Finite Automata

INTRODUCTION:

Theory of Computation (TOC):

It describes the basic ideas and models underlying computing.

Computation :-

Computation is executing an algorithm ie, it involve taking some inputs and performing required operation on it to produce an output.

Computation: Input x → Function → Output $y = f(x)$

Computation is a sequence of steps that can be performed by computer.

→ TOC suggests various abstract models of computation, represented mathematically.

→ Computer which performs computation are not actual computers, they are abstract machines.

→ Our focus is on abstract machines that can be defined mathematically.

Examples of abstract machines:

1) Turing Machines (powerful as real computers) - Universal model.

2) Finite Automata (Simple) - Restricted model.

Applications of TOC:

1. Compiler design
2. Robotics
3. Artificial Intelligence
4. Knowledge Engineering.

(i) FINITE AUTOMATA:-

-> It recognises regular languages only. It was developed by "Scott Robin" in 1950 as a model of a computer with limited memory.

-> It receives its input as a string, usually from an input tape. It delivers no output at all except an indication of whether the input is acceptable or not.

-> Hence used for decision making ~~system~~ problems.

Applications of Finite Automata:

1. It is an useful tool in the design of Lexical Analyser - A part of compiler that gets characters into tokens such as variable name and keyword.
2. Text editor
3. Pattern Matching
4. File Searching problem
5. Text processing. (occurrence of one string in a file).

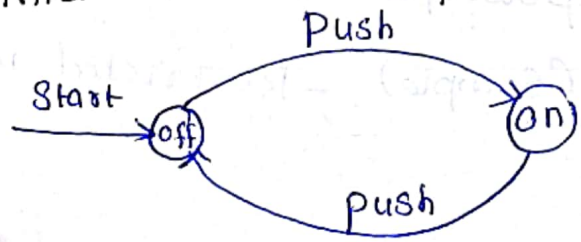
Limitations:

1) It can recognise only simple Languages (regular)

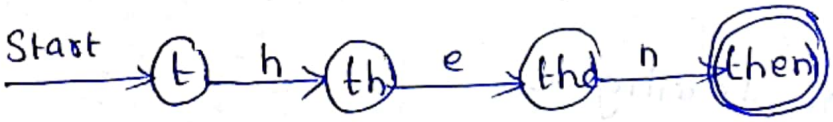
2) FA can be designed for decision making problems.

Example:

(i) Finite automaton for an on/off switch - Digital systems



2. Lexical analysis - Recognising a string "then"



(II) CONTEXT - FREE LANGUAGES:

(i) It allows richer syntax than regular languages.

(ii) Context-free grammars can be recognised by computing devices like pushdown automata

(iii) Pushdown automata is a finite automata with an auxillary memory in the form of stack.

IV It is used in the design of parsers.

(III) TURING MACHINES:

1) It was invented by British Mathematician - "Alan Turing".

2) It is a most powerful abstract model.

3) It has infinite amount of tape memory accessible in both directions, that is left or right.

4) It can recognise recursively enumerable languages.

5. It simulates digital computers in terms of power.

6. Some problems are theoritically solvable and are not practically solvable due to non polynomial time. If any function is not solvable by turing machine, it cannot be computed by digital computer.

FINITE STATE SYSTEMS

The Finite Automaton (FA) is a mathematical model of a system, with discretes inputs and outputs and a finite number of memory configuration called states and a set of transitions from state to state that occurs on input symbols from alphabet Σ.

Examples of finite automaton models are:

1) Software for designing digital circuits - Silicon compilers.
2) Lexical analyser of a compiler.
3) Searching for keywords in a file (or) on the web Text editors.

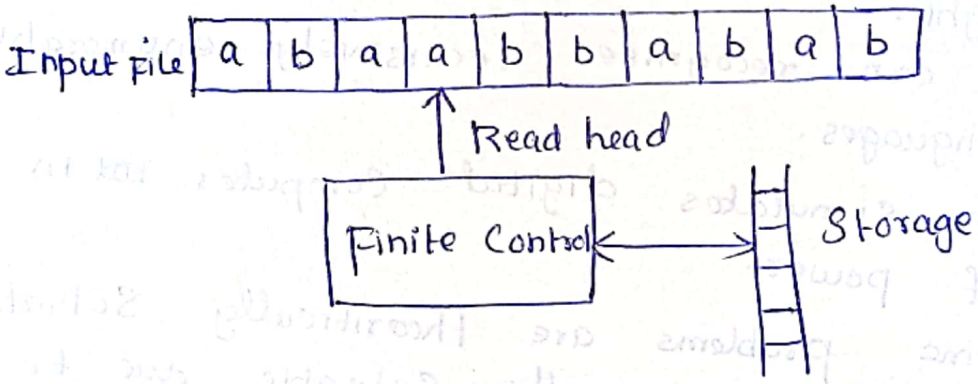
The FA is also classified as:

(i) Deterministic Finite Automata (DFA)
(ii) Non Deterministic Finite Automata (NFA).

DETERMINISTIC FINITE AUTOMATA (DFA)

A finite State automata is the simplest and most restricted model of a computer, we begin by looking at Deterministic Finite Automata (DFA).

BASIC DFA:



DFA is a language recogniser that has:

1. An input file containing an input string.
2. A finite control - a device that can be in a finite number of States.
3. A reader - a sequential reading device
4. A program

How DFA works?

(i) Initialisation:
Reader (read head) should be over the leftmost Symbol. Finite control is in Start State.

(ii) Single Step:

Reader reads current Symbol then, reader moves to the next Symbol to right.

Control enters a new State that (deterministically) depends on the current State and current Symbol. There may be no desired next State, in which case the machine Stops. The machine repeats this action.

(iii) No current Symbol:

All Symbols have been read then, if control is in final State, the input String is accepted. Otherwise, the input String is not accepted.

DFA SPECIFICATION:

A DFA M is formally defined specified by a five tuple

$$M = (Q, \Sigma, S, F, \delta), \text{ where}$$

Q - a finite, non empty set of States.

Σ - a finite, non empty set of input alphabets

$S \in Q$ - a start State.

$F \subseteq Q$ - a set of final States Subset of Q

δ - a transition function, defined as

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

Current State	Current Symbol	Next State
p	σ	$\delta(p, \sigma)$

The transitions are represented in the form of transition table or transition diagram

Example:

Transition Table of DFA:

DFA is specified by

$M = (Q, \Sigma, \delta, S, F)$ where

$Q = \{q_0, q_1\}$, $S = q_0$ $F = \{q_1\}$

$\Sigma = \{a, b\}$ and δ is given by

	States	Inputs	
		a	b
→	q_0	q_0	q_1
*	q_1	q_1	q_0

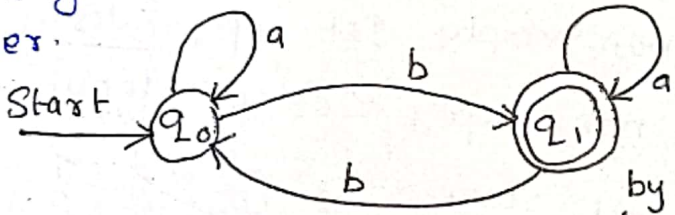
→ Starting State

* Ending State. or Final State

(O) → Final State.

Transition Diagram of DFA:

"Transition Diagram" associated with DFA is a directed graph whose vertices corresponding states of DFA. The edges are the transitions from one State to another.



Start State S is represented by →

Final states are represented by * or double circle.

PROPERTIES OF TRANSITION FUNCTION (δ) 10

(I) $\delta(q, \in) = q$

This means the State of the system can be changed only by an input symbol else remains in original state.

(II) For all String w and input Symbol a

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

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

(III) The transition function δ can be extended to $\hat{\delta}$ or δ that operates on States and Strings (as opposed to States and Symbols).

Basic : $\bar{\delta}(q, \epsilon) = q$

Induction : $\bar{\delta}(q, xa) = \delta(\bar{\delta}(q, x), a)$

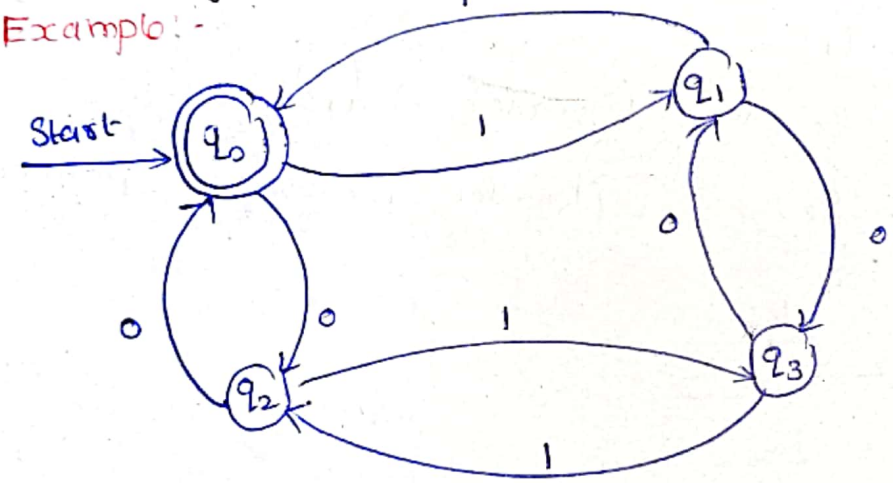
LANGUAGE OF A DFA:

A String x is said to be accepted by DFA $M = (Q, \Sigma, S, F, \delta)$, if $\bar{\delta}(q_0, x) = P$ for some P in F.

Method: A finite automata accepts a string $w = a_1 a_2 \ldots a_n$ if there is a path in the transition diagram which begins at a start state ends at an accepting state with the sequence of labels $a_1, a_2 \ldots a_n$.

* the Language accepted by finite Automata (A) is $L(A) = \{ w : \bar{\delta}(q_0, w) \in F \}$ where F is a final state
* the language accepted by finite automatds are called "regular Language".

Example:-



Transition diagram

The DFA for the above transition is represented as: $M = (Q, \Sigma, S, F, \delta)$ where

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

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

$S = \{q_0\}$

$F = \{q_0\}$

δ = transition function is represented as transition table.

States	Inputs	
	0	1
→ * q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Suppose 110101 is input to M, check the validy of the input.

→ We note that finite automata is in start state and reads from left most-

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

$\delta(q_1, 1) = q_0$ (Reader reads next symbols)

$\delta(q_0, 0) = q_2$ (Reader moves one position right)

$\delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_1$

$\delta(q_1, 01) = q_0$

Since q_0 is a final state, the given string is accepted.

Example 2:

Let $M = (Q, \Sigma, S, F, \delta)$ be a DFA with

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

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

$S = \{q_0\}$

$F = \{q_0\}$

The same operation $\delta(q_0, 110101)$ can be written directly from a transition table. That is:

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

q_0: Starting State

q_0: Final State

Example: 2

Describe the Language accepted by DFA

$M = (\{q_0, q_1, q_2\}, \{0,1\}, q_0, \{q_1\}, \delta)$, where δ is given by

States	Inputs	
	0	1
→ q_0	q_0	q_1
* q_1	q_0	q_2
q_2	q_2	q_1

(i) $\delta(q_0, 01) = \delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

∴ Accepting.

(ii) $\delta(q_0, 010) = \delta(q_0, 0) = q_0$

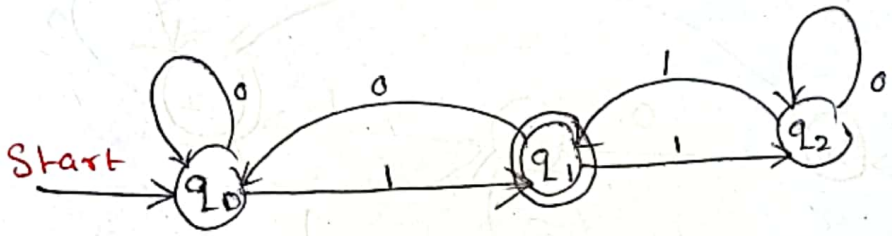
$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_0$

∴ No accepting State.

Solution:

The transition diagram is



$\delta(q_0, 0) = q_0$ Non accepting State

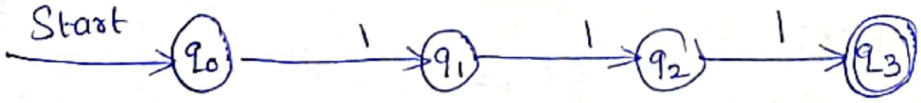
$\delta(q_0, 01) = q_1$ $\delta(q_0, 1) = q_1$ Accepting State

$\delta(q_0, 010) = \delta(q_0 10) = \delta(q_1, 0) = q_0$ Non accepting State.

Hence the above DFA accepts 01, 101, 0111 ... That is odd number of ones at the end of String.

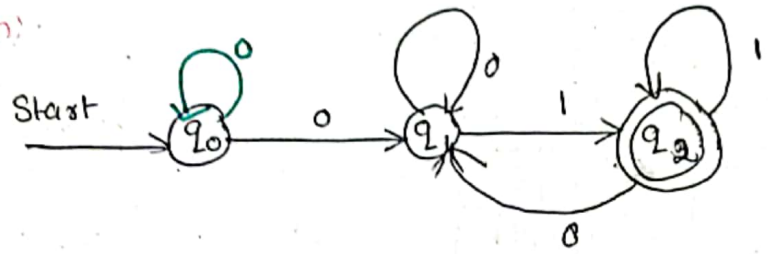
Example 3: Design DFA which accepts only input 111 over the input set $\Sigma = \{1\}$

Solution



Example 4: Design FA which accepts only those strings which start with 0 and end with 1 over the input set $\Sigma = \{0, 1\}$

Solution:



Example 5:

Design FA and Transition table which accepts odd number of 1's and any number of 0's.

Solution:

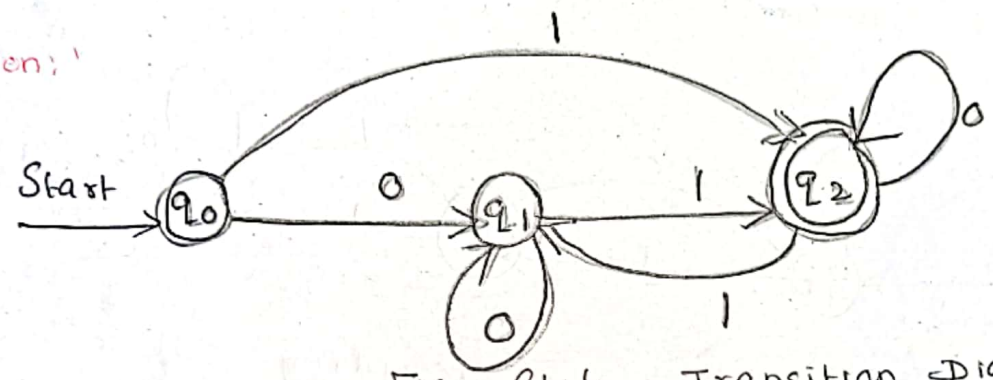


Fig: State Transition Diagram

States	Inputs	
	0	1
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_1

Transition Table

FA = $A = (Q, \Sigma, \delta, S, F)$

Transition State:

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_2$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_1$

Check : 0110 & 1110

0110

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

$\delta(q_1, 0) = q_1$ ∴ Reached non final state so it is not accepted

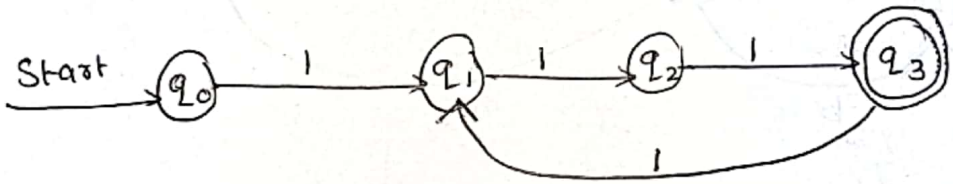
1110

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

∴ Reached final state. So it is accepted.

Example: 6 Design DFA which accepts the unary number which is divisible by 3 over $\Sigma = \{1\}$.

Solution:



Check:

① String 111

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

=> Reached final state => String accepted

2) 1111 => $\delta(q_0, 1) = q_1$, $\delta(q_1, 1) = q_1$, $\delta(q_2, 1) = q_3$, $\delta = (q_3, 1) = q$

=> Reached non final state.

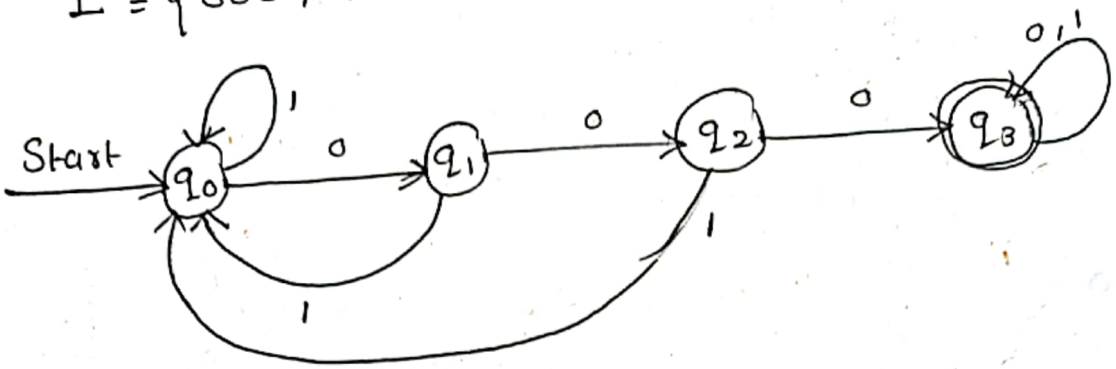
=> not accepted.

Example: 7

Design a DFA to accept strings over $\Sigma = \{0, 1\}$ with three consecutive 0's.

