	q_2	0	100	10	q_0
10	q_2	1	101	0	q_1



Transition diagram

DFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$ | $\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_1$ | $\delta(q_2, 0) = q_0$, $\delta(q_2, 1) = q_1$.

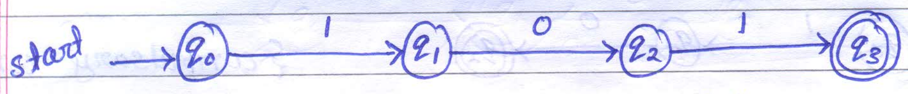
(iv) q_0 is the initial state.

(v) $F = \{q_2\}$.

Q3) Construct a DFA for the string of 0's and 1's ends with substring 101.

Solution :-

Read 101 starting from initial state. suppose after reading 101 it will enter into final state.



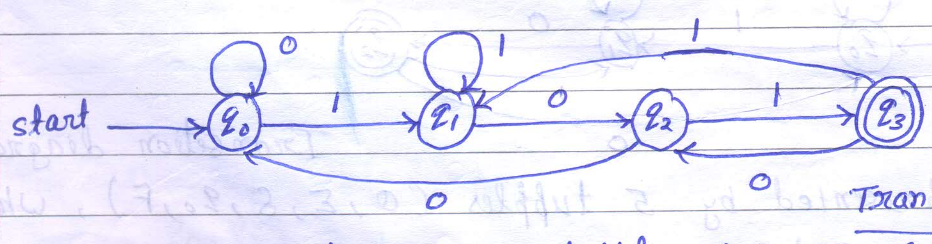
compulsory condition.

(i) when requires 101 at end then goto on q_0.
(ii) when requires 01 at end then goto on q_1.
(iii) when requires 1 at end then goto on q_2.
(iv) when 101 is available at end then goto on q_3.

⇒ Find the Next state :-

Note :- Apply all input symbol on each state in DFA.

string read from starting state to current state	current state	Input on current state	string read from starting state to Next state	(logic) Requirement end with 101	Next state
ε	q_0	0	0	101	q_0
1	q_1	1	11	01	q_1
10	q_2	0	100	101	q_0
101	q_3	0	1010	1	q_2
101	q_3	1	1011	01	q_1



Transition diagram

DFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$, (ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as

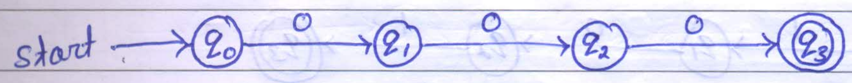
$\delta(q_0, 0) = q_0$	$\delta(q_1, 0) = q_2$	$\delta(q_2, 0) = q_0$	$\delta(q_3, 0) = q_2$
$\delta(q_0, 1) = q_1$	$\delta(q_1, 1) = q_1$	$\delta(q_2, 1) = q_3$	$\delta(q_3, 1) = q_1$

(iv) q_0 is the initial state. (v) $F = \{q_3\}$.

Q4) construct a DFA for string of 0's and 1's having three consecuti zero's as a substring.

Solution :-

Read 000 starting from initial stat. suppose after reading 000 it will enter into final stat.



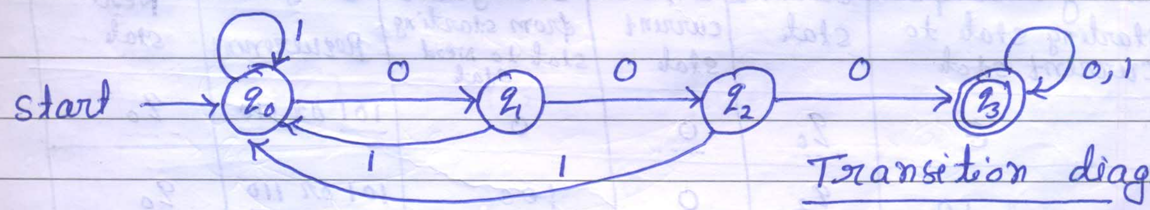
compulsory condition

(i) when it requires three consecutive zero's then goto on q_0.
(ii) when it requires 00 then goto on q_1.
(iii) when it requires 0 then goto on q_2.
(iv) when three consecutive zero's are available then goto on q_3.

=> Find the Next stat :-

Note : Apply all input symbol on each stat in DFA.

string read from starting stat to current stat	current stat	Input on current stat	string read from starting stat to Next stat	Logic Requirement (Three consecutive zero's)	Next stat
ε	q_0	1	1	000	q_0
0	q_1	1	01	000	q_0
00	q_2	1	001	000	q_0
000	q_3	0	0000	—	q_3
000	q_3	1	0001	—	q_3



Transition diagram

DFA is Represented by 5-Tupples $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2, q_3\}$ (ii) $\Sigma = \{0, 1\}$

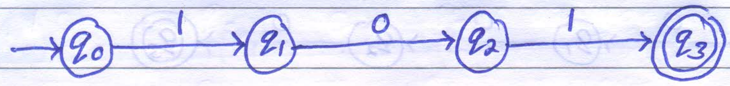
(iii) δ is defined as $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$ | $\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_0$ | $\delta(q_2, 0) = q_3$, $\delta(q_2, 1) = q_0$ | $\delta(q_3, 0) = q_3$, $\delta(q_3, 1) = q_3$

(iv) q_0 is the initial stat. (v) $F = \{q_3\}$.

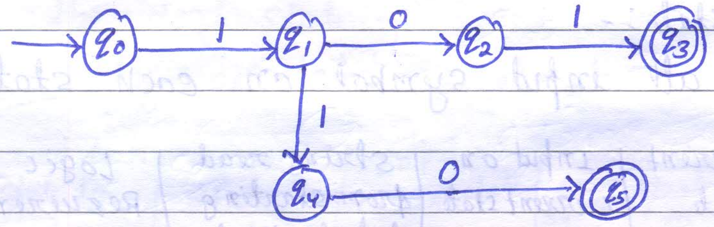
Q5) Construct a DFA for string of 0's and 1's ends with substring 101 or 110.

Solution:-

Reads 101 starting from initial state. Suppose after reading 101 it will enter into final state.



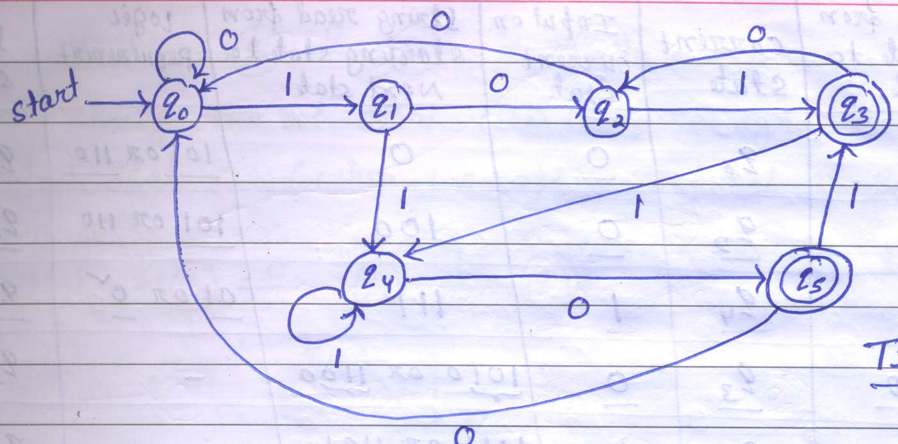
or also read 110 starting from initial state. 1 is already read on q_0 so read 10 from q_1 and enter into another final state.



