THEORY OF AUTOMATA COMPUTATION

SECTION-A

FINITE AUTOMATA AND REGULAR EXPRESSIONS:

- → INTRODUCTION
- → Basic Definitions
- → Finite state Machines/systems
  - ↳ Deterministic Finite Automata [DFA]
  - ↳ Non-Deterministic Finite Automata [NDFA]
- → Equivalence of DFA and NFA (NDFA)
- IMP. → Conversion of NFA to DFA
- → Finite Automata with ϵ-Moves

⇒ Regular Expressions:

- IMP. → Regular Expression Conversion

Introduction to Machines :-

- → Concept of Basic Machines
- → Properties & Limitations of FSM.
- → Moore & Mealy Machines.
- → Equivalence of Moore & Mealy Machines (m/c)
- IMP. → State and Prove Arden's Method (Thm).

FLA LECTURE NOTES BY:
KIRTI SHARMA
ASSISTANT PROFESSOR

Scanned by CamScanner

Introduction :-

→ Computation Problems :-

Theoretically, in Computer Science, a computational problem is a mathematical object representing a collection of questions that computers might want to solve.

For example, the problem of factoring "Given a +ve integer n, find a non-trivial Prime factor of n." is a computational problem.

↳ The field of algorithms studies methods of solving computational problems efficiently.

↳ The complementary field of computational complexity attempts to explain why certain computational problems are intracable for computers.

⇒ Types of computational Problems :

→ Decision Problem : is a computational problem where the answer for every instance is either Yes or No.

Eg. "Given a +ve integer n, determine if n is Prime".

eg. $L = \{2, 3, 5, 7, 11, ----\}$.

→ Search Problem :- the answers can be arbitrary strings.

for eg- factoring is a search problem where the instances are +ve integers and the solutions are collections of Primes.

Represented as Relation as all instance-solution pairs.

$$R = \{(4,2), (6,2), (6,3), (8,2), (8,4), (9,3), ----\}.$$

which consists of all pairs of numbers (n, p) where p – is a nontrivial Prime factor of n.

Scanned by CamScanner

→ Counting Problem: asks for no. of solutions to a given search problem. For eg. the counting problem associated with Primality is "Given a +ve integer n, count No. of Non-trivial Prime factors of n."

→ Optimization Problem:- for finding the "best possible" solution among set of all possible solutions to a search problem. eg. – Maximum independent set Problem: "Given a graph G, find an independent set of G of maximum size".

Computational Models:

In computability Theory and computational Complexity theory a model of Computation is the definition of set of allowable operations used in computation and their respective costs.

↳ Used for measuring complexity of an algorithm in 'execution time' or 'memory space': by assuming a certain model of computation, it is possible to analyze the computational resources required or to discuss the limitations of algorithms or computers.

Some examples of Models include-

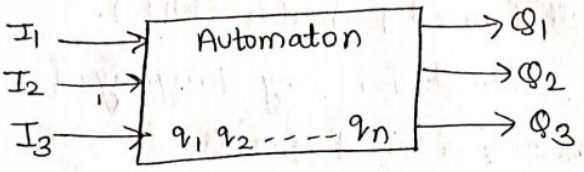
Turing M/cs, Recursive functions, Lambda Calculus, & Production Systems.

Scanned by CamScanner

Automaton :-

"Automata Theory is an abstract model of computation."

- It is used for solving the computational problems.
- "Automata Theory is the study of self-operating Virtual-Machine to help in logical understanding of input and output process with or without the intermediate stage of computation".



↳ An Automaton gets one input every time step that is picked up from a set of symbols or letters which is called an 'Alphabet' and finite sequence of symbols is called 'Word' at each instance of time of run of input word. The automaton is in one of its state.

↳ At each time step, when the automaton reads the symbols of input word one after another and transitions from state to state according to transition function.

↳ Once the input words has been read the automation is said to have stopped and state at which automata has stopped is called "final state".

↳ Depending on final state, it's said that automation either accepts / Rejects an input word.

The set of all words which is accepted by automata called "language recognized by the automation".

Scanned by CamScanner

⇒ **Language**:- A language is a set of words; i.e. finite strings of letters, symbols or tokens.

↳ The set from which these letters are taken is called 'alphabet' over which language is defined.

↳ A formal language is often defined by means of a "formal Grammer" (formation Rules).

↳ Words that belong to a formal language are sometimes called "well-formed formulas" (WFF).

→ A Hierarchy is defined for any language (or formal language)

Alphabet → a set of symbols

{a, b}

Sentences - are strings of symbols.

a, b, aa, ab, ba, abb, ---

Language - is a set of sentences

L = {aaa, aab, abaa, bbb}

Grammer - is a finite list of rules defining a language.

S → aA	B → bB
A → bA	B → aF
A → aB	F → ϵ

Strings-

Scanned by CamScanner

→ Strings "An alphabet is a non-empty finite set of symbols. denoted by Σ.

eg. $\Sigma = \{a, b, c\}$ is an alphabet.

↳ A string is a finite sequence of symbols. (U or w)

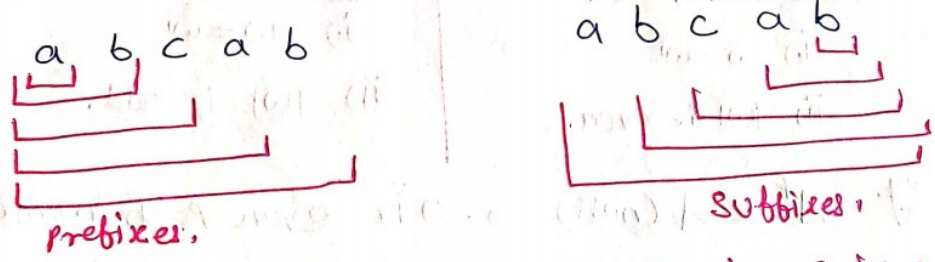
eg. $U = abcab$ is a string on $\Sigma = \{a, b, c\}$.

The empty string (no symbol at all) denoted by λ or ϵ.

- A part of string is a substring.

bca is a substring of abcab.

Note :- A beginning of a string (up to any symbol) is a prefix & an ending is a suffix.



** "A string is a prefix & suffix of itself. λ or ϵ is a prefix & suffix of any string."

⟹ operations on strings :-

1.) Finding the length : $\Sigma = \{a, b\}$

$w = abba$

$|w| = 4$

2.) Concatenation :

$w = abc$, $v = ab$

$wv = \underbrace{abc}_{w}\underbrace{ab}_{v}$

* → $\lambda w = w\lambda = w$.

Scanned by CamScanner

3) Power:- $w^0 = \lambda$ (null string)

$w^1 = w \Rightarrow w^2 = ww \Rightarrow w^3 = w.w^2 = w.w.w$

$w^n = w.w^{n-1} = w.w.\ldots$ (n-times)

4) Reverse:- $w^R \rightarrow$ w in reverse order

$w = abc$

$w^R \rightarrow cba$

5.) Palindrome : $|w| = |w^R|$

Word & its reversal have same value.

$w = aba$
$w^R = aba$

Even Palindrome:

i.) $w = w^R$

ii) $|w|$ is even.

Odd Palindrome:

i.) $w = w^R$

ii) $|w|$ is odd.

eg $\lambda^R = \lambda$ (null) $\rightarrow$ 0 is even palindrome.

$a^R = a$ (odd(1)) $\rightarrow$ odd Palindrome.

eg No. of palindrome of length 8 over $\Sigma = \{0,1\}$.

$\hookrightarrow 2^4 = 16.$

*** No. of Palindromes of length n over $\Sigma = k$ is

$$k^{\lceil n/2 \rceil}.$$

6.) Kleen Star / Kleen's closure Σ^*

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

Σ^* = the set of all strings which can be conducted by using the symbols from Σ including λ.

eg $\Sigma = \{a, b$

$\hookrightarrow \Sigma^* = \{$

it is

eg. $\Sigma =$

$a^* =$

) Kleen Plus

be cons

Σ^+

$\therefore$

we can

Scanned by CamScanner

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

↳ $\Sigma^* = \{\lambda, a, b, aa, bb, ba, ab, aaa, aba, -----\}$

it is a universal language.

eg. $\Sigma = \{a\}$

$a^* = \{a^*\} = \Sigma^* = \{\lambda, a, a^2, a^3, ----\}$

1.) Kleen Plus / +ve closure (Σ^+) is the set of all strings which can be constructed using the symbols of Σ excluding λ.

$\Sigma^+ = \{a, b, aa, bb, aaa, ----\}$

$\therefore \boxed{\Sigma^* - \Sigma^+ = \{\lambda\}}$

we can say,

$\Rightarrow [\Sigma^+ \cup \{\lambda\} = \Sigma^*] \Rightarrow \{\Sigma^* \cup \{\lambda\}\} = \Sigma^*$

→

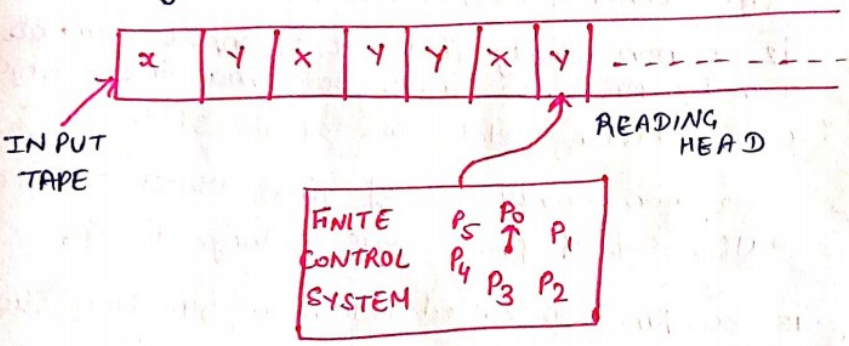
Scanned by CamScanner

A.

FINITE AUTOMATA

Finite Automation is called "finite" because no of possible states and no. of letter in the alphabet are both finite and "automation" because the change of the state is totally governed by the input.

It is deterministic, what state is next is automatic not will-full, just as the motion of the hands of clock is automatic, while the motion of hands of a human is presumably the result of desire and thought.



Here, $P_0, P_1, P_2, P_3, P_4, P_5$ are states in Finite control system

x and y are input symbols.

→ At regular interval the automation reads one symbol from the input tape and then enters in a new state that depends only on the current state and the symbol just read.

→ After reading an input symbol, reading head moves one square to the right on the input tape, so that on the next move, it will read the symbol in next tape square.

Repeat it again and again.

The automation then indicates approval or disapproval.

Scanned by CamScanner

↳ If it winds up in one of a set of final states the input strings is considered to be accepted.

The language accepted by the machine is the set of strings, it accepts.

Definition:-

DETERMINISTIC FINITE AUTOMATA (DFA):

A deterministic Finite Automata is a quintuple

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

where,

Q : is a non-empty finite set of states present in finite control. $(q_0, q_1, q_2, \ldots)$

Σ : is a non-empty finite set of input symbols which can be passed to finite state machine. $(a, b, c, \ldots)$

q_0 : is a starting state, one of the state in Q.

F : is a non-empty set of final states or accepting states, set of final states belongs to Q.

δ : is a function called transition function that takes two arguments a state and a input symbol, it returns a single state. $\delta: Q \times \Sigma \to Q$.

Let 'q' is the state and 'a' be input symbol passed to the transition function as:

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

q' is output of the function.

A single state q' may be q. It can be:

$\delta(q,a) \longrightarrow q'$ (✓)

but q' may be same as q. $(q' = q)$

$\delta(q,a) \longrightarrow q'$, q'' (X)

Sometimes, FA (F
Recognizer.
TRANSITION DI
1.) Initial
ii.) Final
"ac
Desi
S
S

Scanned by CamScanner

Sometimes, FA (Finite Automata) is also called as 'Language Recognizer'.

TRANSITION DIAGRAMS:

1.) Initial State:

"Accepting Nothing".

ii.) Final State:

"accepting Everything".

2.) An alphabet Σ of possible input letters from which input strings are formed.

3) A finite set of transitions (labelled edge) that show how to go from some states to some others, based on reading specified substrings of input letters (Null string).

Design a DFA which accepts strings that ~~starts~~ with a over $\Sigma = \{a, b\}$.