Solution:

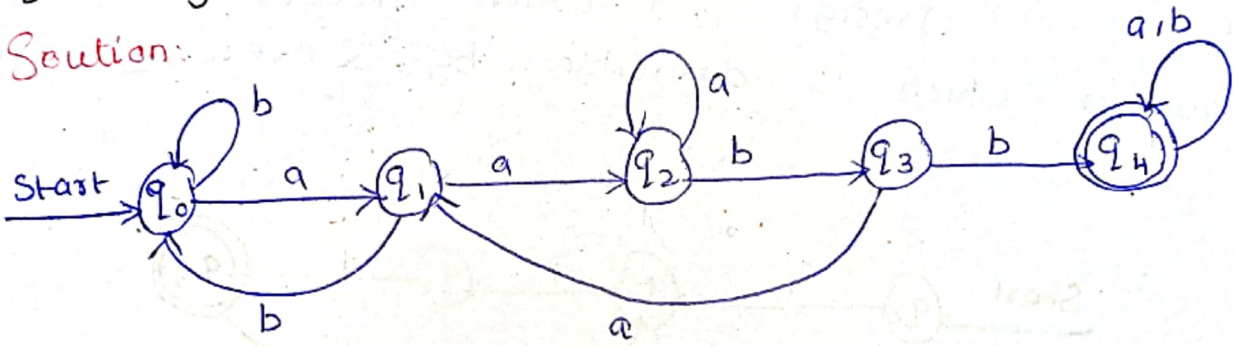
$L = \{000, 1000, 01000, 101000, \ldots\}$



Example 8:

Construct a DFA over $\Sigma = \{a, b\}$ containing a Substring aabb

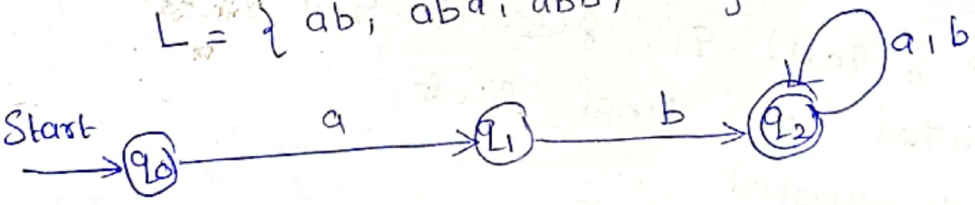
Soution:



Example 9:

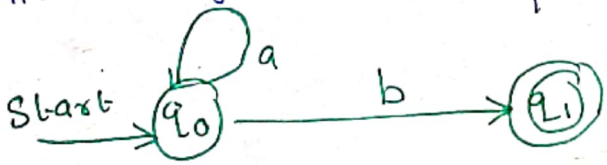
Construct a DFA over $\Sigma, \{a, b\}$ containing a Substring with the prefix ab

$L = \{ab, aba, abb, \ldots\}$



Example:

Design a DFA that accepts the strings defined by the language $L = \{a^n b : n \geq 0\}$



$L = \{b, ab, aab, aaab, \ldots\}$

NON DETERMINISTIC FINITE AUTOMATA (NFA)

DFA VS NFA

In DFA

1) Each symbol causes a move (eventhough the state of the machine remains unchanged after the move)
2) The next state is completely determined by the current state and current symbol.

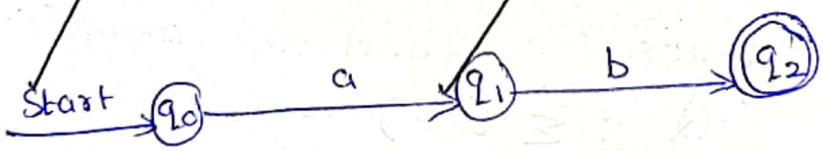
Where as in NFA

1) The machine can move without consuming any symbols and sometimes there is no possible moves and sometimes there are more than one possible moves.
2) The state is only partially determined by the current state and input symbol.

Example:

$L = \{(ab)^n \mid n > 0\}$. The corresponding DFA and NFA are:

DFA



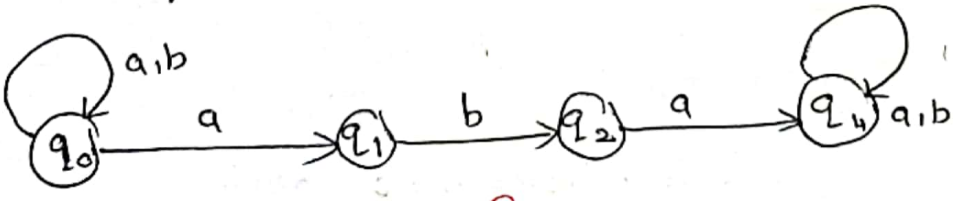
Example:

Construct the NFA and DFA for

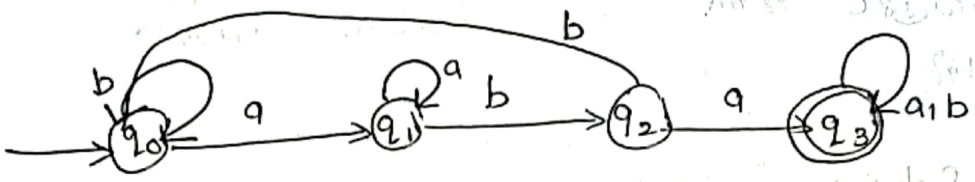
$L = \{$ input mode of $\Sigma = \{a, b\}$ and have aba as substring $\}$

$\therefore L = \{aba, aaba, baba, \ldots\}$

NFA:



DFA:



Check:

aaba string

$\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_1$, $\delta(q_1, b) = q_2$, $\delta(q_2, b) = q_3$

∴ Reached final state.

∴ DFA accepted.

Definition of NFA:

The Non deterministic Finite Automa (NFA) is defined by a five tuple $M = (Q, \Sigma, \delta, \$, F)$, where.

Q - finite non empty set of states

Σ - Finite set of input alphabet

$S \in Q$ - Start state, belongs to Q

$F \subseteq Q$ - Set of final states, subset of Q.

δ - transition function mapping

$$Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$$

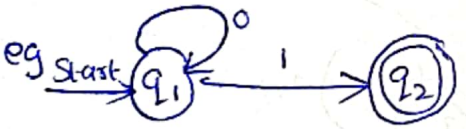
TRANSITION TABLE

The transition table can be used to view the transition function of either an NFA or DFA. The only difference is that each entry in a table for the NFA is a set.

Eg

States	i/p 0	1
→ q_0	$\{q_0, q_1\}$	$\{q_1\}$
* q_1	ϕ	$\{q_1\}$

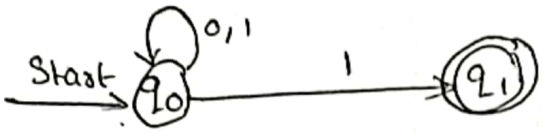
DIFFERENCE BETWEEN DFA and NFA:

	DFA	NFA
1)	The transition function δ returns exactly one State	δ is a transition function that takes a State and input Symbol as arguments but returns a zero, one or more States.
2)	$Q \times \Sigma \rightarrow Q$.  $\delta(q_1, 0) = \{q_1\}$ (Q, Σ, Q) $\delta(q_1, 1) = \{q_2\}$	$Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$  $\delta(q_1, 0) = \{q_1, q_2\}$ $\delta(q_1, 1) = q_2$ $\delta(q_1, 0) = q_1$ $\delta(q_1, 0) = \epsilon$ $\therefore 2^Q = 2^2 = 4$ State transitions

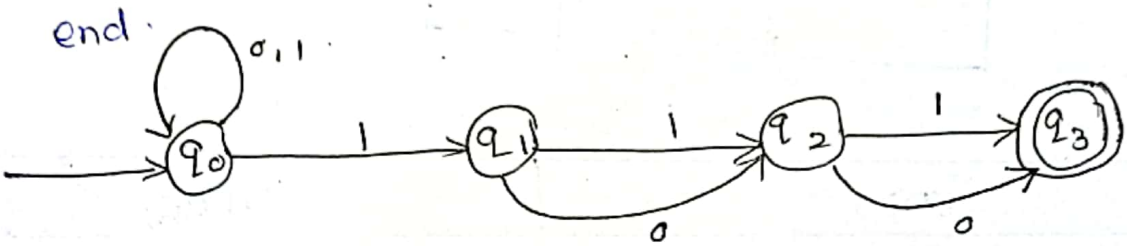
Example: obtain an NFA for a language consisting of all strings over $\{0,1\}$.

Solution:

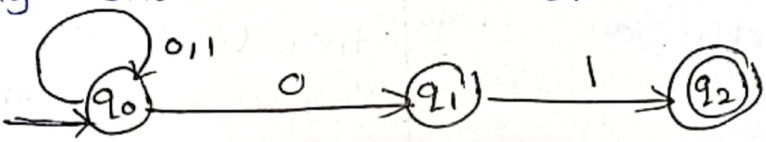
(a) String should end in a 1



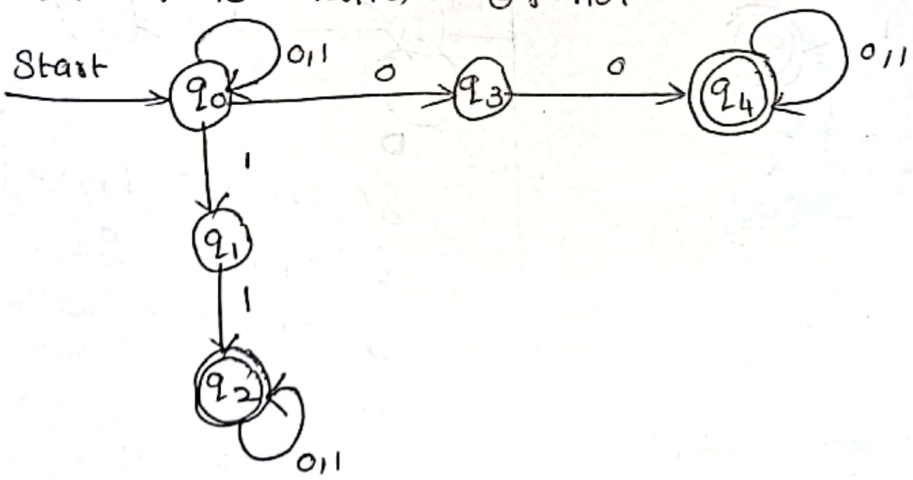
b) Containing a '1' in the third position from the end.



c) String should end in 01



Example: Consider the given NFA to check whether $w = 01001$ is valid or not



Solution:

$M = (Q, \Sigma, S, F, \delta)$ is a NFA, where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$ $\quad S = q_0$ $\quad \delta = Q \times \Sigma$ to 2^Q

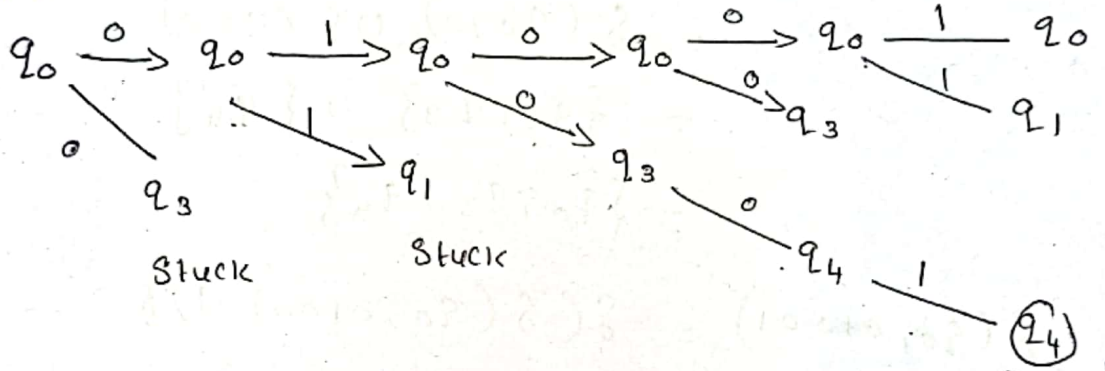
$\Sigma = \{0,1\}$ $\quad F = \{q_2, q_4\}$

Transition Table:

States	Inputs	
	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\{\}$	q_2
q_2	q_2	q_2
q_3	q_4	$\{\}$
q_4	q_4	q_4

Transition State for $w = 01001$

~~$\delta(q_0, 0)$~~ = Method : 1



Hence Accepted

$\therefore q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_4 \xrightarrow{1} q_4$

$\therefore$ Hence the given string is accepted.

Method 2: $w = 01001$

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

$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$

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

$= \delta(q_0, 1) \cup \delta(q_3, 1)$

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

$= \{q_0, q_1\}$

$\delta(q_0, 010) = \delta(\delta(q_0, 01), 0)$

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

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

$= \{q_0, q_3\} \cup \phi$

$= \{q_0, q_3\}$

$\delta(q_0, 0100) = \delta(\delta(q_0, 010), 0))$

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

$= \delta(q_0, q_3) \cup \delta(0$

$= \delta(q_0, 0) \cup \delta(q_3, 0)$

$= \{q_0, q_3\} \cup \{q_4\}$

$= \{q_0, q_3, q_4\}$

$\delta(q_0, 01001) = \delta(\delta(q_0, 0100), 1)\}$

$= \delta(q_0, q_3, q_4, 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1), \delta(q_4, 1)$

$= \{q_0, q_1\} \cup \{\phi\} \cup \{q_4\}$

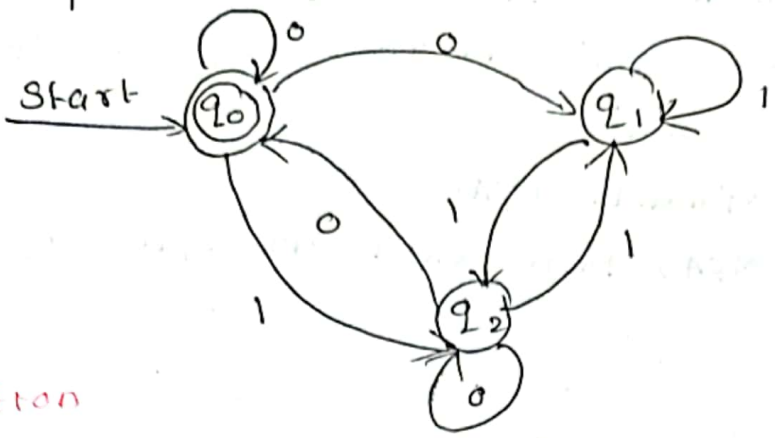
$= \{q_0, q_1, q_4\}$

$\delta(q_0, 01001) \cap F = \{q_0, q_1, q_4\} \cap \{q_4\}$

$= \{q_4\}$ = final state.

∴ Hence the given string is accepted.

Example: For the NFA shown, check whether the input string 0100 is accepted or not?



Solution

The Transition table δ is:

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

Input String = 0100

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

$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$

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

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

$= q_2 \cup \{q_1, q_2\}$

$= \{q_1, q_2\}$

$\delta(q_0, 010) = \delta(\delta(q_0, 01), 0)$

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

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

$= \phi \cup \{q_0, q_2\}$

$= \{q_0, q_2\}$

$\delta(q_0, 0100) = \delta(\delta(q_0, 010), 0)$

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

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

$= \{q_0, q_1\} \cup \{q_0, q_2\} = \{q_0, q_1, q_2\}$

$\therefore \delta(q_0, 0100) \cap F = \{q_0, q_1, q_2\} \cap \{q_0\}$

$= \{q_0\}$ = Final state.

$\therefore$ Hence the given string is accepted.

Equivalence of DFA and NFA

* As every DFA is an NFA, the class of languages accepted by NFA's includes the class of languages accepted by DFA's.
* DFA can Simulate NFA.
* For every NFA, there exist an equivalent DFA.

Theorm:

For every NFA, there exists a DFA which Simulates the behavior of NFA. If L is the set accepted by NFA, then there exists a DFA which also accepts L.

Proof:

Let $M = (Q, \Sigma, q_0, F, \delta)$ be NFA accepting L.

We construct DFA

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

1.) $Q' = 2^Q$

any State in Q' is denoted by $[q_1, q_2, \ldots q_i]$ where $q_1, q_2, \ldots q_i \in Q$)

2) $q_0' = [q_0]$

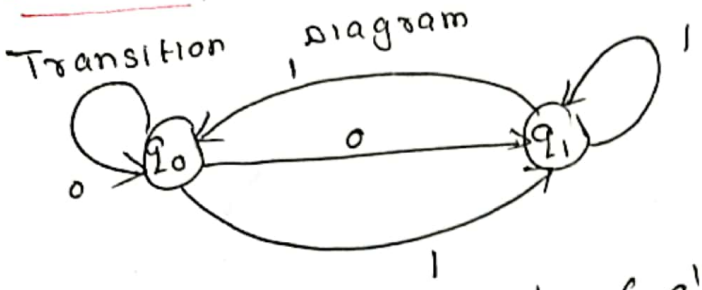
3) F' is set of final states.

Example: Construct the DFA equivalent to the NFA $M = (\{q_0, q_1\}, \{0,1\}, \delta\{q_0, \{q_1\})$ and δ is defined as

States	Inputs	
	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_0, q_1\}$

Solution:

Transition diagram



We construct DFA $M' = (Q', \{0,1\}, \delta', [q_0], F')$ accepting $L(M)$ as follows:

$Q' = 2^Q$ (All subsets of $Q = \{q_0, q_1\}$)

$\therefore$ Subset $= \{\phi, [q_0], [q_1], [q_0, q_1]\}$

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

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

$\delta'([q_1], 0) = \phi$ — since $\delta(q_1, 0) = \phi$

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

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

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

$\therefore$ DFA transition Table is.

	States	Inputs 0	Inputs 1
→	$[q_0]$	$[q_0, q_1]$	$[q_1]$
*	$[q_1]$	ϕ	$[q_0, q_1]$
*	$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

$\therefore$ set of final states of DFA $M' = \{[q_1], [q_0, q_1]\}$

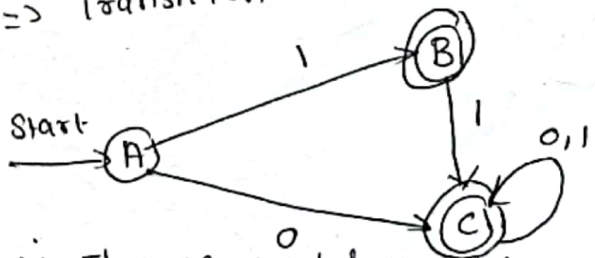
Starting state $= [q_0]$

$[q_0] \to A$ $[q_1] \to B$

$[q_0, q_1] \to C$

Rewrite the DFA table. => Transition ~~States~~ diagram

	States	0	1
→	A	C	B
*	B	ϕ	C
*	C	C	C



$\therefore$ This is accepted in DFA.