(i) when requires 101 or 110 at end then goto on q_0.
(ii) when requires 1 at end then goto on q_2. ~~by reading 1~~
(iii) when requires 0 at end then ~~requires~~ goto on q_4. ~~by reading 0~~
(iv) when ~~requires — or 10 at end then goto on q_3 by —~~.
101 is available then goto on q_3.
(v) when 110 is available then goto on q_5.

Note :- Apply all input symbol on each state in DFA.

string read from starting state to current state	current state	Input on current state	string read from starting state to Next state	C(logic) Requirement	Next state.
ε	q_0	0	0	101 or 110	q_0
10	q_2	0	100	101 or 110	q_0
101	q_3	0	1010	1 or 110	q_2
101	q_3	1	1011	01 or 0	q_4
11	q_4	1	111	01 or 0	q_4
110	q_5	0	1100	101 or 110	q_0
110	q_5	1	1101	- or 10	q_3



Transition diagram.

DFA is Represented by five Tupples $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as

$\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_4$

$\delta(q_2, 0) = q_0$, $\delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_2$, $\delta(q_3, 1) = q_4$

$\delta(q_4, 0) = q_5$, $\delta(q_4, 1) = q_4$

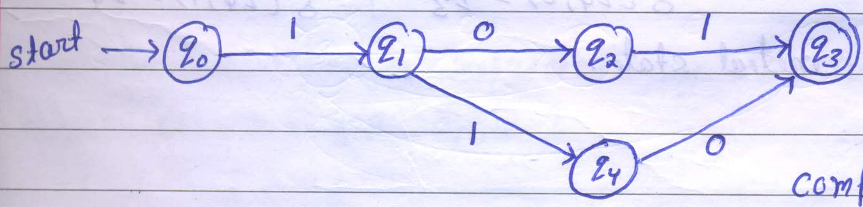
$\delta(q_5, 0) = q_0$, $\delta(q_5, 1) = q_3$

(iv) q_0 is the initial state.

(v) $F = \{q_3, q_5\}$.

Q6) Construct a DFA for string of 0's and 1's having 101 or 110 as a substring.

Solution:- Reads 101 or 110 starting from initial state. suppose after reading 101 or 110 it will enter into final state.



compulsory condition.

(i) when requires 101 or 110 then goto on q_0.

(ii) when requires 1 then goto on q_2.

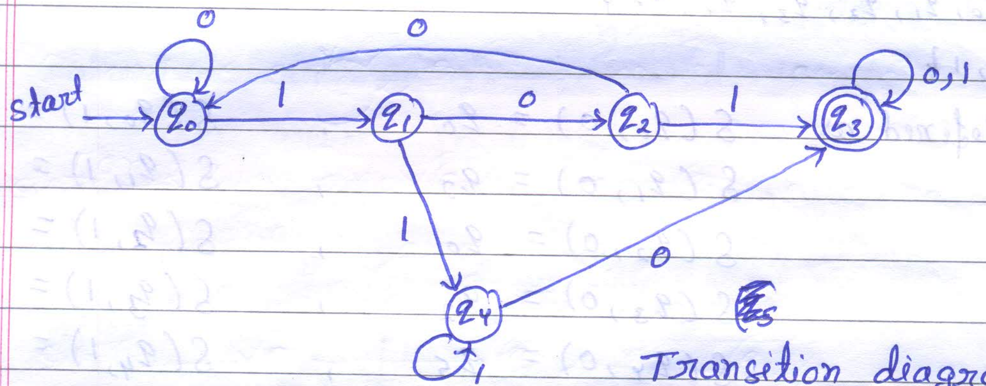
(iii) when requires 0 then goto on q_4.

(iv) when ~~requires~~ 101 or 110 is available then goto on q_3.

Note:- Apply all input symbol on each state in DFA.

⇒ Find the Next state :-

String read from starting state to current state	Current state	Input on current state	String read from starting state to Next state	Logic Requirement	Next state
ϵ	q_0	0	0	101 or 110	q_0
10	q_2	0	100	101 or 110	q_0
11	q_4	1	111	01 or 0	q_4
101 or 110	q_3	0	1010 or 1100	-	q_3
101 or 110	q_3	1	1011 or 1101	-	q_3



Transition diagram

DFA is represented by 5 tupple $(Q, \Sigma, \delta, q_0, F)$, where

(i) $Q = \{q_0, q_1, q_2, q_3, q_4\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as

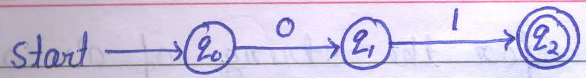
$\delta(q_0, 0) = q_0$	$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_2$	$\delta(q_1, 1) = q_4$
$\delta(q_2, 0) = q_0$	$\delta(q_2, 1) = q_3$
$\delta(q_3, 0) = q_3$	$\delta(q_3, 1) = q_3$
$\delta(q_4, 0) = q_3$	$\delta(q_4, 1) = q_4$

(iv) q_0 is the initial state.

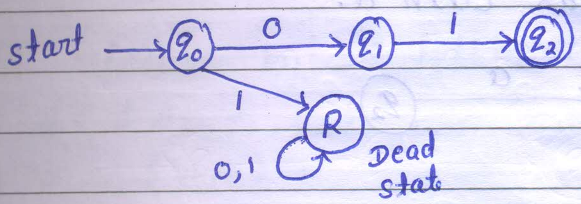
(v) $F = \{q_3\}$.

Q7) Construct a DFA which accepts a language L over $\Sigma = \{0, 1\}$ in which every string start with 0 and ends with 1.

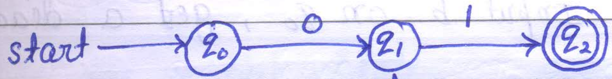
Solution:- Here condition is that every string start with 0 and ends with 1.



Here string can not start with 1. so ignore the input 1 on initial state q_0. (when we apply Input 1 on q_0, get a dead state.)



Ignore the dead state because we can not reach to the final state from dead state.

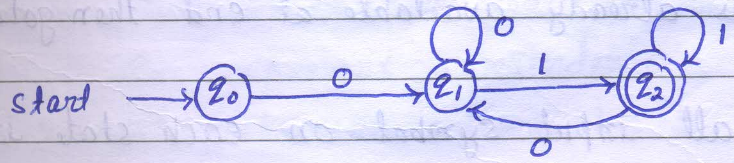


(i) when it requires 1 at end then goto on q_1.

(ii) when 1 is already available at end then go to on q_2.

⇒ Find Next state:-

string read from initial state to current state	current state	Input on current state	string read from initial state to Next state	Requirement end with 1.	Next state
0	q_1	0	00	1	q_1
01	q_2	0	010	1	q_1
01	q_2	1	011	-	q_2



Transition diagram

Finite Automata is represent by 5 Tupples $(Q, \Sigma, \delta, q_0, F)$,

Where (i) $Q = \{q_0, q_1, q_2\}$

(ii) $\Sigma = \{0, 1\}$

(iii) δ is defined as $\delta(q_0, 0) = q_1$ | $\delta(q_1, 0) = q_1$ | $\delta(q_2, 0) = q_1$

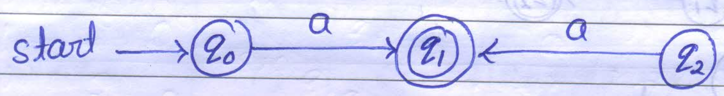
$\delta(q_1, 1) = q_2$ | $\delta(q_2, 1) = q_2$

(iv) q_0 is the initial state.