Step 1: A DFA is a quintuple:

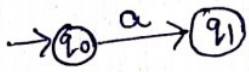
$\Rightarrow (Q, q_0, \Sigma, F, \delta)$ where, $Q = \{q_1, q_0\}$

$q_0 = q_0$

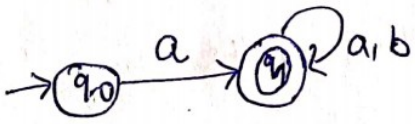
$F = q_1$

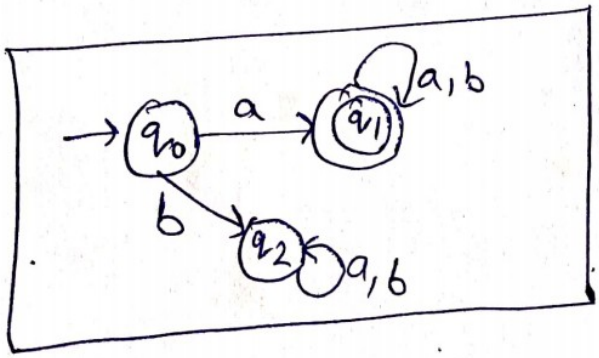
$\Sigma = \{a, b\}$

Step 2:



$L = \{a, aa, ab, aba, ---\}$





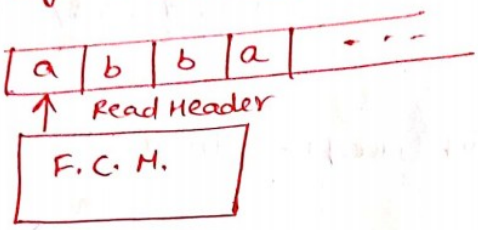
TRANSITION DIAGRAM

Scanned by CamScanner

Step 3: Transition Diagram: we have $\Sigma = \{a, b\}$

States \ alphabets	a	b
→ q_0	q_1	q_2
(q_1)	q_1	q_1
q_2	q_2	q_2

Step 4: A string ex. 'abba' is accepted by this DFA or Not:



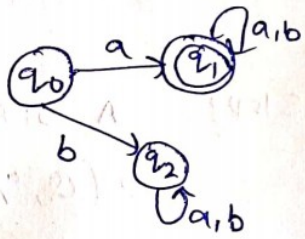
Using δ :-

$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_1$

$\delta(q_1, b) = q_1$

$\delta(q_1, a) = q_1$



So, final transition state is 'q_1' & 'q_1' in transition diagram is Final State. Hence, the string 'abba' is accepted by DFA.

Elements of
↳ States
↳ State t
another
↳ Rule
transi
↳ Inp
gen
St

1. States
↳

(i) S
(ii)
(iii)
IV)

Scanned by CamScanner

Elements of Finite state systems: 4 main elements:

↳ States which define the behavior and may produce actions
↳ State transitions, which are movement from one state to another.
↳ Rule or conditions which must be met to allow a state transition.
↳ Input Events, which are either externally or internally generated. These may possibly trigger the rules & lead to state transitions.

Not:

(1.) **State:**

↳ A state is a complete set of properties, transmitted by an object to an observer via one or more channels.
- Any change in the nature or quantity of such properties in a state is detected by an observer & thus a transmission of information occurs.

(i) **Start State:** An initial state or condition of a finite state Machine.

(ii) **Accepting State:** If a FSM finishes an i/p string & is in an accepting state, the string is accepted or considered to be valid.

(iii) **Next State:** ↳ The state immediately following current state;

iv) **Universal State:** A state in an alternating Turing M/c, from which m/c only accepts all possible moves leading to acceptance.

v) **Existential State:** ↳ A state in a nondeterministic T.M. from which the m/c accepts any move that leads to acceptance, is the existential state.

vi) **Dead/Trap State:** → A nonfinite state of a FSM, whose transitions on every input symbol terminates on itself.

Transition:

↳ "Transition is the act of passing from one place/state to the next."

Scanned by CamScanner

↳ A change from one place or state or subject to another

Transitions are represented in the following ways

↳ state diagram or Transition diagram.

↳ State Transition Table.

↳ Transition functions.

↳ State diagram :-

A diag. consisting of circles to represent states & directed line segments to represent transitions b/w states, is called a state diagram.

Start of the process

final / Accepting / Absorption state (end of process)

Transition.

↳ For a FSM, a state diagram is a directed graph where,

a) Each edge is a transition between two states.

↳ For DFA, NFA and Moore M/C, i/p is labeled on each edge.

↳ For a mealy M/C, i/p & o/p is labelled on each edge.

b) Each vertex is a state

↳ for a Moore M/C, o/p is signified for each state

↳ State Transition Table :-

A state Transition table can be described, in general, as :- = any F.A.

→ Tabular representation of transitions that take two arguments & return a value.

→ Rows correspond to states & columns correspond to inputs.

→ Entries correspond to next states.

→ the start state is marked with an arrow (→).

→ the accepting states are marked with a star. (*).

State Transition Table format.

I/Ps / States	Present I/Ps		--- ---	
	a1	a2	--- ---	an
S_0				
S_1				

S_n				

↳ A Finite Automaton has-

a.) a finite set of states, one of which is designed as the initial state or start state and some of which are designated as final states.

b.) An alphabet Σ of possible input symbols.

c) a finite set of transitions that informs each state and each symbol of the i/p alphabet, about next state.

Scanned by CamScanner

Finite Automata :-

According formal definition,

"A finite automation is a list of five objects : set of states, input alphabet, rules for moving, start & accept states."

⇒ Deterministic Finite Automata (DFA) :-

"A DFA is a finite state machine where for each pair of states and input symbol; there is a unique next state

Elements of DFA :-

The DFA exhibits 5 characteristics :

a) A finite set of states Q.

b) an Alphabet Σ of possible i/p symbols.

c) a transition function δ such that $\delta(x,1)=y$ where $x, y \in Q$ and $1 \in \Sigma$.

d) the initial state $q_0 \in Q$.

e) the set of final states (F), where $F \subseteq Q$.

"The term deterministic refers to the fact that on each input, there is one and only one state to which the automaton can transit from its current state".

Ordered Quintuple Specification of DFA :-

Formally, a DFA, M is a five-tuple :

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

where,

a) Q is a finite set of states of finite automata.

b) Σ is a finite set of input symbols called "alphabet",

c) δ: $Q \times$

d) $q_0 \in Q$

e) $F \subseteq$

of from

State

written

wher

Thus,

Des

T

Scanned by CamScanner

Regular Language :- for R.L.; (Finite) . (DFA/NFA).

c.) $\delta: Q \times \Sigma \to Q$, is the transition function,

d.) $q_0 \in Q$ is the start state.

e.) $F \subseteq Q$ is the set of accept states.

If from a state P, the exists a transition going to state 'q' on an input symbol 'a', then this is written as:

$$\boxed{\delta(P, a) = q}$$

Where, δ – is a func. whose domain is a set of ordered pair (P, a)

P – a state

a – input symbol.

Thus, 'δ' → defines a mapping whose domain will be a set of ordered pairs of the form (P, a) & whose range will be a set of states. i.e

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

Description of a DFA's

The transitions of DFA can be represented using transition diagram or table.

→ Extended transition function for DFA's

for DFA, $M = (Q, \Sigma, \delta, q_0, F)$ the function 'δ' is extended as –

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

and is defined recursively as follows :-

a) For any state q of Q

$$\delta(q, \epsilon) = q$$

This means that DFA stays in the same state q when it reads an empty string at q.

Scanned by CamScanner

b) for any state q of Q, any string $x \in \Sigma^*$ with a as the last symbol of 'x' and $a \in \Sigma$.

$$\delta(q, xa) = \delta(\delta(q, x), a).$$

Language accepted by DFA's

The lang. accepted by a DFA, $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings on Σ accepted by M, i.e;

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

A lang. is said to be rejected by DFA if $M = (Q, \Sigma, \delta, q_0, F)$ such that

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

* A machine may accept several strings but it recognises only one language.

Design of DFA's

The basic design strategy for DFA is as follows :-

a) Understand the language properties for which the DFA has to be designed.

b) Determine the state set required.

c) Identify the initial, accepting and Dead state of DFA.

d) For each state, decide on the transition to be made for each character of the input string.

e) obtain the transition table and diagram for DFA.

f) Test the DFA obtained on short strings.

Scanned by CamScanner

Regular Language :- For R.L., it is (finite). (DFA / NFA).

Ex.- To design a DFA that accepts set of all strings that contain 0's or 1's and end in "00". (R.L.)

Soln:- We are required to design a DFA for the regular expr (R.L.)

$r = (0+1)^*00\ldots$ (R.L.)

On other words, the DFA should be designed to accept the language of r; (R.L.)

i.e. $L(M) = \{00, 100, 1100, 0000, \ldots\}$

where, M is a DFA.

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

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

q_0 = initial state

q_2 = final state and

$\delta: Q \times \Sigma \longrightarrow Q$ is given by

Transition diagram :-

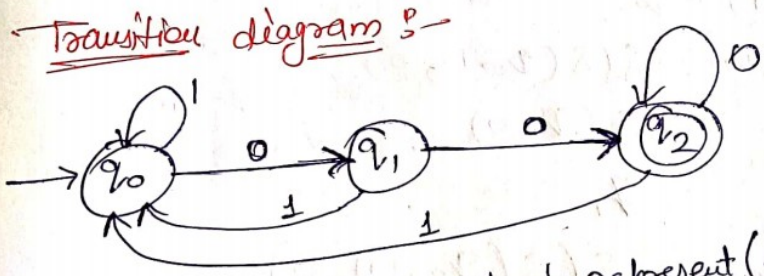


Fig:- State Transition to represent $(L(M) = \{w \in \Sigma^* \mid w$ ends with $00\}$

Transition Table :-

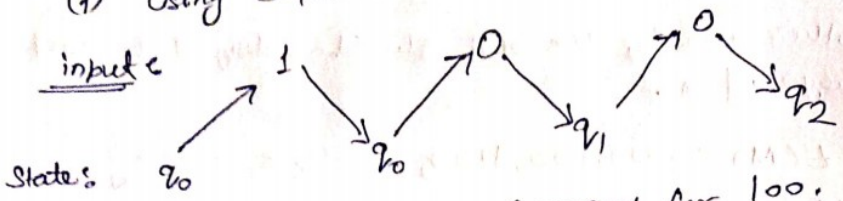
Q	Σ	Present I/Ps	
		0	1
→ q_0		q_1	q_0
q_1		q_2	q_0
* q_2		q_2	q_0

Scanned by CamScanner

DFA action for the input string ε

1.) To show that the string 100 is accepted by DFA:

(i) Using Sequence state diagram ε-



Sequence state diagram for 100.

Since, we encounter the end of the input and we are in the final state, we say that string is accepted by machine M. Thus, 100 is in L(M).

(ii) Using extended transition function-

Consider,

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

$$\delta(q_0, 10) = \delta(\delta(q_0, 1), 0)$$
$$= \delta(q_0, 0)$$
$$= q_1$$

$$\delta(q_0, 100) = \delta(\delta(q_0, 10), 0)$$
$$= \delta(q_1, 0)$$
$$= q_2$$

Since, after scanning the entire string, we reach at the final state q_2, the given string 100 is thus accepted by the DFA M. Thus, 100 is in L(M).

(iii) Using vdash function (⊢):

Scanned by CamScanner

$(q_0 100) = \vdash M(q_0, 00)$

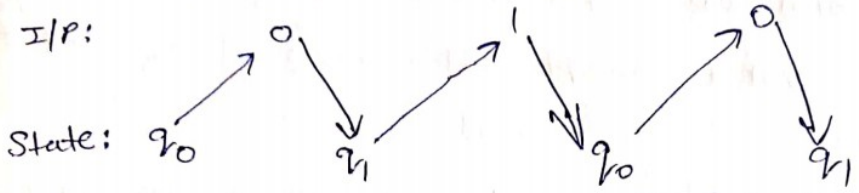
$\vdash M(q_1, 0)$

$\vdash M(q_2, e)$ → means 'reached end of i/p'.

Since, we reached the end of input (e) and we r in final state q_2, hence 100 is accepted by M. Thus, 100 is in L(M).

2) To show that the string 010 is rejected by DFA:—

(i) Using state (sequence. state) diagram:



Since, we encounter the end of i/p & q_1 is not the final state, we say that the string 010 is rejected by the m/c.

(ii) Using extended transition functions:

$\delta(q_0, 0) = q_1$

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

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

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

$= \delta(q_0, 0)$

$= q_1$

Since, after scanning the entire string, we did not reach the final state q_2, hence the string 010 is rejected by DFA 'M'.

=

Scanned by CamScanner

Q.1.) To design a DFA that accepts a set of even number of a's.

2.) Design a DFA that accepts odd number of 1's.

3.) Design a DFA that
→ starts with 0 and has odd number of 0's.
→ starts with 1 and has even number of 1's.

4.) Design a DFA that accepts even number of a's & b's.

5.) Design a DFA to accept odd number of a's and b's.

6.) Design a DFA to accept odd number of a's and even number of b's.

7.) Design a DFA that contains set of all strings ending with 3 consecutive zeros, over the $\Sigma = \{0, 1\}$.

8.) Design a DFA, to accept the language $L = \{awa | w \in (a+$
over $\Sigma = \{a, b\}$,

9.) Design a DFA to accept the set of all strings of a
starting with string 'ab'.

10.) Design a DFA, over $\Sigma = \{0, 1\}$, that contains set of strings of 0's & 1's except those containing substring '110'.

11.) Design a DFA over $\Sigma = \{0, 1\}$, that contains set of 0's except those containing substring 001.

* Design a DF
accepted must
1.) w = 'a'
Sol:
L
i.)
iii)
II)
III

Scanned by CamScanner

Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with w.