Construct a DFA for the given NFA

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

δ is given as:

States	Input	
	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
* q_2	ϕ	$\{q_0, q_1\}$

$2^3 = 8$

We construct $M' = (Q', \{0,1\}, [q_0], \delta', F')$

Q' contains all subsets of $Q = \{q_0, q_1, q_2\}$

Solution: $= \{\phi, q_0, q_1, q_2, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

Step 1:

$\delta'(q_0, 0) = \{q_0, q_1\}$ ✓

$\delta'(q_0, 1) = \{q_2\}$ ✓

Step 2:

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

$\delta'([q_0, q_1], 1) = \{q_2\} \cup \{q_1\} = \{q_1, q_2\}$ ✓

Step 3:

$\delta'(q_2, 0) = \{\phi\}$ ✓

$\delta'(q_2, 1) = \{q_0, q_1\}$ ✓

Step 4:

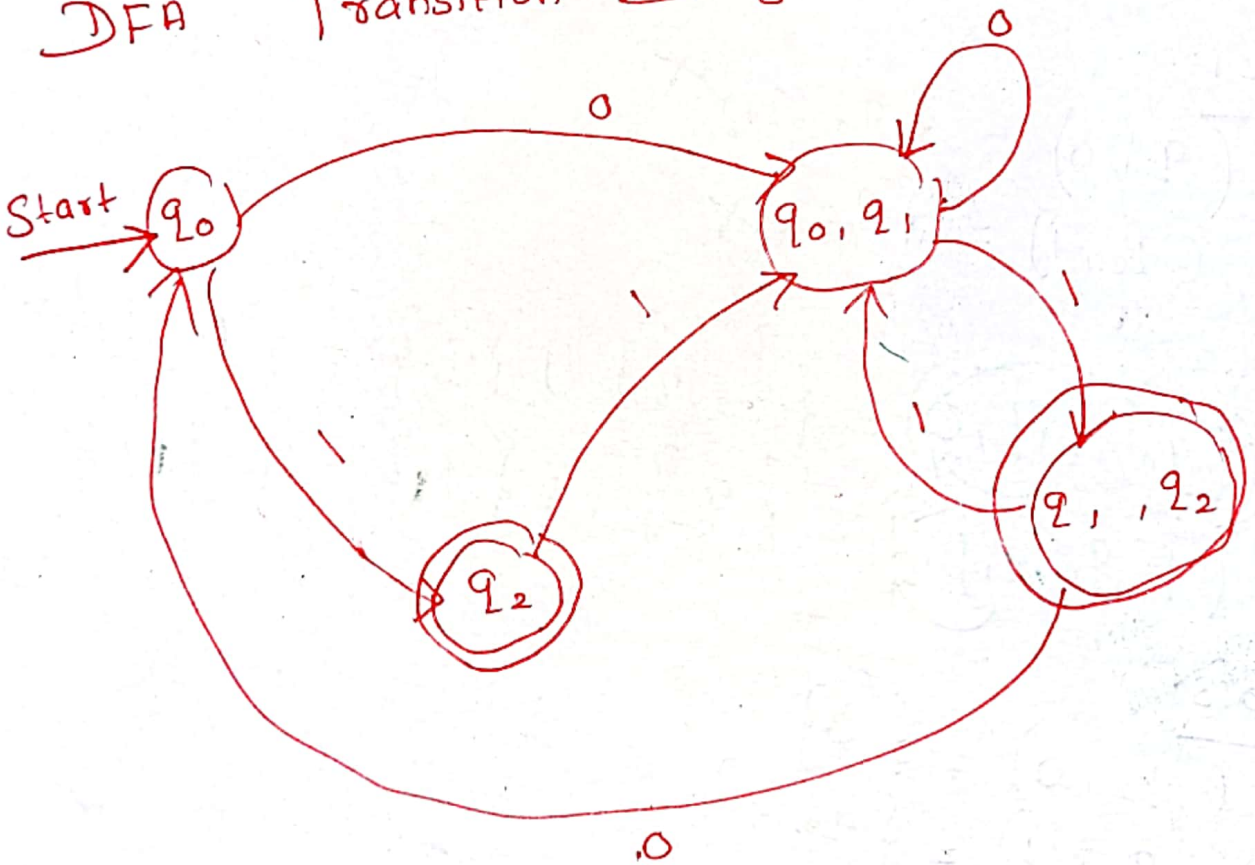
$\delta'([q_1, q_2], 0) = \{q_0\} \cup \{\phi\} = \{q_0\}$ ✓

$\delta'([q_1, q_2], 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$ ✓

DFA Transition Table:

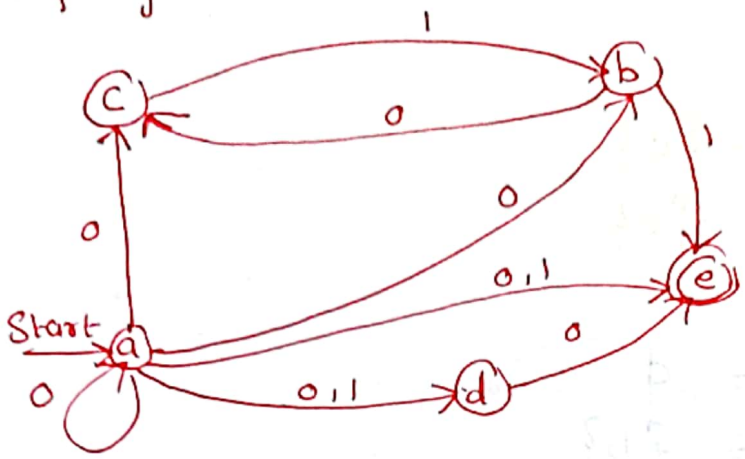
	δ	Input	
		0	1
→	q_0	$\{q_0, q_1\}$	q_2
*	q_2	ϕ	$\{q_0, q_1\}$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
*	$\{q_1, q_2\}$	q_0	$\{q_0, q_1\}$

DFA Transition Diagram:



1. Convert an NFA to DFA given NFA $M = \{\Sigma, Q, \delta, q_0, F\}$

$\Sigma = \{0, 1\}$, $Q = \{a, b, c, d, e\}$, $F = \{e\}$.



Solution:-

	States	Inputs	
		0	1
→	a	{abcde}	{de}
	b	{c}	{e}
	c	{ϕ}	{b}
	d	{e}	{ϕ}
*	e	{ϕ}	{ϕ}

The sets are:

{ϕ {a} {b} {c} {d} {e} {ab} {ae} ... {abcde}

$Q' = 2^Q = 2^5 = 32$

$\delta'([a], 0) = \{abcde\}$ — New1

$\delta'([a], 1) = \{de\}$ — New2

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

$\delta'([abcde, 1) = \{bde\}$ — New3

$\delta'([de], 0) = \{e\}$ — New4

$\delta'([de], 1) = \phi$

$\delta'([bde], 0) = \{ce\}$ — New5

$\delta'(bde], 1) = \{e\}$

$\delta'([e], 0) = \phi$

$\delta'([e], 1) = \phi$

$\delta'([ce], 0) = \phi$

$\delta'([ce], 1) = b$ — New State $\rightarrow b$

$=$

$\delta'([b], 0) = \{c\}$ — New State $\rightarrow 7$

$\delta'([b], 1) = \{e\}$

$=$

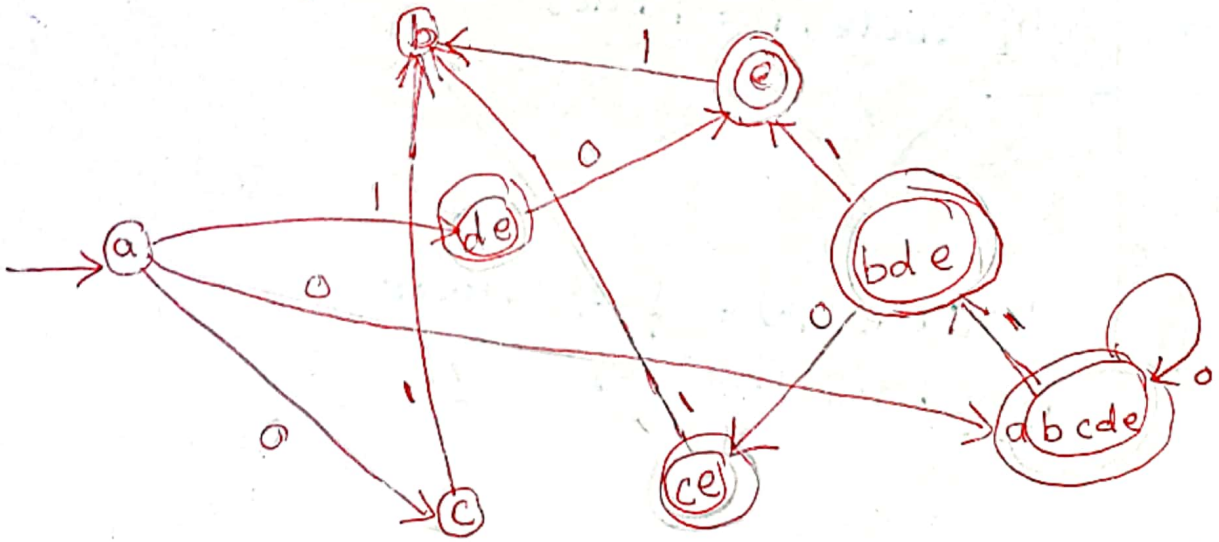
$\delta'([c], 0) = \phi$

$\delta'([c], 1) = \{b\}$

Transition Table of DFA :

	States	0	1
a →	a	abcde	de
	b	c	e
	e	ϕ	b
*	e	ϕ	ϕ
*	de	e	ϕ
*	bde	ce	e
*	abcde	abcde	abde
*	ce	ϕ	b

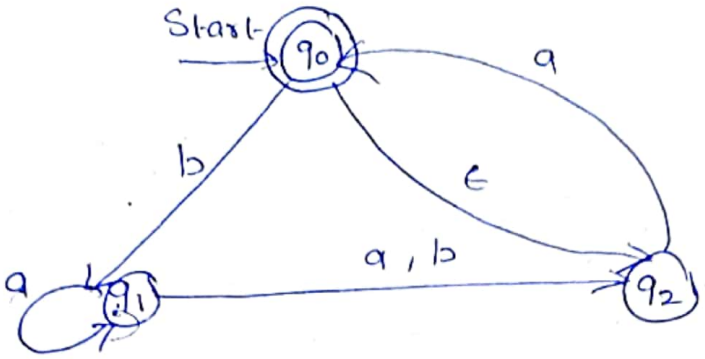
Transition diagram :



Part B-7

Convert the given NFA with epsilon to DFA

with Epsilon



States	Inputs		
	a	b	ε
q_0	ϕ	q_1	q_2
q_1	$\{q_1, q_2\}$	q_2	ϕ
q_2	q_0	ϕ	ϕ

Solution:

Step 1:

ε-closure of q_0

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

$\therefore$ A → New State.

Step 2:

move$(A, a) = \{q_0\}$ ——— A.

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

move$(A, b) = \{q_1\}$

$\therefore$ ε-closure$(A, b) = \{q_1\}$ ——— B

B – New State.

Step 3:

move$(B, a) = \{q_1, q_2\}$

ε-closure$(B, a) = \{q_1, q_2\}$ ——— C

move$(B, b) = \{q_2\}$

ε-closure$(B, b) = \{q_2\}$ ——— D.

$\therefore$ New State = C, D

Step 4:

move$(C, a) = \{q_0, q_1, q_2\}$

$\therefore$ ε-closure$(C, a) = \{q_0, q_1, q_2\}$ ——— E

move$(C, b) = \{q_2\}$ ——— D.

ε-closure$(C, b) = \{q_2\}$

E – New State.

Step 5:

move (D, a) = { q_0 }

ϵ-closure (D, a) = { q_0, q_2 } — A

move (D, b) = ϕ

∴ ϵ-closure(D, b) = ϕ

Step 6:

move(E, a) = { q_0 q_1 q_2 }

ϵ-closure (E, a) = { $q_0 q_1 q_2$ } — E

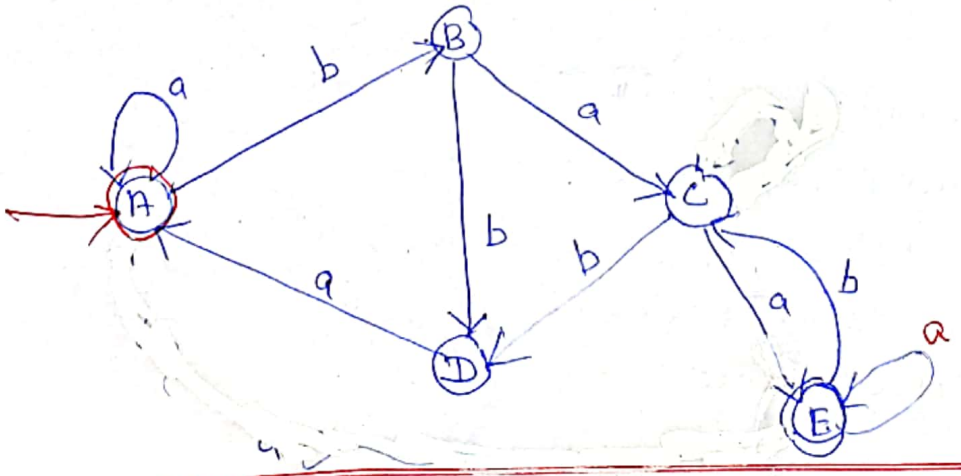
move (E, b) = { q_1, q_2 }

∴ ϵ-closure (E, b) = { q_1, q_2 } — C

Transition ~~diag~~ Table:

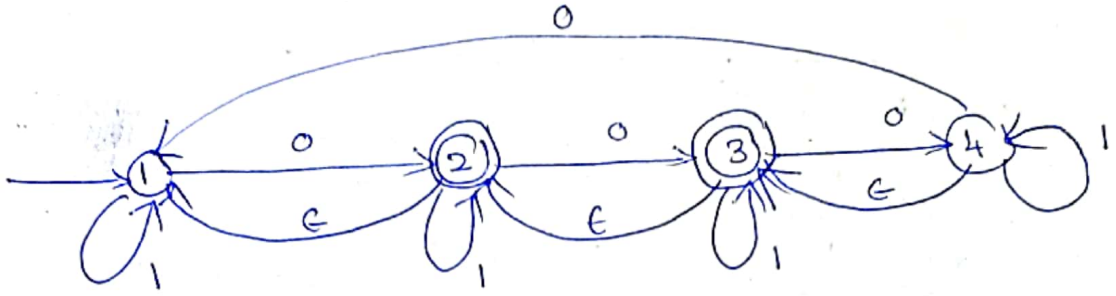
δ	a	b
→*A	A	B
B	C	D
C	E	D
D	A	ϕ
*E	E	C

Transition Diagram for DFA



8.

Convert the given NFA with epsilon to DFA



Solution:

ϵ-closure(1) = {1} ——— A

New State - A

Step 1:

move(A, 0) = {2}

ϵ-closure(move(A,0)) = {1, 2} ——— B

move(A, 1) = {1} ——— A

ϵ-closure(move(A,1)) = {1}

States	Inputs		
	0	1	ϵ
1	2	1	ϕ
2	3	2	1
3	4	3	2
4	1	4	3

Step 2: B -> New State

move(B, 0) = {2, 3}

ϵ-closure(move(B,0)) = {1, 2, 3} ——— C

move(B, 1) = {1, 2}

ϵ-closure(move(B,1)) = {1, 2} ——— B

C - New State

Step 3:

move(C, 0) = {2, 3, 4}

ϵ-closure(move(C,0)) = {1, 2, 3, 4} ——— D

move(C, 1) = {1, 2, 3}

ϵ-closure(move(C,1)) = {1, 2, 3} ——— C

D - New State.

Step 4:

move(D, 0) = {1, 2, 3, 4}

ϵ-closure(move(D,0)) = {1, 2, 3, 4} ——— D.

move(D, 1) = {1, 2, 3, 4}

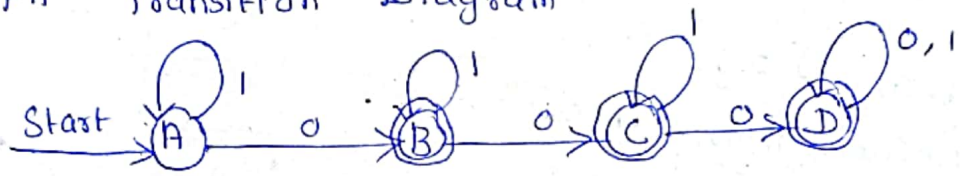
ϵ-closure(move(D,0)) = {1, 2, 3, 4} ——— D.

∴ No New State.

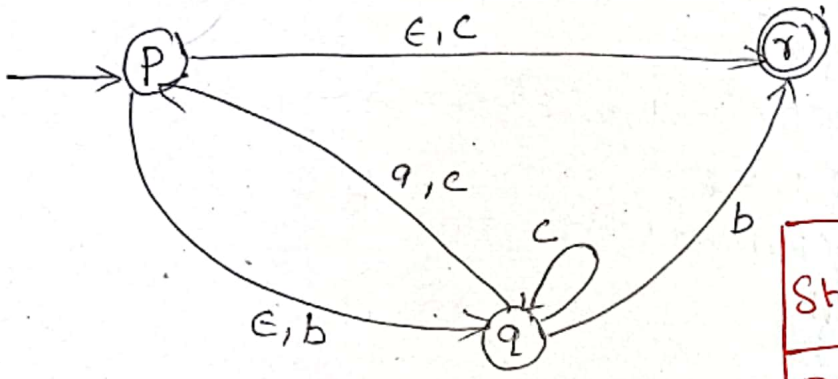
DFA Transition ~~Diagram~~ Table.

	New States &	Inputs 0	1
→	A	B	A
*	B	C	B
*	C	D	C
*	D	D	D

DFA Transition Diagram



part c 3. Convert the given NFA with epsilon to NFA without epsilon.