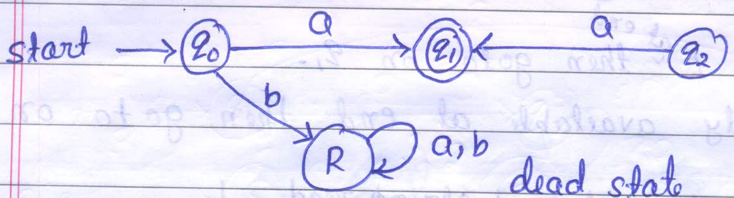
(v) $F = \{q_2\}$.

Q8) construct a DFA for the string of a's and b's start and ends with a.

Solution:- Here condition is that strings start with a and ends with a.

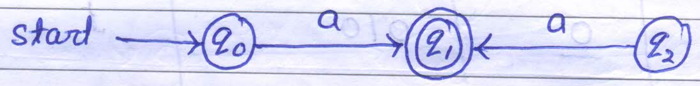


we can not start the string with b so ignore the input b on initial state q_0.
(when we apply Input b on q_0, get a dead state)



dead state

Ignore the dead state because we can not reach to the final state from dead state.

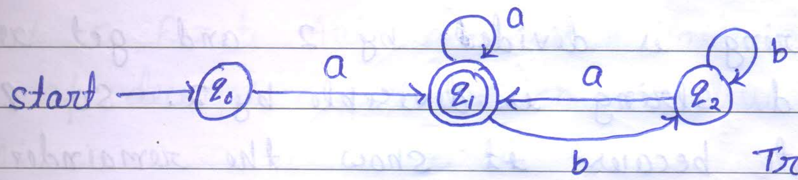


(i) when requires a at end then goto on q_2.
(ii) when a is already available at end then goto on q_1.

Note: Apply all input symbol on each state in DFA.

Find Next state:-

String read from initial state to current state	current state	Input on current state	String read from initial state to Next state	Requirement	Next state
a	q_1	a	aa	–	q_1
a	q_1	b	ab	a	q_2
ab	q_2	b	abb	a	q_2



Transition diagram

DFA represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2\}$

(ii) $\Sigma = \{a, b\}$

(iii) δ is defined as $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_1$, $\delta(q_2, a) = q_1$ | $\delta(q_1, b) = q_2$, $\delta(q_2, b) = q_2$.

(iv) q_0 is the initial state.

(v) $F = \{q_1\}$.

Q9) construct a Finite Automata which will accept those strings of decimal digits which are divisible by 2.

Solution:-

Here $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

(i) Assume S is the initial state.
Two Remainders 0 and 1 occurs, when string is divided by 2. so Two states are required named q_0 and q_1, where

(ii) state q_0 represent remainder 0.

(iii) state q_1 represent remaindar 1.

Inputs is divided into two subset according remainder property.

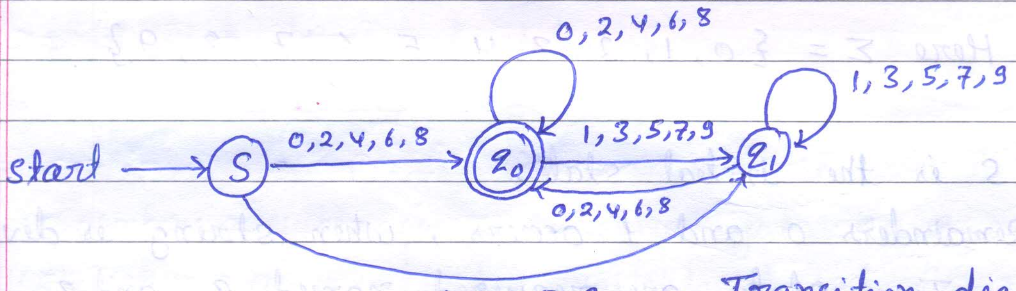
(i) $\Sigma_0 = \{0, 2, 4, 6, 8\}$. Σ_0 contains the no. which gives remainder 0, when no. is divided by 2.

(ii) $\Sigma_1 = \{1, 3, 5, 7, 9\}$. Σ_1 contains the no. which gives remainder 1, when no. is divided by 2.

all input of ~~same~~ subset have the same Transition. so solve any one input and apply on all inputs.

when string is divided by 2 and get remainder 0. then said string is divisible by 2. so q_0 is the final state because it show the remainder 0.

Read string from initial to current state	current state	any one Input on current state from subset	string from initial to next state	Remainder when string is divided by 2.	Next state
ϵ	S	2	2	0	q_0
ϵ	S	3	3	1	q_1
2	q_0	2	22	0	q_0
2	q_0	3	23	1	q_1
3	q_1	2	32	0	q_0
3	q_1	3	33	1	q_1



Transition diagram.

DFA is represented by 5 Tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{S, q_0, q_1\}$

(ii) $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

(iii) δ is defined as in Transition diagram.

(iv) S is the initial state.

(v) $F = \{q_0\}$.

Q10) construct a Finite Automata which will accept those strings of decimal digits which are divisible by 3.

Solution:-

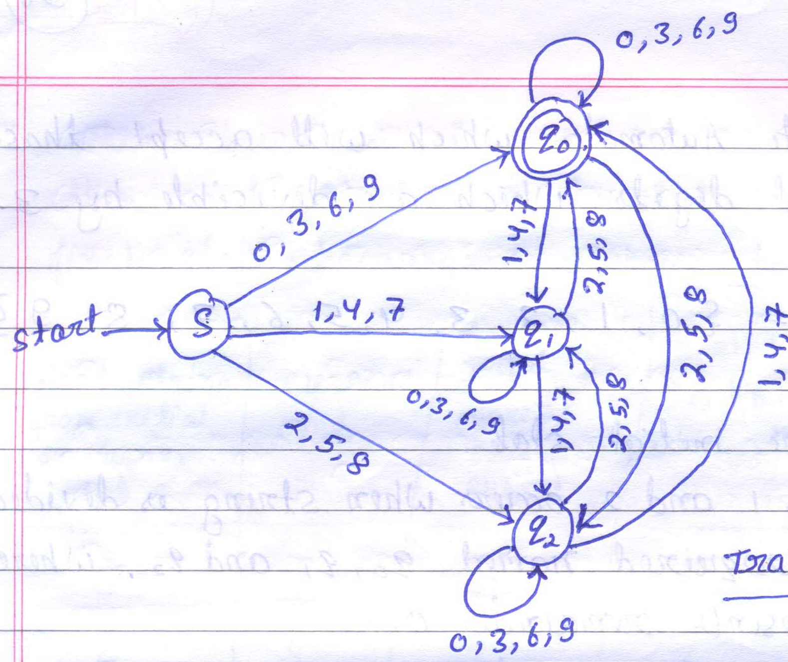
Here $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

(i) Assume S is the initial state.
Three remainders 0, 1 and 2 occurs when string is divided by 3.
so Three states are required named q_0, q_1 and q_2, where
(ii) state q_0 represents remainder 0.
(iii) state q_1 represents remainder 1.
(iv) state q_2 represents remainder 2.

