Date - 08.09.23

* Theory Of Computation :-

It is a branch of computer science that deals with how efficiently a problem can be solved on a model of computation using an algorithm, i.e it is concerned with how problems can be efficiently solved by using algorithms.

The main purpose of theory of computation to develop a formal mathematical model of computation that reflects the real world computers.

⊙ Branches of study under theory of computation:-
→ Complexity theory (easy or hard)
→ Computability theory (Soluable or not solvable)
→ Automata theory

⊙ Automata Theory :-
derived from automata → related to automation

(i) Automata theory is the study of abstract computational devices.

(ii) It mainly deals with the logic of computation with respect to simple machines referred to as automata.

(iii) Automata enables scientists to understand how machines compute the functions & solve problems.

- Sets — (i) order doesn't matter (ii) no repeatition
- Sequence — (i) order matters (ii) repeatition possible
- Tuples — no. of elements in sequence
- Relations & Functions

Date - 09.09.23

* Alphabets is a finite set of symbols and it is denoted by the capital greek letter Σ (sigma)

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

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

$\Sigma = \{a, b, c, d, \ldots, z\}$

$\Sigma = \{\#, 2, \Delta\}$

⊙ Symbols - Symbols are the entities or individual objects which can be any character, special character, or any picture.

Denoted as typewriter font -

$\Gamma = \{0, 1, a, b\}$

⊙ Strings - A string over an alphabet is a finite collection of symbols from the alphabet

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

A string over Σ can be 001110

w → used to denote strings

$w = 001110$

$|w|$ → string length

w^R → string in reverse order

$w^R = 011100$

$w = w_1 w_2 w_3 w_4$

$w^R = w_4 w_3 w_2 w_1$

⊙ Language - A language L over alphabet Σ is any set of strings made up of symbols from alphabet (Σ)

A language which is formed over Σ is finite or infinite

Example :-

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

L_1 = set of all strings with length 2

= $\{aa, ab, ba, bb\}$ → Finite

L_2 = set of all strings starts from a

= $\{a, aa, aaa, ab, aba, \ldots\}$ → Infinite

ε → empty string

ϕ → empty language

The set of all strings over Σ is denoted by Σ^*

Set theory operations of L :-

1. Union
2. Intersection
3. Concatenation
4. Closure / Kleenstar - The closure / kleenestar of a language L is the language L^* obtained by uniting the L^k for multiple times where k is the language obtained by concatenating to itself for k times

$$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \ldots$$

$$L^k = \underbrace{LLL \ldots L}_{\leftarrow k \text{ times} \rightarrow}$$

→ Kleene star / closure of a language.

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \ldots$$

Positive closure of a language -

$$L^+ = L^1 \cup L^2 \cup L^3 \cup L^4 \ldots$$

Date - 14.09.23

⊙ Closure/Kleene Star :- It is the set of all possible string of all possible lengths including the empty string ε.

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

Σ^0 ↓ set of all strings of length zero (ε)

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

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\ldots\}$

⊙ Positive Closure / Kleene Star -

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

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

$\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, \ldots\ldots\}$

* Complement of language (L) :-

$$\overline{L} = L^* - L$$

* Boolean Logic :-

1. AND ∧ (Conjuction)
2. OR ∨ (Disjuction)
3. NOT ¬
4. XOR ⊕
5. Equality ↔
6. Implication →

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$	$x \leftrightarrow y$	$x \rightarrow y$ $(\neg x \vee y)$
0	0	0	0	0	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	0
1	1	1	1	0	1	1

(*)

* Types Of Proof :-

1) Proof by Induction
2) Proof by Contradiction
3) Proof by Construction

* Proof by Construction :-

① For each even no. $n \geq 2$ there exists a 3-regular graph with n nodes.

G — degree of each node is same.

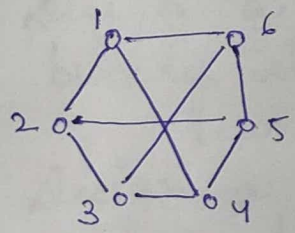
Proof :-

① Our goal is to create or find a formula for creating a 3-regular graph for any $n \geq 4$ where n is even.

Example - Let $n = 6$

$G = (V, E)$ $\quad V = 1, 2, 3, 4, 5, 6$ (name of nodes)

$$E = \{\{i, i+1\} : 1 \leq i \leq n-1\} \cup \{\{n, 1\}\}$$
$$\cup \{\{i, i+\tfrac{n}{2}\} : 1 \leq i \leq n/2\}$$



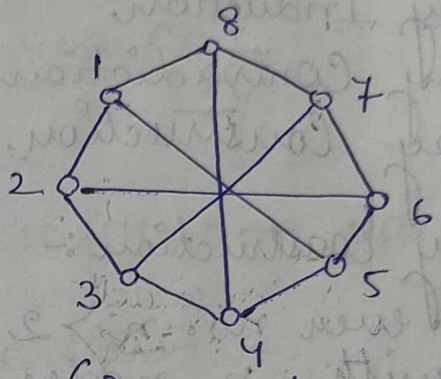
⇒ adjacent vertices ∪ first & last vertex ∪ opposite vertices

Date - 15.09.23

(*) $n = 8$

$V = \{1, 2, 3, 4, 5, 6, 7, 8\}$

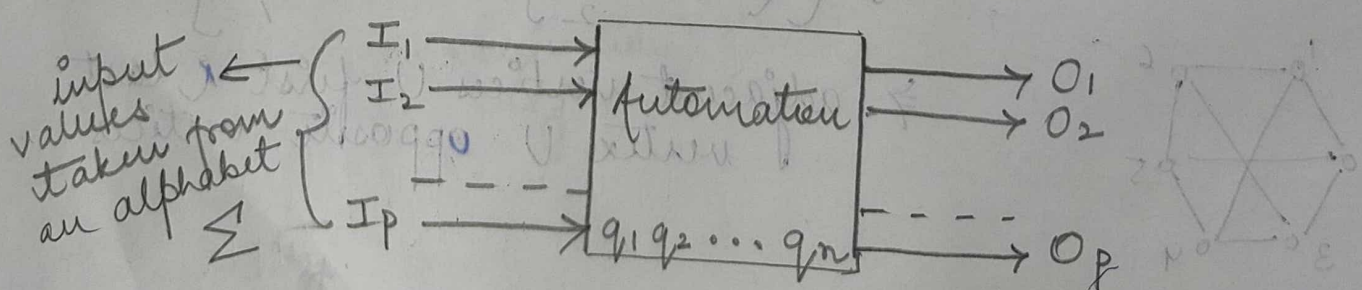
$E = \{(1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,1), (1,5), (2,6), (3,7), (4,8)\}$



(3-regular graph)

* Introduction to Finite Automata (FA)
Finite Automation / Finite State Machine

An automaton is an abstract computing device or machine. An automaton is defined as mathematical model of a system with discrete inputs and outputs. The system can be of a finite number of internal states.

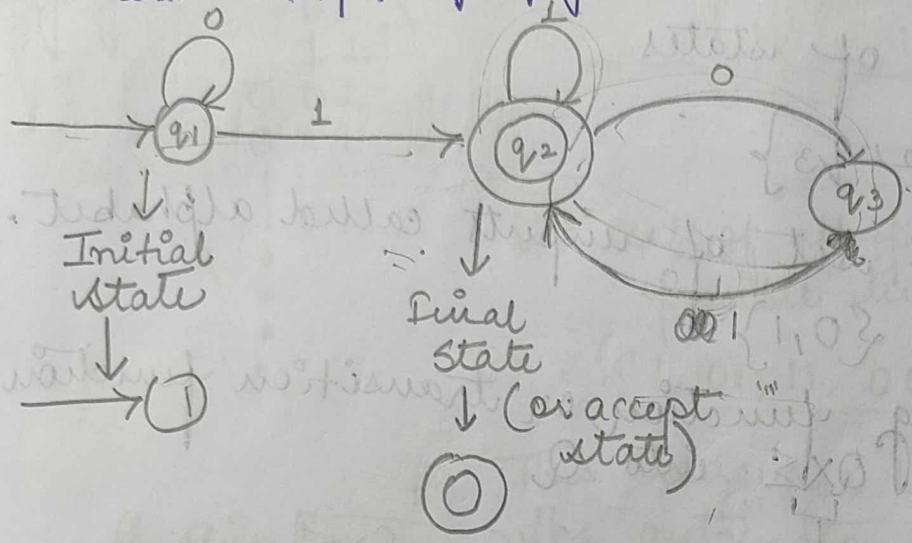


(A model of discrete automaton)

Definition of FA – Finite Automaton consists of a finite set of states and set of transitions from state to state that occur on input symbols chosen from an alphabet Σ.

The following figure depicts a finite automaton

M_1 :-



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

$w = 1101$

↓

reaches at final state so, its accepted

Pattern 1) ending with 1

Pattern 2) even no. of 0s after last 1

Pattern

$w_1 = 0101$

$w_2 = 01$

$w_3 = 1$

$w_4 = 1100$

$w_5 = 0100$

$w_6 = 011$

$w_7 = 01000$

$w_8 = 110000$

$w_9 = 01111$

$w_{10} = 11111$

$M_1 = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_2\})$

i/p / o/p	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Q. Design a finite automaton, which accepts all the binary strings ending at 1.

Q. Design a finite automaton that accepts all binary string having even no.s of 0's after the last 1

$\{0, 1\}$ → 1, 01, 011, 0111, 11

A finite automaton M is defined as a 5 tuple.

$$M = (Q, \Sigma, \delta, q_0, f) \text{ where}$$

1. Q is a finite set of states.
$$Q = \{q_1, q_2, q_3\}$$

2. Σ is the finite set of inputs called alphabet.
$$\Sigma = \{0, 1\}$$

3. δ is a mapping function or transition function which maps $\delta : Q \times \Sigma$ into Q.

$Q \times \Sigma$ — cross product / cartesian product

$Q \times \Sigma$ into Q

$Q \times \{0,1\} \rightarrow Q$

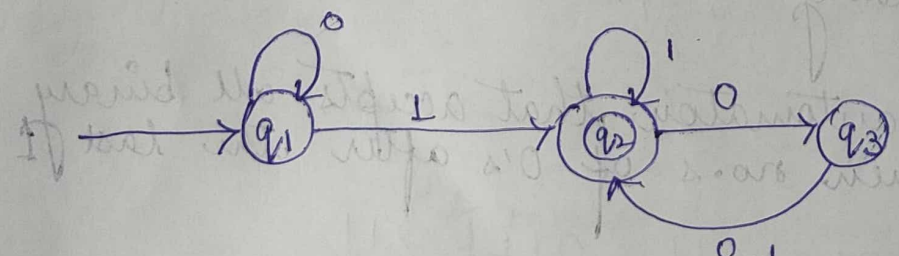
4. $q_0 \in Q$ is the initial state.
$$q_0 = q_1$$

5. $f \subseteq Q$ is the set of final states.
$$f = \{q_2\}$$

Date – 16.09.23

$\Sigma = \{a, b\}$?? $\{0, 1\}$

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



Example :-

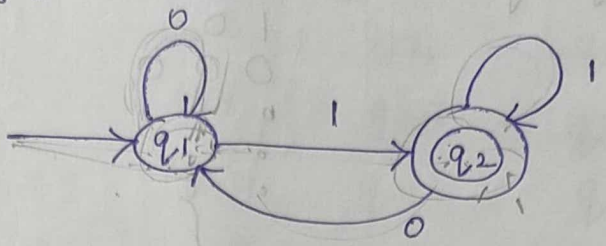
1. Design a FA that accepts all strings that ends at 1 only.