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

Solution:

ϵ-closure (p) = {p, q, r}

ϵ-closure (q) = {q}

ϵ-closure (r) = {r}

Step1:

move (p, a) = {p}

ϵ-closure move(p, a) = {p, q, r}

move (p, b) = {q, r}

ϵ-closure move (p, b) = {q, r}

move (p, c) = {p, q, r}

ϵ-closure move(p, c) = {p, q, r}

Step 2:

move (q, a) = {p}

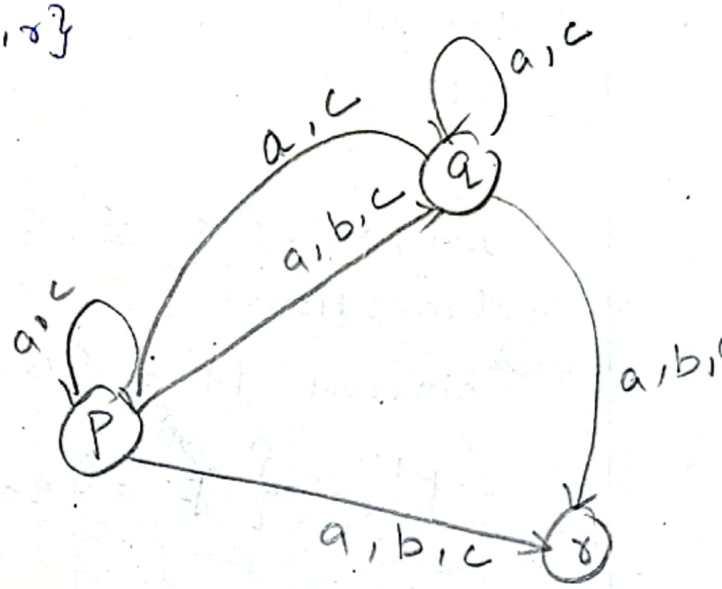
ε-closure move(q, a) = {p, q, r}

move (q, b) = {r}

ε-clos move(q, b) = {r}

move(q, c) = {p, q}

ε-clos move(q, c) = {p, q, r}



Step 3:

move(r, a) = {φ}

ε-clos-move(r, a) = {φ}

move(r, b) = {φ}

ε-clos-move(r, b) = {φ}

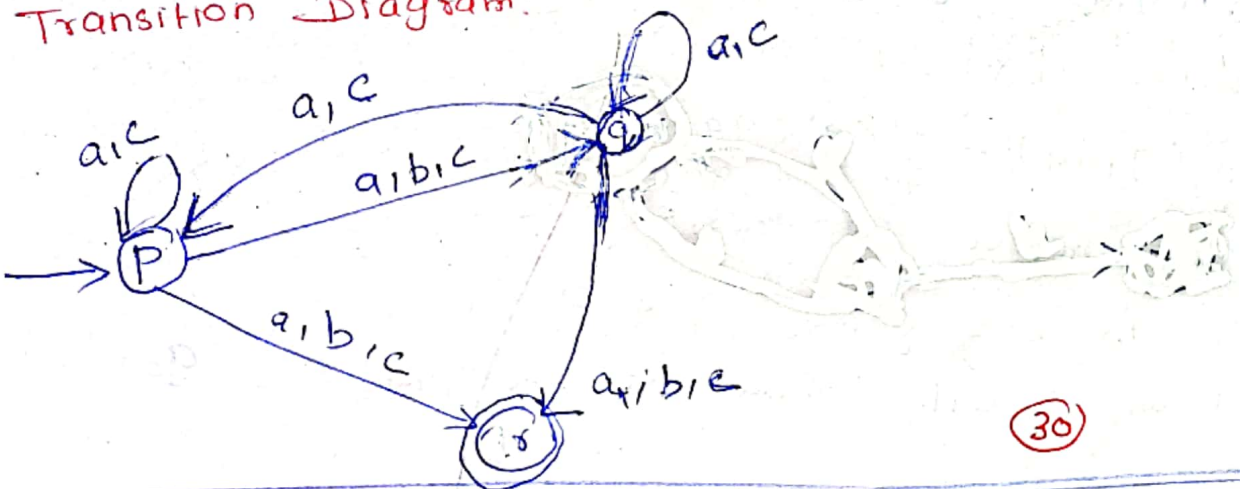
move(r, c) = {φ}

ε-closu-move(r, c) = {φ}

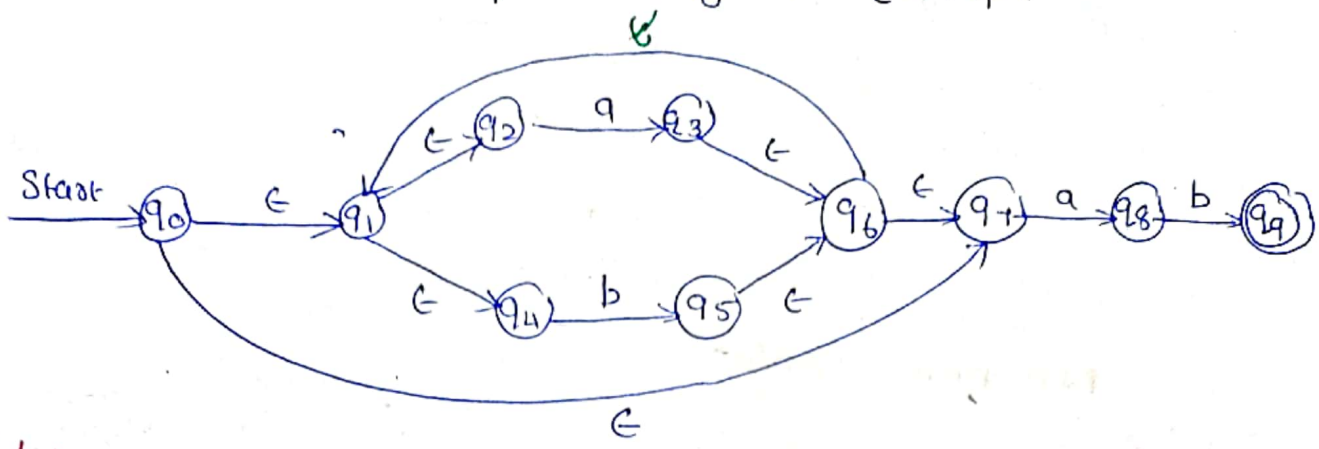
Transition Table:

	δ	a	b	c
→	p	p, q, r	q, r	p, q, r
	q	p, q, r	r	p, q, r
*	r	φ	φ	φ

Transition Diagram:



Construct the DFA for the given ∈-NFA.



Solution:

Step 1:

∈-closure (q0) = {q0, q1, q2, q4, q7} → A

∴ A New State.

Step 2: (for new state A)

move (A, a) = {q3, q8}

∈-closure (q3) = {q3, q6, q7, q1, q2, q4}

∈-closure (q8) = {q8}

∴ ∈-closure [move(A, a)] = {q1, q2, q3, q4, q6, q7, q8} — B.

move (A, b) = {q5}

∈-closure (q5) = {q5, q6, q1, q2, q4, q7}

∴ ∈-closure [move (A, b)] = {q1, q2, q4, q5, q6, q7} — C.

∴ B, C — New States.

Step 3: (for new state B)

move (B, a) = {q3, q8}

∴ ∈-closure [move(B, a)] = {q1, q2, q3, q4, q6, q7, q8} — B.

move (B, b) = {q5, q9}

∈-closure (q3) = {q3, q6, q7, q1, q2, q4}

{q8} = {q8}

∈-closure [move [B, a]] = {q1, q2, q4, q5, q6, q7, q9} — D.

D - new state

∈- q5 = {q5}.

∈- {q9} = {q9}

Step 4: (for new state C):

move (C, a) = {q3, q8}

∈-closure [move(C, a) = {q1, q2, q3, q4, q6, q7, q8} — B

move (C, b) = {q5}

∈-closure [mov (C, b)] = {q1, q2, q4, q5, q6, q7} → C

No new states

Step 5 (for the new State D):

move (D, a) = {q3, q8}

ε-closure [move (D, a)] = {q1, q2, q3, q4, q6, q7, q8} → B

move (D, b) = {q5}

ε-closure [move [D, b)] = {q1, q2, q4, q5, q6, q7} → C

No new States

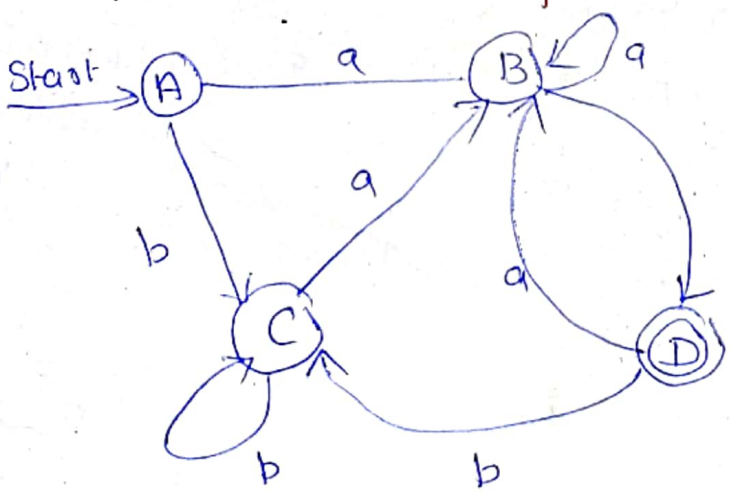
Step 6:

* Construction terminates, because no new states are generated from Step 4 and Step 5
* From the above steps (1 to 5) transition table. and transition diagram of DFA are constructed.

Transition Table:

New States	a	b
→ A	B	C
B	B	D
C	B	C
(D)	B	C

Transition diagram for DFA



part B
(5) If L is accepted by an NFA with ϵ- transition then show that L is accepted by an NFA without ϵ-transition.

Solution:

Proof:

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

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

$$F' = \begin{cases} F = \{q_0\} & \text{if } \epsilon\text{-closure contains of F state} \\ F & \text{otherwise.} \end{cases}$$

M' is a NFA without ϵ moves. The δ' function can be denoted by δ'' with some input. For example, $\delta'(q,a) = \delta''(q,a)$ for some q in Q and a from Σ. We will apply the method of induction with input X. The X will not be ϵ because.

$$\delta'(q, \epsilon) = \{q_0\}$$

$\delta''(q, \epsilon) = \epsilon$-closure (q_0). Therefore we will assume length of string to be 1.

Basic: $|x| = 1$. Then x is a symbol a.

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

Induction: $|x| > 1$ Let $x = wa$

$$\delta'(q, wa) = \delta'(\delta'(q_0, w), a)$$

by inductive hypothesis.

$$\delta'(q_0, w) = \delta''(q_0, w) = P.$$

now we will show that $\delta'(p,a) = \delta(q_0, wa)$

Part-B
5

but $\delta'(p,a) = \bigcup_{q \text{ in } p} \delta'(q,a) = \bigcup_{q \text{ in } p} \delta''(q,a)$

as

$p = \delta''(q_0, w)$

we have $\bigcup_{q \text{ in } p} \delta''(q,a) = \bigcup_{q \text{ in } p} \delta''(q_0, wa)$

Thus by definition δ''

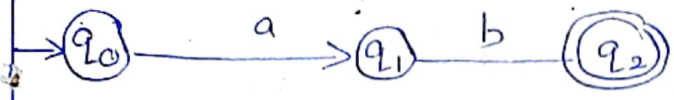
$$\boxed{\delta'(q_0, wa) = \delta''(q_0, wa)}$$

Rule for conversion:

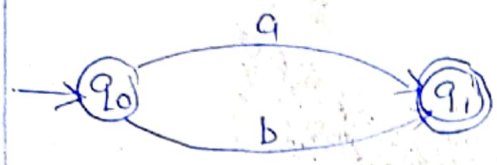
$\delta'(q,a) = \epsilon\text{-closure}(\delta(\delta'(q,\epsilon),a))$

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

Convert DFA to RE

1) 

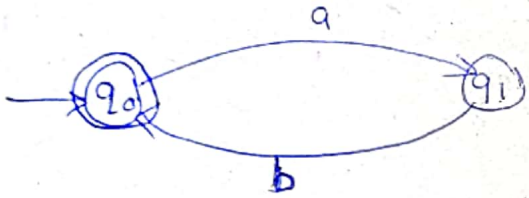
RE = ab

2. 

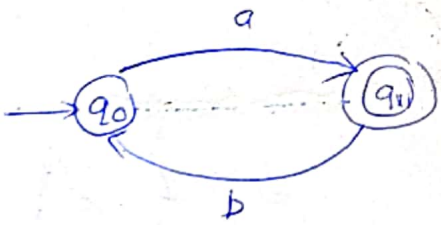
RE = a + b

3. 

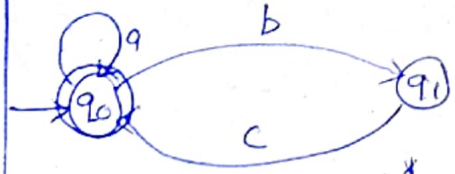
RE = ab^*c

4. 

RE = $(a.b)^*$

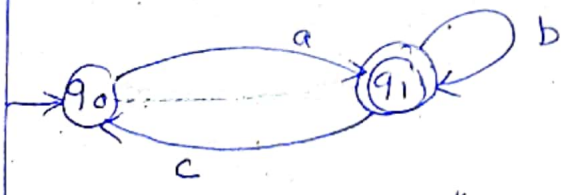
5. 

RE = $a(ba)^*$

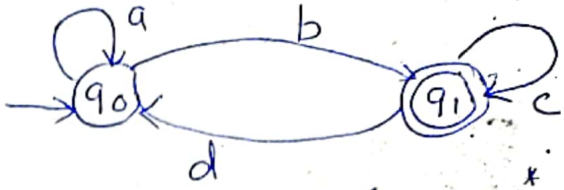
6. 

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

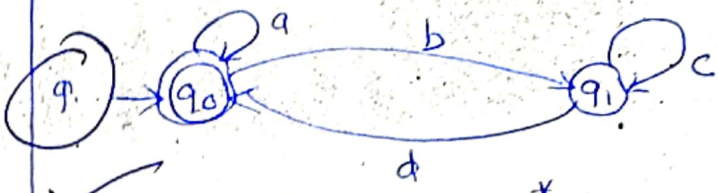
$= (a + bc)^*$

7. 

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

8. 

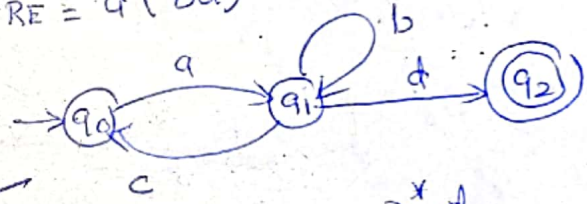
RE = $a^*b(c^* + da^*b)^*$



$= RE = (a + bc^* d)^*$

10) 

RE = $a(ba)^*c$

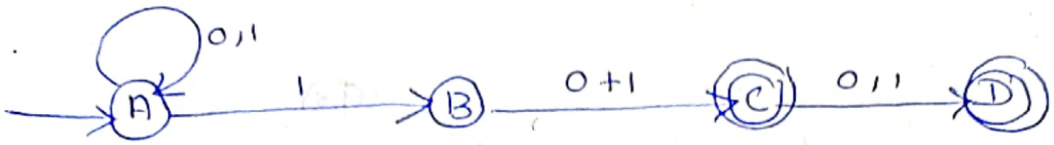
11) 

RE = $a(b + ca)^* d$

$(ab^* c)^* ab^* + d$

Part B
10

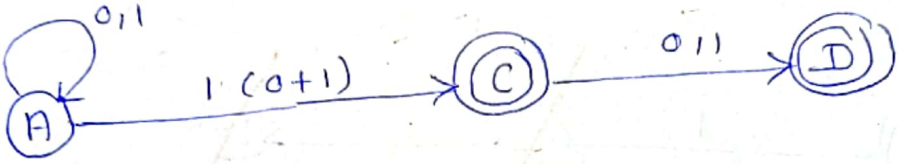
Convert the following NFA into regular expression.



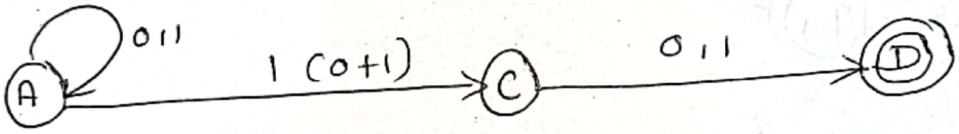
Solution:

Part B
10

First we eliminate state B and find out the RE

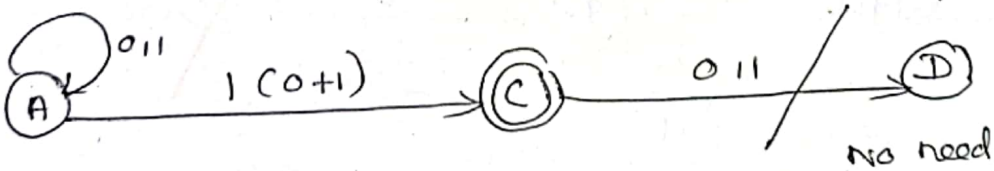


Since there are two final states we can take C to eliminate first and having D as the final state. Then.



$$\therefore RE_1 = (0+1)^* 1(0+1)(0+1)$$

Now C as the final state. Then.



No need bcz already we reached final state.

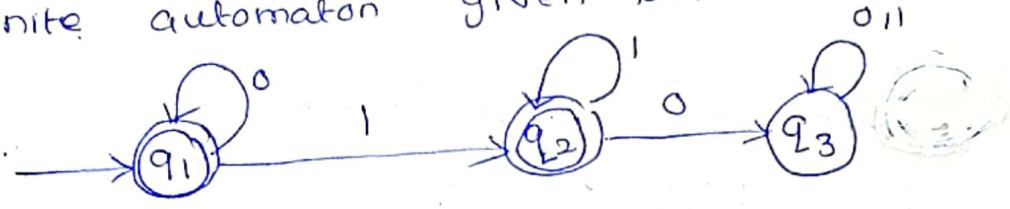
$$\therefore RE_2 = (0+1)^* 1(0+1)$$

∴ The required regular expression is $RE_1 + RE_2$

$$\therefore \boxed{RE = (0+1)^* 1(0+1)(0+1) + (0+1)^* 1(0+1)}$$

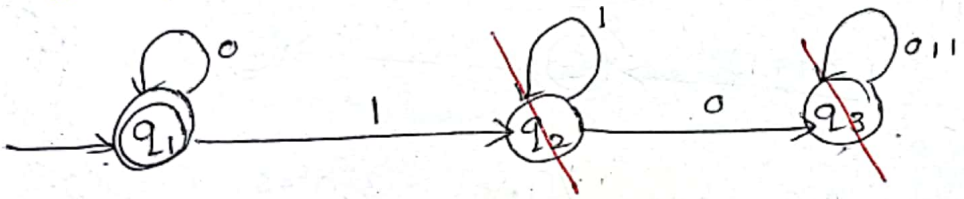
part B
15

Find the regular expression corresponding to the finite automaton given below. (Using State elimination method)

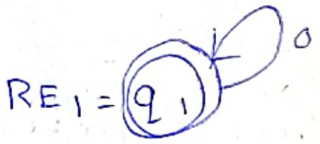


Solution:

There are two final states we can take q_1 as the final state. Then

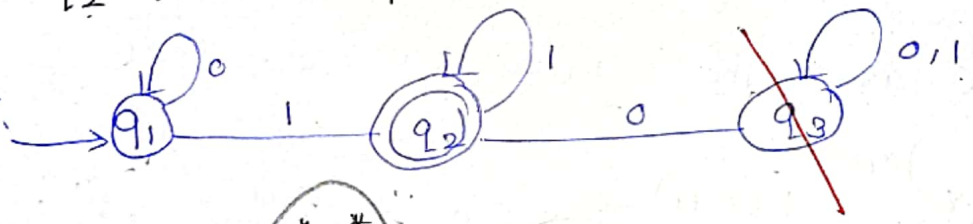


q_1 is the final state ∴ remove q_2 & q_3 state.