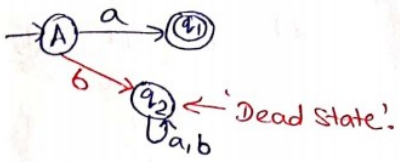
i) $w = 'a'$

Sol: Possible Language;

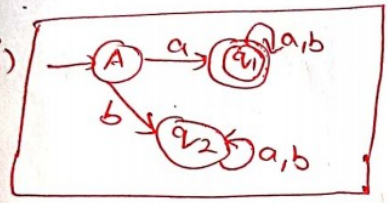
$L = \{a, aa, ab, aaa, \ldots\ldots\}$

i)

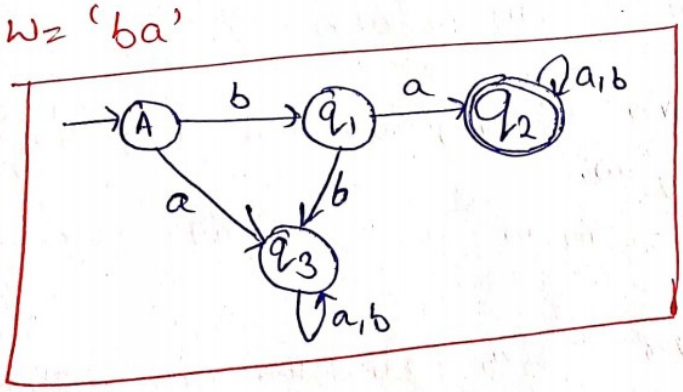
ii)



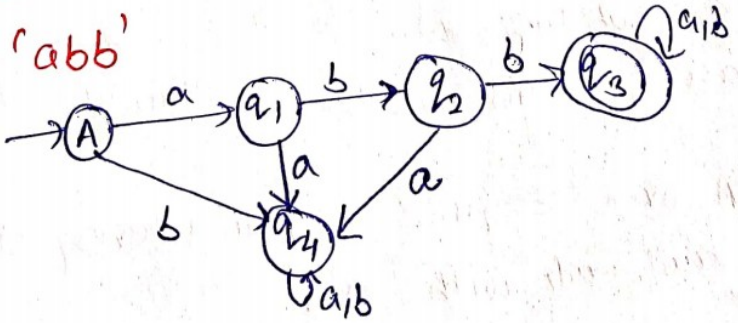
iii)



II) $w = 'ba'$



III) $w = 'abb'$



* NOTE:- (n-2) states in MDFA.

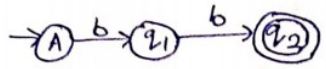
Scanned by CamScanner

Design a MDFA over $\Sigma=\{a,b\}$ such that every string accepted must ends with a substring w.

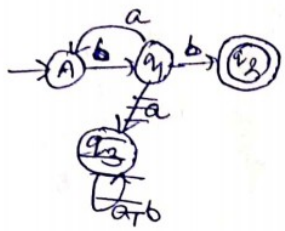
i.) $w='bb'$

$L=\{bb, abb, aabb, bbb, ----\}$

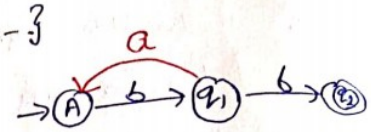
1.)



2.)

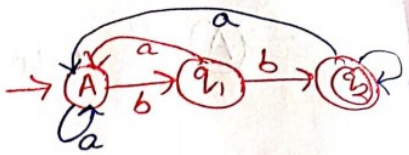


3.)



No Need of dead

4.)



PRACTICE PROBLEMS

TRY IT YOURSELF!

(1.) Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted must start with a substring w.

i.) $w=ba$ ii) $w=abb$

(2.) Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted ending with a substring w.

i.) $w=bb$ ii) $w=ab$ iii) $w=bab$

(3.) Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted contains a substring w.

i.) $w=aa$ ii) $w=ba$ iii) $w=bb$

(4.) Design a DFA over $\Sigma=\{a,b\}$ such that every string accepted must starts and ends with 'a'.

Scanned by CamScanner

5.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start and ends with same symbol.

6.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with 'a' ending with 'b' and vice-versa

7.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start with 'aa' or 'bb'.

8.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must ends with 'aa' or 'bb'.

9.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must contains a substring 'aa' or 'bb'.

10.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must

a) $|w| = 2$ b) $|w| \geq 2$ c.) $|w| \leq 2$

d.) $|w|_a = 2$ e.) $|w|_a \geq 2$ f.) $|w|_b \leq 2$.

11.) Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must

a.) $|w| = 2 \pmod 3$ b.) $|w| = 3 \pmod 4$

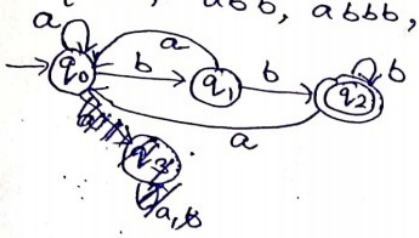
c.) $|w| = 1 \pmod 5$ d.) $|w|_b = 2 \pmod 3$

e.) $|w|_a = 3 \pmod 4$ f.) $|w|_b = 1 \pmod 5$

Scanned by CamScanner

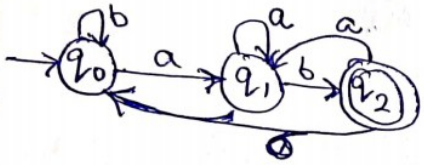
i.) ending with bb.

$L = \{ bb, abb, abbb, bbb, babb, \ldots \}$



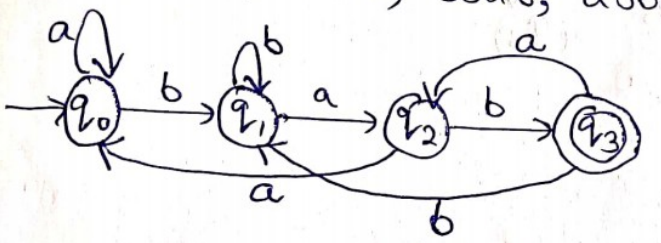
ii) ending with $w = ab$;

$L = \{ ab, aab, bab, aaab, bbab, baab, \ldots \}$



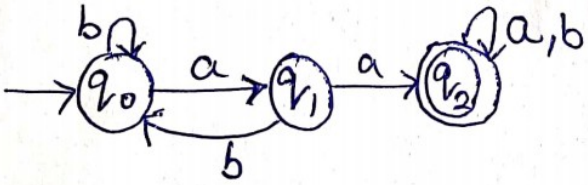
iii) ending with bab.

$L = \{ bab, abab, bbab, abbab, \ldots \}$



iv) Contains a substring $w = aa$.

$L = \{ aa, baa, aab, baab, aaa, \ldots \}$

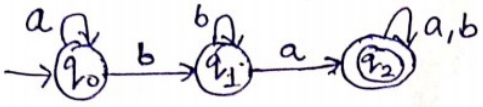


v) ~~EOL~~

Scanned by CamScanner

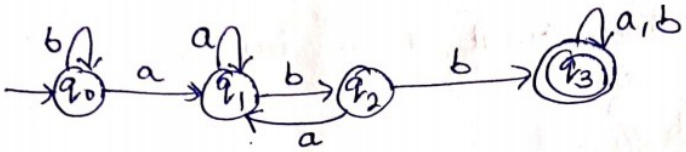
v) Containing substring $w = ba$.

$L = \{ ba, bba, aba, bab, aba, baa, abab, bbab, \ldots \}$



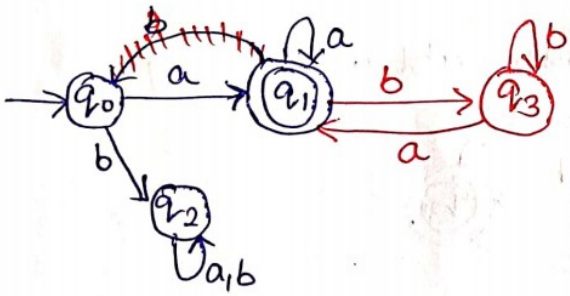
vi) Containing substring abb.

$L = \{ abb, babb, abba, abbb, \ldots \}$



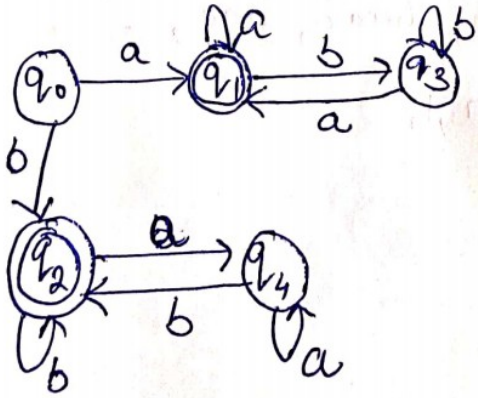
vii) Starts & ends with 'a'.

$L = \{ a, aa, aaa, aba, abba, \ldots \}$



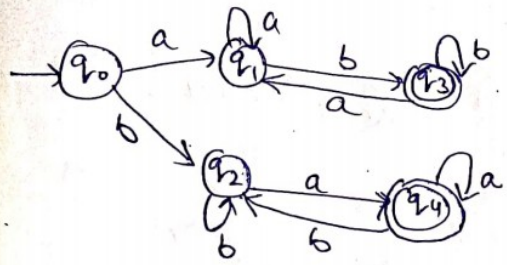
viii) Start & end with same symbol $\Sigma = \{a, b\}$.

$L = \{ a, b, aba, bab, baab, babb, abaa, baabab, \ldots \}$



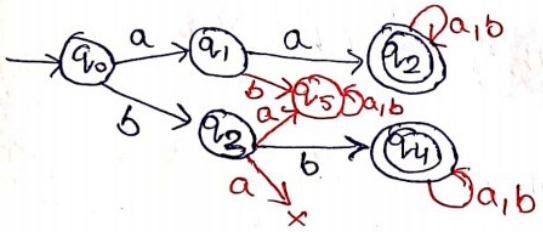
Scanned by CamScanner

x) starts with 'a', ~~and~~ ending with 'b' and vice-versa.



) starts with either 'aa' or 'bb'.

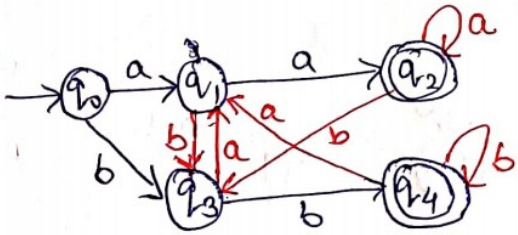
L = {aa, bb, aab, bba, aabb, aaba, bbba,



or

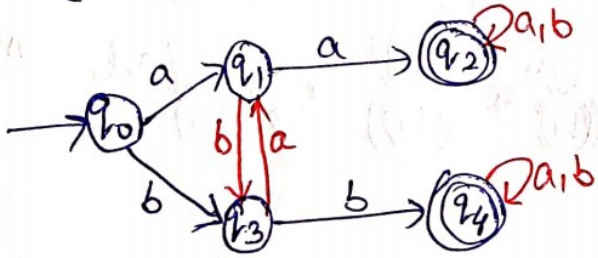
xi) ending with either 'aa' or 'bb'.

L = { aa, bb, baa, bbb, aaa, abb, babb, ---- }

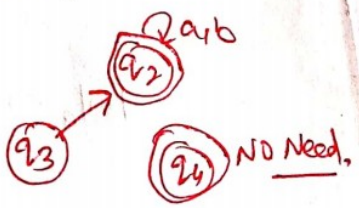


xii) contains substring either 'aa' or 'bb'.

L = { aa, bb, baab, aaa, bbb, abbb, baaa, babb, ---}

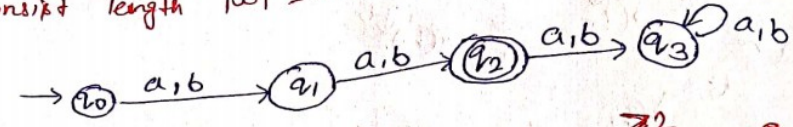


or

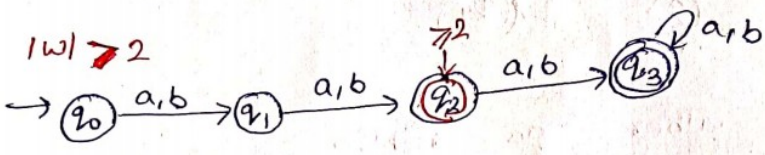


Scanned by CamScanner

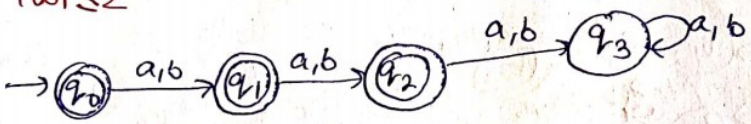
xiii) ~~Contain substring either 'aa' or 'bb'~~
Consist length $|w| = 2$.



xiv) $|w| \geq 2$



(xv) $|w| \leq 2$



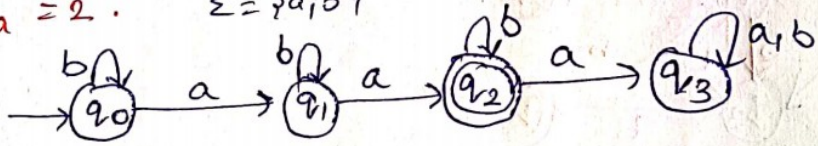
Generally, $|w| = n$; No. of states required?