Input is divided into three subset according remainder that comes when Input is divided by 3.
(i) Σ_0 contains no., which gives remainder 0. $\Sigma_0 = \{0, 3, 6, 9\}$
(ii) Σ_1 contains no., which gives remainder 1. $\Sigma_1 = \{1, 4, 7\}$
(iii) Σ_2 contains no., which gives remainder 2. $\Sigma_2 = \{2, 5, 8\}$

q_0 is the final state because it represent remainder 0.
all input of subset have same Transition. so solve by any one input and apply on all inputs.

Read string from initial to current state	current state	any one input from subset on current state	string from initial to next state	Remainder when string is divided by 3.	Next state
ϵ	S	0	0	0	q_0
ϵ	S	1	1	1	q_1
ϵ	S	2	2	2	q_2
3	q_0	0	30	0	q_0
3	q_0	1	31	1	q_1
3	q_0	2	32	2	q_2
1	q_1	0	10	1	q_1
1	q_1	1	11	2	q_2
1	q_1	2	12	0	q_0
2	q_2	0	20	2	q_2
2	q_2	1	21	0	q_0
2	q_2	2	22	1	q_1



Transition diagram

DFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{S, q_0, q_1, q_2\}$

(ii) $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

(iii) δ is defined as in Transition diagram.

(iv) S is the initial state.

(v) $F = \{q_0\}$.

Q11) construct a Finite Automata which will accept those strings of binary no. which are divisible by 2.

Solution:-

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

(i) Assum S is the initial state.

Two Remainder's are Possible when string is divided by 2

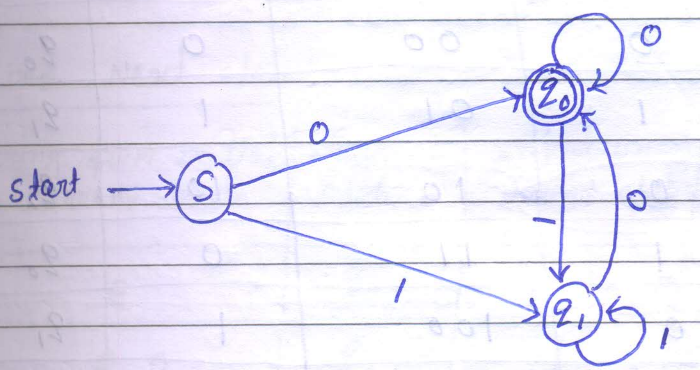
So two states are required, named q_0 and q_2, where

(ii) State q_0 represent remainder 0.

(iii) State q_1 represent remainder 1.

q_0 is the final state because it represents remainder 0

Read string from initial to current state	current state	Input on current state	string from initial to next state	Remainder when we divide by 2.	Next state
ϵ	S	0	0	0	q_0
ϵ	S	1	1	1	q_1
0	q_0	0	00	0	q_0
0	q_0	1	01	1	q_1
1	q_1	0	10	0	q_0
1	q_1	1	11	1	q_1



Transition diagram

DFA is represent by 5 Tupples $(Q, \Sigma, \delta, q_0, F)$, where
(i) $Q = \{S, q_0, q_1\}$
(ii) $\Sigma = \{0, 1\}$
(iii) δ is defined in Transition diagram.
(iv) S is the initial state.
(v) $F = \{q_0\}$

Q12) construct a Finite Automata which will accept those strings of binary number which are divisible by 3.

Solution:-

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

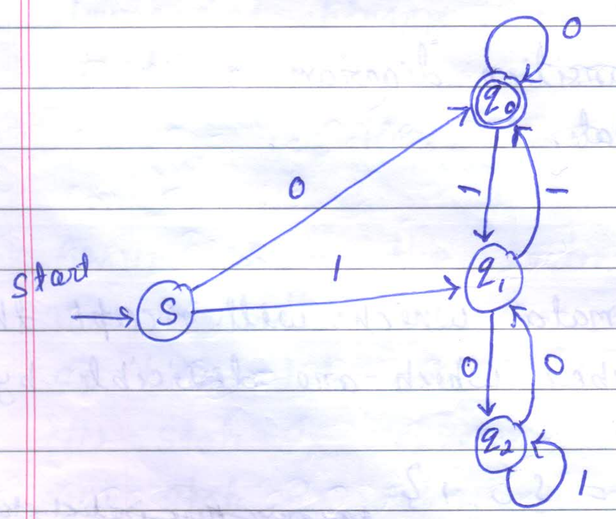
(1) Assume S is the initial state. ~~because the initial state is~~

Three Remainder's are Possible when string is divided by 3. So Three states are required named q_0, q_1 & q_2. where

(ii) state q_0 represent remainder 0.
(iii) state q_1 represent remainder 1.
(iv) state q_2 represent remainder 2.

q_0 is the Final state because it represent remainder 0

string from initial to current state	current state	Input on current state	string from initial to next state	Remainder when divide by 3	Next state
ε	S	0	0	0	q_0
ε	S	1	1	1	q_1
0	q_0	0	00	0	q_0
0	q_0	1	01	1	q_1
1	q_1	0	10	10	q_2
1	q_1	1	11	0	q_0
10	q_2	0	100	1	q_1
10	q_2	1	101	10	q_2

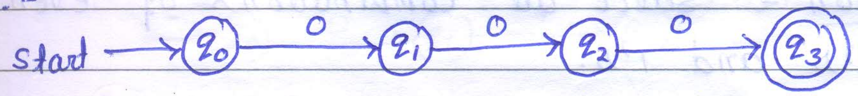


Transition diagram.

DFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ wh
(i) $Q = \{S, q_0, q_1, q_2\}$
(ii) $\Sigma = \{0, 1\}$
(iii) δ is defined in Transition diagram.
(iv) S is the initial state.
(v) $F = \{q_0\}$.

Q13) Design a DFA that accepts strings with 0's and 1's such that the no. of zero's is divisible by 3.

Solution:-

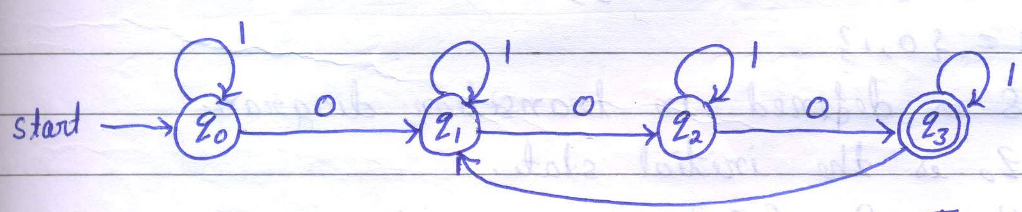


Here no. of zero's 000 (3) is divisible by 3.

(i) if it requires Three 0's to get final state then goto on q_0.
(ii) if it requires Two 0's to get final state then goto on q_1.
(iii) if it requires one 0' to get final state then goto on q_2.
(iv) if No. of 0's in string already divisible by 3, then goto on q_3.

Find Next state:-

string from initial to current state	current state	Input on current state	string from initial to Next state	Requirement of No. of zero's for divisible by 3.	Next state
ε	q_0	1	1	000	q_0
0	q_1	1	01	00	q_1
00	q_2	1	001	0	q_2
000	q_3	0	0000	00	q_1
000	q_3	1	0001	–	q_3



Transition diagram

DFA is represented by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

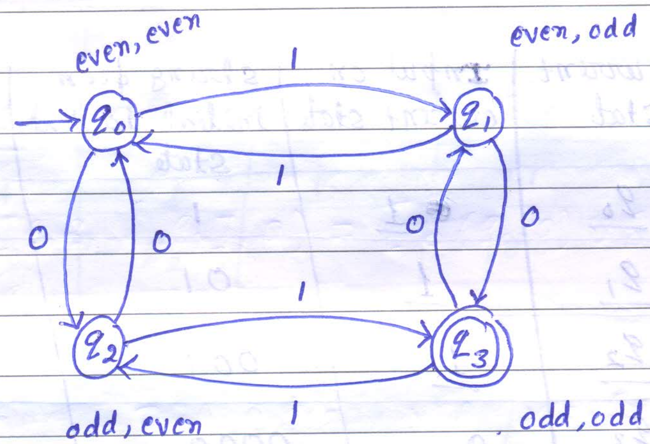
(i) $Q = \{q_0, q_1, q_2, q_3\}$
(ii) $\Sigma = \{0, 1\}$
(iii) δ is defined in Transition diagram.
(iv) q_0 is the initial state.
(v) $F = \{q_3\}$.

Q14) Design a DFA which accepts set of all strings containing odd no. of 0's and odd no. of 1's.

Solution:- Select all combinations of even and odd nu of 0's and 1's.

(i) q_0 represent even no. of 0's and even no. of 1's.
(ii) q_1 represent even no. of 0's and odd no. of 1's.
(iii) q_2 represent odd no. of 0's and even no. of 1's.
(iv) q_3 represent odd no. of 0's and odd no. of 1's.

q_3 is the final state because it represent odd no. of 0's and odd no. of 1's.



DFA represent by 5 tupples $(Q, \Sigma, \delta, q_0, F)$ where

(i) $Q = \{q_0, q_1, q_2, q_3\}$
(ii) $\Sigma = \{0, 1\}$
(iii) δ is defined in transition diagram.
(iv) q_0 is the initial state.
(v) ~~q_3 is~~ $F = \{q_3\}$.

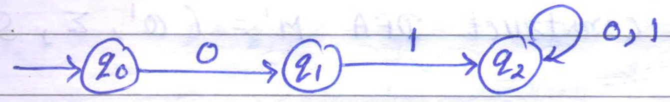
Q15) Design a DFA which accepts set of all binary strings

~~Solution:- Ans~~

Answer)

Q16) Design a DFA that recognizes the set of all string of 0 and 1's starting with prefix 01.

Answer)



Q17)

Convert NFA into DFA :-

Let given NFA is $M = (Q, \Sigma, \delta, q_0, F)$.
Now to construct DFA $M' = (Q', \Sigma, \delta', q_0', F')$.

Procedure :-

Step 1 :- $Q' = 2^Q$, the set of all subsets of Q.

Step 2 :- $q_0' = [q_0]$, the initial state.

Step 3 :- F' is the set of all subsets of Q which are having final state of given NFA.

Step 4 :- construction of δ'.

$$\delta([q_0 q_1 \ldots q_n], a) = [\delta(q_0, a) \cup \delta(q_1, a) \cup \ldots \delta(q_n, a)]$$

Q1) Construct DFA for NFA $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$ where δ is given by :-

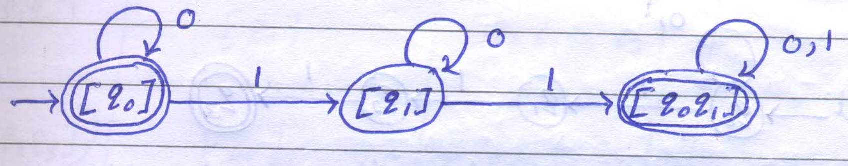
state/input	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$

Solution :- DFA $M' = (Q', \Sigma, \delta', q_0', F')$

(i) $Q' = 2^Q = \{\phi, [q_0], [q_1], [q_0 q_1]\}$
(ii) $[q_0]$ is the initial state. $q_0' = [q_0]$.
(iii) $F' = \{[q_0], [q_0 q_1]\}$, states which having q_0 (final state of N
(iv) δ' is defined by state table :-

state/input	0	1
$\rightarrow [q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0 q_1]$
$[q_0 q_1]$	$[q_0 q_1]$	$[q_0 q_1]$

Transition Table.



Transition diagram.

Q2) Find a DFA equivalent to NFA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ δ is given by.

state/input	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
q_2	–	$\{q_0, q_1\}$

Transition Table.

Solution:- Construction of DFA $M' = (Q', \Sigma, \delta', q_0', F')$.

(i) $Q' = 2^Q = \{\phi, [q_0], [q_1], [q_2], [q_0 q_1], [q_0 q_2], [q_1 q_2], [q_0 q_1 q_2]\}$

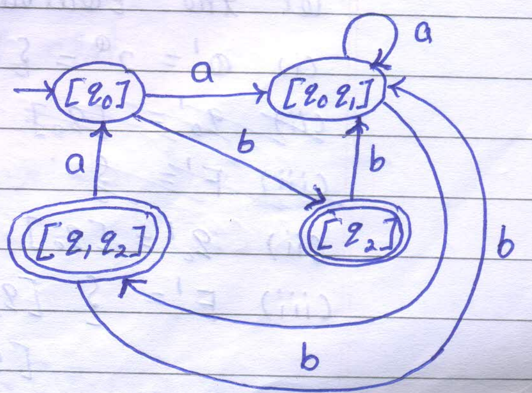
(ii) $[q_0]$ is the initial state. $\{ q_0' = [q_0]$.

(iii) $F' = \{[q_2], [q_0 q_2], [q_1 q_2], [q_0 q_1 q_2]\}$: states which having q_2.

(iv) δ' is defined by state Table:-

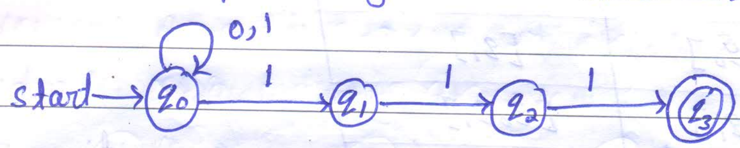
state/input	a	b
$\rightarrow [q_0]$	$[q_0 q_1]$	$[q_2]$
$[q_0 q_1]$	$[q_0 q_1]$	$[q_1 q_2]$
$[q_2]$	–	$[q_0 q_1]$
$[q_1 q_2]$	$[q_0]$	$[q_0 q_1]$

Transition Table

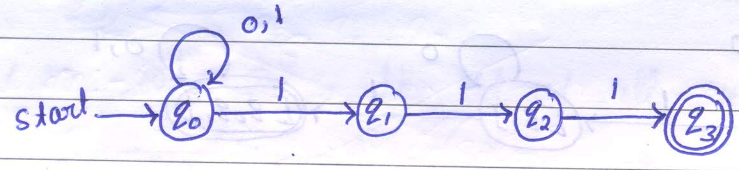


Transition diagram

Q3) convert the following NFA into equivalent DFA.



Solution:-



Transition diagram of NFA

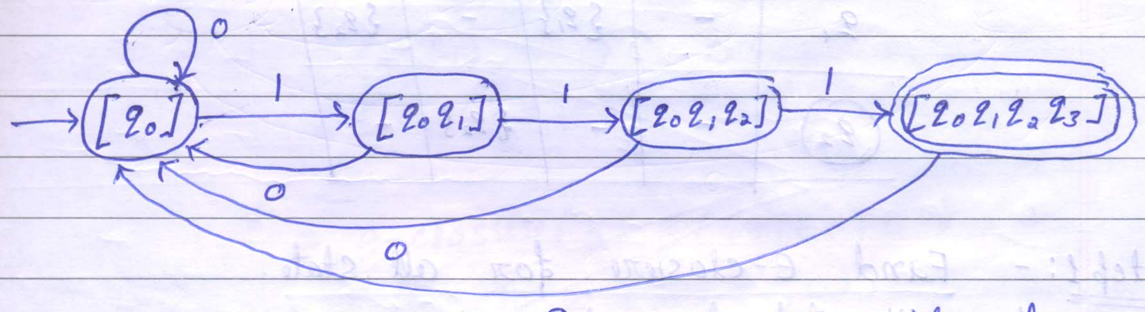
state / input	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{-\}$	$\{q_2\}$
q_2	$-$	$\{q_3\}$
q_3	$-$	$-$

Transition Table.

so given NFA ~~be~~ M is $(Q, \Sigma, \delta, q_0, F)$.
where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, q_0 is initial state
$F = \{q_3\}$.

let the Equivalent DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where
(i) $Q' = 2^Q = \{\phi, [q_0], [q_1], [q_2], [q_3], [q_0q_1], [q_0q_2], [q_0q_3],$
~~(ii) $q_0' = [q_0]$~~ $[q_1q_2], [q_1q_3], [q_2q_3], [q_0q_1q_2], [q_0q_1q_3],$
~~(iii) $F' = \{$~~ $[q_0q_2q_3], [q_1q_2q_3], [q_0q_1q_2q_3]\}$
(ii) $q_0' = [q_0]$
(iii) $F' = \{[q_3], [q_0q_3], [q_1q_3], [q_2q_3], [q_0q_1q_3], [q_0q_2q_3],$
$[q_1q_2q_3], [q_0q_1q_2q_3]\}$
(iv) δ' ~~is~~ defined by Transition/state Table:-

state/input	0	1
$\rightarrow [q_0]$	$[q_0]$	$[q_0 q_1]$
$[q_0 q_1]$	$[q_0]$	$[q_0 q_1 q_2]$
$[q_0 q_1 q_2]$	$[q_0]$	$[q_0 q_1 q_2 q_3]$
$[q_0 q_1 q_2 q_3]$	$[q_0]$	$[q_0 q_1 q_2 q_3]$



Transition diagram

construction of NFA without ϵ-moves from NFA with ϵ-moves

let $M = (Q, \Sigma, \delta, q_0, F)$ is NFA with ϵ-moves then
$M' = (Q, \Sigma, \delta', q_0, F')$ is NFA without ϵ-moves.

Procedure :-

step 1 :- Find ϵ-closure for all state.

step 2: Find Extended transition function.

$$\hat{\delta} : Q \times \Sigma^* \longrightarrow 2^Q$$

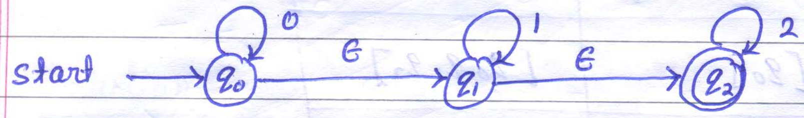
$$\hat{\delta}(q_0, \epsilon) = \epsilon\text{-closure}(q_0)$$

$$\hat{\delta}(q_0, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), a))$$

step 3 :- set of Final states F' consists of all states whose ϵ-closure contains a final state in F.

~~Construction of NFA without ϵ-moves from NFA with ϵ-moves~~

Q1) Convert NFA with ϵ-moves in figure given below to equivalent NFA without ϵ-moves.



Solution:-

state/input	0	1	2	ϵ
$\rightarrow q_0$	$\{q_0\}$	-	-	$\{q_1\}$
q_1	-	$\{q_1\}$	-	$\{q_2\}$
q_2	-	-	$\{q_2\}$	-

Step 1:- Find ϵ-closure for all state.
from the defination of ϵ-closure,

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

Step 2:- Find Extended Transition Function

$\Rightarrow \hat{\delta}(q_0, 0) = \epsilon\text{-closure}(\delta(\underline{\hat{\delta}(q_0, \epsilon)}, 0))$

$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0))$
$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$
$= \epsilon\text{-closure}(\{q_0\} \cup \phi \cup \phi)$
$= \epsilon\text{-closure}(q_0)$

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

similarly :

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

$\hat{\delta}(q_0, 2) = \{q_2\}$

$\Rightarrow \hat{\delta}(q_1, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 0))$

$= \epsilon\text{-closure}(\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon\text{-closure}(\phi \cup \phi)$

$= \epsilon\text{-closure}(\phi)$

$\hat{\delta}(q_1, 0) = \phi$

similarly : $\hat{\delta}(q_1, 1) = \{q_1, q_2\}$

$\hat{\delta}(q_1, 2) = \{q_2\}$

$\Rightarrow \hat{\delta}(q_2, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(q_2, 0))$

$= \epsilon\text{-closure}(\phi)$

$\hat{\delta}(q_2, 0) = \phi$

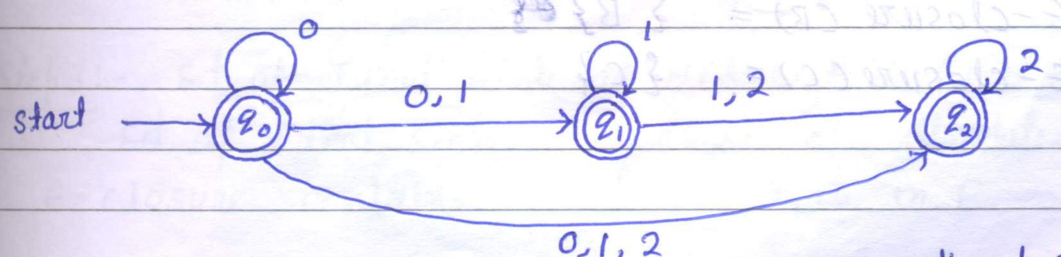
similarly : $\hat{\delta}(q_2, 1) = \phi$

$\hat{\delta}(q_2, 2) = \{q_2\}$

step 3:- set of Final state of NFA with ϵ $(F) = \{q_2\}$.

set of Final state F' consists of all states whose ϵ-closure contains a final state in F.

then set of Final state of NFA without ϵ $(F') = \{q_0, q_1, q_2\}$.

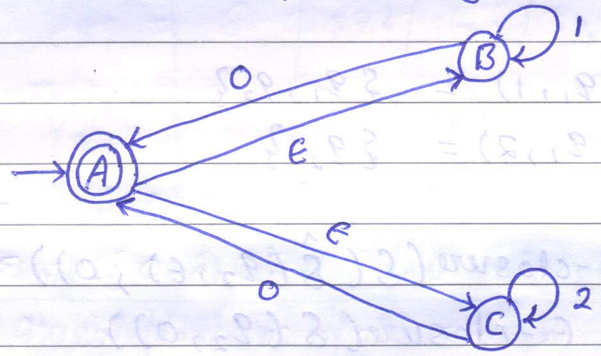


Transition diagram of NFA without ϵ.

NFA represented by 5 tupples $(Q, \Sigma, \delta', q_0, F')$, where

(i) $Q = \{q_0, q_1, q_2\}$
(ii) $\Sigma = \{0, 1, 2\}$
(iii) q_0 is the initial state.
(iv) $F' = \{q_0, q_1, q_2\}$.
(v) δ' is defined in Transition diagram.

Q2) Convert the following into NFA without ϵ-move.



Solution:-

state/input	0	1	2	ϵ
→(A)	–	–	–	{B, C}
B	A	B	–	–
C	A	–	C	–

Step 1:- Find ϵ-closure for all state.

ϵ-closure$(A) = \{A, B, C\}$
ϵ-closure$(B) = \{B\}$
ϵ-closure$(C) = \{C\}$

F'),

ove.

Step 2:- Find Extended Transition Function:-

$\Rightarrow \quad \hat{\delta}(A,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(A,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(\{A,B,C\},0))$
$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0))$
$= \epsilon\text{-closure}(\phi \cup \{A\} \cup \{A\})$
$= \epsilon\text{-closure}(A)$

$\hat{\delta}(A,0) = \{A,B,C\}$

similarly : $\hat{\delta}(A,1) = \{B\}$
$\hat{\delta}(A,2) = \{C\}$

$\Rightarrow \quad \hat{\delta}(B,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(\{B\},0))$
$= \epsilon\text{-closure}(A)$

$\hat{\delta}(B,0) = \{A,B,C\}$

Similarly : $\hat{\delta}(B,1) = \{B\}$
$\hat{\delta}(B,2) = \phi$

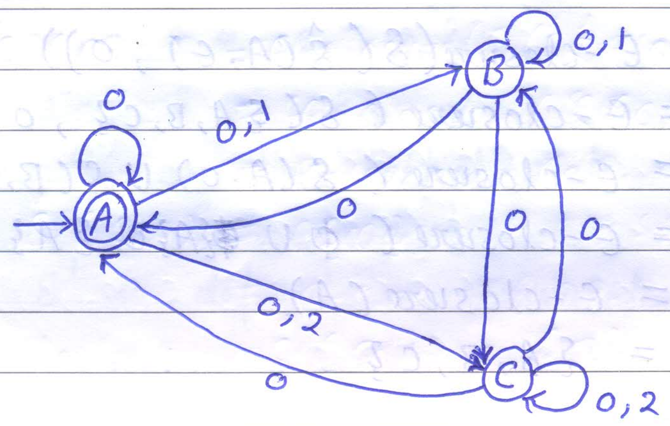
$\Rightarrow \quad \hat{\delta}(C,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(C,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(C,0))$
$= \epsilon\text{-closure}(A)$

$\hat{\delta}(C,0) = \{A,B,C\}$

similarly : $\hat{\delta}(C,1) = \phi$
$\hat{\delta}(C,2) = \{C\}$

Step 3:- Set of Final state of NFA with ϵ $(F) = \{A\}$.
Set of Final state F' consists of all states whose ϵ-closure contains a final state in F.

then set of Final state of NFA without ϵ $(F') = \{A\}$.



NFA without ϵ (Transition diagram)

NFA represented by Five Tupples $(Q, \Sigma, \delta', q_0, F')$ where

(i) $Q = \{A, B, C\}$

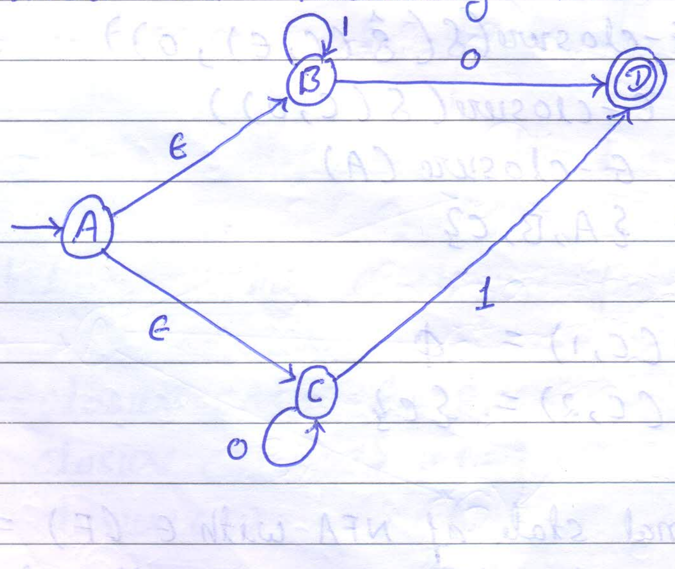
(ii) $\Sigma = \{0, 1, 2\}$

(iii) A is the initial state then $q_0 = \{A\}$.

(iv) $F' = \{A\}$.

(v) δ' is defined in Transition diagram.

Q3) Convert the Following NFA into NFA without ϵ-moves



solution:-

state/input	0	1	ϵ
$\rightarrow A$	-	-	$\{B, C\}$
B	$\{D\}$	$\{B\}$	-
C	$\{C\}$	$\{D\}$	-
D	-	-	-

Transition Table.

step 1:- Find ϵ-closure for all state.

$\epsilon\text{-closure}(A) = \{A, B, C\}$
$\epsilon\text{-closure}(B) = \{B\}$
$\epsilon\text{-closure}(C) = \{C\}$
$\epsilon\text{-closure}(D) = \{D\}$

step 2:- Find Extended Transition Function.

$$\Rightarrow \hat{\delta}(A,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(A,\epsilon), 0))$$
$$= \epsilon\text{-closure}(\delta(\{A,B,C\}, 0))$$
$$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0))$$
$$= \epsilon\text{-closure}(\phi \cup \{D\} \cup \{C\})$$
$$= \epsilon\text{-closure}(\{C, D\})$$
$$= \epsilon\text{-closure}(C) \cup \epsilon\text{-closure}(D)$$
$$= \{C\} \cup \{D\}$$
$$\hat{\delta}(A,0) = \{C, D\}.$$

similary : $\hat{\delta}(A, 1) = \{B, D\}$.

$$\Rightarrow \hat{\delta}(B,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,\epsilon), 0))$$
$$= \epsilon\text{-closure}(\delta(B,0))$$
$$= \epsilon\text{-closure}(D)$$
$$\hat{\delta}(B,0) = \{D\}.$$

Similarly : $\hat{\delta}(B,1) = \{B\}$

$\Rightarrow \hat{\delta}(C,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(C,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(C,0))$
$= \epsilon\text{-closure}(C)$
$\hat{\delta}(C,0) = \{C\}$

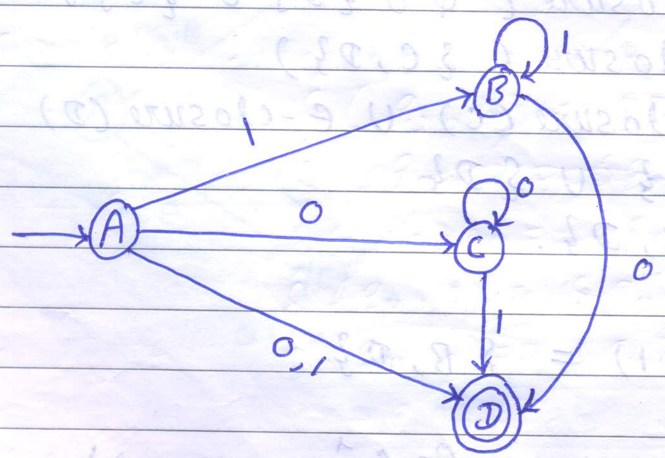
Similarly : $\hat{\delta}(C,1) = \{D\}$.

$\Rightarrow \hat{\delta}(D,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(D,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(D,0))$
$= \epsilon\text{-closure}(\phi)$
$\hat{\delta}(D,0) = \phi$

Similarly : $\hat{\delta}(D,1) = \phi$

Step 3 :- Set of Final state of NFA with ϵ $(F) = \{D\}$.

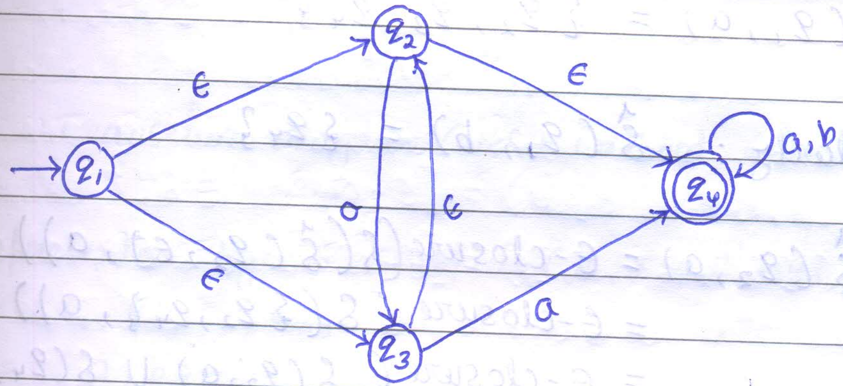
Set of Final state F' consists of all states whose ϵ-closure contains a final state in F.
then set of Final state of NFA without ϵ $(F') = \{D\}$.



Transition diagram (NFA without ϵ).

NFA represent by 5 Tupples $(Q, \Sigma, \delta', q_0, F')$, where
(i) $Q = \{A, B, C, D\}$.
(ii) $\Sigma = \{0, 1\}$
(iii) $q_0 = \{A\}$.
(iv) $F' = \{D\}$
(v) δ' is defined in Transition diagram.

Q4) convert the Following NFA with ϵ-move into NFA without ϵ-move.



Solution:-

state/input	a	b	ϵ
$\rightarrow q_1$	—	—	$\{q_2, q_3\}$
q_2	$\{q_3\}$	—	$\{q_4\}$
q_3	$\{q_4\}$	—	$\{q_2\}$
q_4	$\{q_4\}$	$\{q_4\}$	—

Step 1:- Find ϵ-closure for all state.

$\epsilon\text{-closure}(q_1) = \{q_1, q_2, q_3, q_4\}$
$\epsilon\text{-closure}(q_2) = \{q_2, q_4\}$
$\epsilon\text{-closure}(q_3) = \{q_3, q_2, q_4\} = \{q_2, q_3, q_4\}$
$\epsilon\text{-closure}(q_4) = \{q_4\}$.

Step 2:- Find Extended Transition Function.

$$\Rightarrow \hat{\delta}(q_1, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), a))$$
$$= \epsilon\text{-closure}(\delta(\{q_1, q_2, q_3, q_4\}, a))$$
$$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a))$$
$$= \epsilon\text{-closure}(\phi \cup \{q_3\} \cup \{q_4\} \cup \{q_4\})$$
$$= \epsilon\text{-closure}(\{q_3, q_4\})$$
$$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_4)$$
$$= \{q_2, q_3, q_4\} \cup \{q_4\}$$
$$\hat{\delta}(q_1, a) = \{q_2, q_3, q_4\}$$

Similarly: $\hat{\delta}(q_1, b) = \{q_4\}$.

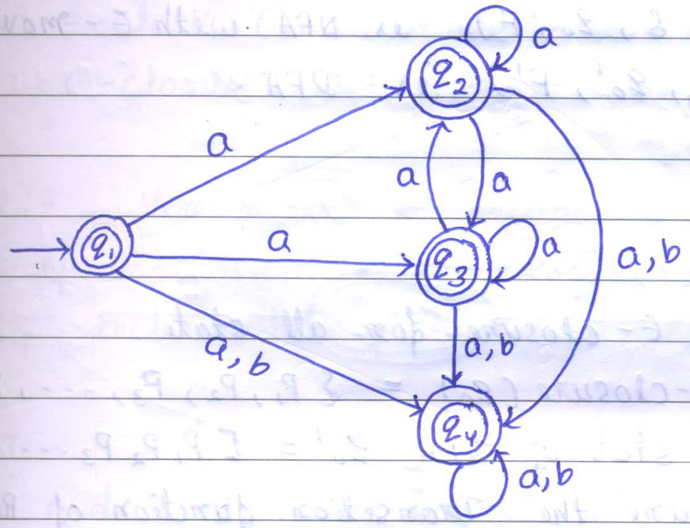
$$\Rightarrow \hat{\delta}(q_2, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2, \epsilon), a))$$
$$= \epsilon\text{-closure}(\delta(\{q_2, q_4\}, a))$$
$$= \epsilon\text{-closure}(\delta(q_2, a) \cup \delta(q_4, a))$$
$$= \epsilon\text{-closure}(\{q_3\} \cup \{q_4\})$$
$$= \epsilon\text{-closure}(\{q_3, q_4\})$$
$$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_4)$$
$$= \{q_2, q_3, q_4\} \cup \{q_4\}$$
$$\hat{\delta}(q_2, a) = \{q_2, q_3, q_4\}.$$

Similarly :

$$\hat{\delta}(q_2, b) = \{q_4\}$$
$$\hat{\delta}(q_3, a) = \{q_2, q_3, q_4\}$$
$$\hat{\delta}(q_3, b) = \{q_4\}$$
$$\hat{\delta}(q_4, a) = \{q_4\}$$
$$\hat{\delta}(q_4, b) = \{q_4\}$$

Step 3:- Set of Final state of NFA with ϵ (F) = $\{q_4\}$.
Set of Final state F' consists of all states whose ~~Production~~

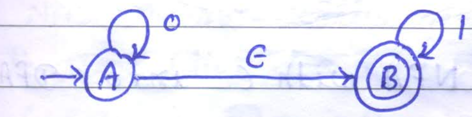
ε-closure contains a final state in F.
then set of Final state of NFA without ε $(F') = \{q_1, q_2, q_3, q_4\}$.

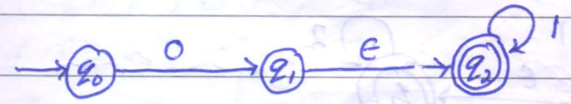


Transition diagram (NFA without ε).

NFA represented by 5 Tupples $(Q, \Sigma, \delta', q_0, F')$, where
(i) $Q = \{q_1, q_2, q_3, q_4\}$
(ii) $\Sigma = \{a, b\}$
(iii) $q_0 = \{q_1\}$
(iv) $F' = \{q_1, q_2, q_3, q_4\}$
(v) δ' is defined in Transition diagram.

Q5) convert the Following NFA with ε-move into NFA without ε-move.

ⓐ


ⓑ


ⓒ