$RE1 = 0^*$

Now q_2 as the final state:



$\therefore RE_2 = 0^* 1 1^*$

$\therefore RE = RE_1 + RE_2$

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

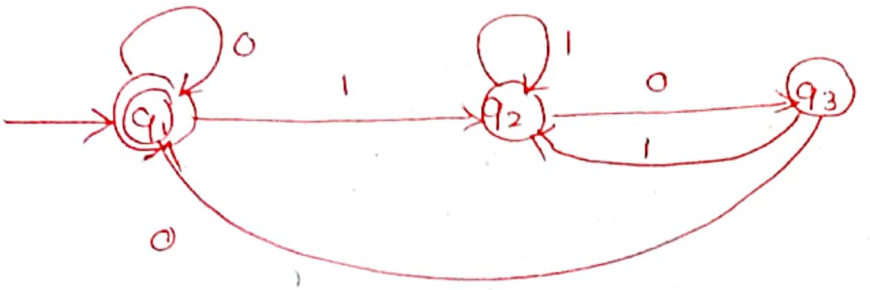
$= 0^* (\epsilon + 1 1^*)$

$\boxed{RE = 0^* 1^*}$

$\boxed{\epsilon + RR^* = R^*}$

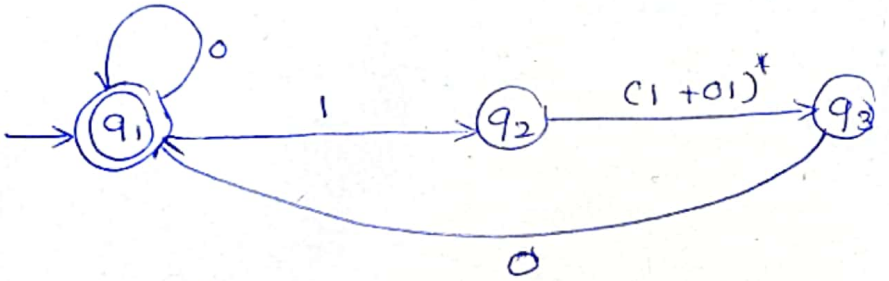
part c 4

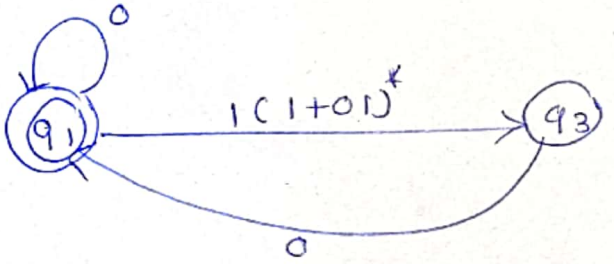
Find the regular expressions corresponding to the finite automatons given below. Using State elimination method



Solution: Step 1:

Eliminate q_2





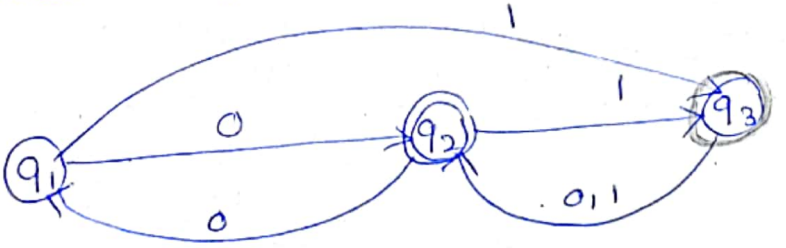
Step 2:

Eliminate q_3

$\therefore$ RE = $(0+1(1+01)^*00)^*$

part c 5

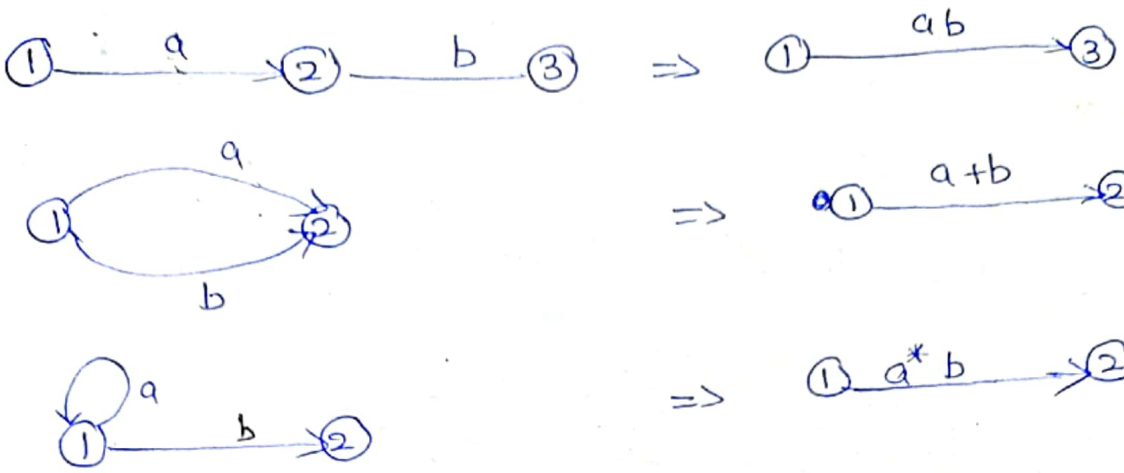
Find the regular expressions corresponding to the finite automatons given below.



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

DFA to Regular expression:

(State Elimination method)



There are three basic operations of regular expression.

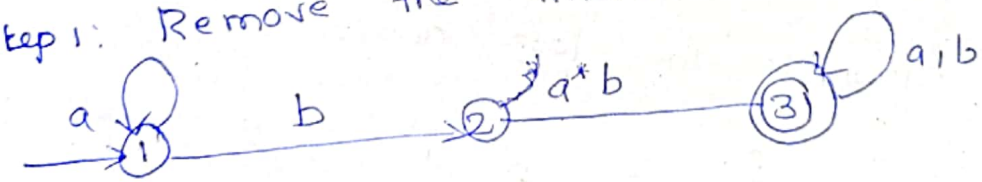
1. Concatenation
2. parallel edge
3. Closer things.

=> Eliminate all the intermediate State.

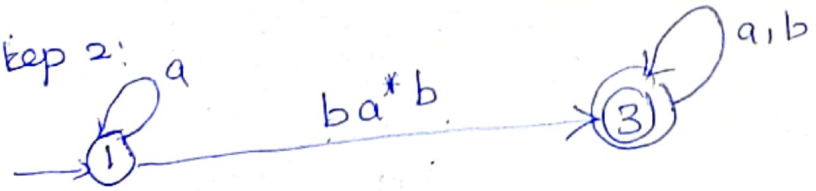
Eg)

Solution:

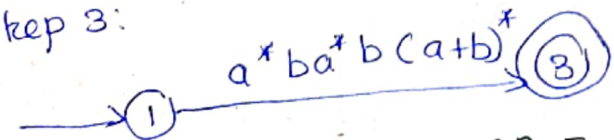
Step 1: Remove the intermediate State.



Step 2:



Step 3:

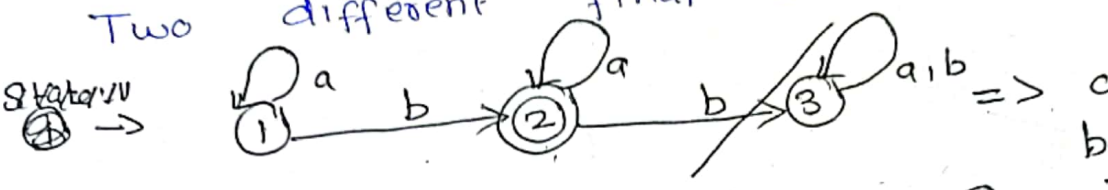


Final Regular expression = $a^* ba^* b(a+b)^*$

Example 2: Convert DFA to RE

Solution:

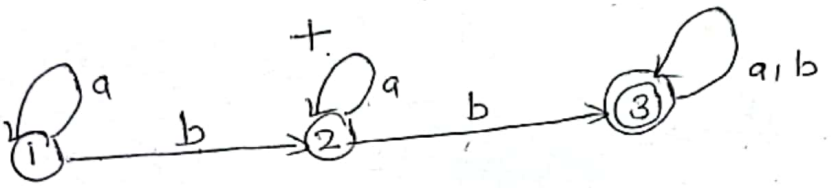
Two different final State.



$\Rightarrow a^* b a^*$

because after reaching the final state no effect in the next state.

+



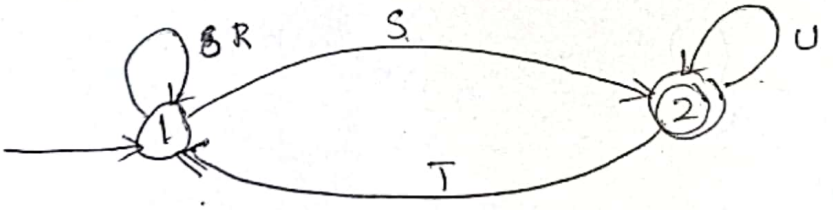
$$a^* b a^* b (a+b)^*$$

Result:

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

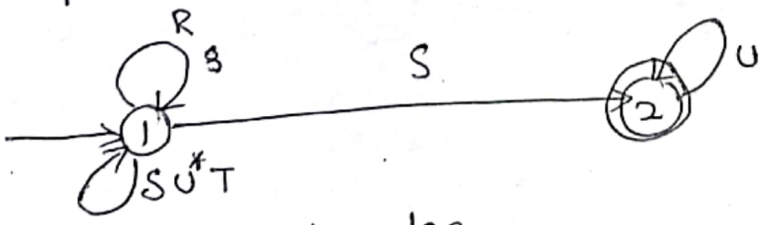
Example : 3:

Convert DFA to RE

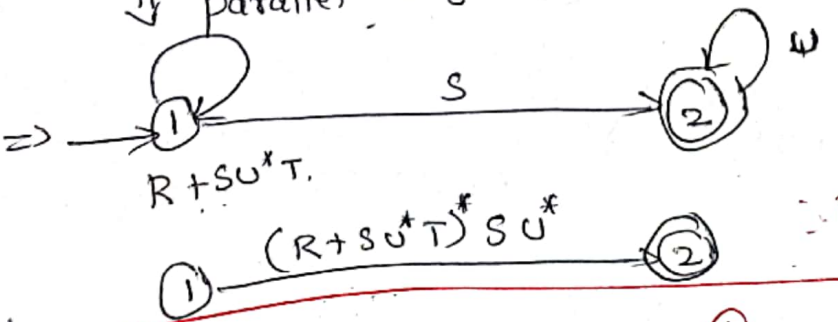


Solution:

First eliminate the loop only.



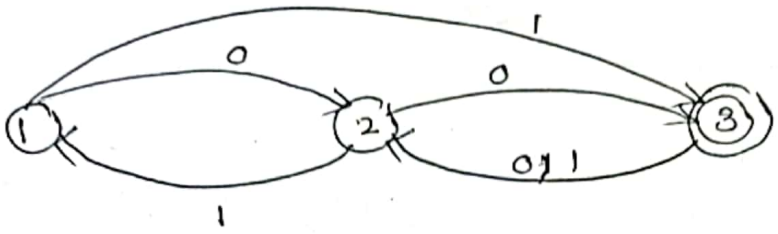
⇓ parallel edge.



$\therefore RE = (R + SU^*T)^* SU^*$

Example 4:

Convert DFA to RE(State Elimation method)



0,1 => 0+1
0,1 => 0/1

Solution:

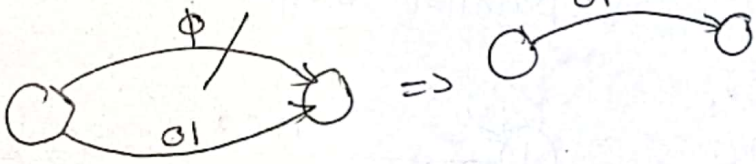
First eliminate the intermediate state.

Step: Find the direct path from 1 to 3.

Direct path			path through 2
11	–	$\phi + 01$	$1 \xrightarrow{0} 2 \xrightarrow{1} 1$
13	–	$1 + 00$	$1 \to 2 \to 3$
31	–	$\phi + (0+1)1$	$3 \to 2 \to 1$
33	–	$\phi + (0+1)0$	$3 \to 2 \to 3$

Eg:

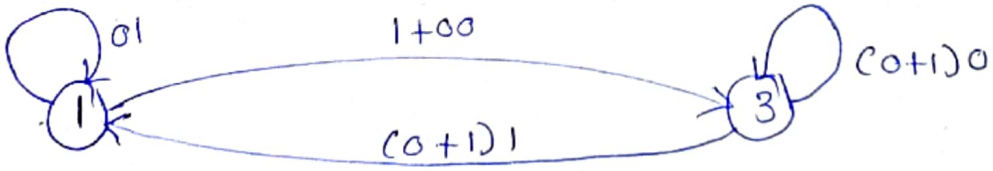
$\phi + 01 \Rightarrow$



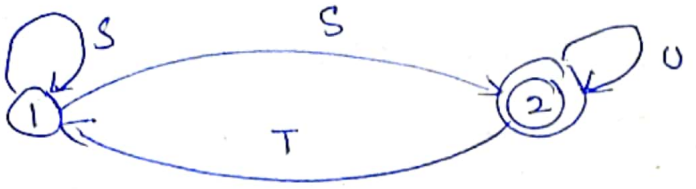
∴ Direct path o/p

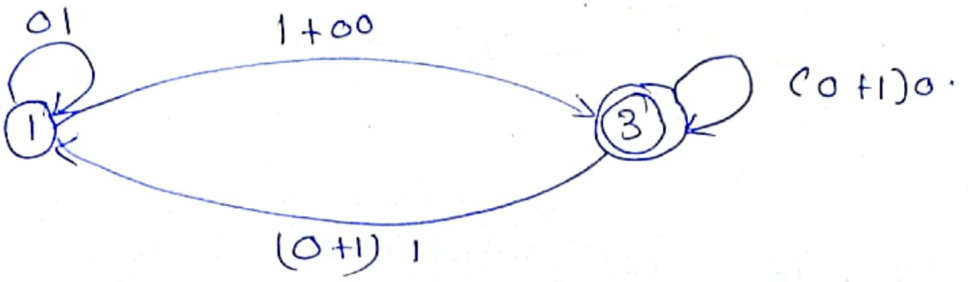
11	–	01
13	–	$1 + 00$
31	–	$(0+1)1$
33	–	$(0+1)0$

Redefined the transition diagram.
Based on the eliminating state.



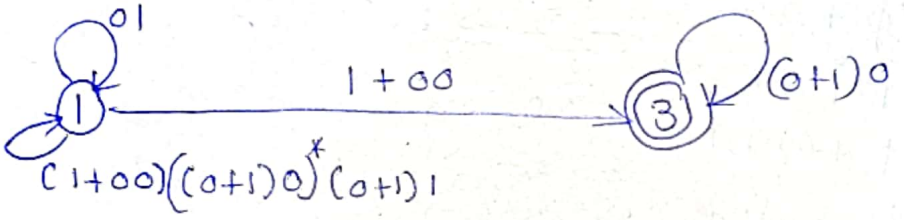
It is ~~natural~~ Similar to



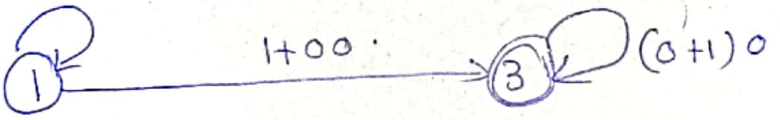


Step 1:

Eliminate the loop.



⇓ parallel edge.



$(01) + (1+00)((0+1)0)^*(0+1)1$

⇓

$$RE = [01 + (1+00)((0+1)0)^*(0+1)1]^* (1+00)((0+1)0)^*$$

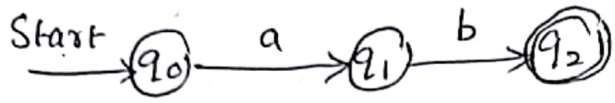
part B

13

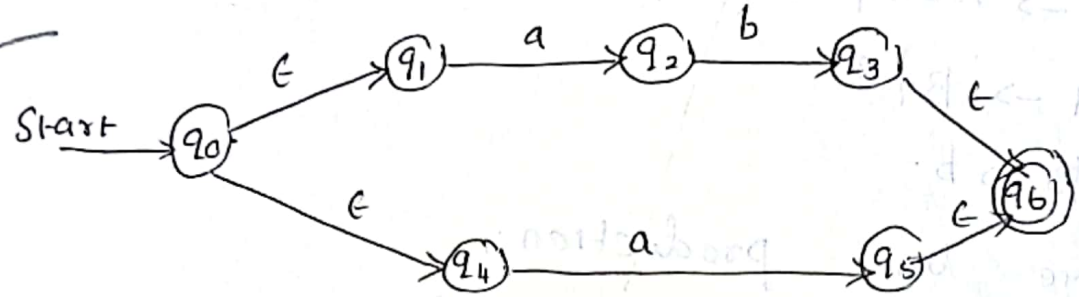
Construct Finite Automata equivalent to the regular expression $(ab + a)^*$.