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

Solⁿ - Input strings :-

Language ←

1
01
11
001
011
111
101
.
.
.



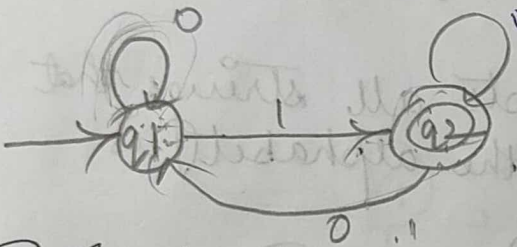
L = set of all possible strings of all possible lengths ending at 1.

$= \{1, 01, 11, 001, \ldots\ldots\ldots\}$

* This language L is recognized by the automaton.

↓

All strings of L are accepted by the automaton.



⊙ Regular Language - A language is called as a regular language if some finite automata recognizes it.

eg - automaton M_1

$A = \{ w/w$, ends at $1 \}$

$L(M_1) = A$

For the above example :-

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

2. Design an automaton that accepts an empty string ε & any string that ends at 0 only.

Soln- Language = { 0, 00, 10, 000, ... }

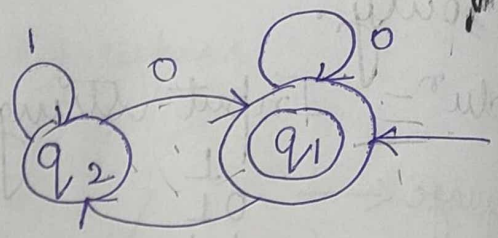


fig - state diagram/ state transition diagram

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

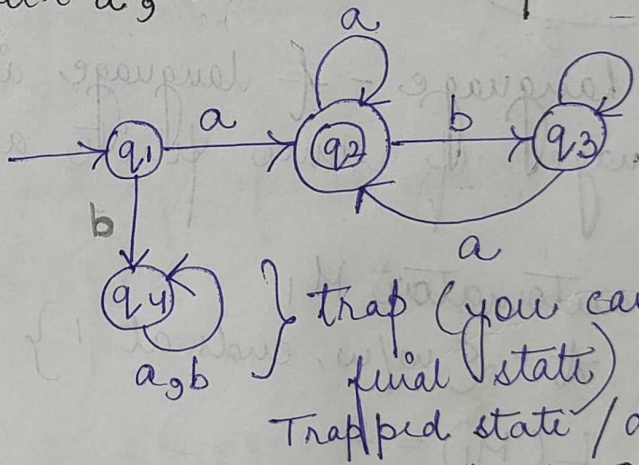
δ mapping -

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

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

3 (a) Design an automaton that accepts all strings that start and end with a, where the alphabet is $\Sigma = \{a, b\}$

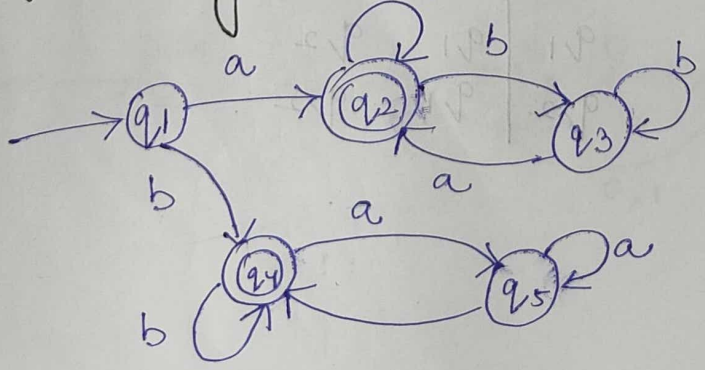
Soln - L = { a, aa, aba, aaa, ... }



} trap (you can't go to final state)

Trapped state / dead state

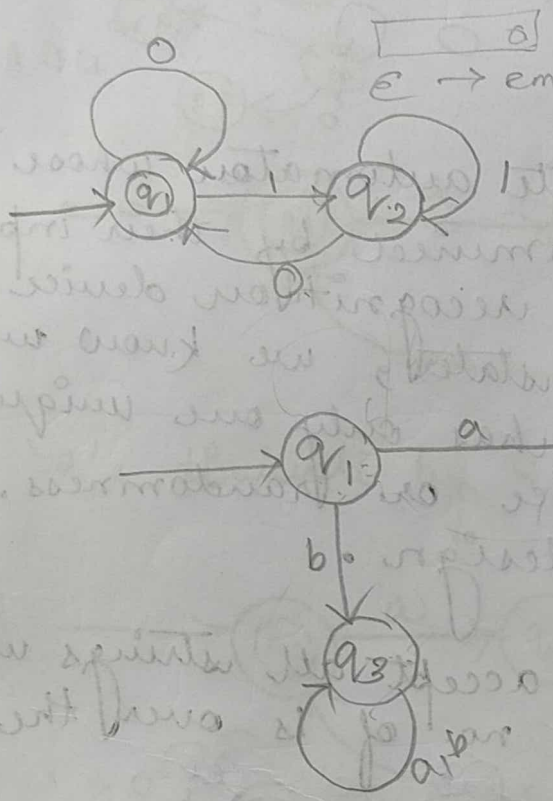
(b) All string start & end at a OR start & end at b.



$M = (\{q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \delta, q_1, \{q_2, q_4\})$

Mapping technique : (δ) :

	a	b
q_1	q_2	q_4
q_2	q_2	q_3
q_3	q_2	q_3
q_4	q_5	q_4
q_5	q_5	q_4



ex :-

$w_1 = abbaba$

$\Rightarrow \delta(q_1, abbaba)$

$\Rightarrow \delta(q_2, bbaba)$

$\Rightarrow \delta(q_3, baba)$

$\Rightarrow \delta(q_3, aba)$

$\Rightarrow \delta(q_2, ba)$

$\Rightarrow \delta(q_3, a)$

$\Rightarrow \delta(q_2, \varepsilon)$

$= q_2$

ex :-

$w_2 = abbab$

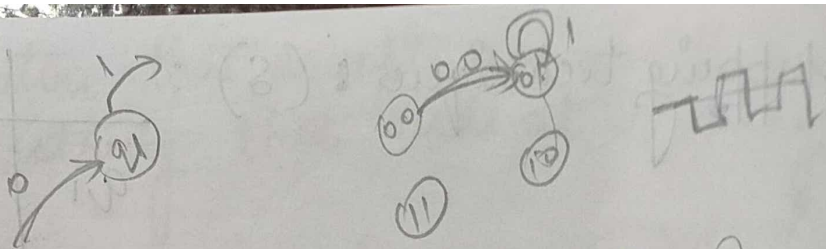
↓

Rejected or trapped.

Transition Mapping

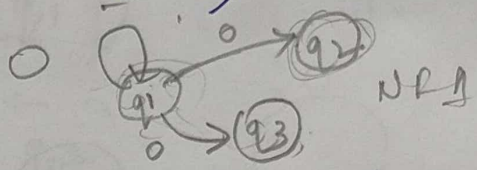
Date - 21.09.23

Q Design



* Types of FA :-
  - (i) FA with output
    - (a) Mealy machine
    - (b) Moore machine
  - (ii) FA without output
    - (a) Deterministic FA (DFA) (all previous example)
    - (b) Non-deterministic FA (NFA)
    - (c) ε NFA

Not in syllabus



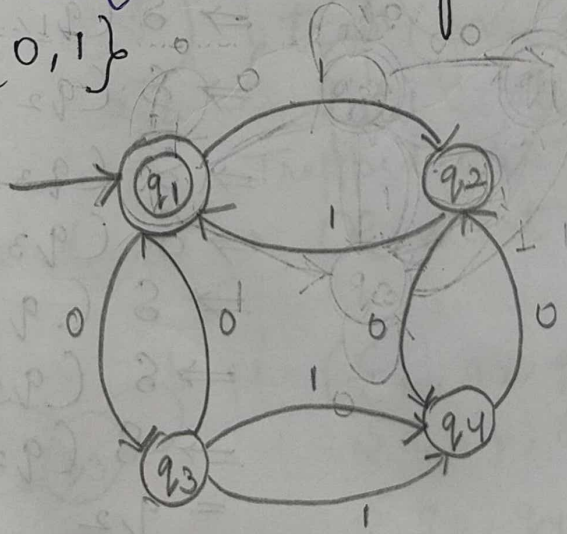
* DFA :-

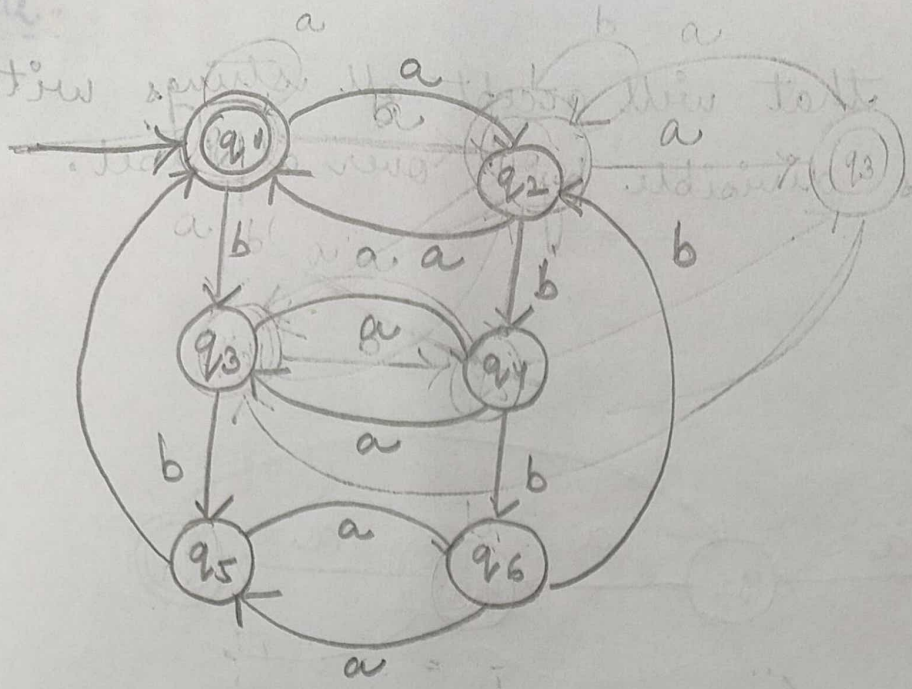
DFA is nothing but a finite automata whose operation is completely determined by their input. It is simply a language recognition device. In DFA, given a current state, we know what the next state will be. It has only one unique next state. It has no choice or randomness. It is simple and easy to design.

Q. Construct a DFA that will accept all strings with even no. of 0s & even no. of 1s over the alphabet $\Sigma = \{0, 1\}$

Soln -

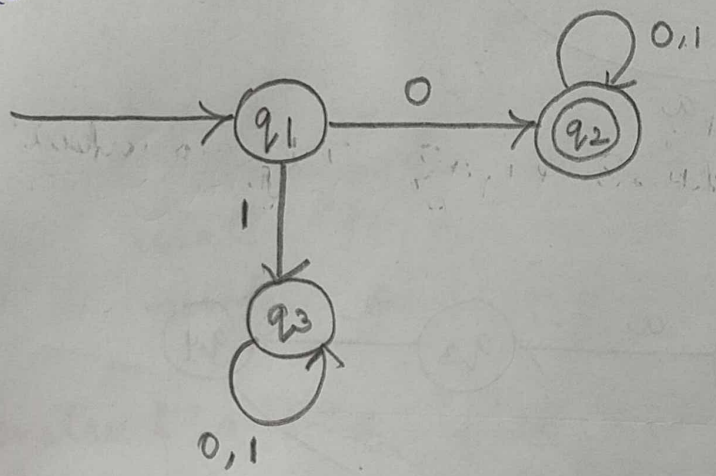
00
0110
0000
010000
0110
1001
1100
0011
0101
1010
11
1111





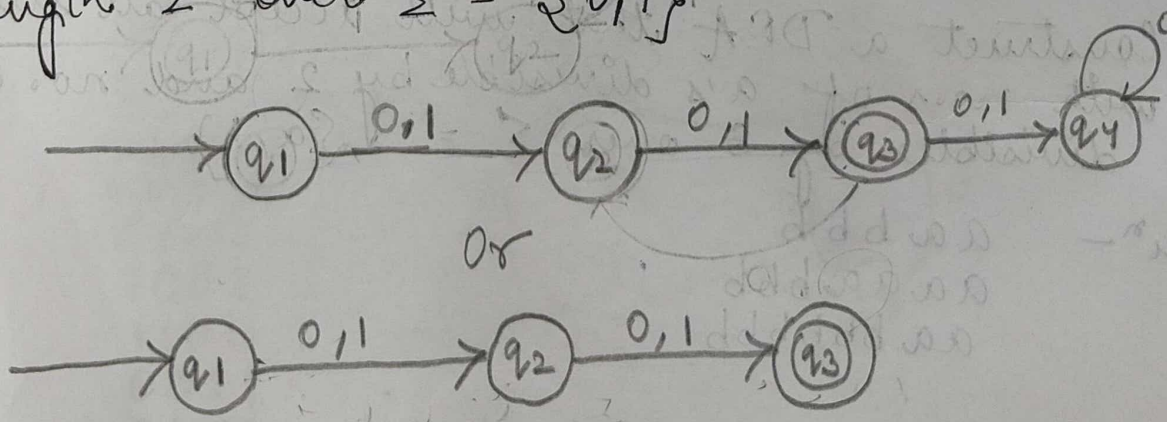
Q. Construct a DFA that accept all strings that starts with a 0 over $\Sigma = \{0,1\}$

Solⁿ-



Q Construct a DFA that will accept all strings of length 2 over $\Sigma = \{0,1\}$

Solⁿ-

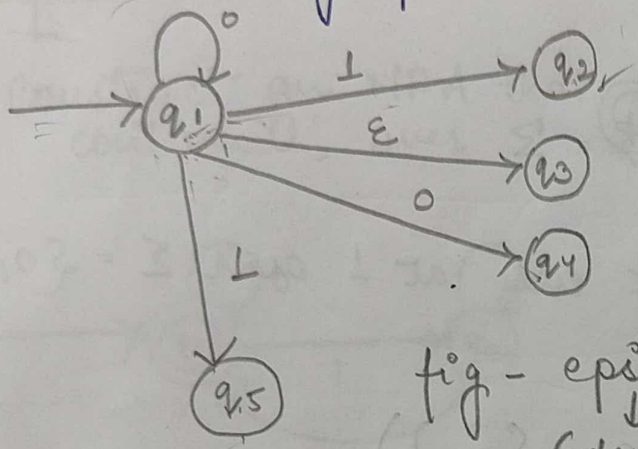


Date - 23.09.23

* Non-Deterministic FA (NFA) :-

→ In NFA, for some given state an input symbol, the transition function δ may lead to a set of states that is a subset of Q.

→ In a NFA, several choices may exist for the next state at any point.



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

1110

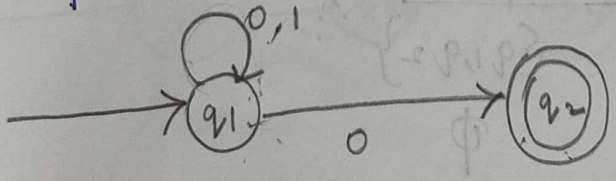
fig - epsilon NFA
(denotes empty string/no input)

→ An NFA is defined as a 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where
Q is the finite set of states;
Σ is the finite set of input symbols.
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the final state
δ is the mapping function or transition function that maps $\delta : Q \times \Sigma \to 2^Q \approx P(Q)$, i.e all possible states of Q.

$\{q_1, q_2\}\{0, 1\}$
$(q_1, 0) \to q_2$

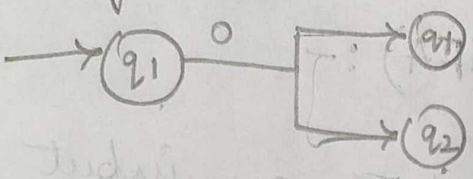
$Q \times \Sigma \to Q$
$Q \times \Sigma \to 2^Q$

Example :-



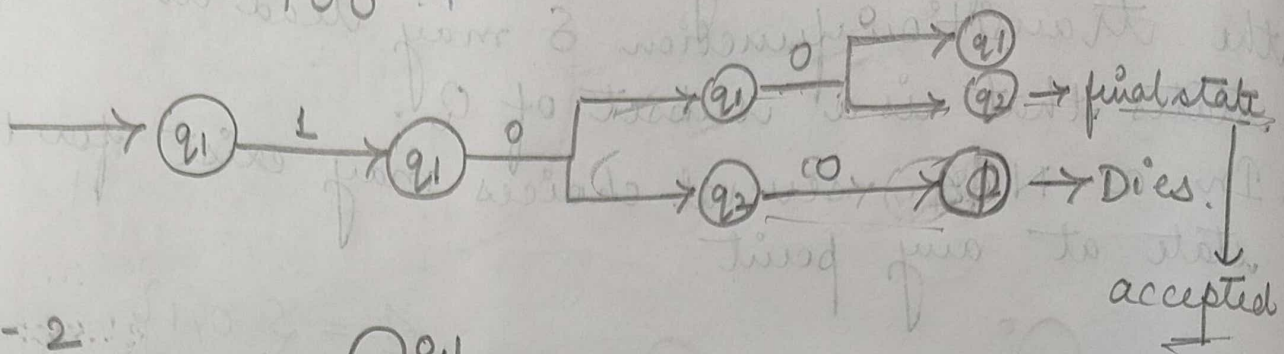
N_1 will take strings ending at 0 over $\Sigma = \{0, 1\}$
(NFA N_1)

String i/p - 0

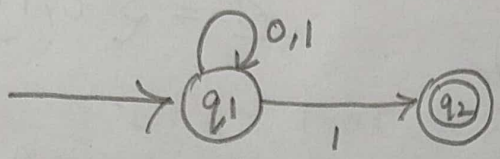


⇒ string accepted

Example 1- w = 100



Example - 2



(NFA N_2)

N_2 will accept all strings that ends at 1 over $\Sigma = \{0, 1\}$

$N_1 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$

δ mapping :-

	0	1
q_1	$\{q_1, q_2\}$	q_1
q_2	ϕ	ϕ

$N_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$

δ mapping :-

	0	1
q_1	q_1	$\{q_1, q_2\}$
q_2	ϕ	ϕ

$Q = \{q_1, q_2\}$

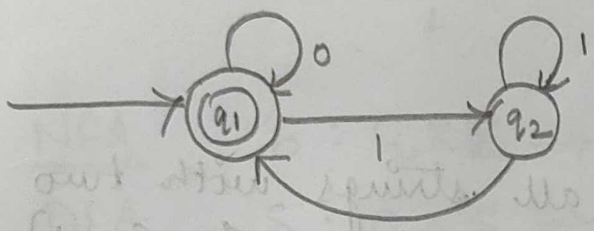
$q_1 \xrightarrow{1} q_1, q_2, \{q_1, q_2\}, \phi$ (4 possibilities)

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

$q_1 \xrightarrow{0} q_1, q_2, q_3, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}$
ϕ (8 possibilities)

No. of possibilities for a i/p symbol to move $= 2^{Q}$
(no. of elements in Q)

Q Construct an NFA that will accept all strings that contain '0', over $\Sigma = \{0, 1\}$



NFA is also a DFA.

Date - 28/09/23

$N = (Q, \Sigma, \delta, q_0, f)$

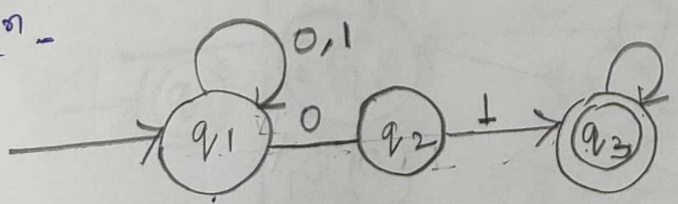
Example -

$Q = \{q_1, q_2\}$

① Construct an NFA that accepts all strings with '01' over $\Sigma = \{0, 1\}$.

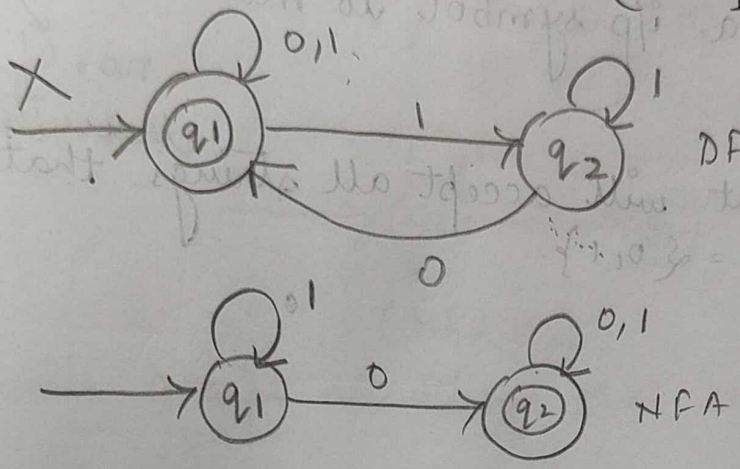
01
001
101
0100

Soln -



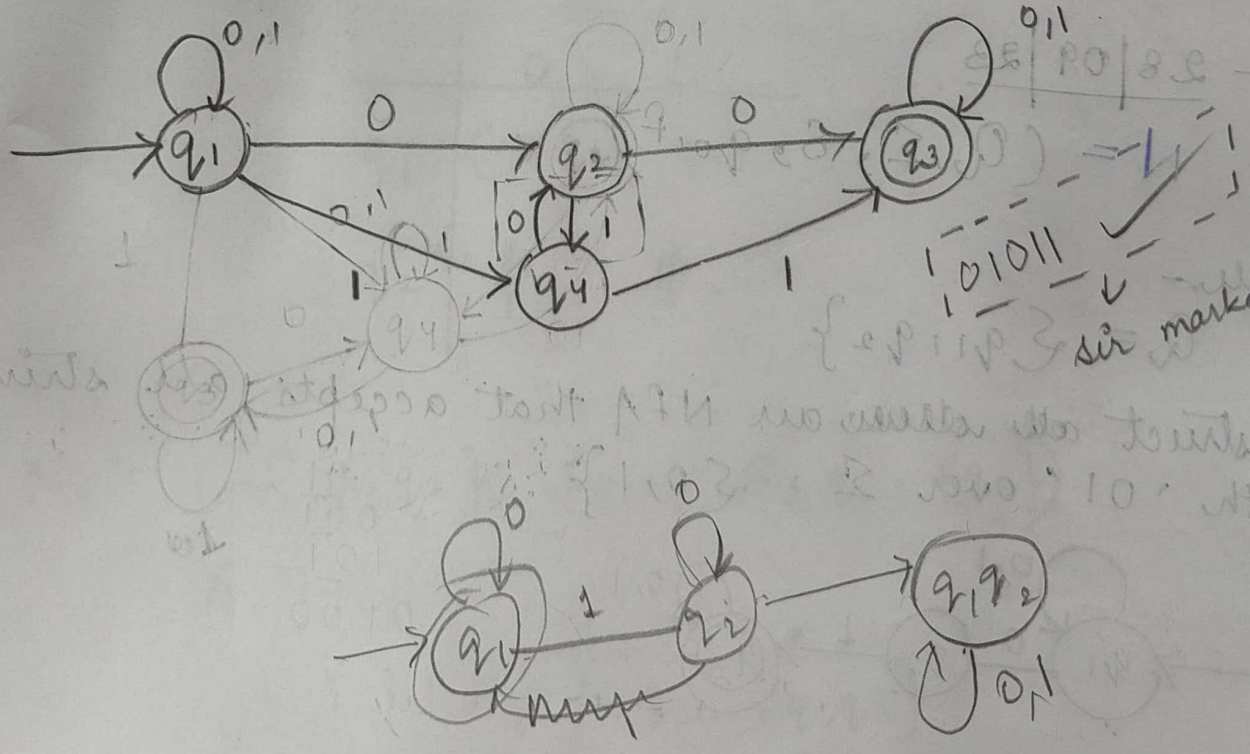
	0	1
q_1	$\{q_1, q_2\}$	q_1
q_2	ϕ	q_3
q_3	q_3	q_3

(2) Construct an NFA that accepts all strings with a '0' over $\Sigma = \{0, 1\}$



(3) Construct the NFA will accept all strings with two consecutive 1's or two consecutive 0's over $\Sigma = \{0, 1\}$

00, 11, 0010, 1100, 0011, 00111, 1101.

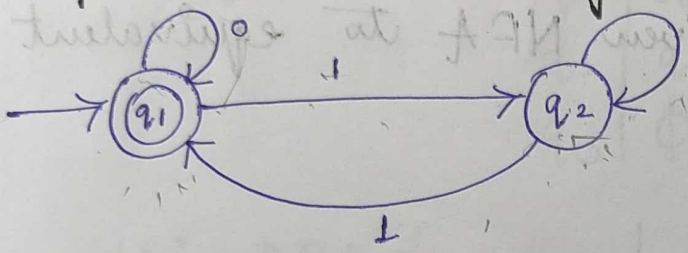


Date - 30.09.23

* Equivalence of NFAs and DFAs.
(i.e Conversion from NFA to DFA)

→ Nondeterminism is a generalization of Determinism, so every deterministic finite automaton (DFA) is automatically a non-deterministic finite automata (NFA) i.e every DFA is an NFA but not vice-versa.

Example :- Convert the given (NFA) to the equivalent (DFA).



NFA $\quad \delta : Q \times \Sigma \to 2^Q$

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

1 ⊙ Transition mapping table of the NFA. δ NFA

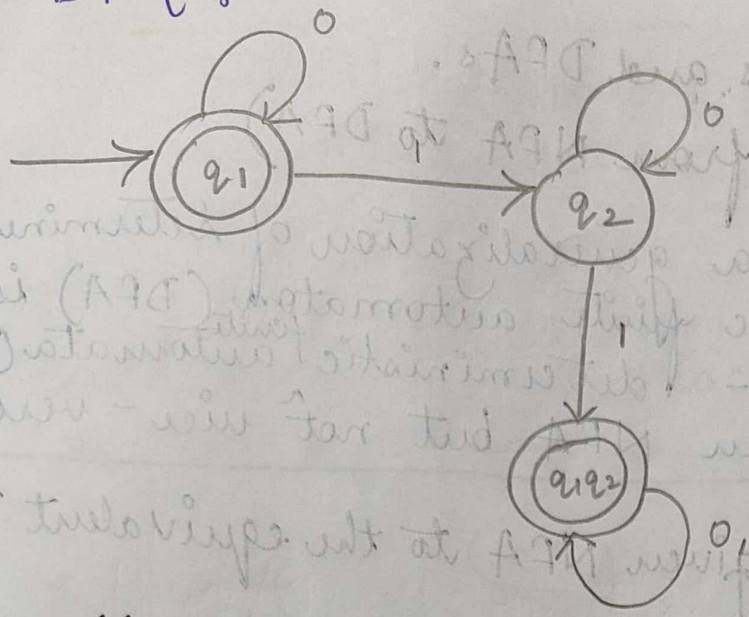
	0	1
→ q1	q1	q2
q2	q2	{q1, q2} ⇒ q1q2 (combined state)

2 ⊙ Transition mapping table of the DFA

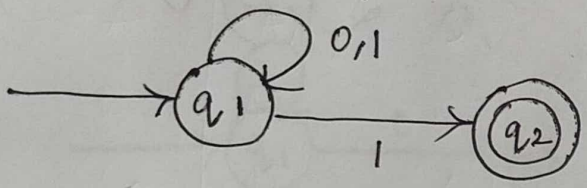
final state, whenever q1 is involved.

	0	1
→ q1	q1	q2
q2	q2	q1q2 → combined new state
q1q2 (union of values from NFA)	q1q2	q1q2

DFA :-



Example - Convert the given NFA to equivalent DFA



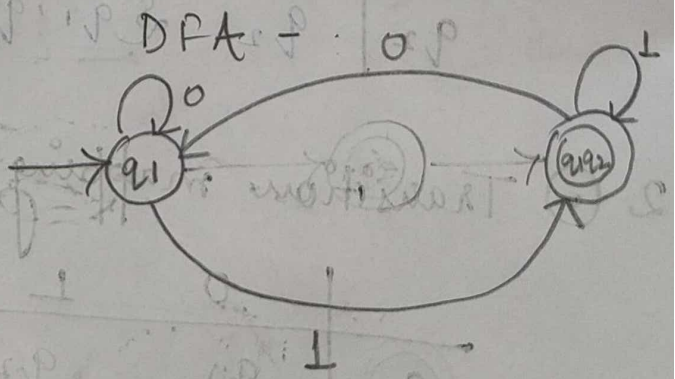
Mapping of NFA

	0	1
q_1	q_1	$\{q_1, q_2\}$
q_2	ϕ	ϕ

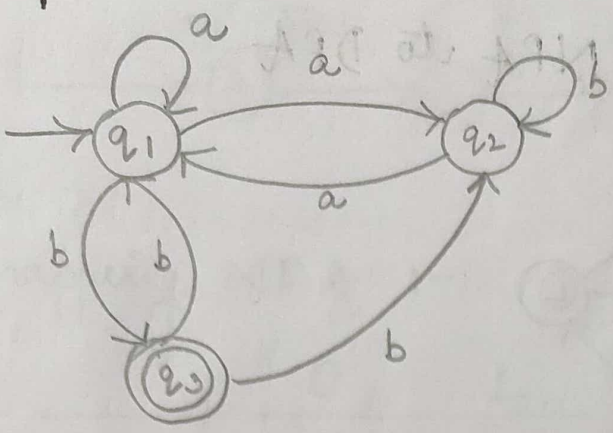
Mapping of DFA

	0	1
q_1	q_1	q_1q_2
q_2	ϕ	ϕ
q_1q_2	q_1	q_1q_2

DFA :-



Example - Convert this NFA to DFA:-



δ mapping NFA :-

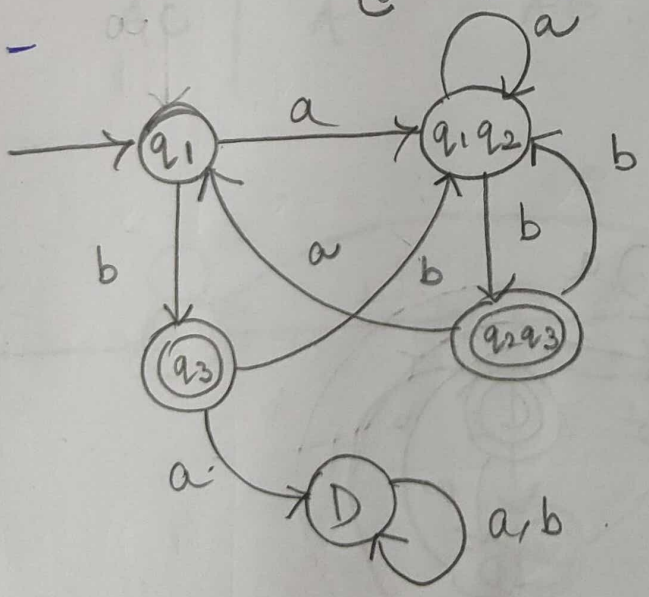
	a	b
q_1	$\{q_1, q_2\}$	q_3
q_2	q_1	q_2
q_3	ϕ	$\{q_1, q_2\}$

δ mapping DFA :-

		a	b
①	q_1	q_1q_2	q_3
	q_2	q_1	q_2
④	q_3	D	q_1q_2
②	q_1q_2	q_1q_2	q_2q_3
③	q_2q_3	q_1	q_1q_2
⑤	D	D	D

→ consider states coming in table sequentially

DFA -

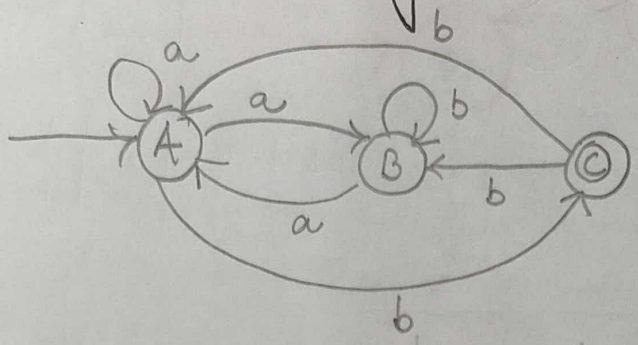


⇓

	a	b
q_1	q_1q_2	q_3
q_1q_2	q_1q_2	q_2q_3
q_2q_3	q_1	q_1q_2
q_3	D	q_1q_2
D	D	D

Date - 05.10.23

Example - Convert the given NFA to DFA.



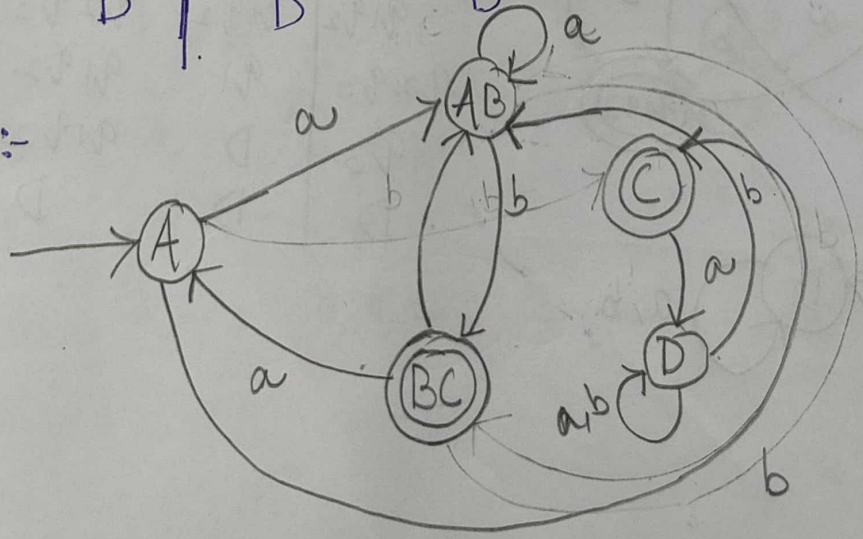
δ mapping / Transition table of NFA is

	a	b
→ A	{A, B}	C
B	A	B
C	ϕ	{A, B}

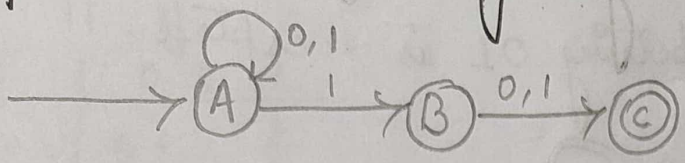
δ mapping / Transition table of DFA is

	a	b
→ A	AB	C
AB	AB	BC
BC	A	AB
C	D	AB
D	D	D

DFA :-



Example: convert the given NFA to DFA. strings where 2nd last symbol is 1.



Soln -

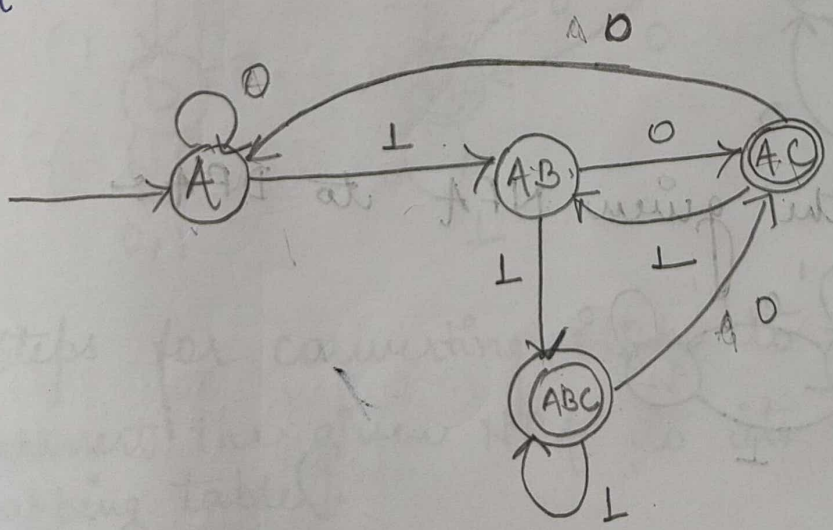
δ mapping NFA :-

	0	1
A	A	{A,B}
B	C	C
C	ϕ	ϕ

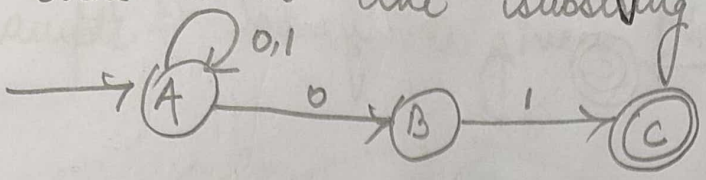
δ mapping DFA :-

	0	1
A	A	AB
AB	AC	ABC
ABC	AC	ABC
AC	A	AB

DFA -



* Example - Convert the given NFA to DFA ends with the substring 01 is accepted.



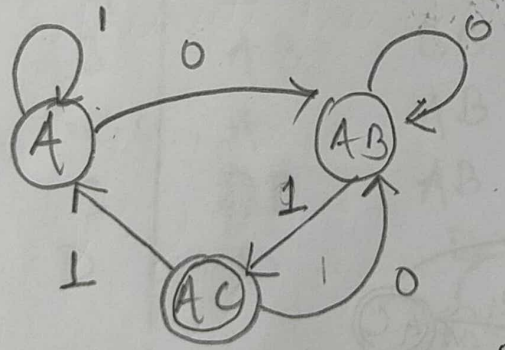
δ mapping NFA

	0	1
A	{A, B}	A
B	ϕ	C
C	ϕ	ϕ

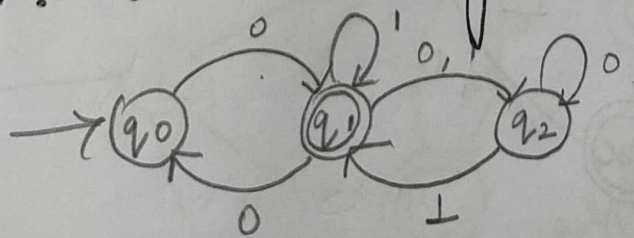
δ mapping DFA

	0	1
A	AB	A
AB	AB	AC
AC	AB	A

DFA :-



Example :- Convert the given NFA to DFA :-



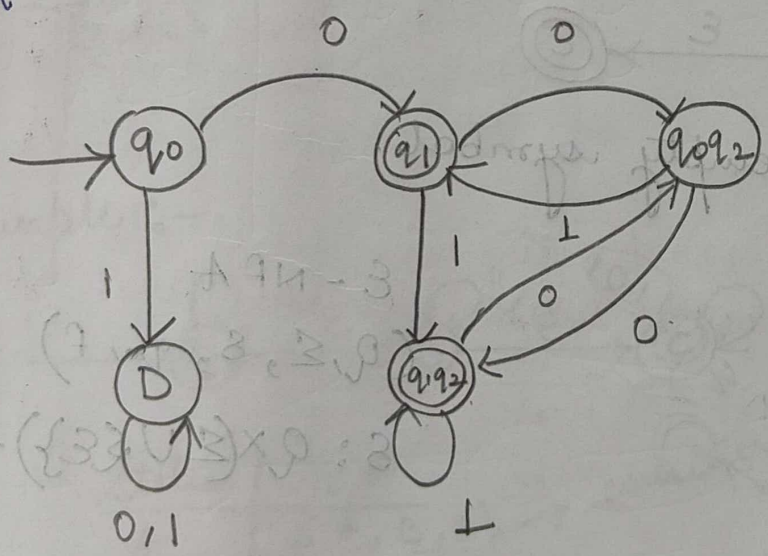
δ mapping NFA –

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

δ mapping DFA –

	0	1
q_0	q_1	D
q_1	$q_0 q_2$	$q_1 q_2$
$q_0 q_2$	$q_1 q_2$	q_1
$q_1 q_2$	$q_0 q_2$	$q_1 q_2$
D	D	D

DFA :-

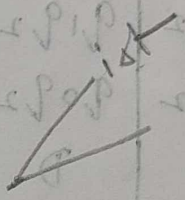


* Steps for converting NFA to DFA :-

① Convert the given NFA to its equivalent transition mapping table.

② Create the DFA's initial state.

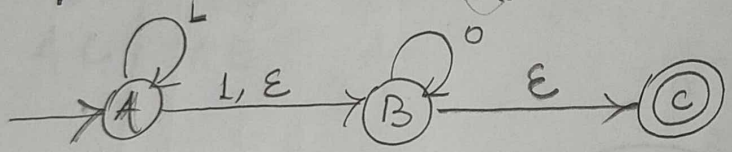
③ Create the DFA's transition mapping table.
④ Create the DFA's final states.
⑤ Simplify the DFA, i.e
   (i) remove unreachable states.
   (ii) merge equivalent states (states that have the same transition rules for all input symbols can be merged into single state).
   (iii) Remove dead states.
⑥ Repeat steps 3 to 5 until no further simplification is possible



* Date - 06/10/23

* Epsilon NFA (ε - NFA) :-

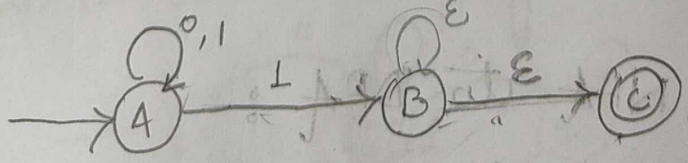
$\Sigma = \{0, 1\}$



→ ε represents the empty symbol.

NFA
$(Q, \Sigma, \delta, q_0, F)$
$\delta : Q \times \Sigma \to 2^Q$

ε - NFA
$(Q, \Sigma, \delta, q_0, F)$
$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$



States	ε
A	A
B	{B, C}
C	C

① Every state goes to itself with the ε input.

* Conversion from ε-NFA to NFA :-

→ The procedure for converting any ε-NFA to its equivalent NFA is that for each state that we have, check where does this state go on ε* (epsilon closure). Then the set of states that we get there have to be checked on which state does they go on getting a particular input and the set of states that we get here have to be again checked on to which state do they go on ε* again.

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

	ε*	input (0)	ε*
t	x	P	P
		Q	P, M
	y	M	Φ
		N	N

Note

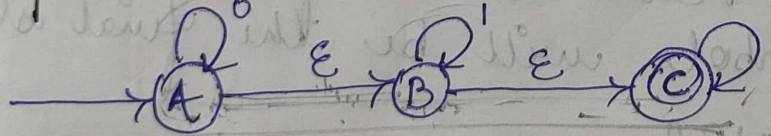
ε* (epsilon closure)

ε* means all the states that can be reached from a particular state only by seeing the ε symbol.

	0
A	{P, M, N}

Example :-

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



consider the whole path

States	ε*
A	{A, B, C}
B	{B, C}
C	C

	ε^*	0	ε^*
A	A	A	A, B, C (union)
	B	ϕ	ϕ
	C	C	C
B	B	ϕ	ϕ (union)
	C	C	C
C	C	C	C

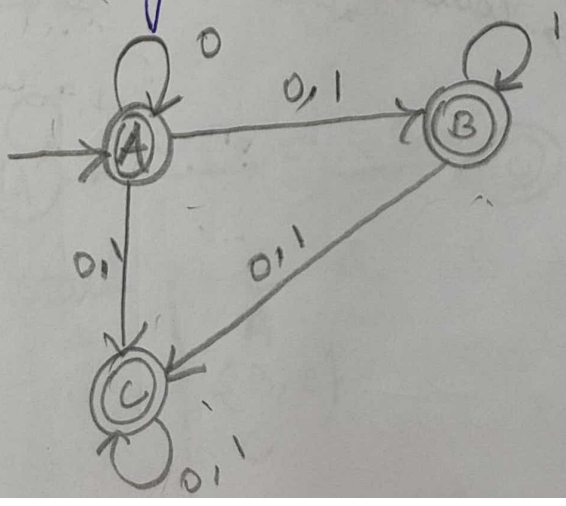
	ε^*	1	ε^*
A	A	ϕ	ϕ
	B	B	B, C
	C	C	C
B	B	B	B, C
	C	C	C
C	C	C	C

δ (Transition Mapping of the NFA)

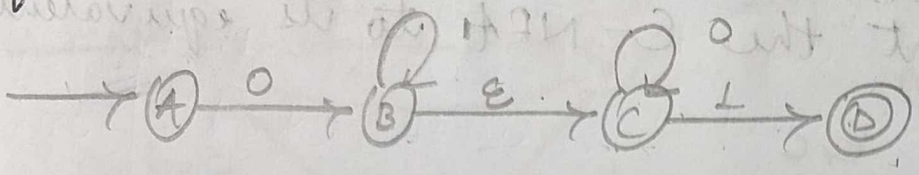
	0	1
→ A	{A, B, C}	{B, C}
B	C	{B, C}
C	C	C

In this example all A, B and C are the final states as in ε-NFA there is a path from all the states to the final state for ε.

Note -
A state which can reach to the final state of ε-NFA on seeing an ε symbol will be the final state of the regular NFA.



Example: Convert the following ε - NFA to its equivalent NFA:



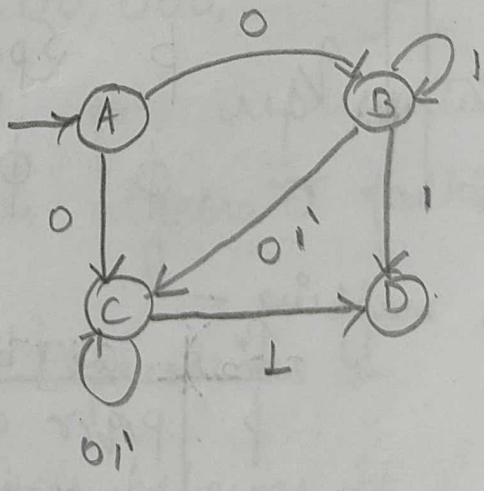
States	ε*
A	A
B	{B, C}
C	C
D	D

	ε*	0	ε*
A	A	B	{B, C}
B	B	ϕ	ϕ
	C	C	C
C	C	C	C
D	D	ϕ	ϕ

	ε*	1	ε*
A	A	ϕ	ϕ
B	B	B	{B, C}
	C	D	D
C	C	D	D
D	D	ϕ	ϕ

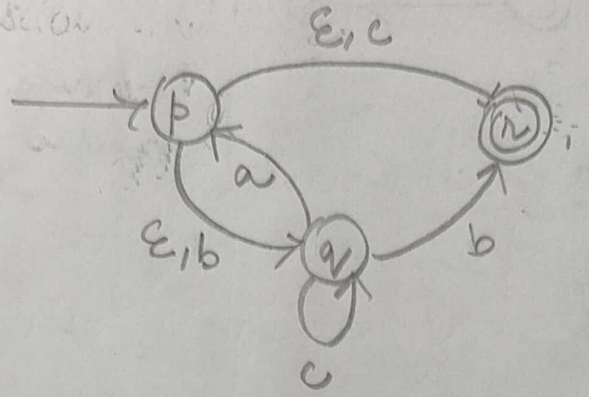
δ mapping –

	0	1
→ A	{B, C}	ϕ
B	C	{B, C, D}
C	C	D
(D)	ϕ	ϕ



Date - 7/10/23

Example - Convert the ε-NFA to its equivalent state.

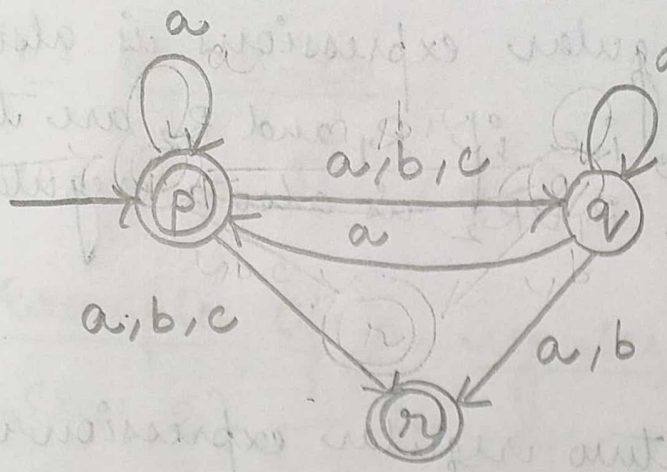


States	ε*
p	{p, q, r}
q	{q}
r	{r}

States	ε*	a	ε*	ε*	b	ε*	ε*	c	ε*
p	p	ϕ	ϕ	p	q	q	p	r	r
	q	b	{p, q, r}	q	r	r	q	q	q
	r	ϕ	ϕ	r	ϕ	ϕ	r	ϕ	ϕ
q	q	b	{p, q, r}	q	r	r	q	q	q
r	r	ϕ	ϕ	r	ϕ	ϕ	r	ϕ	ϕ

δ mapping -

	a	b	c
p	p, q, r	q, r	r, q
q	p, q, r	r	q
r	ϕ	ϕ	ϕ



Any language recognized by a finite state machine is a regular language.

* Regular Expression :-

→ Regular Expressions are the expression built up by using regular operations ($\cup$, ., *)

→ The regular expressions are used to describe the regular languages.

→ The value of a regular expression is a regular language, i.e regular expression generates the regular languages.

eg - $(0 \cup 1)0^* \approx (0+1)0^*$

$\approx (0+1).0^*$

$\approx (0 \cup 1).0^*$

$0^* = \{\varepsilon, 0, 00, 000, \ldots\}$

→ Regular expressions are used for representing certain sets of strings in an algebraic fashion

⊙ Certain Rules about Regular Expressions :-

1) Any terminal symbol i.e symbols belongs to Σ including ε and ϕ are regular expressions.

2) The union of two regular expressions is also a regular expression, i.e if R_1 and R_2 are two regular expressions then $R_1 \cup R_2$ is also a regular expression.

3) The concatenation of two regular expression is also a regular expression, i.e if R_1 and R_2 are two regular expression then $R_1 \circ R_2$ is also a regular expression. $(\equiv R_1 R_2)$

4) The iteration or closure of a regular expression is also a regular expression, i.e if R is a regular expression, then R^* is also a regular expression.

* Formal Definition Of Regular Expression :-

Say that R is a regular expression if R is

1) 'a', for some symbol 'a' in the alphabet Σ
2) ε
3) ϕ
4) $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions.
5) $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions.
6) R_1^*, where R_1 is a regular expression.

Date - 12.10.23

Example → R = (0U1)1* → Regular Expression

* Examples of a regular expression :-

1) {0, 1, 2} → language
   R = 0U1U2 ≈ 0+1+2 → expression
2) {ε, a}
   R = εa = a
3) {a, b, ab, aaa, bb}
   R = a+b+ab+aaa+bb
4) {ε}
   R = ϕ
5) {0, 00, 000, 0000, ----}
   R = 0+
6) {ε, 0, 00, 000, ------}
   R = 0*
7) {ε, 0, 00, 000, ---, 10}
   R = 0* + 10

** Identities of Regular expressions :-

1) ϕUR = RUϕ = R
2) ϕR = Rϕ = ϕ (Concatenating the empty language to any set yields the empty set ϕ)
3) ϕR + Rϕ = ϕ
4) εR = Rε = R
5) ε* = ε

6) $\phi^* = \varepsilon$

7) $R + R = R$

8) $R^* R^* = R^*$

9) $R \cdot R^* = R^* R$

10) $(R^*)^* = R^*$

11) $\varepsilon + R \cdot R^* = \varepsilon + R^* R = R^*$

12) $(PQ)^* P = P(QP)^*$

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

14) $(P+Q).R = PR + QR,$

or

$R.(P+Q) = RP + RQ$

Example:- $L = \{w \mid$ length of w is atleast 2 over $\Sigma = \{a, b\}\}$.

$R = aa + ab + ba + bb + \ldots\ldots$

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

atleast 2 - at most 3

$R = aa + ab + ba + bb + aaa + aab + abb + bbb + bba + baa + bab + aba$

$R = (a+b)(a+b) + (a+b)^3$

$R = (a+b)^2 + (a+b)^3$

Example :-

$L = \{ w \mid$ length of w is atmost $2 \}$

$= \{ \varepsilon, a, b, aa, ab, bb, ba \}$

$R = \varepsilon + a + b + aa + ab + bb + ba$

$R = \varepsilon + (a+b) + (a+b)^2$

$R = (\varepsilon + a + b)(\varepsilon + a + b)$

$L(R) \rightarrow R$ is the regular expression that

Date - 13.10.23

① $\Sigma = \{a, b\}$

$L = \{ w \mid$ length of w is atleast $2 \}$

$= aa + ab + ba + bb + aaa + \ldots$

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

⊙ Examples on Regular Language :-

1) $R = (0+1)1^*$ → all string that may start from 0 or 1 & followed by any no. of 1s.

2) $R = (0+1)^* 00 (0+1)^*$ → it denotes all strings of 0s and 1s with atleast 2 consecutive 0s.

3) $R = (1+10)^*$ → it denotes all strings of 0s and 1s begins with 1 and not having two consecutive zeros.

4) $R = (0+1)^* 011$ → It denotes all strings that start from 0s or 1s but ends with 011

5) $R = 0^*10^*10^*$ → It denotes all strings having exactly two 1s.

6) $R = 01 + 01$ → It will denote 01 (string) only. ($L = \{01\}$)

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

1) $\Sigma^* 1 \Sigma^*$ → It denotes all strings having at least one 1.

2) 0^*10^* → It denotes all strings having exactly one 1.

3) $\Sigma^* 001 \Sigma^*$ → It denotes all strings having 001 as a substring.

Example:- Prove that the regular expression $\varepsilon + 1^*(011)^*(1^*(011)^*)^* = (1+011)^*$

Proof :-

$$LHS = \varepsilon + \overbrace{1^*(011)^*}^{R}\overbrace{(1^*(011)^*)^*}^{R^*}$$

$$= (1^*(011)^*)^* \quad (\because \varepsilon + RR^* = R^*)$$

$$= (1+011)^* \quad (\because (P^*Q^*)^* = (P+Q)^*)$$

$$= RHS$$

Example 2:- Prove that the regular expression $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ is equal to $0^*1(0+10^*1)^*$

Proof -

$$LHS = (1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$$

$$= (1+00^*1)\{\varepsilon+(0+10^*1)^*(0+10^*1)\}$$

$$= (1+00^*1).(0+10^*1)^* = (\varepsilon.1+00^*1).(0+10^*1)^*$$

$$= 1(\varepsilon+00^*).(0+10^*1)^*$$

$$= 1(00^*)^*(0+10^*1)^*$$

$$= (\varepsilon+00^*)1.(0+10^*1)^*$$

$$= 0^*1(0+10^*1)^*$$

* Date - 14.10.23

◎ Arden's Theorem :-

It states that -

"If P and Q are two regular expressions over Σ, and if P does not contain ε, then the following equation $\boxed{R = Q + RP}$ has a unique solution, that is, $R = QP^*$."

Proof :- Given $R = Q + RP$ ——— (1)

Substitute $R = QP^*$ in equ^n (1)

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

$$R = Q(\varepsilon + P^*P)$$

$$R = QP^* \quad (\text{As } \varepsilon + R.R^* = R^*)$$

(Proved that QP^* is a solution now we will verify its uniqueness)

Now to prove that,

$R = QP^*$ is the unique solution to $R = Q + RP$

Given, $R = Q + RP$

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

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

$$= Q + QP + RP^2 \quad \text{——— (2)}$$

$$= Q + QP + (Q + RP)P^2$$

$$= Q + QP + QP^2 + RP^3 \quad \text{——— (3)}$$

$= Q + QP + QP^2 + (Q+RP)P^3$

$= Q + QP + QP^2 + QP^3 + RP^4$ ——— (4)

⋮

$= Q + QP + QP^2 + QP^3 + \ldots + QP^n + RP^{n+1}$

$= Q(\varepsilon + P + P^2 + P^3 + \ldots + P^n + P^*P^{n+1})$

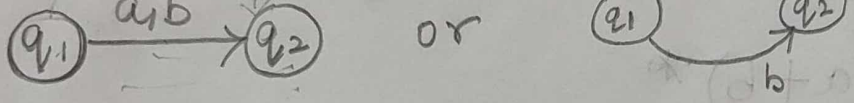
$= Q(\varepsilon + P + P^2 + P^3 + \ldots + P^n + P^*P^{n+1})$

$= QP^*$

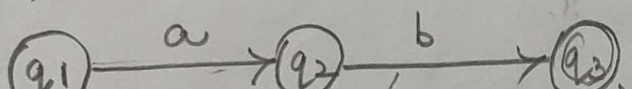
⊙ Conversion of Regular Expression to Finite Automata:

#. Rules :-

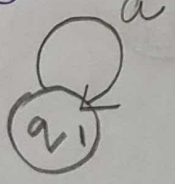
1) $R = (a+b)$



2) $R = ab$

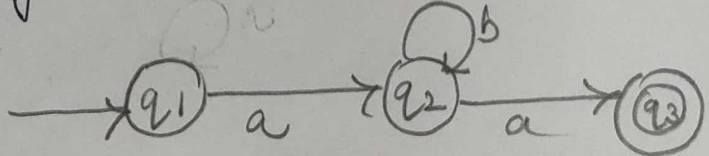


3) $R = a^*$

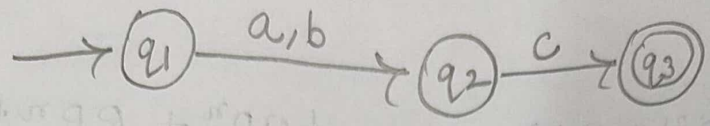


Example :-
Convert the following regular expression into equivalent FA

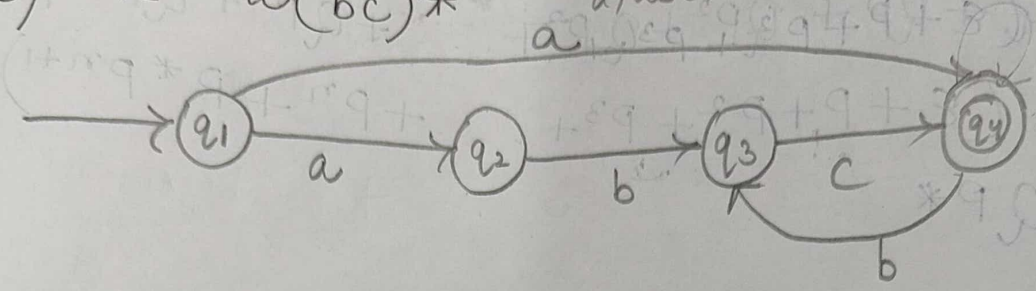
① $R = ab^*a$



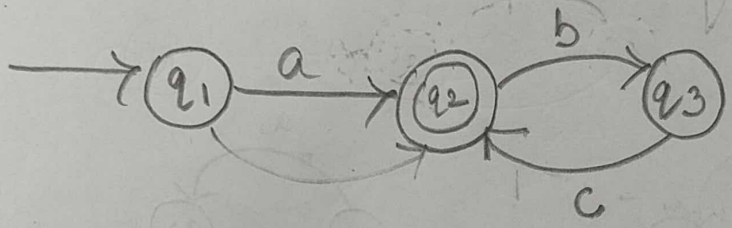
2) $R = (a+b)c$



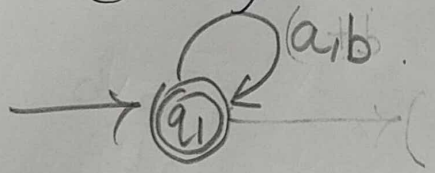
3) $R = a(bc)^*$



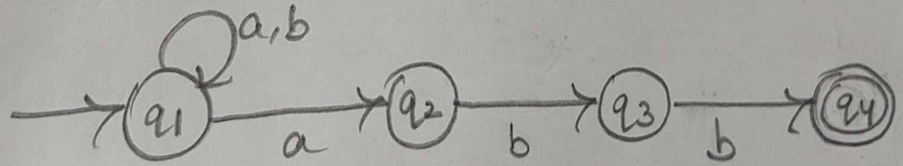
or



4) $R = (a+b)^*$



5) $R = (a+b)^* abb$



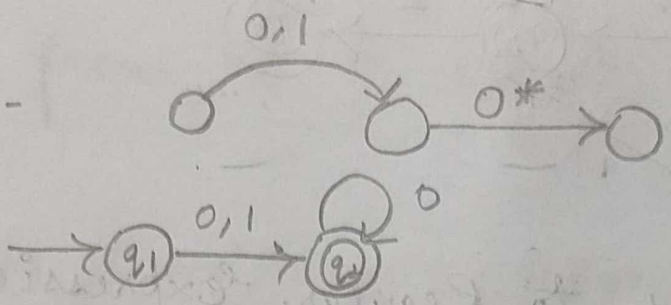
Date - 19.10.23

1) Convert the following Regular expression to FA :-

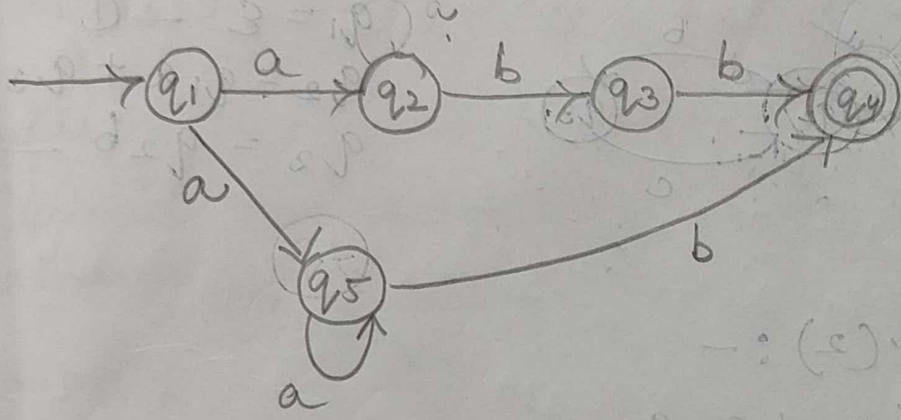
$R = (0+1)\,0^*$

Soln -

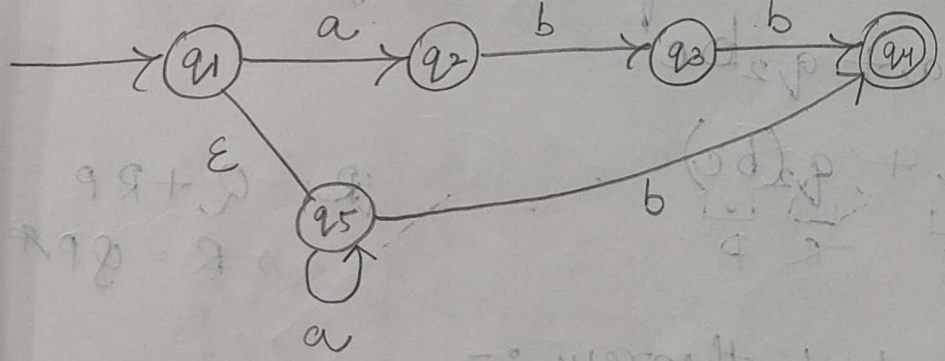
The FA is -



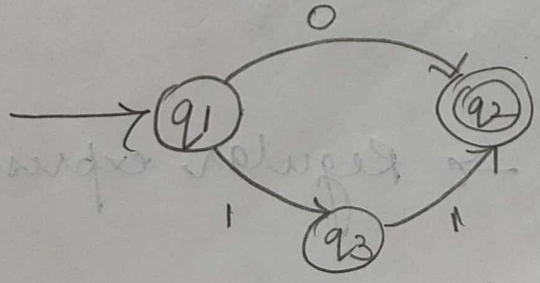
$R = abb + a+b$



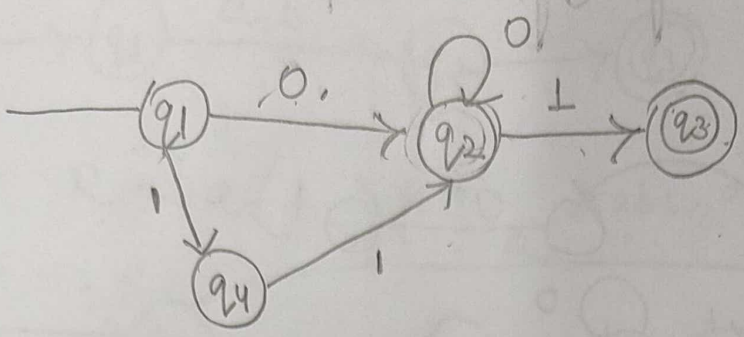
$R = abb + a^*b$



$R = 0 + 11$

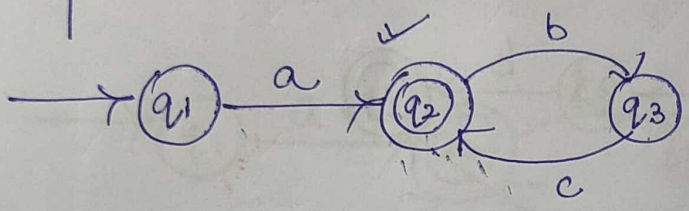


$$R = (0+11)\,0^*1$$



* Convert FA to Regular expression:-

Example -



$q_1 = \varepsilon$ — (1)

$q_2 = q_1 a + q_3 c$ — (2)

$q_3 = q_2 b$ — (3)

Now solve equ^n (2) :-

$$q_2 = q_1 a + q_3 c$$

Putting the values of q_1 & q_3 from (1) & (3)

$$q_2 = \varepsilon a + q_2 bc$$

$$\Rightarrow \underset{R}{q_2} = \underset{Q}{a} + \underset{R}{q_2}\underset{P}{(bc)}$$

$R = Q + RP$

$\Rightarrow R = QP^*$

From Arden's theorem :-

$$R = QP^*$$

$$R = a(bc)^*$$

$$q_2 = a(bc)^* \leftarrow \text{Regular expression}$$

* If multiple final states are present just add the regular expressions derived for each final state.

Date - 26.10.23

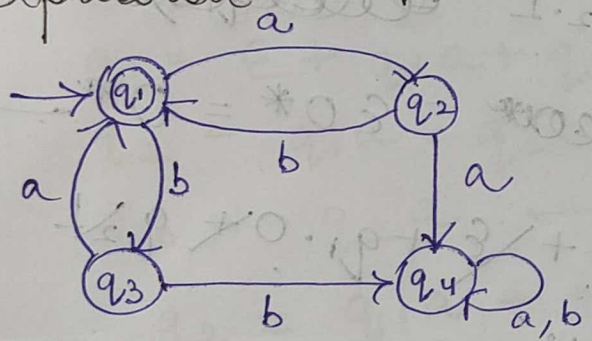
* Conversion from FA to Regular expression :-

⊙ Steps :-

1) Find out the transition or edge which comes from other state to that particular state.
2) Then find out solution for the state which is in final state & leave other states as it is.

Example :-

Q1. Convert the following FA to equivalent Regular expression.



$q_1 = \varepsilon + q_2 b + q_3 a$ ——— (1)

$q_2 = q_1 a$ ——— (2)

$q_3 = q_1 b$ ——— (3)

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$ ——— (4)

Solving equ^n (1) —

$q_1 = \varepsilon + q_1 ab + q_1 ba$

$q_1 = \varepsilon + q_1 (ab+ba) =$

q_1 ↓ R, ε ↓ Q, q_1 ↓ R, $(ab+ba)$ ↓ P

$R = QP^*$

$R = \varepsilon (ab+ba)^*$

$q_1 = \varepsilon (ab+ba)^*$

$q_1 = \varepsilon + q_1 \cdot 0 = \varepsilon 0^* = 0^*$

$q_2 = q_1 1 + q_2 1$

$q_2 = \varepsilon 0^* 1 + q_2 1 1^*$

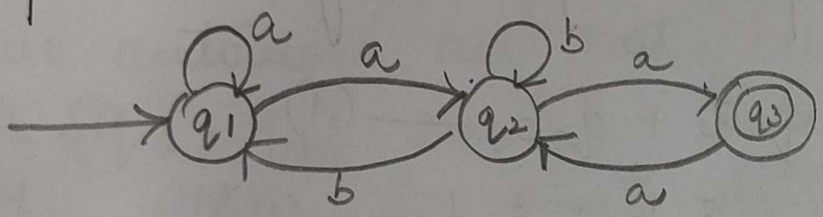
$q_2 = \varepsilon 0^* 1 1^*$

$q_2 = 0^* 1 1^*$

$q_1 + q_2 = 0^* + 0^* 1 1^*$

$= 0^* + 0^* 1+$

Q3. Convert the following FA to equivalent Regular expression.



$q_1 = \varepsilon + q_1 a + q_2 b$ — (1)

$q_2 = q_1 a + q_2 b + q_3 a$ — (2)

$q_3 = q_2 a$ — (3)

$q_2 = q_1 a + q_2 b + q_2 a a$

$q_2 = q_1 a + q_2 (b + aa)$

$q_2 = q_1 a (b + aa)^*$ — (4)

$q_1 = \varepsilon + q_1 a + q_1 a (b + aa)^* b$

$q_1 = \varepsilon + q_1 \{ a + a(b + aa)^* b \}$

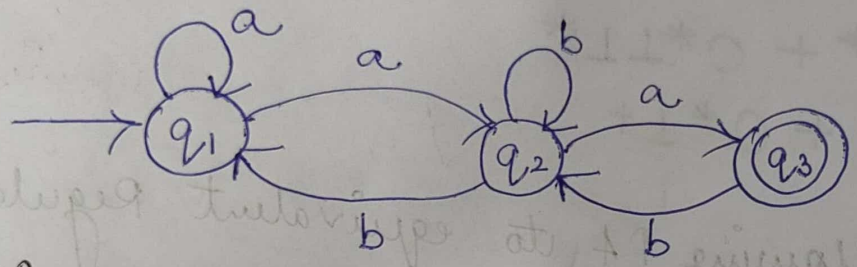
$q_1 = \varepsilon (a + a(b + aa)^* b)^*$

$q_1 = (a + a(b + aa)^* b)^*$ — (5)

$$q_2 = (a + a(b+aa)^*b)^*a \cdot (b+aa)^* \quad \text{—(6)}$$

$$q_3 = (a + a(b+aa)^*b)^*a(b+aa)^*a$$

Date - 27.10.23



Q. Find the regular expression for the given FA

$$q_1 = \varepsilon + q_1a + q_2b \quad \text{—(1)}$$

$$q_2 = q_1a + q_2b + q_3b \quad \text{—(2)}$$

$$q_3 = q_2a \quad \text{—(3)}$$

$$q_2 = q_1a + q_2b + q_2ab$$

$$q_2 = q_1a + q_2(b+ab)$$

$$= q_1a \cdot (b+ab)^* \quad \text{—(4)}$$

$$q_1 = \varepsilon + q_1a + q_1a(b+ab)^*$$

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

$$= (a + a(b+ab)^*)^* \quad \text{—(5)}$$

$$q_2 = (a + a(b+ab)^*)^* a \cdot (b+ab)^*$$

$$q_3 = (a + a(b+ab)^*)^* a \cdot (b+ab)^* a$$ Ans

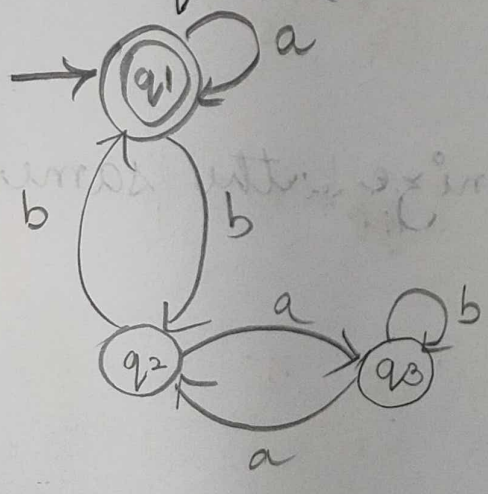
* Equivalence Of Two Finite Automata :-

⊙ Rules :-

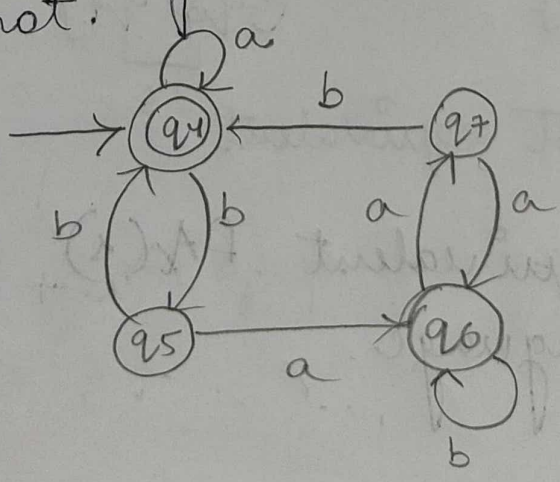
1) For any pair of states $\{q_i, q_j\}$ the transition for input $a \in \Sigma$ is defined by $\{q_a, q_b\}$ where $\delta(q_i, a) = q_a$ and $\delta(q_j, a) = q_b$. The two automata are not equivalent if for a pair $\{q_a, q_b\}$, one is intermediate state & the other is final state.

2) If initial state is the final state of one automata, then in second automaton also initial state must be final state for them to be equivalent.

→ Example - find whether the given two automata are equivalent or not.



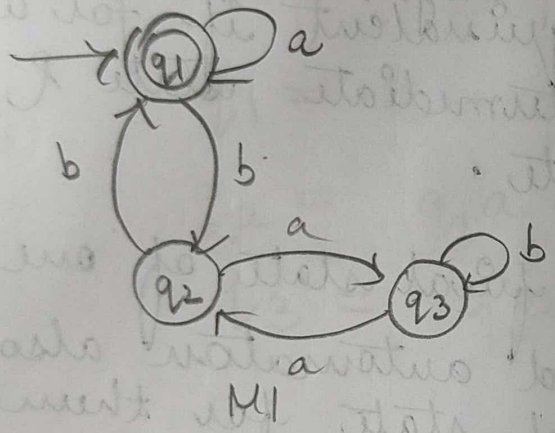
M1



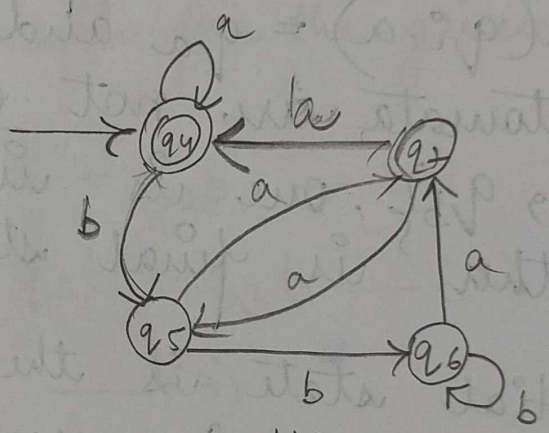
M2

States	a	b
$\{q_1, q_4\}$	$\{q_1, q_4\}$ (FS, FS)	$\{q_2, q_5\}$ (IS, IS)
$\{q_2, q_5\}$	$\{q_3, q_6\}$ (IS)	$\{q_1, q_4\}$ (FS)
$\{q_3, q_6\}$	$\{q_2, q_7\}$ (IS)	$\{q_3, q_6\}$ (IS)
$\{q_2, q_7\}$	$\{q_3, q_6\}$ (IS)	$\{q_1, q_4\}$ (FS)

Date - 28.10.23



M1



M2

	a	b
$\{q_1, q_4\}$	$\{q_1, q_4\}$ (FS)	$\{q_2, q_5\}$ (IS)
$\{q_2, q_5\}$	$\{q_3, q_7\}$ (IS)	$\{q_1, q_6\}$ (FS, IS)

$\Rightarrow$ Not equivalent

* Equivalent FA(s) recognize the same language.

Example -

Regular expression M1:-

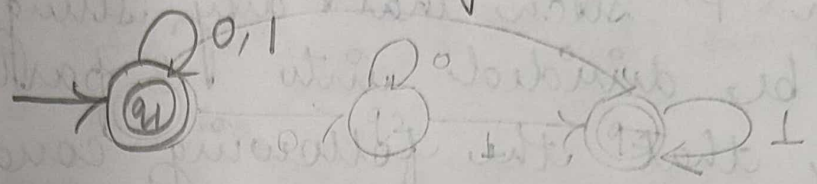
Construct the FA for the following Regular expressions

$R_1 = (0*1*)*$

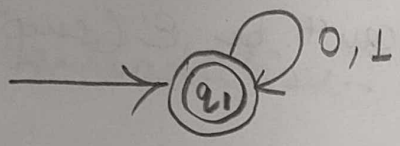
$R_2 = (0+1)*$

and check whether the two generated FA are equivalent.

Solu^n - R_1:-



R_2:-



∴ Equivalent