↳ $n+2$

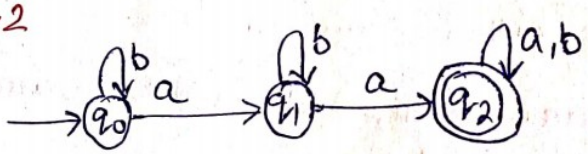
atleast → $n+1$

atmost → $n+2$

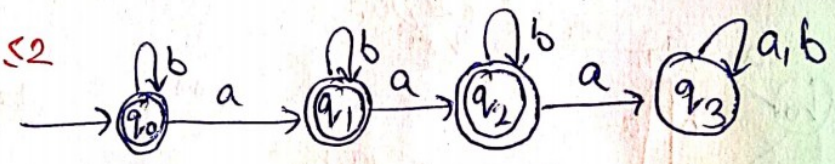
(xvi) $|w|_a = 2$. $\Sigma = \{a, b\}$



$|w|_a \geq 2$ Atleast 2 a's.



$|w|_a \leq 2$ Atmost 2 a's.



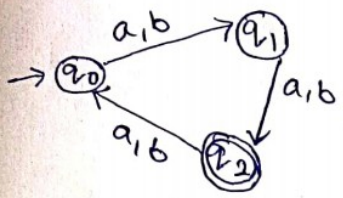
*NOTE → F.A. doesnot hv memory.

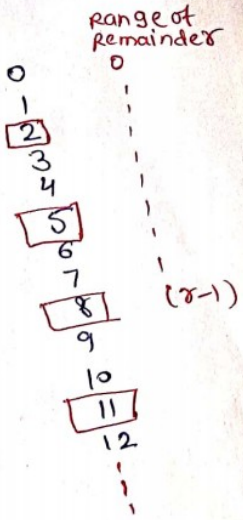
Scanned by CamScanner

ii) string accepted must

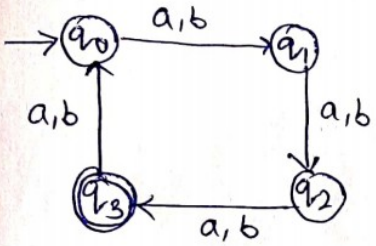
$|w| = 2 \pmod 3$

$|w| = r \pmod n$

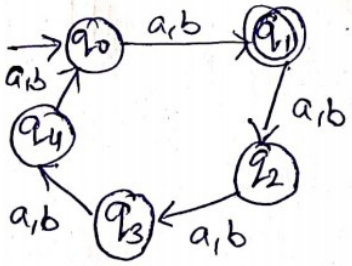




$|w| = 3 \pmod 4$



$|w| = 1 \bmod 5$



XVIII) $|w|_b = 2 \pmod 3$

2 b's.

Scanned by CamScanner

Design a DFA over $\Sigma = \{0, 1\}$, that contains set of strings of 0's & 1's which contain substring. 0101.

Construct a DFA to accept the set of all strings of the form $0^{2n}11$; $n \geq 0$.

Scanned by CamScanner

Non-Deterministic Finite Automata (NFA):-

In the foregoing section, it is observed, that every step of computation proceeds in a unique way the preceding step.

→ In other words,
When a machine (in a given state) reads the next input symbol, then the next state is uniquely determined.
This is called as Deterministic Finite Automata.

→ However, in the case of Nondeterministic machine, multiple choices may exist for the next state at any point.

* Thus, every DFA is automatically a NFA.

* → The term Non-deterministic refers to the fact that for each input, there can be several states to which the automaton can make a transition from its current state.

→ Ordered Quintuple Specification:-

formally, a nondeterministic finite automaton M is a five-tuple:-

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

Where,
a.) Q is a finite set of states.
b.) Σ is a set of input symbols.
c.) δ is a transition func. i.e. $\delta : Q \times \Sigma \rightarrow P(Q)$
d.) q_0 is a specially designed initial state.
e.) $F \subseteq Q$ is a set of final states.

→ In NFA, the transition function δ takes a state & an i/p symbol (or the empty string) and produces the set of possible next states.

Scanned by CamScanner

Design of NFA's :-

The basic design strategy for an NFA is as follows :-

a) Understand the language properties for which the NFA has to be designed.

b) Determine the alphabet & state set required.

c) Identify the initial, accepting & dead states of NFA.

d) Obtain the transitions to be made for each state on each character of the i/p string.

e) Draw the transition table & diagram for NFA.

f) Test the NFA obtained on short strings.

Q.1) Design an NFA that accepts set of all strings over {0,1} that have atleast two consecutive 0's or 1's.

→ we are required to design a NFA for the regular expr $r = (0+1)^*(00+11)(0+1)^*$.

In other words, NFA, M should be designed to accept the language of r.

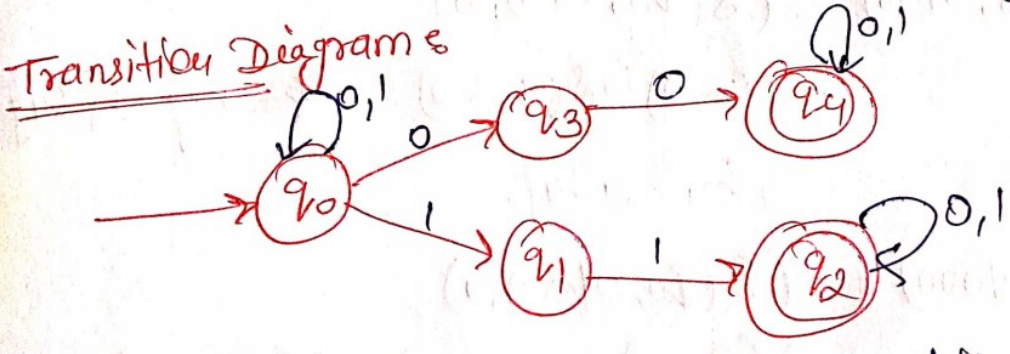
$$L(M) = \{00, 11, 0110, 1001, 10110101, \ldots\}$$

where M is the NFA.

Consider, $\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$.

q_0 = initial state, $\{q_2, q_4\}$ - final states & δ is given by -

Transition Diagram :-



State Diagram to Represent

Scanned by CamScanner

$L(M) = \{w \in \Sigma^* \mid w$ contains atleast two consecutive 0's or 1's$\}$.

* Transition Table :-

δ	Next I/P 0	Next I/P 1
→ q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	ϕ	$\{q_2\}$
* q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	ϕ
* q_4	$\{q_4\}$	$\{q_4\}$

↳ NFA action for the i/p strings

a) To show that the string 0100011 is accepted by the NFA.

(i) Using extended transition function :-

→ $\delta(q_0, \epsilon) = q_0$, $\delta(q_0, 0) = \{q_0, q_3\}$ & $\delta(q_0, 1\} = \{q_0, q_1\}$.

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

$= \delta(q_0, 1) \cup \delta(q_3, 1) = \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$.

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

$= \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_3\} \cup \{\phi\}$

$= \{q_0, q_3\}$.

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

$= \delta(q_0, 0) \cup \delta(q_3, 0) = \{q_0, q_3\} \cup \{q_4\}$

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

→ $\delta(q_0, 01000) = \delta(\delta(q_0, 0100), 0)$

$= \delta(q_0, 0) \cup \delta(q_3, 0) \cup \delta(q_4, 0)$

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

$\delta(q_0, 010$

Now,

↳ Since,

clearly

i.e.

(ii)

Scanned by CamScanner

wo consecutive

by the NFA.

$= \{q_0, q_1\}$,

$\delta(q_0, 01000|1) = \delta(\delta(q_0, 01000), 1)$

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

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

$= \{q_0, q_1, q_2, q_4\}$,

Now, let $P = \{q_0, q_1, q_2, q_4\}$.

→ Since, P contains q_4 & q_2 (which are final states), clearly 01000|11 is accepted by L(M).

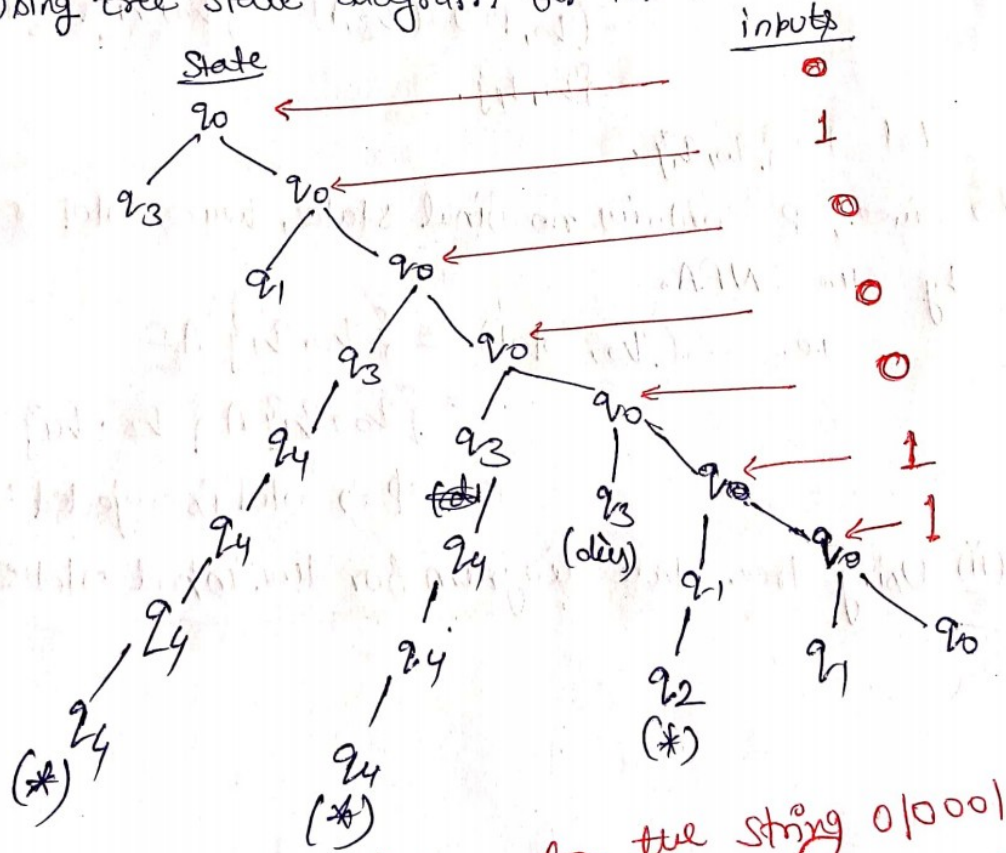
i.e. $\delta(q_0, 01000|11) = \{q_0, q_1, q_2, q_4\} \cap F$

$= \{q_0, q_1, q_2, q_4\} \cap \{q_2, q_4\}$

$= \{q_2, q_4\}$

$= \{\phi\} \Rightarrow 01000|11$ is accepted.

(ii) Using tree state diagram for input = 01000|1



Tree State diagram for the string 01000|11

Scanned by CamScanner

Since, there exists a final path, 01000|1 is accepted.

(b) To show that the string 0101 is rejected by the NFA

(i) Using extended function:-

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

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

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

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

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

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

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

$\rightarrow \delta(q_0, 0101) = \delta(\delta(q_0, 010), 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1) = \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$.

Let $P = \{q_0, q_1\}$.

↳ Since, P contains no final states, hence 0101 is rejected by the NFA.

ie $\delta(q_0, 0101) = \{q_0, q_1\} \cap F$

$= \{q_0, q_1\} \cap \{q_2, q_4\}$

$= \phi \Rightarrow 0101$ is rejected by NFA.

(ii) Using tree state diagram for the input 0101 :-

Scanned by CamScanner

accepted.
by the NFA

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

Let $P = \{q_0, q_1\}$.

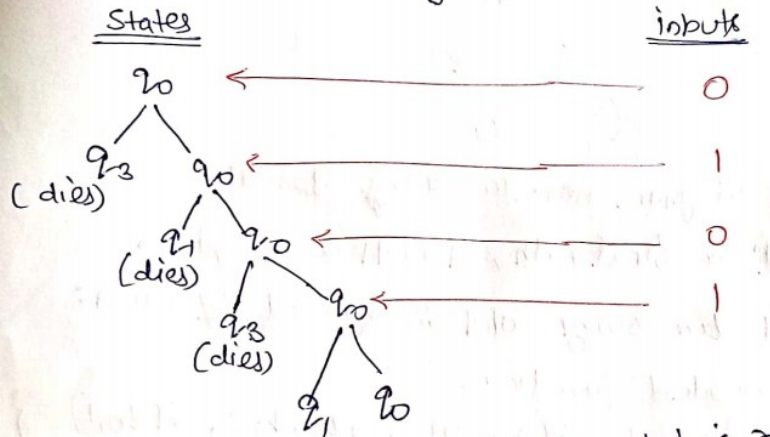
Since, P contains no final states, hence 0101 is rejected by the NFA.

i.e $\delta(q_0, 0101) = \{q_0, q_1\} \cap F$

$= \{q_0, q_1\} \cap \{q_2, q_4\}$

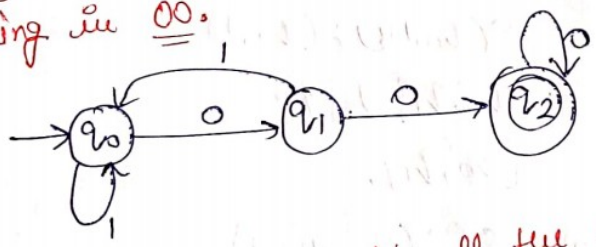
$= \phi \Rightarrow 0101$ is rejected by NFA.

(ii) Using Tree state diagram for the i/p 0101 →



Since, No final paths exist, 0101 is rejected by NFA.

Design an NFA that accepts a set of all strings ending in 00.

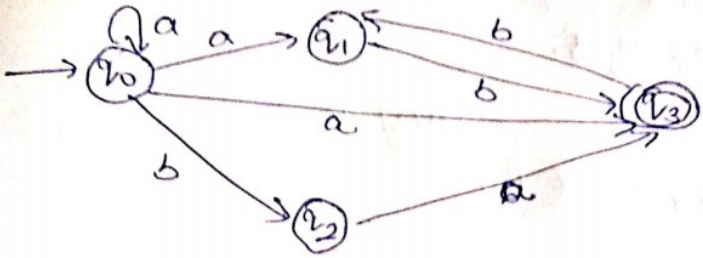


Q. Design an NFA to accept all the strings over $\Sigma = \{a, b\}$, ending in 'aba'.

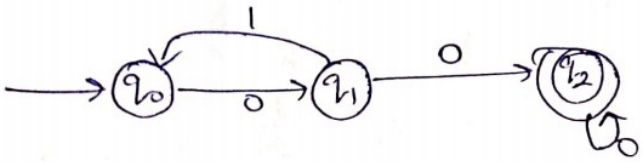
Scanned by CamScanner

2) Design an NFA to accept set of all strings starting with 'a' followed by 'a' or 'b' and ending with 'a' or any number of b's.

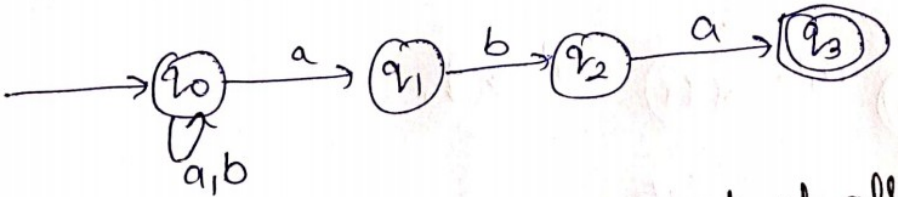
Ans follow steps as in eg. 1. (in every question)



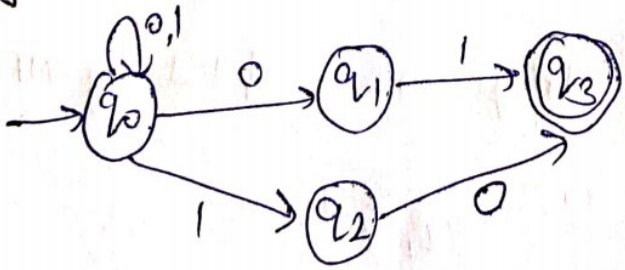
3) Design NFA that accepts a set of all strings ending in '00'.



4) Design NFA to accept all string over the alphabet {a,b} ending with 'aba'.

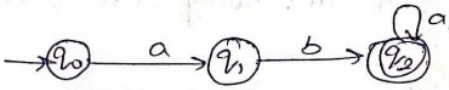


5) Construct an NFA that accepts the set of all strings over {0,1} that start with '0' or '1' and ends with '01' or '10'.

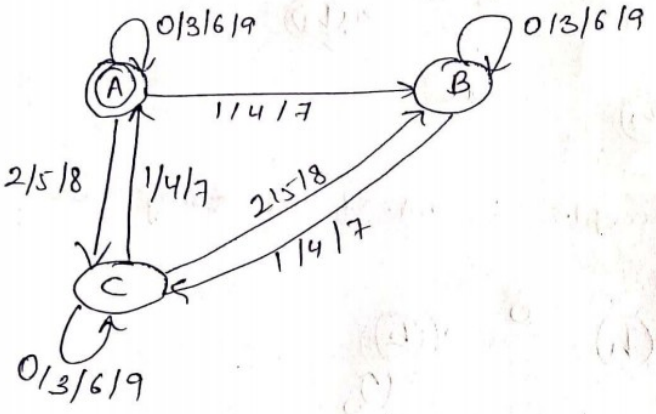


Scanned by CamScanner

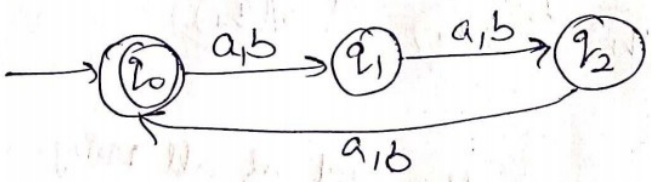
Q6) Given an NFA which accepts all strings with 'ab' over $\{a,b\}$.



Q7) Construct NFA that accept those strings of decimal digits, that are divisible by '3'.

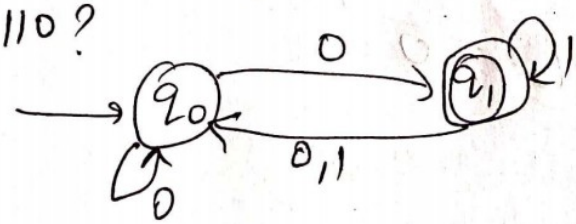


Q8) Construct a NFA that accepts the language $L=\{w: |w| \bmod 3 = 0\}$ on $\Sigma=\{a,b\}$.



Q9) Construct NFA that accepts the language $L=\{w: |w| \bmod 5 \neq 0\}$ on $\Sigma=\{a,b\}$.

10.) Which of the following strings are accepted by NFA:
0|0, 110?

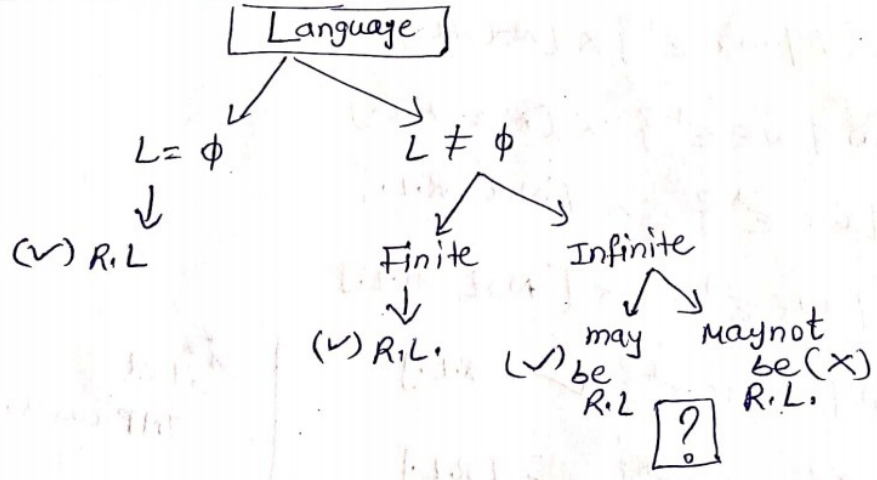


Scanned by CamScanner

Regular Language :- For R.L.; it is compulsory to exist any F.A. (Finite). (DFA / NFA).

1. $L = \{a^m b^n \mid m, n \geq 0\}$ ✓ (R.L)
2. $L = \{a^m b^n c^q \mid m, n, q \geq 0\}$ ✓ (R.L)
3. $L = \{a^{x_1} b^{x_2} \dots z^{x_{26}} \mid x_i \geq 0, 1 \leq i \leq 26\}$ ✓ (R.L.)
4. $L = \{a^m b^n \mid 1 \leq m \leq 500, 1 \leq n \leq 1000\}$ ✓ (R.L.)
5. $L = \{a^n b^n \mid 1 \leq n \leq 10\}$ ✓ (R.L)
6. $L = \{a^n b^n \mid 1 \leq n \leq 2^{29^{th} \text{ prime no}}\}$ ✓ (R.L)
7. $L = \{a^n b^n \mid 1 \leq n \leq 2^{|GATE|}\}$ ✓ (R.L)
8. $L = \{a^n b^n \mid n \geq 0\}$ × (NOT RL)

* NOTE :- Any ϕ (Emtpy) Language is also Regular Language.



→ Finite ~~Comparison~~ ordered can be done by Finite Automata.

Finite Automata doesn't have any infinite Memory to store.

[Comparison is Not Possible using F.A.].

Scanned by CamScanner

(9.) $L = \{a^m b^n \mid m \geq n, n \geq 0\}$ [Not R.L] ×

(10.) $L = \{a^m b^n \mid m, n \geq 0, m \neq n\}$ [Not R.L.] ×

(11.) $L = \{a^m b^n \mid m \text{ is divisible by } n\}$ [Not R.L] ×

(12.) $L = \{a^m b^n \mid m = n^p \mid p \geq 1\}$ [Not R.L.] ×

(13.) $L = \{a^m b^n c^q \mid m = n = q\}$ [Not R.L] ×

(14.) $L = \{a^m b^n c^q \mid m + n = q\}$ [Not R.L] ×

(15.) $L = \{a^m b^n \mid m + n = \text{even}\}$ [~~not~~ R.L] ✓

(16.) $L = \{a^m b^n \mid m + n = \text{odd}\}$ [~~Not~~ R.L.] ✓

	a	b
o + o		e
o + e	o	
e + o	o	
e + e	e	

(17.) $L = \{wcw^r \mid w \in \Sigma^*\}$ × [Not R.L] $\Sigma = \{a, b\}$

abb (c) bba

w ↑ w^r

(18.) $L = \{wcw \mid w \in \Sigma^*\}$ × [Not R.L.]

(19.) $L = \{wcw^r \mid w \in \Sigma^+\}$ × [Not R.L.]

(20.) $L = \{wcw \mid w \in \Sigma^+\}$ × [Not R.L.]

(21.) $L = \{w.w \mid w \in \Sigma^*\}$ × [Not R.L.]

(22.) $L = \{w.w^r \mid w \in \Sigma^+\}$ × [Not R.L.]

(23.) $L = \{wcw^r \mid c, w \in \Sigma^*\}$ ✓ [R.L.]

(24.) $L = \{cww^r \mid c, w \in \Sigma^*\}$ ✓ [R.L.]

(25.) $L = \{ww^rc \mid c, w \in \Sigma^*\}$ ✓ (R.L.)

*** Most IMP cases.

Scanned by CamScanner

IATE: PRACTICE PROBLEMS ON REG. Lang.

26. $L = \{w\, c\, w^r \mid c, w \in \Sigma^+\}$ ✓ [R.L.]

27. $L = \{c w w^r \mid c, w \in \Sigma^+\}$ ✗ [Not R.L.]

28. $L = \{w w^r c \mid c, w \in \Sigma^+\}$ ✗ [Not R.L.]

29. $L = \{w\, c\, w \mid c, w \in \Sigma^*\}$ ✓ [R.L.]

30. $L = \{c\, w w \mid c, w \in \Sigma^*\}$ ✓ [R.L.]

31. $L = \{w w c \mid c, w \in \Sigma^*\}$ ✓ [R.L.]

32. $L = \{w c w \mid c, w \in \Sigma^+\}$ ✗ [Not R.L.]

33. $L = \{c w w \mid c, w \in \Sigma^+\}$ ✗ [Not R.L.]

34. $L = \{w w c \mid c, w \in \Sigma^+\}$ ✗ [Not R.L.]

Scanned by CamScanner

Regular Expressions:

↳ Regular Expression (R.E.) is a way of representing Regular Language.

↳ Expression of strings and operators like

i) * Kleen closure (a^*)

ii) + Positive closure (a^+)

(iii) . concatination ($a.b$)

(iv) + Union ($a+b$)

↳ A language is R.L. if there exist a Regular Exp^r for it.

⇒ "R.E. is said to be valid iff it can be derived from the primitive RE by a finite number of application of the rule r_1^*, r_1^+, $r_1 \cdot r_2$, $r_1 + r_2$."

⇒ "if Σ is a given alphabet then, ϕ, $\in$ / λ, $a \in \Sigma$ are primitive R.E.

#

(1.) $r = \phi$, $L(r) = \{\}$ / ϕ

(2.) $r = \in$, $L(r) = \{\in\}$

(3.) $r = a$, $L(r) = \{a\}$

(4.) $r = a+b$, $L(r) = \{a, b\}$

(5.) $r = a \cdot b$, $L(r) = \{ab\}$

(6.) $r = a+b+c$, $L(r) = \{a, b, c\}$

(7.) $r = (b+a) \cdot b$, $L(r) = \{abb, ab\}$

~~(8.) $r = (b+a)^* \cdot b$~~

(8.) $r = a^+$, $L(r) = \{a, aa, aaa, \ldots\}$

(16.) $r = (ab)^*$; $L(r) = \{ab, abab, \ldots\}$

(9.) $r = a^*$, $L(r) = \{\in, a, aa, \ldots\}$

(10.) $r = (a+ba)(b+a)$, $L(r) =$
$L(r) = \{ab, aa, bab, baa\}$

(11.) $r = (a+\in)(b+\phi) = (a+\in)(b)$
$L(r) = \{ab, b\}$

(12.) $r = (a+b)^2$; $L(r) = \{aa, bb, ab, ba\}$

(13.) $r = (a+b)^*$
$L(r) = \{\in, a, b, aa, ab, bb, \ldots\}$

(14.) $r = (a+b)^*(a+b) = (a+b)^+$
$L(r) = \{a, b, aa, bb, ab, \ldots\}$

(15.) $r = a^* \cdot a^*$
$L(r) = \{\in, a, aa, \ldots\}$

Scanned by CamScanner

$r = \epsilon^*$; $L(r) = \{\epsilon, \epsilon^1, \epsilon^2, \ldots\}$
$L(r) = \{\epsilon\}$

$r = \epsilon^+$; $L(r) = \{\epsilon\}$

$r = \phi^*$; $L(r) = \{\epsilon\}$

$r = \phi^+$; $L(r) = \{\} = \phi$

$r_1 = a^*$

$r_2 = a^* + (aa)^*$

a) $L(r_1) \subseteq L(r_2)$

b) $L(r_1) \supseteq L(r_2)$

✓ c) $L(r_1) = L(r_2)$

d) $L(r_1) \neq L(r_2)$

→ ~~For~~ One language: It is possible to generate more than one R.E. for one Lang.

1) $|w|_a =$

12) $|w$

12

(1.) $r^+ \cup r^* = r^*$

(2.) $r^+ \cap r^* = r^+$

(3.) $r^* . r^+ = r^+$

(4.) $(r^*)^* = r^*$

(5.) $(r^+)^* = r^*$

(6.) $(r^*)^+ = r^*$

(7.) $((r^*)^+)^* . r^+ = r^+$

(8.) $(a+b)^* = (a^* + b)^*$ ✓

(9.) $(a+b)^* = (a+b^*)^*$ ✓

(10.) $(a+b)^* = (a^* + b^*)^*$ ✓

(11.) $(a+b)^* \neq (a.b)^*$ ✗

(12.) $(a+b)^* \neq (a^* . b)^*$ ✗

(13.) $(a+b)^* \neq (a.b^*)^*$ ✗

(14.) $(a+b)^* = (a^* . b^*)^*$ ✓

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

R.E. start with ab

$ab\,(a+b)^*$

② start with bba

$bba\,(a+b)^*$

③ ~~start~~ ends with abb

$(a+b)^* abb$

④ contain a substring aab

$(a+b)^* aab\,(a+b)^*$

⑤ start and ends with a

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

⑥ start & ends with same symbol

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

⑦ start and ends with diff symbol

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

⑧ $|w| = 3$

$(a+b)(a+b)(a+b)$

⑨ $|w| \geq 3$

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

⑩ $|w| \leq 3$

$\epsilon + (a+b) + (a+b)^2 + (a+b)^3$

$\Rightarrow (a+b+\epsilon)^3$

8

Scanned by CamScanner

(11) $|w|_a = 2$
$b^* a b^* a b^*$

(12) $|w|_a \geq 2$
$(a+b)^* a (a+b)^* a (a+b)^*$

(13) $|w|_a \leq 2$
$\epsilon + b^* + b^* a b^* + b^* a b^* a b^* = b^* (a+\epsilon) b^* (a+\epsilon) b^*$

(14) 3rd symbol from left end is 'b'
$(a+b)^2 b (a+b)^*$

(15) 28th symbol from right end is 'a'.
$(a+b)^* a (a+b)^{27}$

(16) $|w| = 0 \pmod 3$
$[(a+b)^3]^*$

(17) $|w| = 2 \pmod 3$
$(a+b)^2 [(a+b)^3]^*$

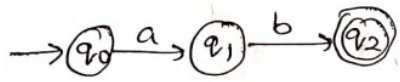
(20) $|w|_b = 0 \pmod 2$
$[(a+b)^2]^* a (a^* b a^* b a^*)^*$
$\Rightarrow a^* (a^* b a^* b a^*)^*$

(21) $|w|_a = 1 \pmod 3$
$b^* a b^* (b^* a b^* a b^* a b^*)^*$

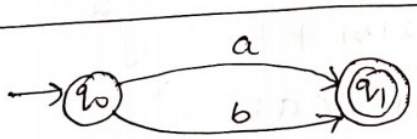
(22) $|w|_b = 2 \pmod 3$
$a^* b a^* b a^* (a^* b a^* b a^* b a^*)^*$

? FA ↓↑ RL RE

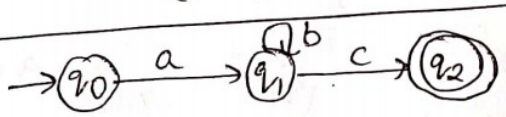
Conversion of F.A. to R.E.

(1)


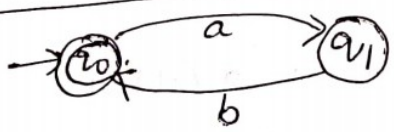
$(a.b)$

(2)


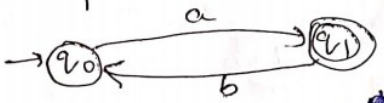
$(a+b)$

(3)


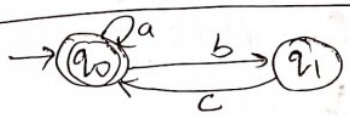
(ab^*c)

(4)


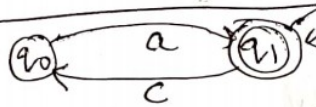
$(ab)^*$

(5)


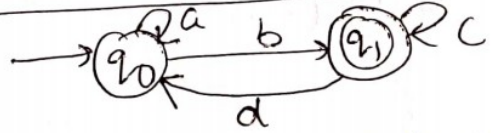
$a(ba)^*$

(6)


$(a+bc)^* = (a^* + (b \cdot c^*))^*$

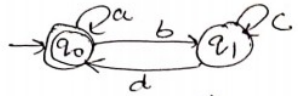
(7)


$a(b+ca)^*$

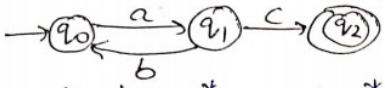
(8)


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

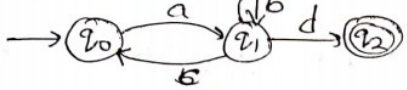
Scanned by CamScanner

(9.) 

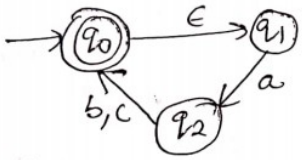
$(a + bc^*d)^*$

(10) 

$a(ba)^*(ab)^*c = a(ba)^*c$

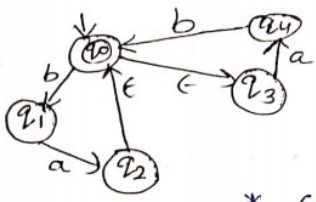
(11.) 

$a(b+ca)^*d$

(12.) 

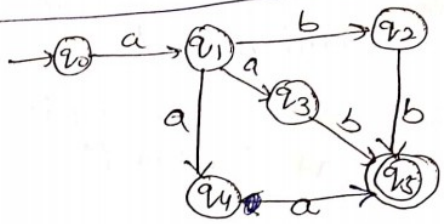
R.E.
↳ $[\epsilon . a(b+c)]^*$

$[a(b+c)]^*$

(13.) 

$R.E. = ((ba\epsilon)^* + (\epsilon ab)^*)^*$

$= ((ba)^* + (ab)^*)^* = (ba+ab)^*$

(14.) 

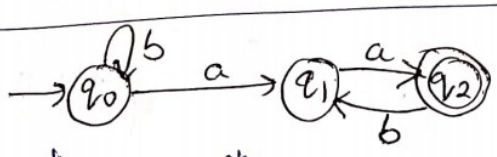
$a(bb + ab + aa)$

(15.) 

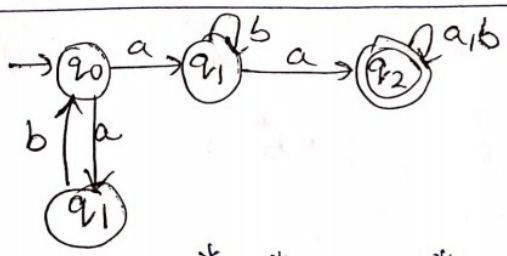
$[a.(a+b) + b(a+b)]^*$

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

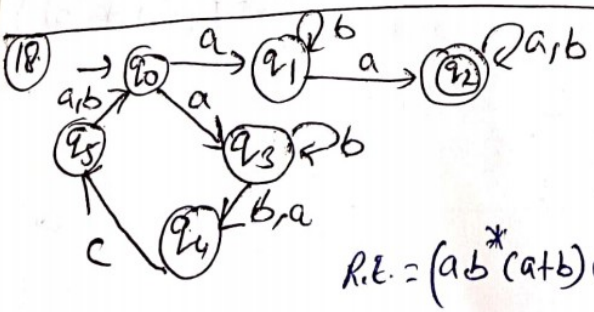
$[(a+b)^2]^*$

(16) 

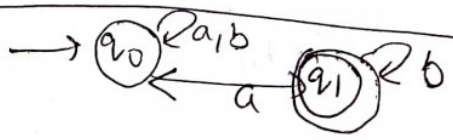
$b^*aa(ba)^*$

(17) 

$R.E. = (aa)^*ab^*a(a+b)^*$

(18) 

$R.E. = (ab^*(a+b)c(a+b))^*ab^*a(a+b)^*$

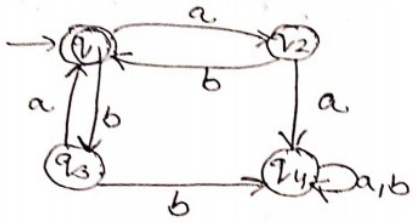
(19.) 

$R.E = \phi$

Find R.E.'s

Scanned by CamScanner

Find R.E. of



$q_1 = q_2 b + q_3 a + \epsilon$

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$

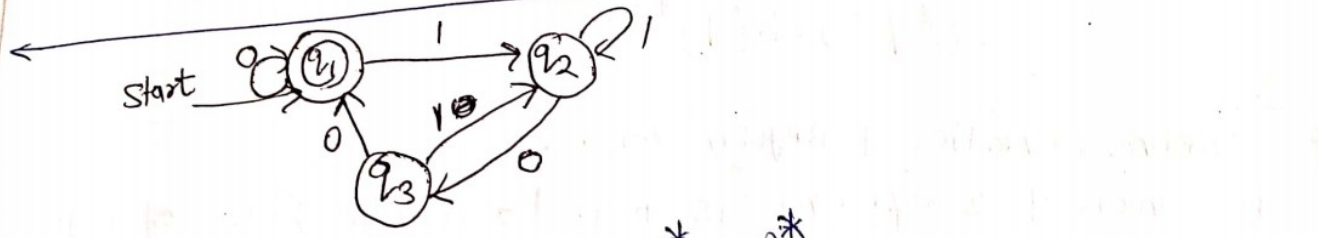
Put q_2 and q_3 in q_1 as

$q_1 = q_1 ab + q_1 ba + \epsilon$

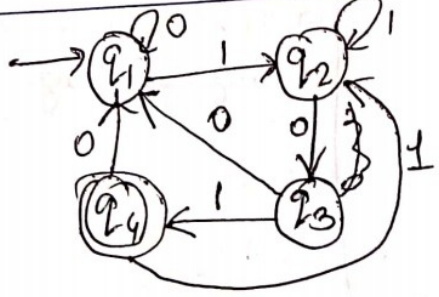
$q_1 = \epsilon + q_1 (ab + ba)$

$$\boxed{q_1 = \epsilon (ab+ba)^*}$$

So, R.E. = $(ab+ba)^*$.



$$R.E = (0 + 1(1+01)^* 00)^*$$



$$\boxed{R.E. = (0+1(1+011)^*(00+010))^* (1(1+011)^* 01)}$$

=

Scanned by CamScanner

Algebraic Laws for R.E.

① Associativity & commutativity. $(P+Q)=(Q+P)$ $(PQ)R=P($

② Identities.

③ Distributive

④ Idempotent

⑤ Law for closure

$(L^*)^*=L^*$

$\phi^*=\epsilon$

⑥ Identities for R.E. :-

→ $\phi+r=r$

→ $\phi r = r\phi = \phi$

→ $\epsilon r = r\epsilon = r$

→ $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

→ $r+r=r$

→ $r^*r^*=r^*$

→ $rr^*=r^*r=r^+$

→ $\epsilon + rr^* = r^* = \epsilon + r^*r$

→ $(r^*)^* = r$

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

→ $(PQ)^*P = P(QP)^*$

→ $(P+Q)R = PR+QR$

→ $R(P+Q) = RP+RQ$

⇒ Prove that

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

Closure Properties of Regular Lang. :

① Union of 2 reg. langs is Reg.

② Intersection of 2 reg. langs is Reg.

③ Complement " reg. " " "

④ difference of 2 reg " " "

⑤ Reversal of reg " " "

⑥ closure " " " " "

⑦ Concatenation " " " " "

⑧ Homomorphism of reg. lang. is Regular.

⑨ Inverse Homomorphism of regular lang. is Regular.

Scanned by CamScanner

(P+Q)R=PQ

Construction of R.E. from DFA:
Using Arden's Theorem:-

Arden's Theorm :→

Let P and Q be two Regular Expression over alphabet Σ, If P does not contain null string ∈, then

$$\boxed{R = Q + RP}$$

has a unique solution that is $R = QP^*$.

Proof:-

Put the value of R in R.H.S.

$$R = Q + RP$$

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

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

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

$$= Q + QP + QP^2 + RP^3$$

$$R = Q(\epsilon + P + P^2 + P^3 + \cdots)$$

$$\boxed{R = QP^*}$$

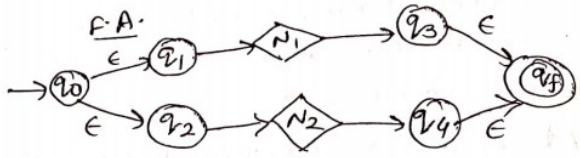
Hence proved.

↳ To use Arden's Theorm some assumptions should be exist regarding Transition systems

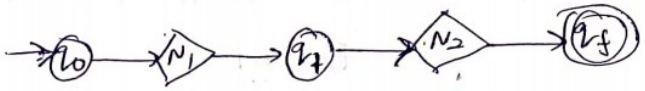
(i) Transition diagram should not have ∈-Transitions.
(ii) It must have only a single initial state.
(iii) It's n states are $q_1, \ldots q_n$.
iv) q_i is final state.
v) w_{ij} denotes R.E. q_i to q_j.

Scanned by CamScanner

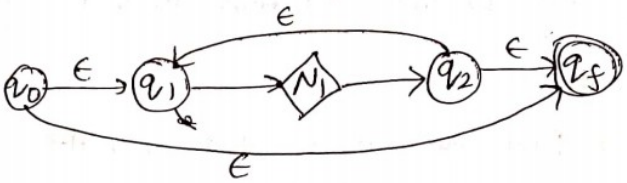
R.E. $(r_1 + r_2)$ → F.A.



$(r_1 . r_2)$ →



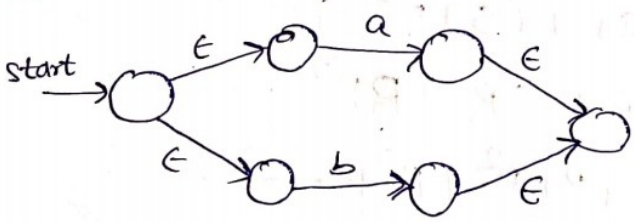
r^* →



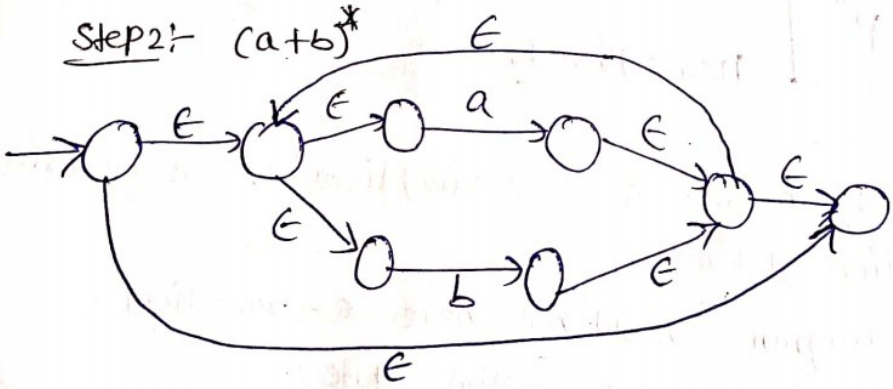
Conversion :-

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

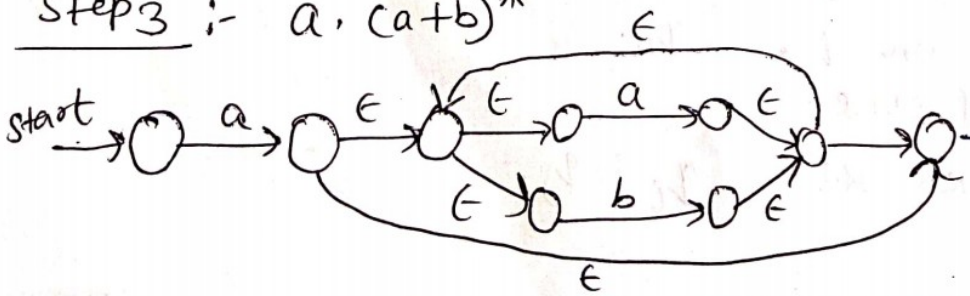
Step 1 :- $(a+b)$



Step 2 :- $(a+b)^*$



Step 3 :- $a . (a+b)^*$



Step 4 :- $a . (a+b)^* . b$

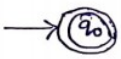
Step 5 : $a . (a+b)^* . b . b$

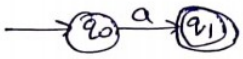
Dug

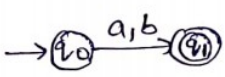
Scanned by CamScanner

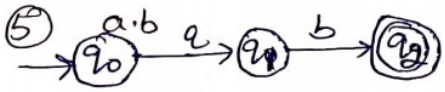
Conversion from R.E. to FA :-

① ϕ 

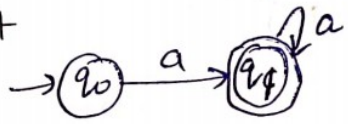
② ϵ 

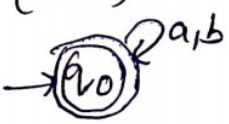
③ a 

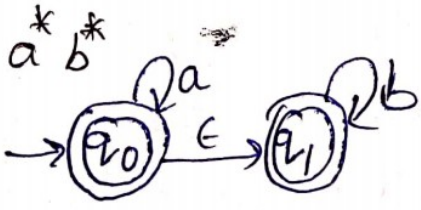
④ a+b 

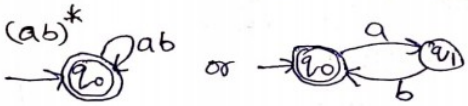
⑤ 

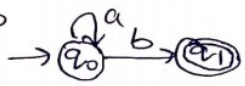
⑥ a^* 

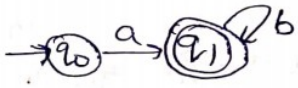
⑦ a^+ 

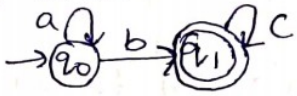
⑧ $(a+b)^*$ 

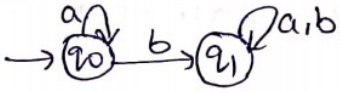
⑨ $a^* b^*$ 

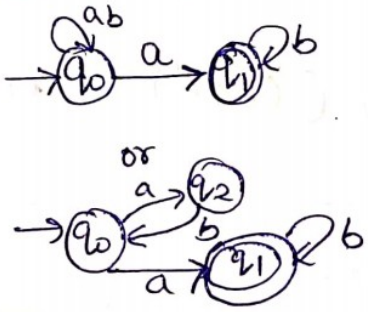
⑩ $(ab)^*$ 

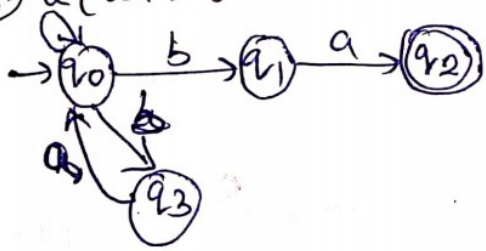
⑪ $a^* b$ 

⑫ $a b^*$ 

⑬ $a^* b c^*$ 

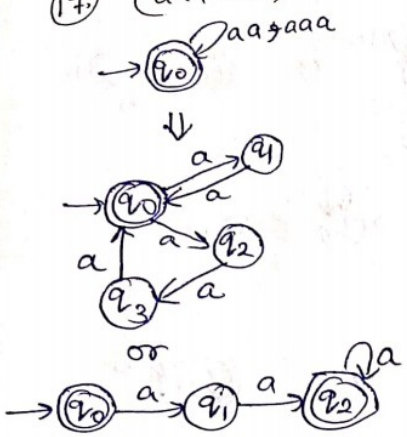
⑭ $a^* b (a+b)^*$ 

⑮ $(a+b)^* a b^*$ 

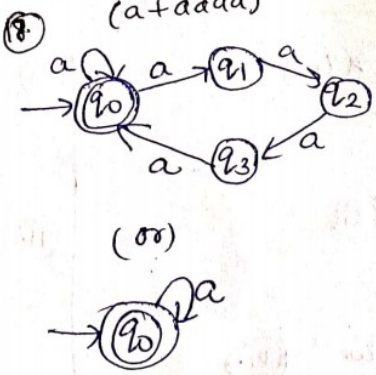
⑯ $a(a+ba)^* ba$ 

Scanned by CamScanner

(17.) $(aa+aaa)^*$



(18.) $(a+aaaa)^*$



Practice Question :-

Conversion from R.E. to FA :-

(1.) $(ab)^* + (a+ab)^* b^* (a+b)^*$

(2.) $[a + ba(a+b)]^* a(ba)^* b^*$

(3.) $b(a+ba+abb)(ba(a+b)^*)$

(4.) $(a+b+ca)((bab^*)+(a+b)^*)^*(ab)^*$

Scanned by CamScanner

Equivalence of Finite Automata and Regular Expressions

↳ Regular Expressions & Finite Automata are equivalent in their descriptive power.

↳ Any R.E. can be converted into a Finite automaton that recognises the language it describes & vice-versa.

↳ "A Regular language is one that is recognised by some finite automata."

↳ "The language accepted by F.A., are the languages denoted by R.E.s."

↳ "Regular Expressions define the class of languages that are accepted by finite automata."

This implies that:-

(i) Every language accepted by a F.A. can be defined by a R.E.

(ii) For every Reg. Expr (R.E.), there is an equivalent NFA with ϵ-transitions.

⟹ Cycle of constructions:-

↳ Regular Expr (RE) → Non-deterministic F.A. (NFA) → DFA → Minimized DFA (MDFA)

↳ NFA → DFA (subset construction)

- DFA → NDFA
- RE → NFA (build an NFA for each term; combine them with ϵ)
- DFA → RE

Scanned by CamScanner

Equivalence of DFA and R.E:-

If L is accepted by a DFA, then L can be expressed by a Regular Language.

⇒ Construction of R.E. from given DFA:

Step 1: For each of states $q_1, \ldots q_n$ in DFA, write down the equations by considering all edges, that enter into that state.

Step 2: for initial state of DFA, the equation is added with ϵ.

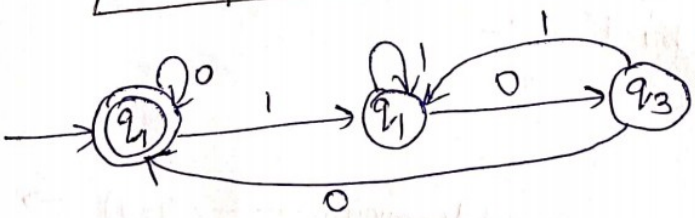
Step 3: Compute the equation for each state.

Step 4: Substitute the results of each state equation into final state equation of DFA to get a RE for DFA.

Ex:- construction of R.E. corresponding to DFA.

Q \ Σ	Present I/Ps 0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_2

Sol:-



Transition diagram

The equ of each state:

$$q_1 = q_1 0 + q_3 0 + \epsilon$$

$$q_2 = q_1 1 + q_2 1 + q_3 1$$

$$q_3 = q_2 0.$$

Substitute q_3 ⇒

Scanned by CamScanner

Substitute q_3 in q_2 :-

$$\Rightarrow q_2 = q_1 1 + q_2 1 + q_2 01$$
$$= q_1 1 + q_2 (1+01)$$
$$= q_1 1 (1+01)^* \quad \text{By using Arden's Thm.}$$

Now,

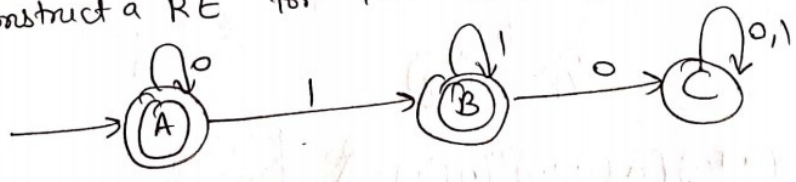
$$q_1 = q_1 0 + q_3 0 + \epsilon$$
$$= q_1 0 + (q_2 0) 0 + \epsilon$$
$$= q_1 0 + q_1 1 (1+01)^* .00 + \epsilon$$
$$= q_1 (0 + 1 (1+01)^* .00) + \epsilon$$
$$= \epsilon (0+1 (1+01)^* .00)^* \quad \text{By using Arden's Thm.}$$

$$\boxed{q_1 = (0+1(1+01)^* 00)^*}$$

Since, q_1 is the final state, RE for DFA will be :-

$$\boxed{RE = (0+1(1+01)^* 00)^*}$$

Ex.2) Construct a RE for the DFA:



Sol:

$$A = A0 + \epsilon$$
$$B = A1 + B1$$
$$C = B0 + C(0+1)$$

Consider:- using Arden's thm:

$$A = A0 + \epsilon \Rightarrow A = \epsilon 0^* \Rightarrow A = 0^*$$

Now, Substitute A in B :-

$$A1 + B1 \Rightarrow 0^* 1 + B1$$

Scanned by CamScanner

$B = 0^*1 + B1$ By using Arden's Thm.

$B = 0^*11^*$

since, A & B are final states in above DFA.

$\Rightarrow A+B = 0^* + 0^*1^*$

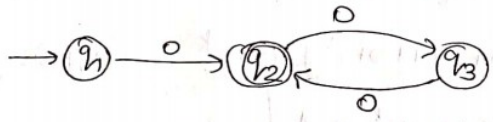
$= 0^* (\epsilon + 1^*)$

$= 0^* (\epsilon + 11^*) = 0^*1^* \quad (\because \epsilon + RR^* = R^*)$

R.E. $= 0^*1^*$

---

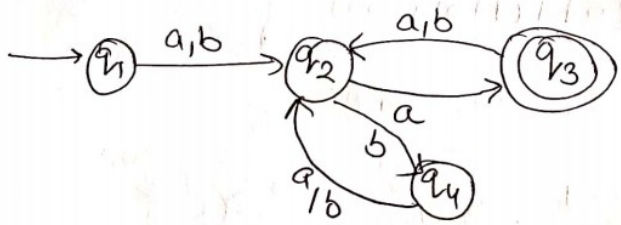
(3) Construct RE for DFA.



Ans. **RE $= 0(00)^*$**

---

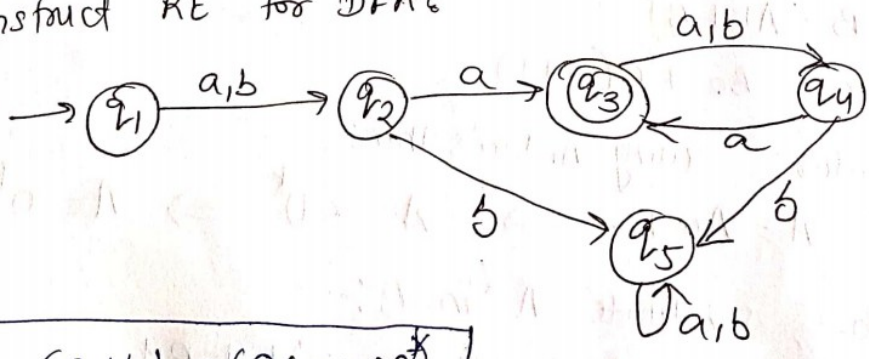
(4) Construct RE for DFA:



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

---

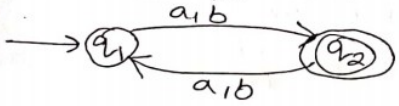
(5) Construct RE for DFA:



R.E. $= (a+b) \cdot a(a(a+b))^*$

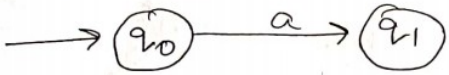
Scanned by CamScanner

Construct a RE for DFA:

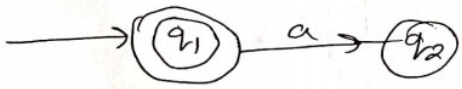


$R.E. = (a+b)(a+b)^*$

No final state

7. 

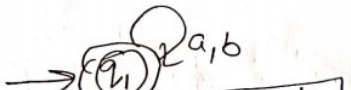
$R.E. = \phi$

8. 

$R.E. = \epsilon$

9. 

$R.E. = b^*$



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

$R.E. = b^*$

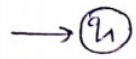
10. 

$R.E. = (ab)^*$

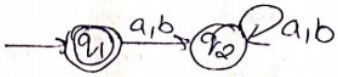
Scanned by CamScanner

Construction of DFA for a given R.E. :-

① R.E. = ϕ



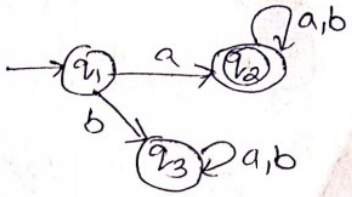
② R.E = ϵ over $\Sigma = \{a, b\}$



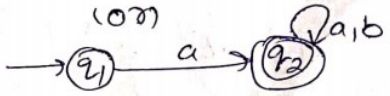
(OR)

③ R.E. = $(a+b)^*$

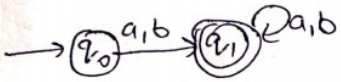
④ ~~R.E~~ = $a(a+b)^*$



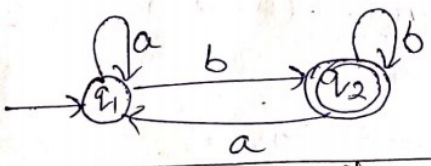
(OR)



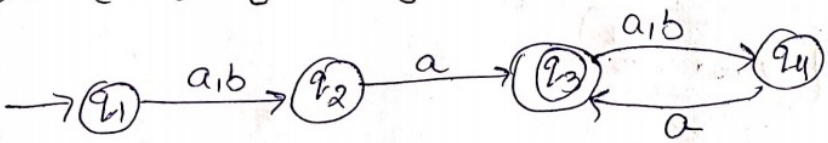
⑤ RE = $(a+b)(a+b)^*$ $\Sigma = \{a, b\}$ except ϵ



⑥ R.E. = $(a)^*(ba)^*b^*$ over $\Sigma = \{a, b\}$ except ϵ



⑦ R.E. = $(a+b)a\,[(a+b)a]^*$



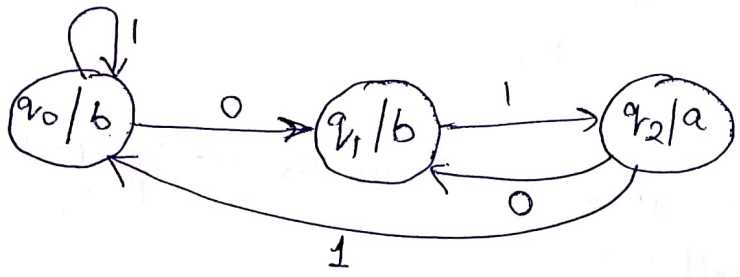
⑧ R.E. = $(a+b)^*aa(a+b)^*$ (TRY YOURSELF)

Scanned by CamScanner

Conversion of Moore to Mealy Machine:-

Eg. Construct a Moore machine that prints 'a' whenever the sequence '01' is encountered in any i/p binary string and then convert it to its equivalent mealy machine.

Moore Machine:-



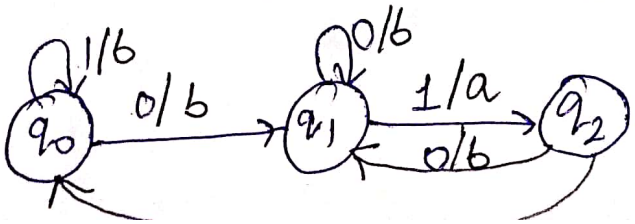
↳ State transition table for the moore machine:-

State	I/P		O/P
	0	1	
q_0	q_1	q_0	b
q_1	q_1	q_2	b
q_2	q_1	q_0	a

↳ State transition table for Mealy Machine:-

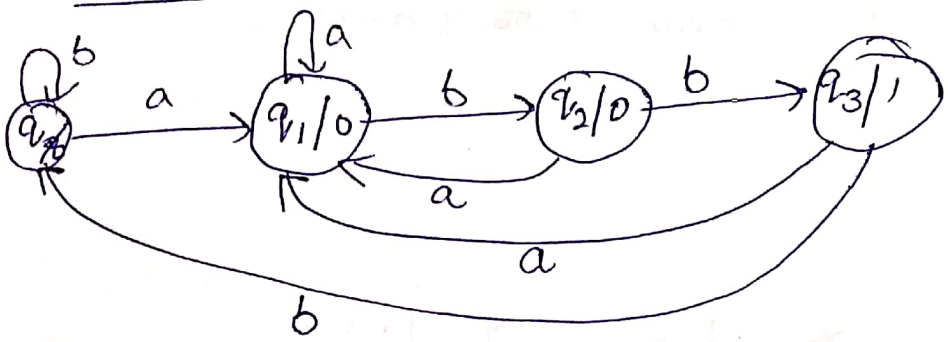
State	I/P & O/P	
	0	1
q_0	q_1, b	q_0, b
q_1	q_1, b	q_2, a
q_2	q_1, b	q_0, b

↳ State transition diagram for mealy M/c:-



eg.2.)
Counts the occurance of 'abb'.

Moore M/C :



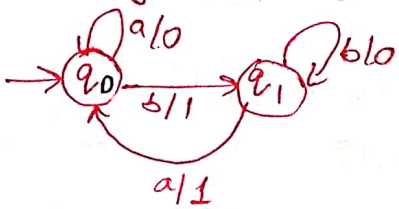
Conversion of the given Moore M/C to Mealy Machine →
TRY it Yourself !

eg.3.) Convert the following Moore M/C table to Mealy M/C :-

Moore M/C Transition Table

State	i/p		o/p
	0	1	
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

Convert Mealy machine into an equivalent Moore M/c.



Sol:- Given the mealy machine $M_e = (\{q_0, q_1\}, \{a, b\}, \{0, 1\}, \delta, \lambda, q_0)$

we construct an equivalent Moore m/c $M_0 = (Q_x, \Sigma, \Gamma, \delta', \lambda', [q_0, b_0])$

where $Q_x = \{[q_0, 0], [q_0, 1], [q_1, 0], [q_1, 1]\}$

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

We need to find out:

$$\delta'([q,b], a) = [\delta(q,a), \lambda(q,a)] \quad \&$$

$$\lambda'(q,b) = b.$$

Thus,

$$\delta'([q_0,0], a) = [\delta(q_0,a), \lambda(q_0,a)]$$
$$= [q_0, 0].$$

$$\delta'([q_0,0], b) = [\delta(q_0,b), \lambda(q_0,b)]$$
$$= [q_1, 1]$$

$$\delta'([q_0,1], a) = [\delta(q_0,a), \lambda(q_0,a)]$$
$$= [q_0, 0]$$

$$\delta'([q_0,1], b) = [\delta(q_0,b), \lambda(q_0,b)]$$
$$= [q_1, 1]$$

$$\delta'([q_1,0], a) = [\delta(q_1,a), \lambda(q_1,a)]$$
$$= [q_0, 1]$$

Similiarly, $\delta'([q_1,0],b) = [q_1,0]$

$$\delta'([q_1,1],a) = [\delta(q_1,a), \lambda(q_1,a)]$$
$$= [q_0,1]$$
$$\delta'([q_1,1],b) = [\delta(q_1,b), \lambda(q_1,b)]$$
$$= [q_1,0].$$

We also find: $\lambda'(q,b) = b$ $\forall$ states i.e.;

$\lambda'[q_0,0) = 0$

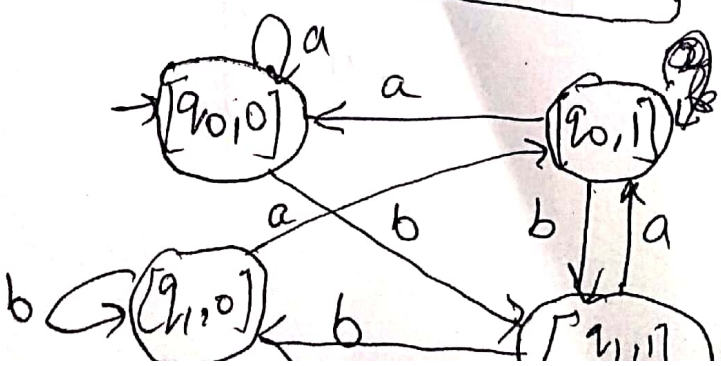
$\lambda'(q_0,1) = 1$

$\lambda'(q_1,1) = 1$

$\lambda'(q_1,0) = 0$

Transition Table :- (for moore M|c)

q_x | Σ	a	b
$[q_0,0]$	$[q_0,0]$	$[q_1,1]$
$[q_0,1]$	$[q_0,0]$	$[q_1,1]$
$[q_1,0]$	$[q_0,1]$	$[q_1,0]$
$[q_1,1]$	$[q_0,1]$	$[q_1,0]$



} equivalent Mo for above Me.

Difference between Moore & Mealy M/c:

(1.) Moore M/Cs gives output $\lambda(q_0)$ in response to input ϵ, so the o/p sequence is $(n+1)$ ~~or~~ not n.

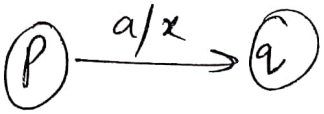
; Mealy M/c does not give output $\lambda(q_0)$ in response to input ϵ, so the o/p sequence is 'n' not $(n+1)$.

(2.) Moore M/c has actions associated with states or moore M/c prints characters, when in state.
Mealy M/c has actions associated with transitions or Mealy M/c prints characters, when traversing an arc.

(3.) The output of moore finite state machine depends only on the current state and doesnot depend on current i/p.
The output of mealy finite state machine depends only on the current ~~state~~ input.

(4.) In moore Machine, if o/p associated with state q is 'x', then q/x is written inside the state circle of transition diagram.
In Mealy m/c, if o/p associated with ~~state~~ edge labelled with letter 'a' is x, it is written as a/x on edge.

M_0



M_e