Solution:

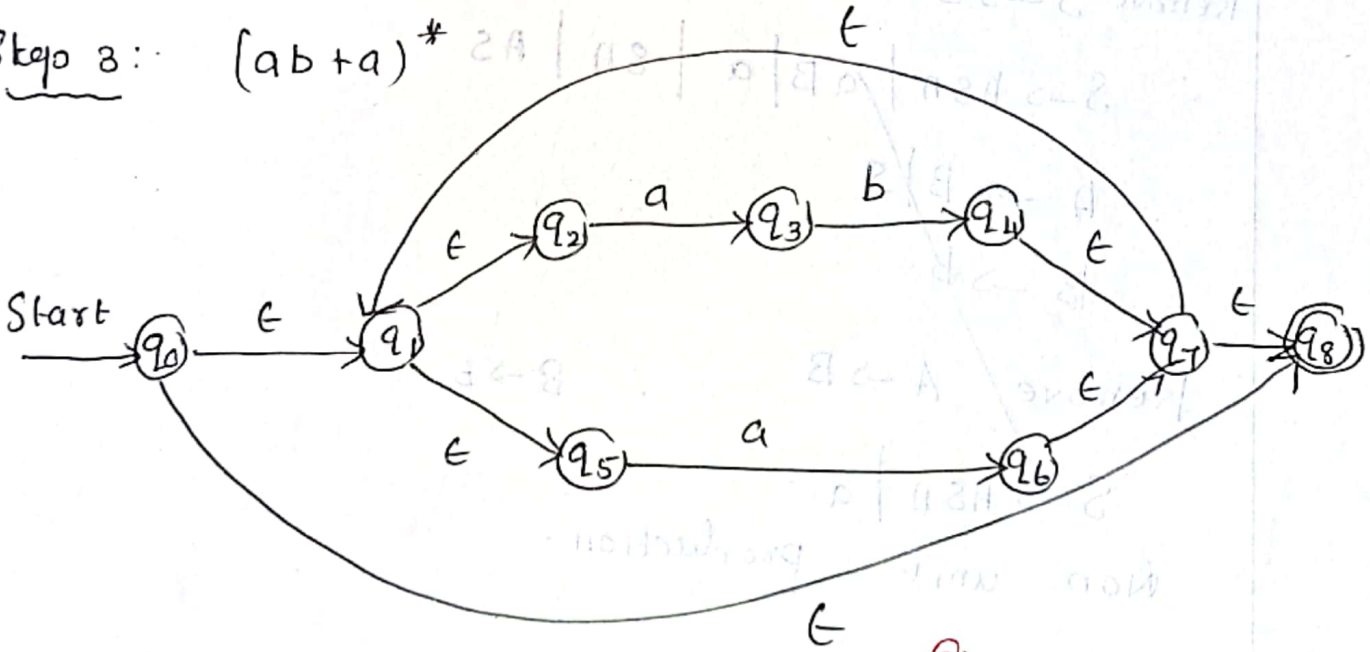
Step 1: ab



Step 2: ab + a



Step 3: $(ab + a)^*$



(9)

Convert the following CFG into CNF

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

(9)

(9)

part B (16)

Write a regular expression for set of strings that consists of alternating 0's and 1's

Solution:

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

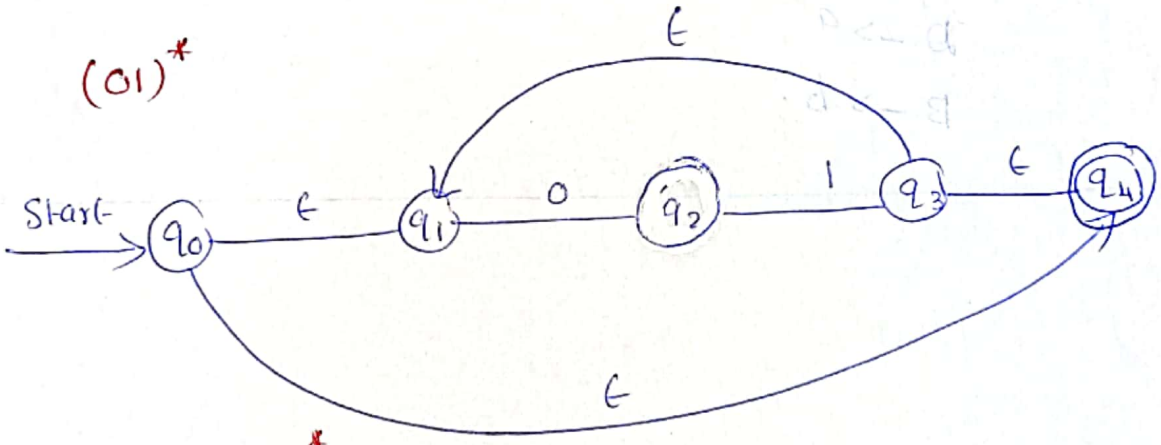
$$RE = (\Sigma + 1)(01)^* (\epsilon + 0)$$

or

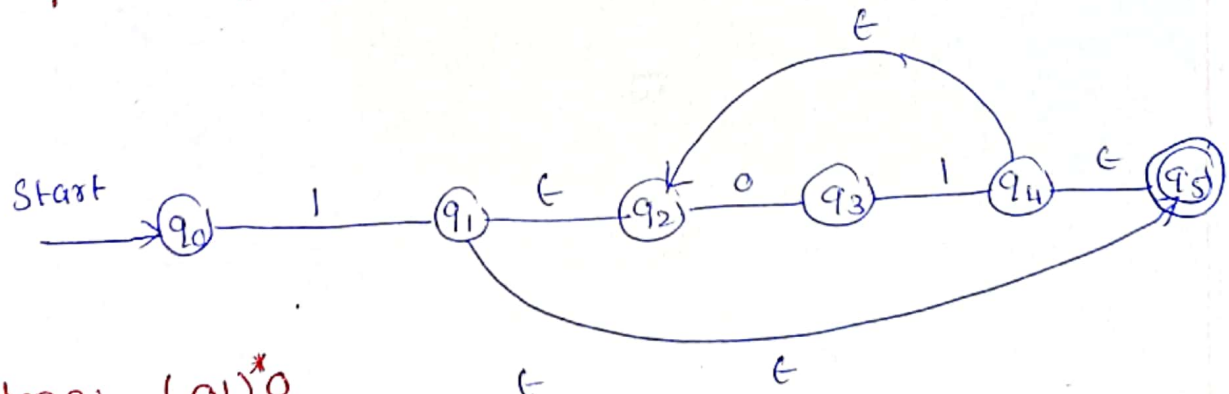
$$RE = (01)^* + 1(01)^* + (01)^*0 + 1(01)^*0$$

Step 1: $(01)^*$

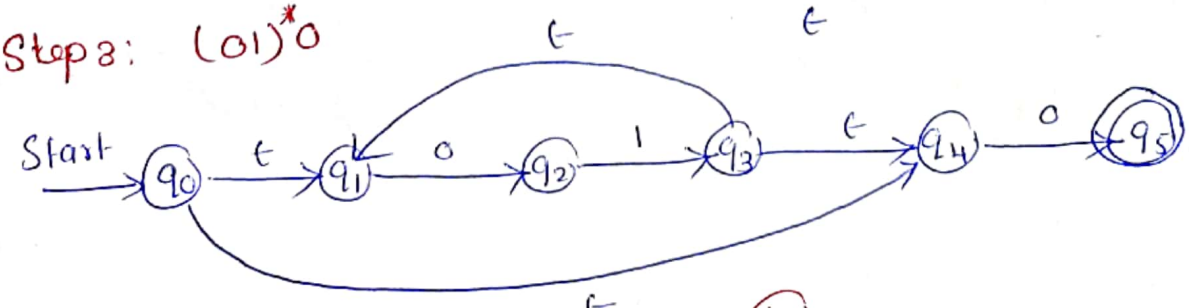
$(01)^*$



Step 2: $1(01)^*$



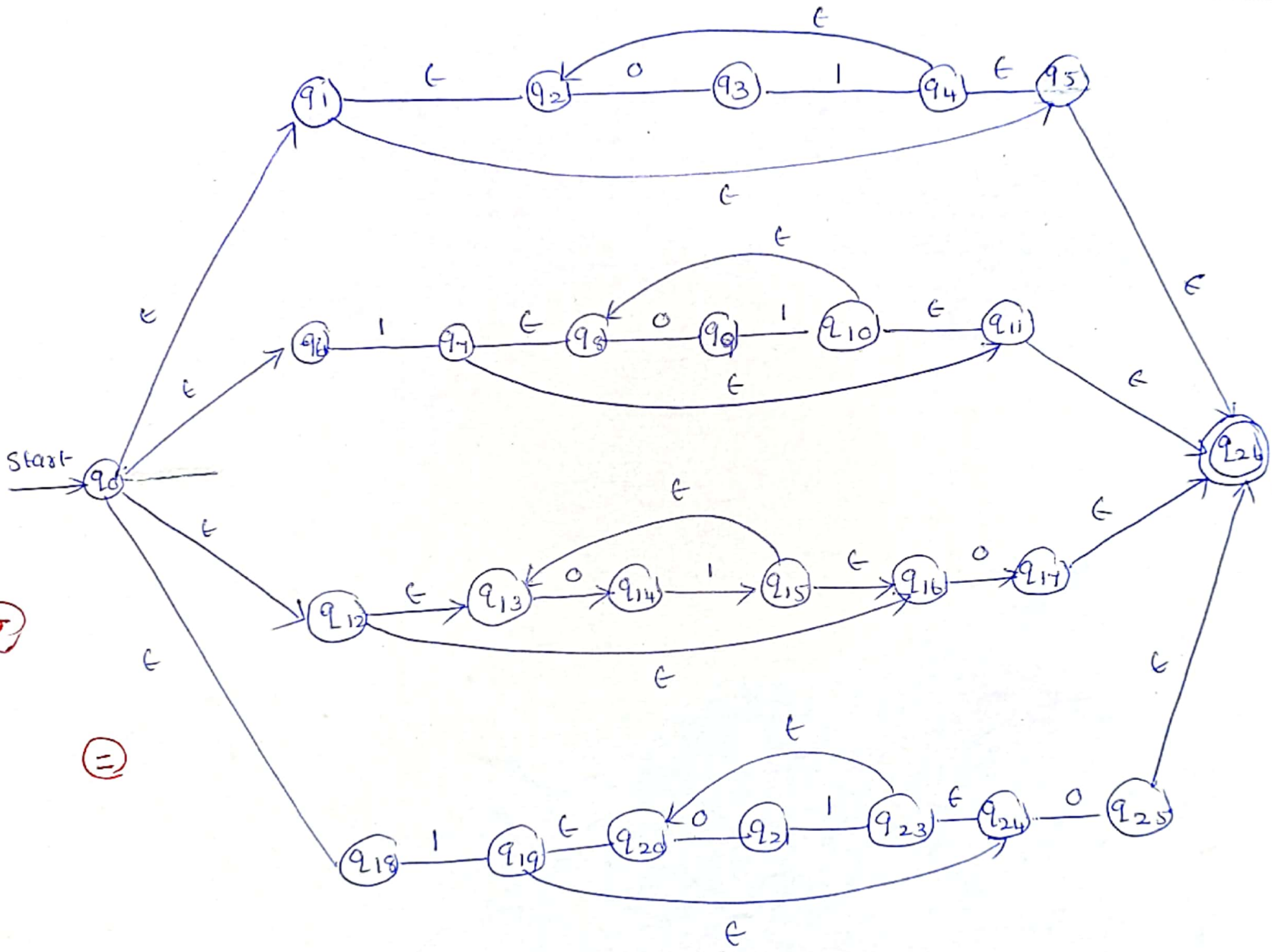
Step 3: $(01)^*0$



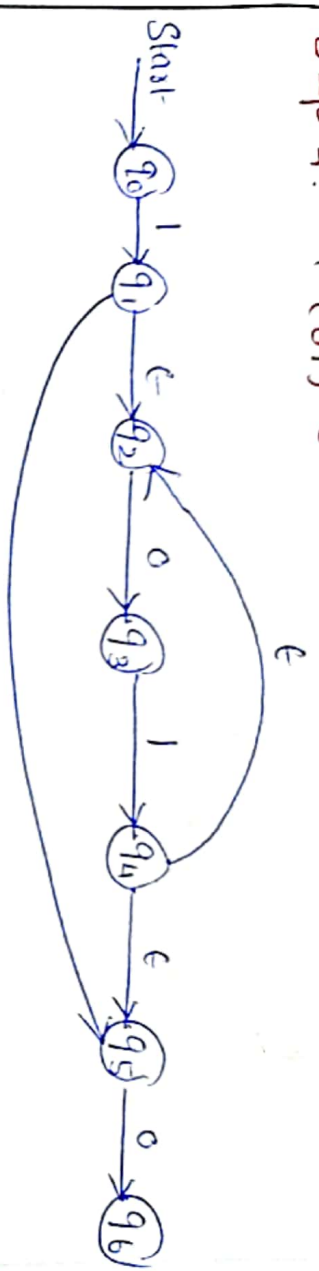
10

5

Step 5: $(01)^* + 1(01)^* + (01)^*0 + 1(01)^*0$



Step 4: $0^*(10)^*$



2. Construct NFA equivalent to

(i) $(0+1)^*(00+11)$

(ii) $((10)+(0+1)^*)01$

(iii) 1^*0+0

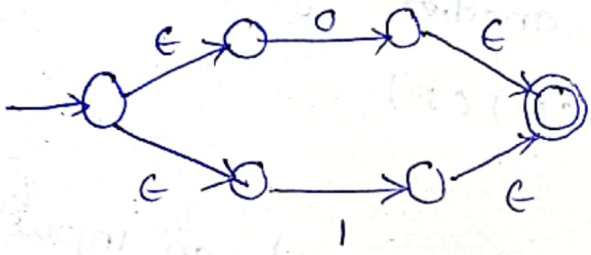
(iv) $((01+001)^*0^*)^*$

Solution:

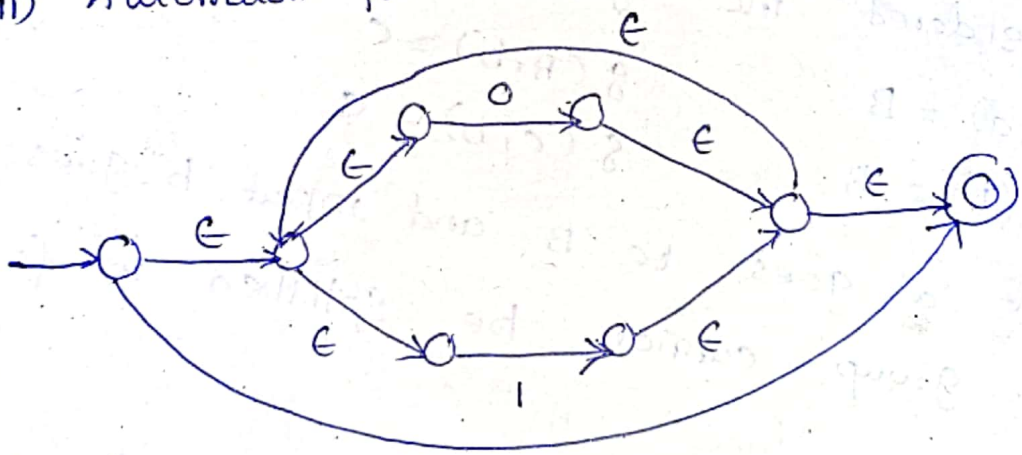
(i) $(0+1)^*(00+11)$

NFA with ϵ transitions:

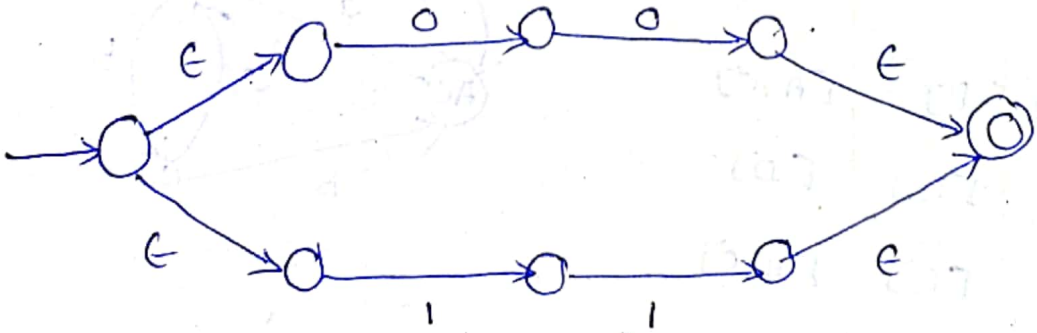
(i) Automaton for $(0+1)$



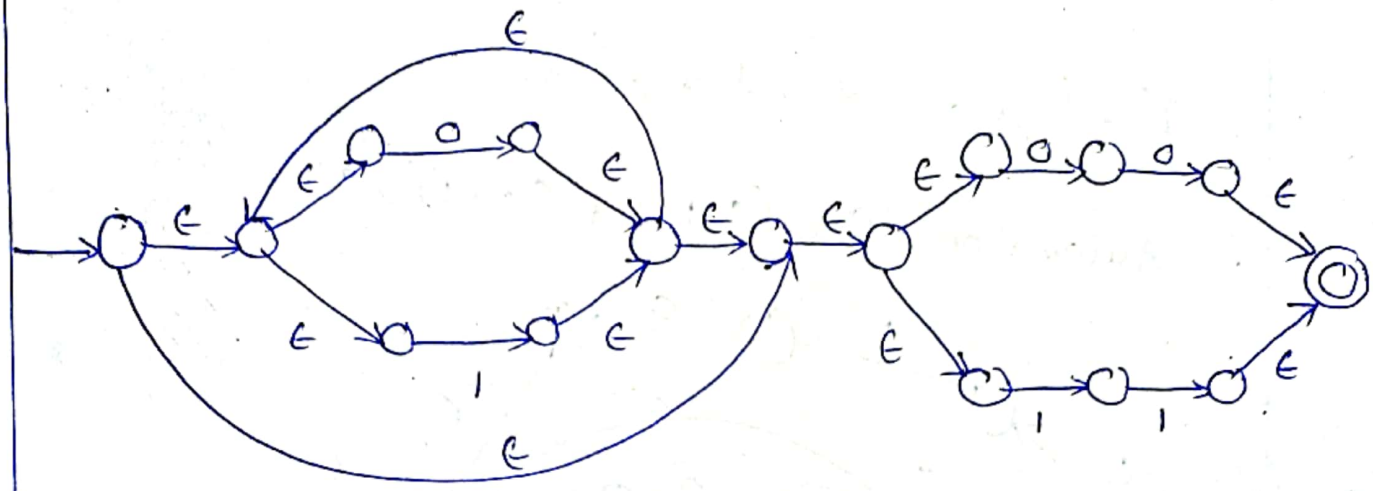
(ii) Automaton for $(0+1)^*$



(iii) Automaton for $(00+11)$



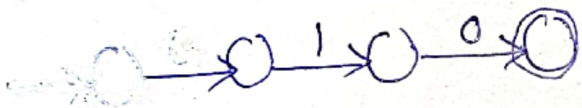
IV Automaton for $(0+1)^*(00+11)$



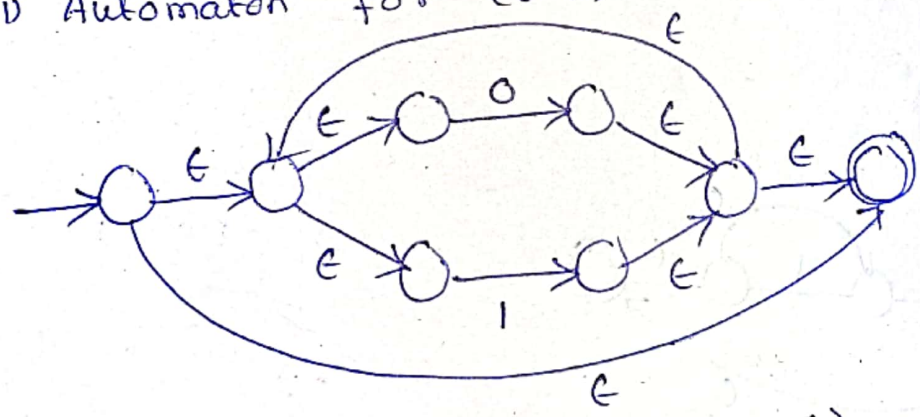
(ii) $((10) + (0+1)^*)01$

Solution:

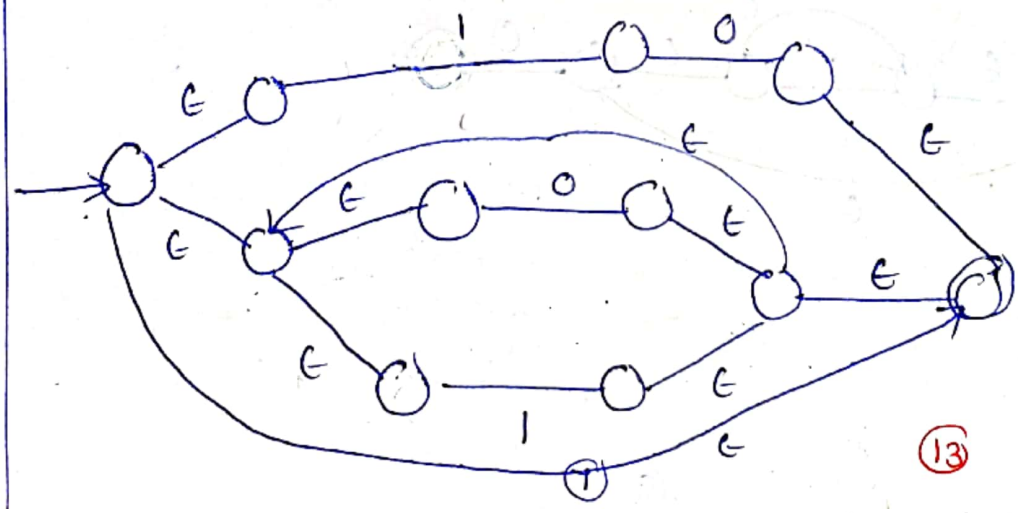
(i) Automaton for (10)



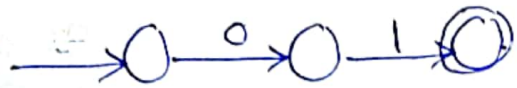
(ii) Automaton for $(0+1)^*$



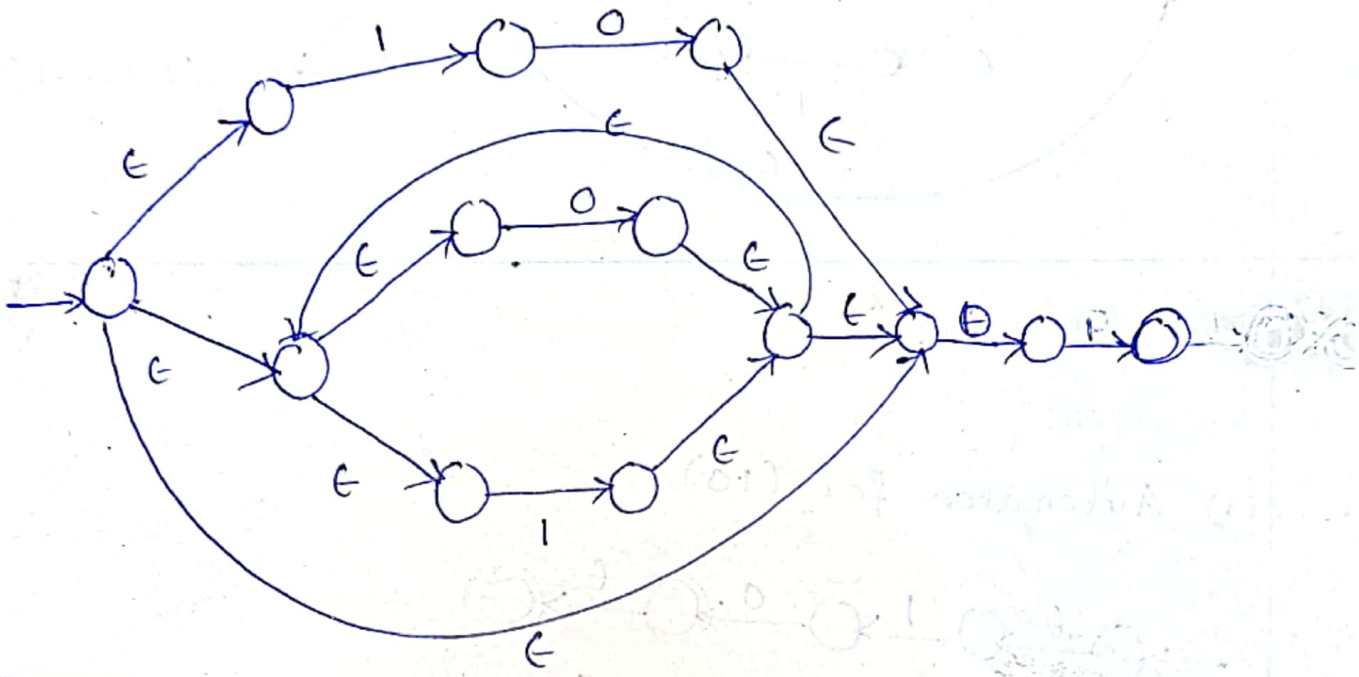
(iii) Automaton for $(10) + (0+1)^*)$



(iv) Automaton for 01



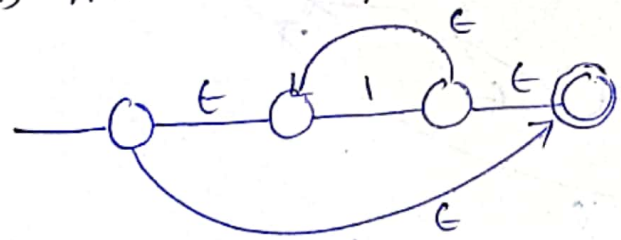
(v) Automaton for $(C 10) + (0+1)^*)01$



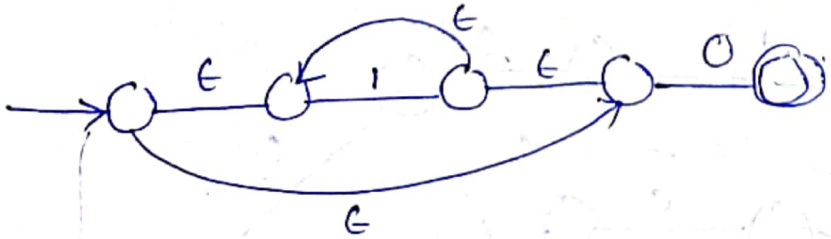
(iii) $1^*0 + 0$

Solution:

(i) Automaton for 1^*

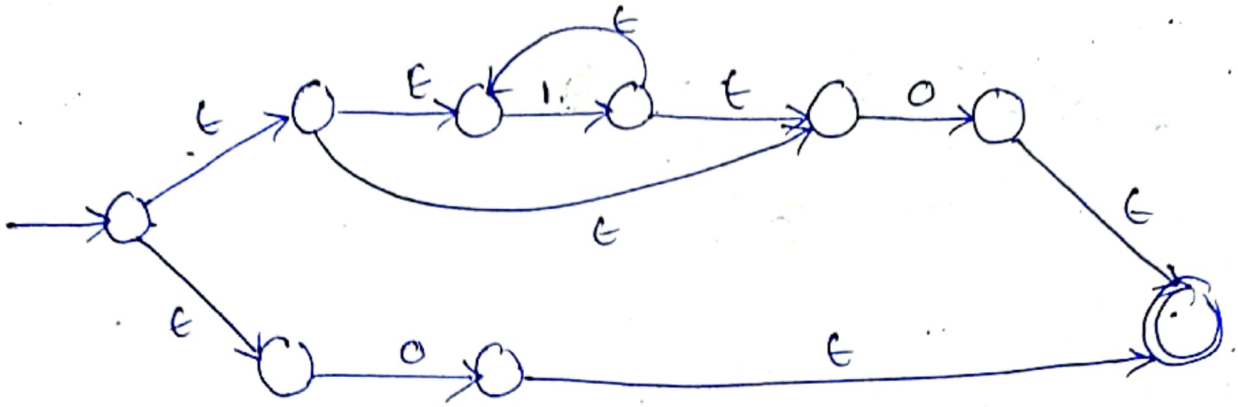


(ii) 1^*0 Automaton



(14)

(iii) Automaton for 1^*0+0

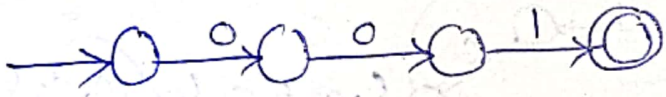


IV $((01+001)^*0^*)^*$

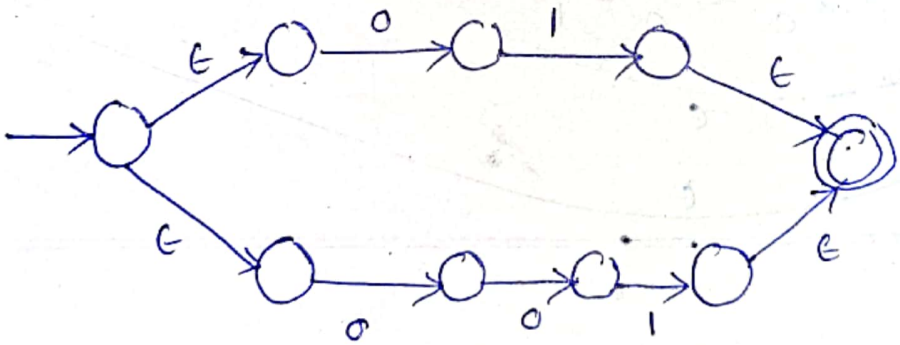
Solution:

(i) Automaton for 01

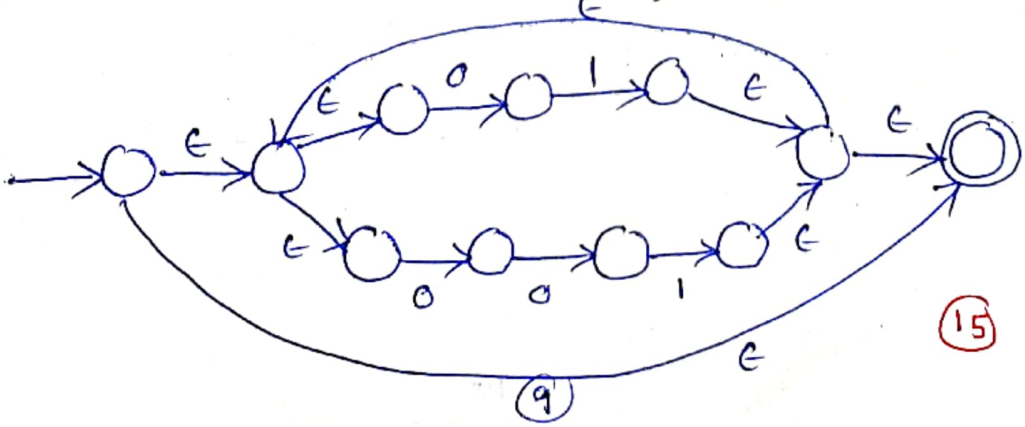
(ii) Automaton for 001



(iii) Automaton for $01+001$



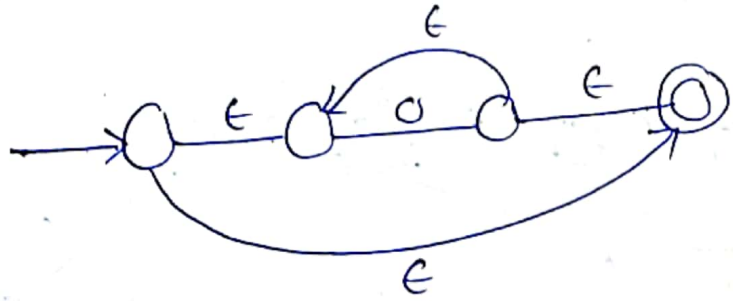
IV Automaton for $(01+001)^*$



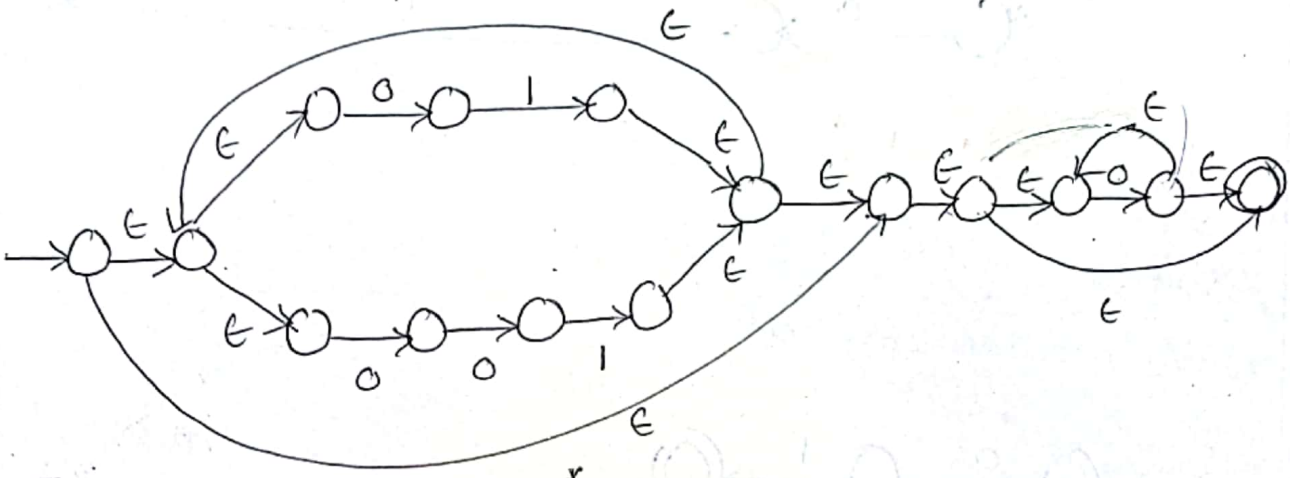
15

9

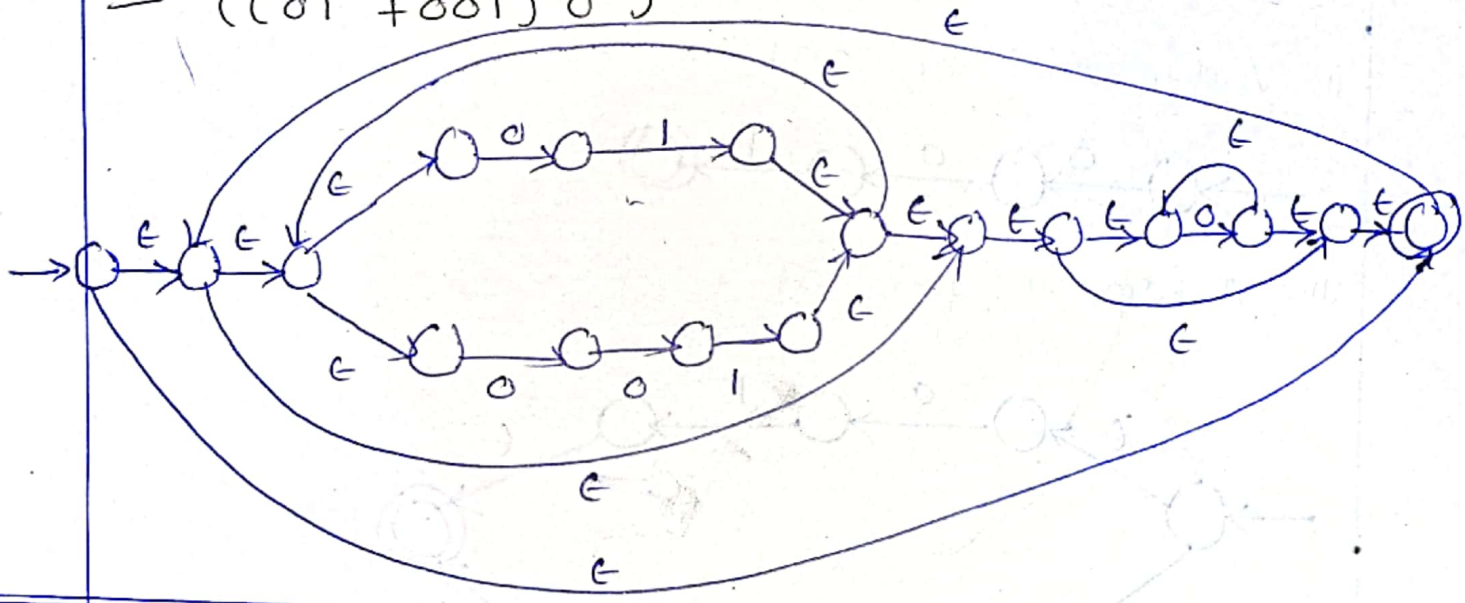
V Automaton for 0^*.



VI Automaton for $((01+001)^*0^*)$

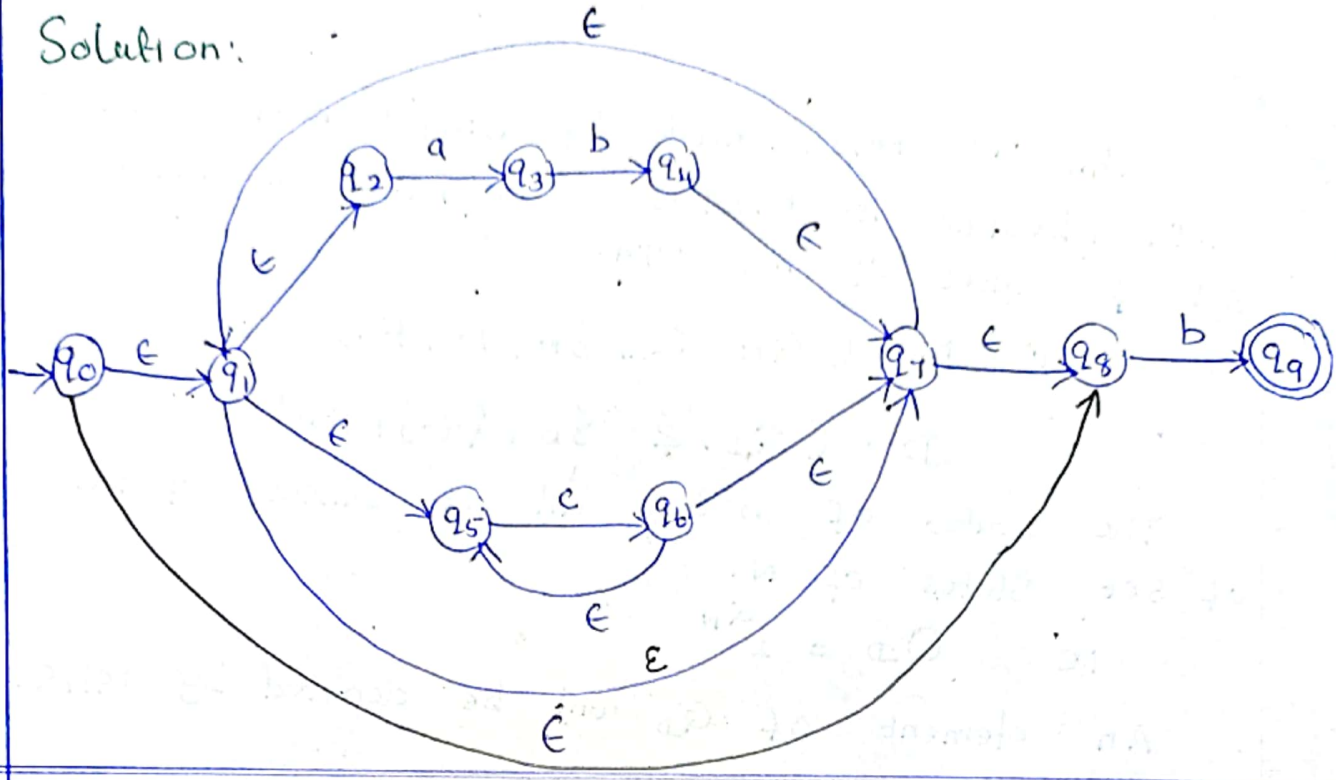


VII $((01+001)^*0^*)^*$



14 Part B Construct transition diagram of a finite automaton corresponding to the regular expression $(a|b+c^*)^* b$

Solution:



2 Show that if L be a set accepted by an NFA then there exists a DFA that accepts L.

Proof:

This is implemented by using 'subset construction' method, because it involves constructing all subsets of the set of states of the NFA.

Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

The states of D are all the subsets of the set of set states of N.

ie $Q_D = 2^{Q_N}$

An element of Q_D will be denoted by $\{q_1, q_2, \ldots q_i\}$ where $q_1, q_2, \ldots q_i$ are in Q.

Define

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

if and only if

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

δ_D is computed by applying δ_N to each state of Q_D represented by $[q_1, q_2, \ldots q_i]$. On applying to each of $q_1, q_2 \ldots q_i$ and taking the union.

We get $[p_1, p_2, \ldots p_i]$ in Q_D.

To Show

$$\delta_D(\{q_0\}, x) = [q_1, q_2, \ldots q_i]$$

if and only if

$$\delta_N(q_0, x) = \{q_1, q_2, \ldots q_i\}$$

This is proved by the method of induction.

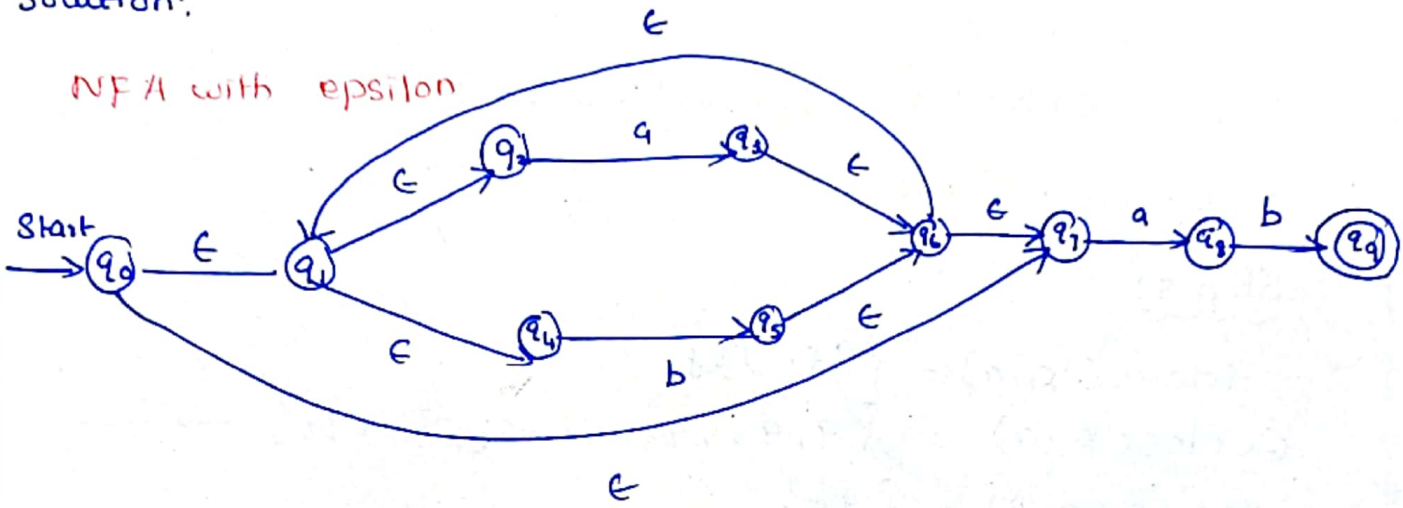
F_D is the set of subsets S of Q_N such that

$$S_n \cap F_N \neq \phi.$$

Construct NFA with epsilon for the RE = (a|b)* ab and convert into DFA and further find minimized DFA.

Solution:

NFA with epsilon



Construct the DFA

Step: 1

ϵ-closure(q_0) = $\{q_0, q_1, q_2, q_4, q_7\}$ ——— A

A – New State

Step: 2:

move(A, a) = $\{q_3, q_8\}$

ϵ-closure(A, a) = $\{q_3, q_6, q_1, q_2, q_4, q_7, q_8\}$ ——— B

move(A, b) = $\{q_5\}$

ϵ-closure(A, b) = $\{q_1, q_2, q_4, q_5, q_6, q_7\}$ ——— C

B, C New State

Step 3:

move(B, a) = $\{q_3, q_8\}$

ϵ-closure(B, a) = $\{q_1, q_2, q_3, q_4, q_6, q_7, q_8\}$ ——— B

move(B, b) = $\{q_5, q_9\}$

ϵ-closure(B, b) = $\{q_1, q_2, q_4, q_5, q_6, q_7, q_9\}$ ——— D.

D New State

Step 4:

move(C, a) = {q3, q8}

ε-clos(C, a) = {q1, q2, q3, q4, q6, q7, q8} ⟶ B

move(C, b) = {q5}

ε-clos(C, b) = {q1, q2, q4, q5, q6, q7} ⟶ C

No New State.

Step 5:

move(D, a) = {q3, q8}

ε-clos(D, a) = {q1, q2, q3, q4, q6, q7, q8} ⟶ B

move(D, b) = {q5}

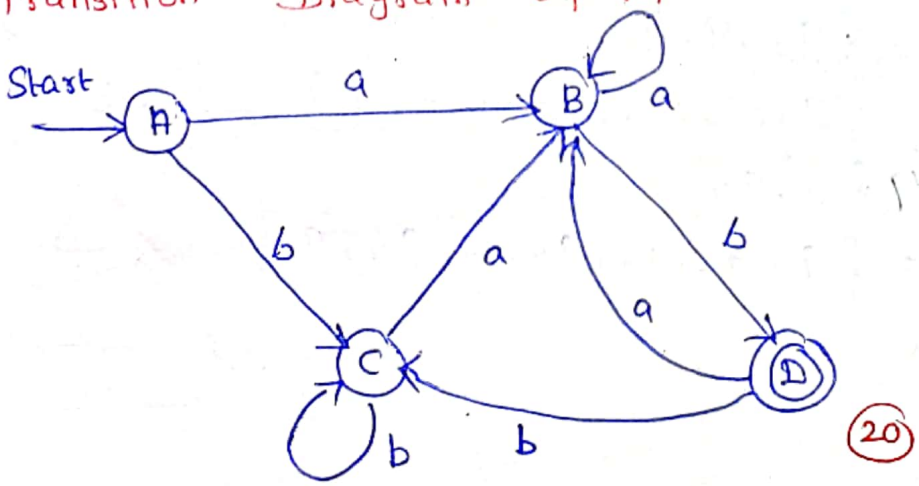
ε-clos(D, b) = {q1, q2, q4, q5, q6, q7} ⟶ C

∴ No new States.

Transition table of DFA:

→	δ	a	b
	A	B	C
	B	B	D
	C	B	C
	(D)	B	C

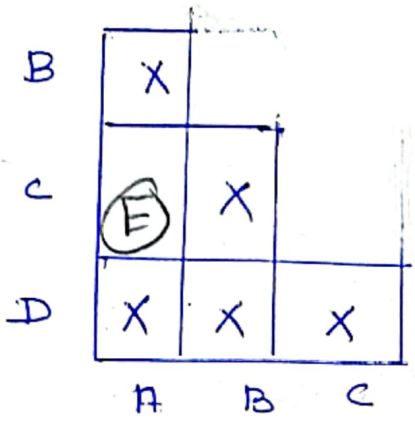
Transition Diagram of DFA



20

Minimized DFA:- (using Myhill – Nerode Theorem or Table Filling method)

Marking X in the Inequivalent States



(D, A) (D, B), (D, C) are marked 'X' because P is distinguishable from q.

Unmarked States

(AB), (BC), (AC)

$\delta(A,a) = B$ $\quad \delta(A,b) = C$

$\delta(B,a) = B$ $\quad \delta(B,b) = D$ } → Already marked

∴ ~~(A,B)~~ marked (A,B)

(B,C)

$\delta(B,a) = B$ $\quad \delta(B,b) = D$

$\delta(C,a) = B$ $\quad \delta(C,b) = C$ } → Already marked

∴ ~~(B,C)~~ mark (B,C)

Now consider (AC)

$\delta(A,a) = B$ $\quad \delta(A,b) = C$

$\delta(C,a) = B$ $\quad \delta(C,b) = C$

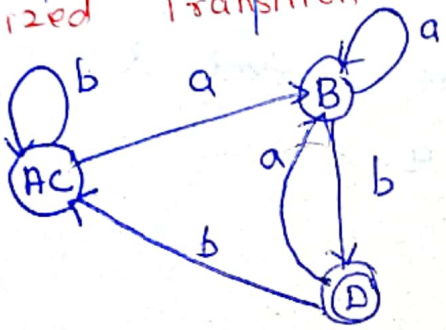
∴ Since no marked state ∴ So (A,C) cannot be marked.

∴ Combined unmarked State into Single State.

Minimized DFA Table.

δ	a	b
→ AC	B	AC
B	B	D
(D)	B	AC

Minimized Transition Diagram



The above example minimized DFA using Equivalent Theorem.

δ	a	b
→ A	B	C
B	B	D
C	B	C
(D)	B	C

Step 1: Initial partition Π consists of two groups:

Nonfinal & final State = (ABC) (D)

Step 2: To construct Πnew from group D: this group consists of a Single State, it cannot be Split further. So (D) is placed in Πnew.

Step 3:

Considering the group (ABC) based on the input Symbol a & b:

$\delta(A,a) = B$ $\quad$ $\delta(A,b) = C$

$\delta(B,a) = B$ $\quad$ $\delta(B,b) = D$

$\delta(C,a) = B$ $\quad$ $\delta(C,b) = C$

→ On input 'a' State has a transition to B

∴ one group for as input 'a' is concerned.

→ On input 'b', the State B has a transition to D, that belongs to another group.

∴ $\Pi_{new} = (AC)(B)(D)$

Step 4:

Considered the group (AC) on input 'a' & 'b'

$\delta(A,a) = B$ $\quad$ $\delta(A,b) = C$

$\delta(C,a) = B$ $\quad$ $\delta(C,b) = C$

Input 'a' goes to B and input 'b' goes to 'C'

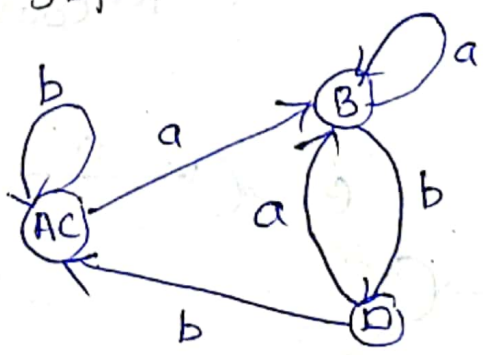
∴ The group cannot be splitted in further.

$\Pi_{final} = \Pi_{new} = (AC)(B)(D)$

Step: 5: Transition Table.

δ	a	b
→ [A,C]	[B]	[A,C]
[B]	[B]	[D]
(D)	[B]	[A,C]

Step: 6: Transition Diagram



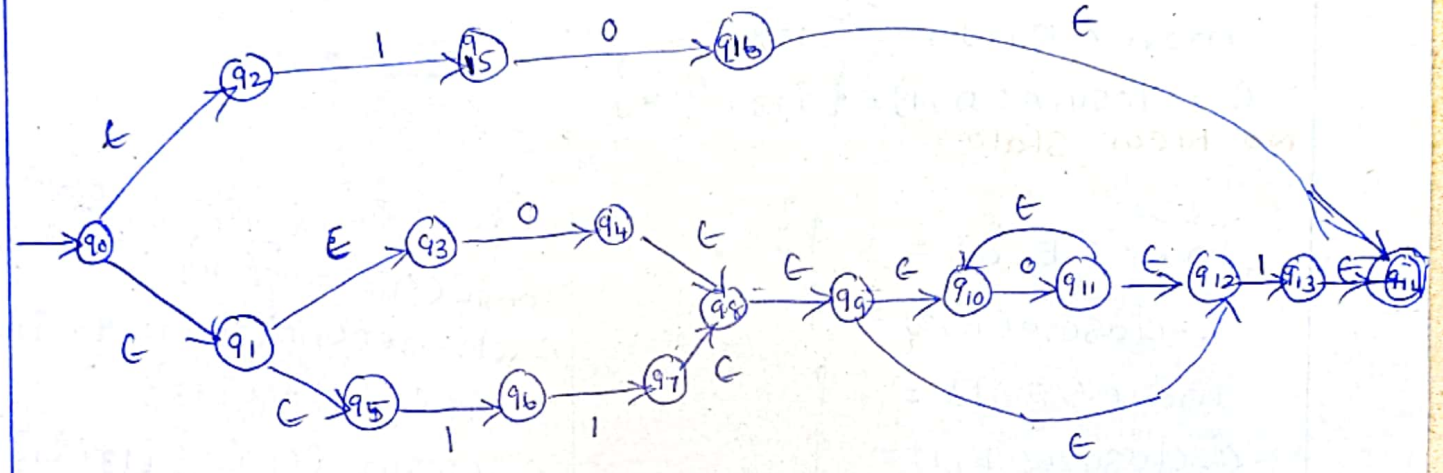
Construct minimized DFA that can be derived from the following regular expression

(i) $10 + (0+11)0^*1$

(ii) $(0+1)^* (00 + 11)(0 + 1^*)$

Solution:

Step 1: The automaton for $10 + (0+11)0^*1$



Step 2: NFA without ε-moves

$\therefore$ ε-closure $(q_0) = \{q_0, q_1, q_2, q_3, q_5\}$ ——— A

Step 3:

move $(A, 0) = \{q_4\}$

ε-closure $(A, 0) = \{q_4, q_8, q_9, q_{10}, q_{12}\}$ ——— B

move $(A, 1) = \{q_{15}, q_6\}$

ε-closure $(A, 1) = \{q_{15}, q_6\}$ ——— C

$\therefore$ B, C New State.

Step 4:

move $(B, 0) = \{q_{11}\}$

ε-closure $(B, 0) = \{q_{11}, q_{10}, q_{12}\}$ ——— D.

move $(B, 1) = \{q_{13}\}$

ε-closure $(B, 1) = \{q_{13}, q_{14}\}$ ——— E

$\therefore$ D, E – New State.

Step: C

move(C,0) = {q16}

ε-closure(C,0) = {q16, q14} ——— F

move(C,1) = {q7}

ε-closure(C,1) = {q7, q8, q9, q10, q12} ——— G

F is a New State.

Step D:

move(D,0) = {q11}

ε-closure(D,0) = {q10, q11, q12} ——— D

move(D,1) = {q13}

ε-closure(D,1) = {q13, q14} ——— E.

No New State

Step E:

move(E,0) = {ϕ}

ε-closure(E,0) = ϕ

move(E,1) = ϕ

ε-closure(E,1) = ϕ

No new State.

Step F:

move(F,0) = {ϕ}

ε-closure(move(F,0)) = {ϕ}

move(F,1) = {ϕ}

ε-closure(move(F,1)) = {ϕ}

No New State.

Step: G

move(G,0) = {q11}

ε-closure(G,0) = {q10, q11, q12} - D

move(G,1) = {q13}

ε-closure(G,1) = {q13, q14} - E

Transition Table of DFA:

New State	Inputs	
	0	1
→ A	B	C
B	D	E
C	F	G
D	D	E
* E	ϕ	ϕ
* F	ϕ	ϕ
G	ϕ	ϕ

Rewrite the Minimized DFA Transition Table:

New State	Inputs	
→	0	1
A	B	C
B	D	E
C	F	G
D	D	E

Transition Diagram for DFA: