Automata theory & computability.
Subject code: 18CS54 CIE Marks: 40
SEE marks: 60

Course Outcomes:
* Acquire fundamental understanding of the core concepts in automata theory and Theory of computation.
* Learn how to translate between different models of computation (Det, Non-det & s/w models)
* Design Grammars & automata for different language classes and become knowledgeable about restricted models of computation and their relative powers.
* Develop skills in formal reasoning and reduction of a problem to a formal model, with an emphasis on semantic precision and conciseness.
* Classify a problem w.r.t different models of computation.

What is Automata theory & computability.

Course learning ~~Outcomes~~: Objectives:
1) Introduce core concepts in Automata & T of C
2) Identify diff formal language classes & their relationships
3) Design grammars & recognizers for diff formal lang.
4) Prove or disprove theorems in AT using their properties
5) Determine the decidability & interactability of computational problems.

Attainment level
>= 70% = 3
= 60% ~~-~~ 70% = 2
~~< 50% = 1~~

Introduction
Computability & modeling
Applications of AT
Diff Natural lang & formal lang.
Various types of automata [Chomsky's hierarchy]
Pictorial representation of FA.
Symbols used in FA
FSM Definition
Example of FSM
Types of FA
Deterministic DFSM definition
Example.
DFA design techniques.

Languages:

A language is a set of strings over finite alphabet Σ.
Formally, a lang. L is subset of Σ^* denoted by $L \subseteq \Sigma^*$

Ex:

* A lang of strings consisting of equal number of 0's and 1's can be represented as,
  $\{\varepsilon, 01, 10, 0011, 1010, 0101 \ldots\}$
* A lang of strings consisting of ~~equal~~ 'n' number of 0's & 1's
  'n' no of
  $\{\varepsilon, 01, 0011, 000111, \ldots\}$
* A lang containing empty string ε is denoted by $\{\varepsilon\}$
* An empty lang is denoted by ϕ.

Formal lang. description.

1) Lang. that contains string consisting of any no of a's & b's
   $L = \{w \in \{a,b\}^*\}$
2) Lang. that contains strings of a's and b's where all a's precede b's.
   $L = \{w \in \{a,b\}^* : \text{all a's precede all b's in } w\}$
   OR
   $L = \{a^i b^j \mid i \geq 1, j \geq 1\}$
3) Lang that contains strings of a's and b's ending with 'a'.
   $L = \{w : w = ya \text{ and every } y \in \{a,b\}^*$
   or
   $L = \{w \in \{a,b\}^* : \text{last character of } w \text{ is } a\}$
4) Empty lang.
   $L = \{\} = \phi$
5) Lang with empty string
   $L = \{\varepsilon\}$
6) Lang. consisting of a's and b's which donot start with b.
   $L = \{w \in \{a,b\}^* : \text{no prefix of } w \text{ starts with } b\}$
7) Lang consisting of any number of a's.
   $L = \{w \in \{a\}^*\}$
   or
   $L = \{a^n : n \geq 0\}$
   or
   $L = \{\varepsilon, a, aa, aaa, \ldots\}$

Cardinality of a language:

The length of a lang. is called cardinality of a lang. i.e., no of numbers of a set is called the cardinality.

Ex: $L = \{\}$. The cardinality of the empty set is zero.

$L = \{\varepsilon\}$. ——————"—————— language is one.

Theorem:- If $\Sigma = \phi$ then Σ^* is countably infinite.

By definition Σ^* is set of strings of length 0, length 1, length 2 and so on.

Since Σ^* contains strings of any length, is it countably infinite. That is all the lang are either finite or countably infinite.

Why theory of computations: [Applications]

* Used to design languages enabling machine/ ~~machine~~ ~~comm~~ and machine communication. [n/w comm protocol]
  Ex: NCP, HTML.
* Used to design and implement modern programming lang such as C/ C++ / Java/ Python etc.
* Used to design vending machines (ATM), comm protocols and also used in building security devices.
* Used to design interactive video games.
* DNA molecules can be analysed with the help of fsm and context free grammars.
* AI programs are used to solve various problems ranging from medical diagnosis to factory scheduling.
* Many natural structures like organic molecules and CN can be modelled as graphs and the solution can be obtained using very efficient graph algorithms.

The variations of power of an alphabet are:

* Kleene closure (Kleene star)
* Kleene plus

The Kleene closure Σ^* is defined as,

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$ which is the set of words of any length.

Ex:- Let $\Sigma = \{0, 1\}$ then Σ^* is,

$\Sigma^0 = \{\varepsilon\}$

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 100, 011, 101, 110, 111\}$

$\vdots$

so

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

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

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

Kleene plus: Σ^+ is defined as,

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$ which is the set of words of any length except the null string i.e., ε, (Σ^0) is not included.

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

$\Sigma^* = \Sigma^+ \cup \varepsilon$ & $\Sigma^+ = \Sigma^* - \varepsilon$

Functions of Strings:

Various string functions used for processing string are:

* Length
* Concatenation
* Replication
* Reversal.

The length of a string w denoted by $|w|$ is the number of symbols in w.

$|\varepsilon| = 0$

$|abcdef| = 6$

A length function is defined as:

$\#_c(w)$ [No. of times 'c' appears in string w]

Ex: $\#_a(abababa) = 4$

$\#_b(abababa) = 3$

$\#_c(abababa) = 0$

Concatenation: uv = ComputerScience $\quad |uv| = |u| + |v| = 8 + 7 = 15$

Replication: The process of repeating a string pattern w by a specified number 'i' is called replication. denoted by w^i.

$w^0 = \varepsilon$

$a^4 = aaaa$

$b^2a^2 = bbaa$

$(ba)^2 = baba$

$a^0b^3 = bbb$

Reversal: $u = a_1 a_2 a_3 \ldots a_n$

$u^R = a_n a_{n-1} a_{n-2} \ldots a_3 a_2 a_1$

Ex: $w = \varepsilon \quad w^R = \varepsilon$

$w = aabb \quad w^R = bbaa$

Theorem 1.1: If w and x are strings then prove that

$$(wx)^R = x^R w^R$$

Base case: $|x| = 0 \Rightarrow x = \varepsilon$

So $(wx)^R = (w\varepsilon)^R = (w)^R = (\varepsilon w)^R = \varepsilon w^R = \varepsilon^R w^R = x^R w^R$

Hence $(wx)^R = x^R w^R$

Inductive hypotheses: For each $n \geq 0$ where $|x| = n$, assume $(wx)^R = x^R w^R$

To prove: To prove that hypothesis is true for $|x| = n+1$

Consider x, when $|x| = n+1$

Let $x = ua$, for some character a and $|u| = n$

$(wx)^R = (w(ua))^R$ replacing x by ua

$= ((wu)a)^R$ associativity of concatenation

$= a(wu)^R$ by definition of reversal

$= a(u^R w^R)$ by induction hypotheses

$= (au^R)w^R$ associativity of concatenation

$= (ua)^R w^R$ by definition of reversal

$= x^R w^R$ replacing ua by x.

Hence proof.

Logic of Computations

Automata theory & Computability:

We will study this in a top down approach, applications first & then the rudiments.

What is an Automata? What is computability? Why study the subject & its applications.

Automata: Its an abstract machine for modeling computations. [Natural lang & formal lang]

An abstract machine (not hard-wired, an abstract m/n) allows us to model the essential parameters and ignore the non-essential parameters.

Modeling - essential is engineering

Eg: Civil Engineer. [stress, strain, parameters & calculations]

What is computable? Add 2 numbers, find roots of a q.e, multiple 2 matrices: All the examples have algorithms

What is not computable? No algorithm. [We know, we program]

[Halting problem of a program]

[Computable, but useless. Ex: Travelling Salesman problem (DAA)

min distance

Time taken is max [NP-problems (non-deterministic polynomial problems)]

Applications of Automata theory:

* In compilers: Lexical analysis [Count = count + 1;]
  Seperating a statement into tokens - LA
  Parser generators [parsing - checking syntax]
* Modeling the circuits: Example: On-off switch

* Word search & translation of Natural languages.
* Parity checkers [even parity i/p : same o/p else info corrupted]
* Video games
* ~~DNA~~
* Security
* Artificial Intelligence.

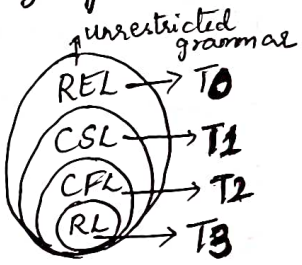
Natural lang - Context sensitive languages.

~~Types of Automata~~: Eg: Charge [Cost, charge of battery, command, positive charge]

~~Different~~ Formal lang → [precise meaning] Not dependent of on context. Eg: C=C+1;

Gist of the subject: A systematic way of depicting the problem so that its solution can be understood and analysed.

Various types of Automata (computer) for recognizing different languages. Chomsky's Hierarchy



DFA - RL [No memory] (Handles tokens seperately)

(Compiler / Intepreter)
PDA - CFL Eg:- C, C++, java

Linear bounded Automata
LBA - CSL } To understand the limitations of DFA & PDA.
TM - REL } (Recursively Enumerable language (Semi-decidable language) not completely decidable - partially.

Finite State Machine (DFSM), M is a quintuple

$(K, \Sigma, \delta, s, A)$

K is a finite set of states

Σ is the i/p alphabets

$s \in K$ is the start state

A is the subset of K, set of accepting/final states

δ is the transition function which maps,

$K \times \Sigma$ to K

When we design a machine FSM, it must include all 5 parameters.

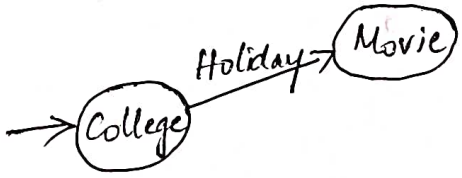
How to draw the transition diagram of FSM:

Notation for ~~start~~ state (1)

Notation for transition $\xrightarrow{b}$

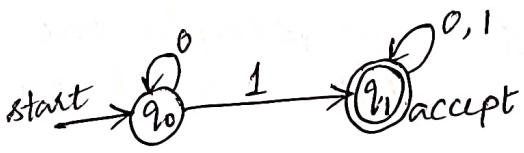
Notation for start state → (2)

Notation for accepting state (3)



Various simple notations used to specify DFA are:

* Transition diagram (Transition graph)
* Transition table



Transition diagram

- Two states q_0 and q_1
- Two i/p symbols 0 & 1
- start state.
- The final states are represented by 2 circles.
- Transition representation from state q on i/p symbol a to state p is represented by a directed edge from state q and ending at state p with label a.

δ	0	1
$\rightarrow q_0$	q_0	q_1
$*q_1$	q_1	q_1

Transition table

two rows q_0 and q_1

Two i/p symbols 0 and 1 correspond to two columns.

direct arrow towards right.

final states are rep by putting stars on side of the states.

Equivalent transition is ~~rep~~ represented by writing p in row q and column a.

Configuration of a DFA:

Configuration of a DFA is which gives following info,

* The current state of the DFA
* The string to be processed.

Initial configuration: State of DFA before processing the string 'w' is called IC.

(q_0, w) → string to be processed.

↓ start state

Language Hierarchy, Computation & FSM:

What are the different types of lang? / Explain the hierarchy of lang classes.

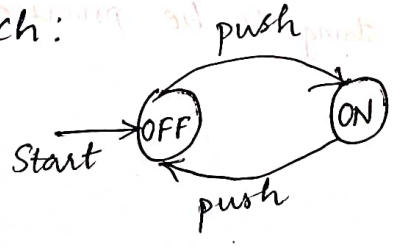
* Regular lang. (Accepted by FSM)
- The FSM starts from the start state and accepts one character at a time and changes the state based on current state & current i/p.
- After entire string is scanned if machine is in a final state then string is accepted by the machine else rejected by the machine.
- Lang accepted by FSM is called RL.

* Context free lang. (Accepted by PDA)
- Non-regular lang. such as lang with balanced parantheses or lang consisting of 'n' a's and 'n' b's ($a^n b^n$).
FSM cannot be constructed, add stack to FSM so that it can remember by pushing onto the stack.
- Such FSM with stack is called PDA.

* Decidable lang. (Accepted by TM)
- For non-context free lang. such as $a^n b^n c^n$ we cannot construct a PDA, we use Turing machines.
- TM can accept all types of lang including regular lang and context free lang.

* Semi decidable lang (Accepted by TM)
- Semi-decidable lang are semi-decided by some Turing machines that halt of on all strings in the lang.

Ex: Electric switch:



Final Configuration: The state of the DFA after processing the string 'w' is called FC. After processing the ~~pr~~ string, no string exists,

$q_0\ (q, \varepsilon)$

Yields-in-one-step: Process of moving from one config to other config based on current state & current i/p using transition function δ is called Yield-in-one-step.

'⊢' denoted.

(q_1, cw)

$\delta(q_0, c) = q_2$

(q_2, w)

$(q_1, cw) \vdash (q_2, w)$

$C_1 \vdash C_2$ in zero or more steps

$C_1 \vdash^+ C_2$ in one or more steps.

Computation: A computation by a FA is a finite seq of configurations $C_0, C_1, C_2 \ldots C_n$

$C_0 \vdash C_1 \vdash C_2 \vdash C_3 \ldots \vdash C_n$

C_0 is the initial config of the form (q_0, w)

C_n is the final config of the form (q_f, ε) → all i/p read.

$(q_0, w) \vdash (q_f, \varepsilon)$ → 'w' accepted by FA

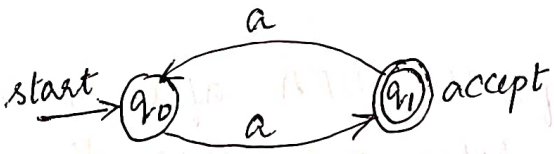
$(q_0, w) \vdash (q_i, \varepsilon)$ → 'w' ~~no~~ rejected by FA.

Language accepted by DFSM:

Let $M = (Q, \Sigma, \delta, q_0, F)$ by a DFA, A string 'w' is accepted by the machine M, $(q_0, w) \vdash (q_f, \varepsilon)$

So, lang accepted by DFA, $L(M)$ states:

$L(M) = \{w \mid w \in \Sigma \text{ and } (q_0, w) \vdash (q_f, \varepsilon)\}$



Consider following DFA, q_0 start state which
q_1 final state & accepts odd no of a's

aaaaa.

$(q_0, aaaaa) \vdash (q_1, aaaa)$
$\vdash (q_0, aaa)$
$\vdash (q_1, aa)$
$\vdash (q_0, a)$
$\vdash (q_1, \varepsilon)$ (final config)

Machine reaches final state q_1 and hence string 'aaaaa' is accepted by DFA.

aaaa

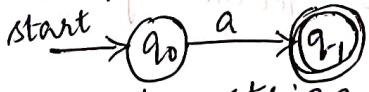
$(q_0, aaaa) \vdash (q_1, aaa)$
$\vdash (q_0, aa)$
$\vdash (q_1, a)$
$\vdash (q_0, \varepsilon)$ (final config)

Hence machine reaches non-final state q_0 and hence string 'aaaa' is rejected by DFA.

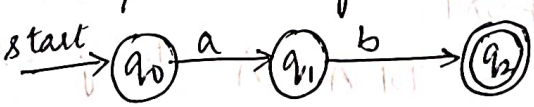
Ex: 1 DFA to accept empty lang $L = \{\phi\}$

2. DFA to accept any empty string $L = \{\varepsilon\}$

3. DFA to accept exactly one 'a'

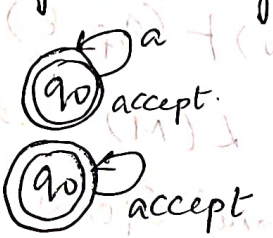


4. DFA to accept string 'ab'



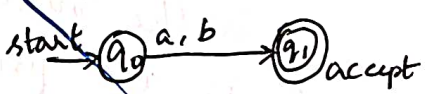
5. DFA to accept string consisting of any no of a's

zero/more a^*



b^*

one 'a' or one 'b'

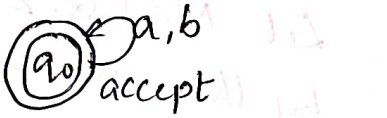


One symbol
2 states

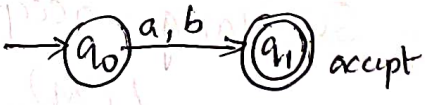
zero or more a's

2 symbol
3 states
Hence n symbols n+1 states.

$(a+b)^*$



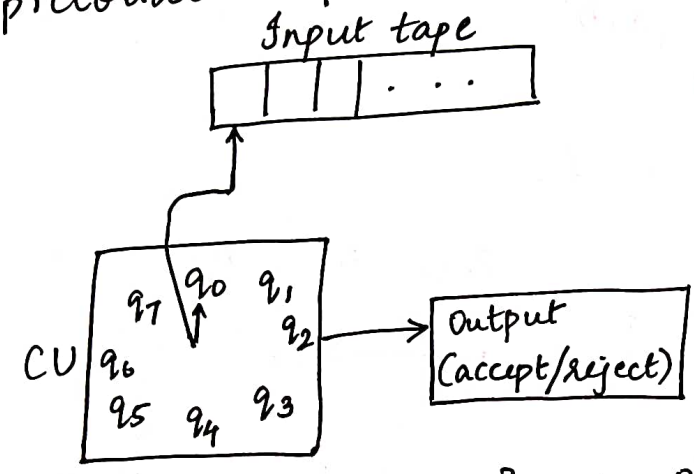
1 a or 1 b



bi.Khodanpur@gmail.com

Using this abstract model, the behavior of the actual system can be understood and built to perform various activities.

The pictorial representation of FA:



Finite automaton has 3 components,

1) Input ~~file~~ tape
2) Control unit
3) Output file.

Input tape: Input tape is divided into cells each of which can hold one symbol. The string to be processed is stored in these cells.

Control Unit: It consists of no of states $q_0, q_1, q_2 \ldots q_n$ one of which is the start state designated as q_0 and atleast one final state. Based on the input symbol and current state of the machine, it may change its state.

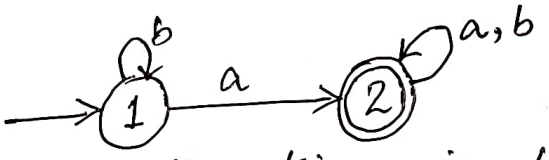
Output: Output maybe accept or reject when end of input is encountered, the control unit maybe in accept or reject state.

Write a FSM to accept L, where

$L = \{w \in \{a,b\}^* \mid w \text{ contains } a\}$

$L = \{a, aa, aaa, baa, baabb \ldots\}$

$\sim L = \{\varepsilon, b, bb, bbb, bbbb \ldots\}$



Accepting all the strings in L.

1) Transition Diagram

2) Transition function - δ

δ	a	b
$\rightarrow 1$	2	1
$*2$	2	2

Define:- $K = \{1, 2\}$

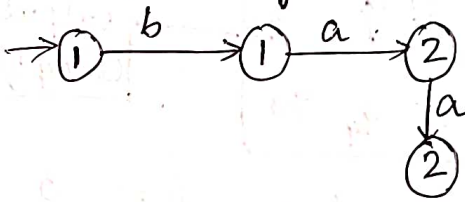
$A = \{2\}$

$s = \{1\}$

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

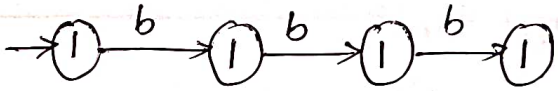
δ = table.

Show string baa is accepted.



final hence accepted.

Show string bbb is not accepted



DFA - RL - RG Type 3 $M = (K, \Sigma, \delta, s, A)$ ↗ initial stack symbol.

PDA - CFL - CFG Type 2 $M = (K, \Sigma, \Gamma, \delta, q_0, Z, F)$

↖ stack alphabets

LBA - CSL - CSG Type 1 M =

TM - REL - UG Type 0 $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$

↗ blank symbol

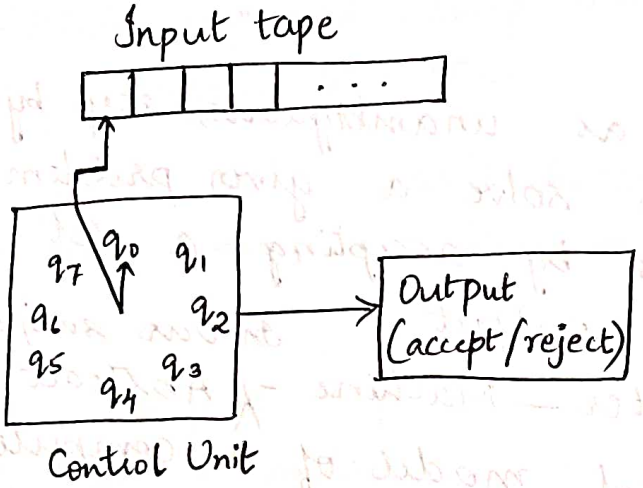
↖ tape alphabets

$M = (K, \Sigma, \delta, q_0, s, A)$

$M = (K\ \Sigma, F\ \delta\ s\ Z\ A)$

$M = (K\ \Sigma\ \Gamma\ b\ \delta\ s\ A)$

The pictorial representation of FA,



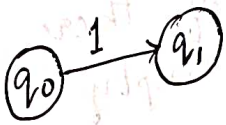
Input tape :- Input tape is divided into cells each of which can hold one symbol. The string to be processed is stored in these cells.

Control Unit :- The machine has some states one of which is the start state designated as q_0 and atleast one final state. Based on the current i/p symbol & current state of the machine, it may change its state. It consists of no of states. $q_0, q_1, q_2 \ldots q_n$

Output :- Output maybe accept or reject. When end of input is encountered, the control unit may be in accept or reject state.

Symbols used in FA:

Symbol	Meaning.
	A circle is used to represe a state. Here, q_0 is a state of the machine.
	A circle with an arrow which is not originating from any node represents the start state of machine.
	Two circles are used to repres a final state. 'q_0' is the

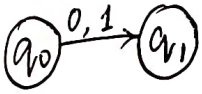


An arrow with label 1 goes from state q_0 to state q_1, indicates a transition from state q_0 on input symbol 1 to state q_1.

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

Indicates machine in state q_0 on reading a 0, it remains in same state q_0.

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



machine in state q_0 on reading 0 or 1 moves to state q_1.

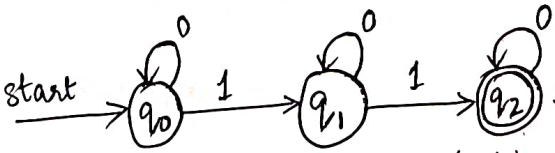
$$\delta(q_0, 0) = q_1$$
$$\delta(q_0, 1) = q_1$$

Types of Finite Automata:

1) Deterministic finite automata
2) Non-deterministic finite automata
3) Non-deterministic finite automata with ε-moves (ε-NFA)

Deterministic finite Automata :-(DFA) **DFSM**

Consider,



States: circles are called Vertices, or nodes or states. Each state is identified by name q_0, q_1 and q_2. states are classified as,

Start state: q_0

final state: q_2

This DFA has 3 states,

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

A DFA can have one or more final states. If there is no final state in DFA, the DFA accepts empty language.

Input alphabets :-

Each edge is labeled with 0 or 1 and represent the input alphabets which can be denoted as,

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

Transitions :- Transition is nothing but change of state after consuming an input symbol.

q_i to q_j on an input symbol a, then.

$$\delta(q_i, a) = q_j$$

for above example,

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

$\delta : \underbrace{(Q \times \Sigma)}$ to Q

Cross product of $Q = \{q_0, q_1, q_2\}$ and $\Sigma = \{0, 1\}$.

Transition function:

$$\delta : Q \times \Sigma \text{ to } Q$$

read as, δ is a transition function which maps $Q \times \Sigma$ to Q.

Ex: the change of state from state 'q' on input symbol 'a' to state 'p' is denoted by

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

where,

δ : is a function called transition function.
q : is a first parameter representing the current state of machine.
a : is the 2nd parameter representing the current input symbol read.
p : is the next state of machine which is returned by the transition function.

From the above discussion, we observe that DFA has 5 components:

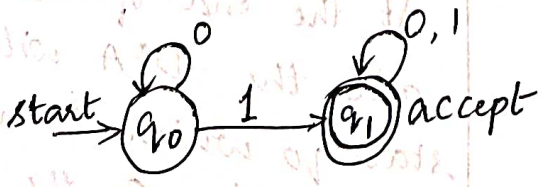
(states, input alphabets, transitions, start state, final state)
↓ ↓ ↓ ↓ ↓
Q Σ δ q_0 F

Definition: The Deterministic Finite Automaton (DFA) is 5-tuple or quintiple indicating five components:

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

where,
M is the name of the machine.
Q is non-empty, finite set of states.
Σ is non-empty, finite set of input alphabets.
$\delta : Q \times \Sigma$ to Q i.e., δ is transition function which is a mapping from $Q \times \Sigma$ to Q.
$q_0 \in Q$ - is the start state.
$F \subseteq Q$ - is set of accepting or final states.

Transition Diagram:-



Deterministic: There is ex on transition for every i/p symbol from the state.
Finite: Finite no of states & edges.
Automaton: is a machine which accept the string or reject the string.

Transition table:

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

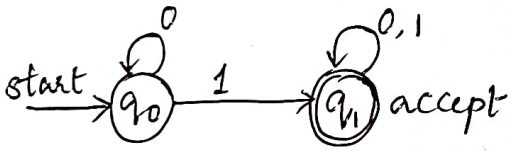
The rows of the table correspond to the states of DFA obtained from Q.

The columns correspond to the input symbols obtained from Σ.

If 'q' is the current state of DFA and 'a' is the curre input symbol, the value returned from $\delta(q, a)$ represent the next state of DFA and is entered in row 'q' and column 'a'

Start state $\rightarrow$

final state $*$



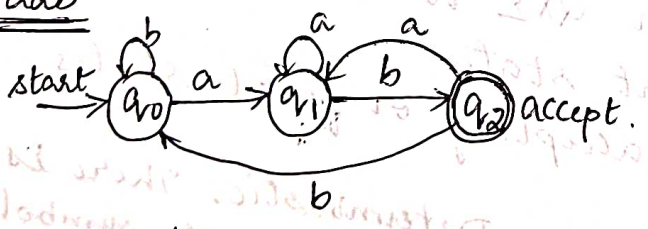
Transition diagram

δ	0	1
$\rightarrow q_0$	q_0	q_1
$*q_1$	q_1	q_1

Transition table.

How a DFA processes strings:

abaab



a b a a b

$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

abb

a b b

$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_0$

At the end of the string abb, the DFA will be in state q_0 which is not the final state. So the string "abb" is rejected by the machine.

abb

Extended transition function of DFA to strings:

Denoted by δ^*.

δ^* describes what happens to a state of machine when the input is a string. (seq. of symbols)

Extended transition function $\delta^* : Q \times \Sigma^*$ to Q is defined recursively as,

a) Basis: $\delta^*(q, \epsilon) = q$ i.e., if machine is in q state & it reads no input then it will still be in state q.

b) Induction: If $w = xa$, where 'a' is last symbol of 'w' and after processing this string if it changes its state from q to p then,

$$\delta^*(q, w) = \delta^*(q, xa) = \delta(\delta^*(q, x), a)$$

OR

$$\delta^*(q, w) = \delta^*(q, ax) = \delta(\delta^*(q, a), x) \quad \text{if } w = ax$$

both 'a' & 'b' are the properties of extended transition function.

where,

δ^* = extended transition function.

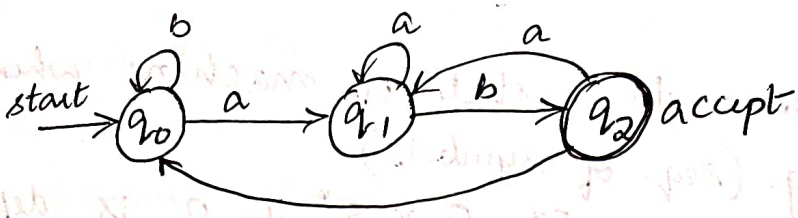
δ = transition function.

q = ~~st~~. current state.

p = next state

w = string.

For the below DFA process the string abaab using extended transition function:



i) for prefix ϵ : $\delta^*(q_0, \epsilon) = q_0$

ii) for prefix a : $\delta^*(q_0, a) = \delta(\delta^*(q_0, \epsilon), a)$
$= \delta(q_0, a)$
$= q_1$

iii) for prefix ab : $\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b)$
$= \delta(q_1, b)$
$= q_2$

iv) for prefix aba : $\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a)$
$= \delta(q_2, a)$
$= q_1$

v) for prefix abaab : $\delta^*(q_0, abaab) = \delta(\delta^*(q_0, aba), a)$
$= \delta(q_1, a)$
$= q_1$

vi) for prefix abaab: $\delta^*(q_0, abaab) = \delta(\delta^*(q_0, abaa), b)$
$= \delta(q_1, b)$
$= q_2$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2$: As final state is q_2 string is accepted.

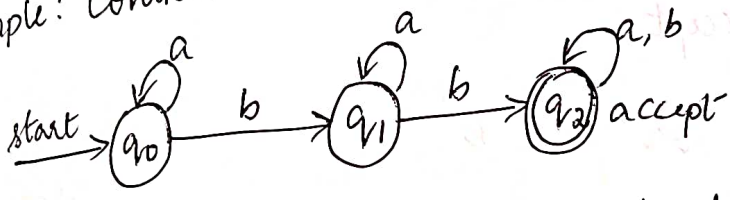
Language accepted by a DFA:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A string 'w' is accepted by the machine M, if it takes the initial state q_0 to final state i.e., $\delta^*(q_0, w)$ should be in F & $w \in \Sigma^*$

Language accepted by DFA is represented as,

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

Example: Consider below DFA for input string abab



$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

As q_2 is final state the string is accepted.

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

Language rejection by DFA:

$M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

Non-acceptance (rejection) of a language is when DFA won't be in final state after processing the input string.

i.e.,

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is not in } F\}$$

For the same DFA example input string abaa

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, a) = q_1$

As q_1 is not the final state the string is rejected.

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1$

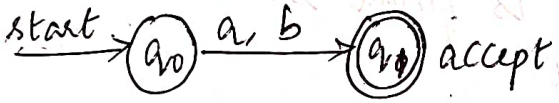
Note:-

a) DFA to accept an empty language $L = \{\phi\}$

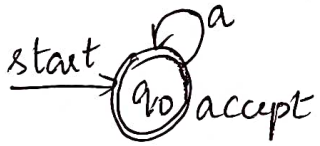
b) DFA to accept an empty string $L = \{\varepsilon\}$

c) DFA to accept exactly one 'a'.

d) DFA to accept one 'a' or one 'b'



e) DFA to accept zero or more a's



DFA design techniques:

Types of problems for which we can construct DFA are,

a) Pattern recognition problems
b) Divisible by 'k' problems
c) Modulo-k counter problems. $|w| \bmod k$
d) $|w| \bmod k$ problems with relational ope
c) $N_a(w) \bmod k$ problems
d) $N_a(w) \bmod k$ problems with logical operato

a) Pattern recognition problems:-

Steps to be followed in designing DFA for pattern recognition problems are,

i) Identify the minimum string
ii) Identify the alphabets.

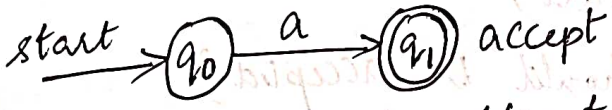
iii) Construct a skeleton of DFA

iv) Identify other transitions that are not defined in step(iii)

v) Construct DFA using transitions in step(iii) & (iv)

Problem 1 :- Draw a DFA to accept string of a's having atleast one 'a' :-

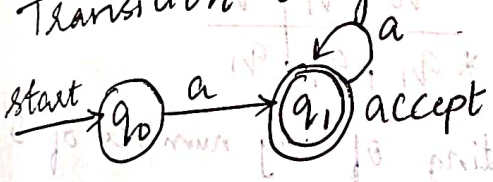
minimum string = a alphabet $\Sigma = \{a\}$



DFA to accept atleast one 'a'

similarly, to accept string of a's i.e., a, aa, aaa, aaaa....

Transition diagram



Transition table.

δ	a
$\rightarrow q_0$	q_1
$* q_1$	q_1

i.e., Language $L = \{a, aa, aaa, aaaa \ldots\}$

or

$L = \{a^n \mid n \geq 1\}$

or

$L = \{w : n_a \geq 1, w \in \{a\}^*\}$

$M = (Q, \Sigma, \delta, q_0, F)$ i.e., DFA for above problem is

$Q = \{q_0, q_1\}$

$\Sigma = \{a\}$

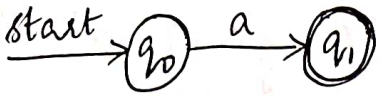
δ = shown above [transition table]

q_0 = start state

$F = q_1$ - final state.

Problem 2: Draw a DFA to accept strings of a's and b's having atleast one 'a' :-

Minimum string is a & $\Sigma = \{a, b\}$



DFA to accept a.

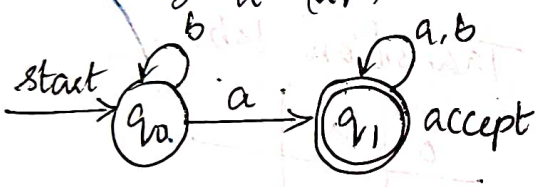
we now want (q_0, b) (q_1, a) (q_1, b)

$q_0 \to b = q_0$ (as the string has no 'a' it cannot go to final state)

$q_1 \to a = q_1$ (seq, aa should be accepted)

$q_1 \to b = q_1$ (seq, ab ——— " ———)

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



δ	a	b
$\to q_0$	q_1	q_0
$* q_1$	q_1	q_1

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

$Q = \{q_0, q_1\}$

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

δ = shown above

q_0 = start state

F = final state. q_1

* string consisting of any number of b's b^*
* followed by one 'a'
* followed by any no of a's and b's denoted by $(a+b)^*$

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

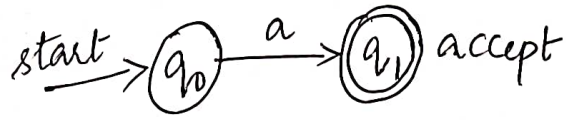
$$L = \{w : n_a(w) \geq 1,\ w \in \{a, b\}^*\}$$

Trap state: Once the machine enters into state q_1, irrespective of any number of inputs of a's and b's, the machine remains in same state q_1 and cannot come out of the state. So, the machine is trapped in state q_1 and hence it is called trap state. Since, q_1 is also a final state, the state q_1 is called "trapped final state"

oblem 3: Draw a DFA to accept strings of a's & b's having exactly one 'a'.

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

$\delta(q_0, a) = q_1$

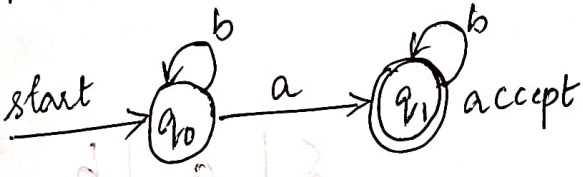


low,

$\delta(q_0, b) = q_0$

$\delta(q_1, a) = -$ (i.e., transition not defined)

$\delta(q_1, b) = q_1$



δ	a	b
$\rightarrow q_0$	q_1	q_0
$* q_1$	-	q_1

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

$Q = \{q_0, q_1\}$

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

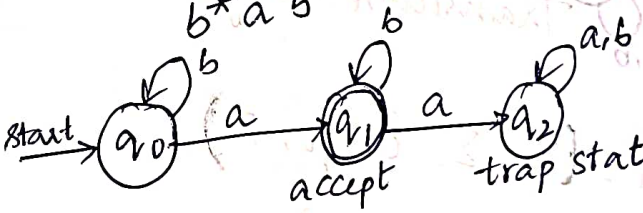
δ : shown above

q_0 = start state

F = final state : q_1

* string consisting of any number of b's b^*
* followed by one 'a'
* followed by any no of b's b^*

$b^* a b^*$



$$L = \{w : n_a(w) = 1,\ w \in \{a, b\}^*\}$$

* There is no transition from state q_1 for the i/p symbol a. but we can include a transition from state q_1 for i/p a, to a non-final reject state q_2

Trap state: A state for which there exists transition to itself for all the i/p symbol of Σ and there is no way to reach the final state is called trap state

also called DEAD STATE

Problem 4: Obtain a DFA to accept strings of a's and b's starting with string ab.

Minimum string = ab $\quad \Sigma = \{a, b\}$

$ab(a+b)^*$

δ

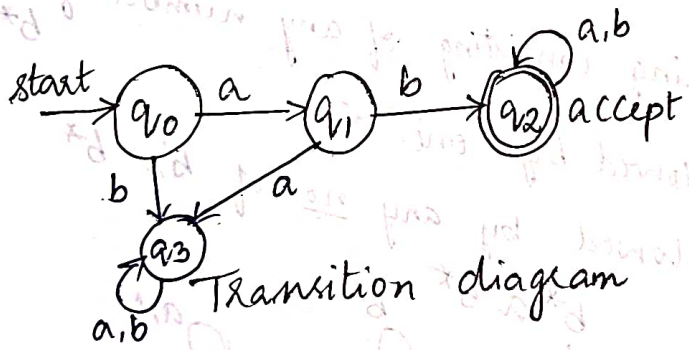
$\delta(q_0, a) = q_1 \quad \delta(q_1, b) = q_2$

$\delta(q_0, b) = -$

$\delta(q_1, a) = -$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_2$



Transition diagram

δ	a	b
$\rightarrow q_0$	q_1	–
q_1	–	q_2
$*q_2$	q_2	q_2

Transition table.

$M = \{Q, \Sigma, \delta, q_0, F\}$

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

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

δ = shown above

q_0 = start state

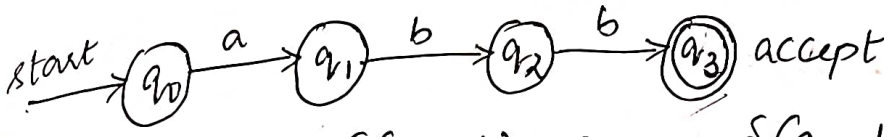
F = final state = q_2

δ	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_3	q_2
$*q_2$	q_2	q_2
q_3	q_3	q_3

$$L = \{ab(a+b)^n : n \geq 0\}$$

Problem 5: Draw a DFA to accept string of a's and b's ending with string abb.

Minimum string abb, $\Sigma = \{a, b\}$



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

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_0$

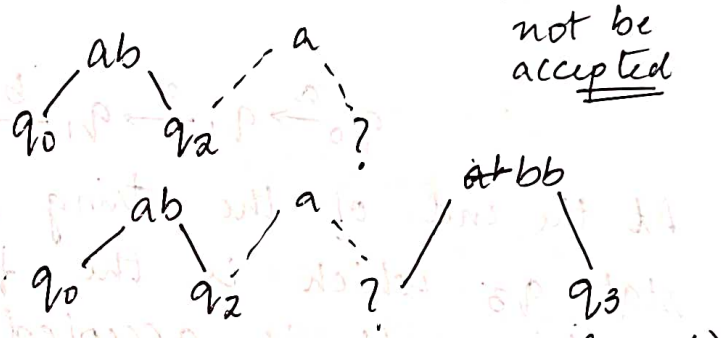
$\delta(q_1, a) = q_1$

$\delta(q_2, a) = q_1$

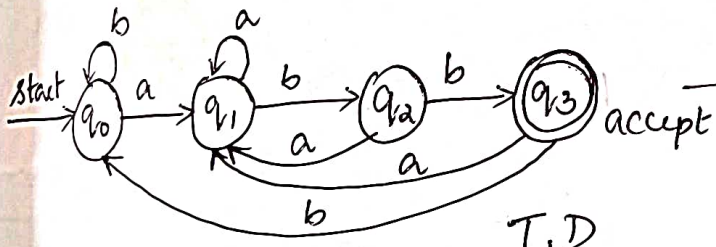
$\delta(q_3, a) = q_1$

$\delta(q_3, b) = q_0$

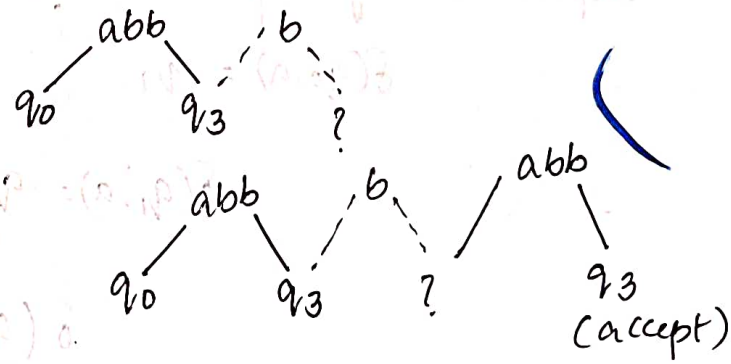
aba should not be accepted



but aba followed by bb resulting in string ababb has to be accepted since it ends with abb.



T.D



δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
$*q_3$	q_1	q_0

T.T

$$L = \{(a+b)^n abb : n \geq 0\}$$

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

Show the sequence of moves made by DFA for the string aabb.

Input String a a b b

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_3$

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{b} q_3$

At the end of the string aabb the DFA is in state q_3 which is the final state.
So string aabb is accepted.

Show the sequence of moves made by DFA for the string aaba.

Input string a a b a

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_1$

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1$

At the end of the string aaba the DFA is not in state q_3 which is the final state. Its in state q_1, so string aaba is rejected.

problem 6: Draw a DFA to accept string of a's and b's which donot end with abb.

The design is same as previous problem but we need to make final state to non-final states and non-final states to final states.

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

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

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

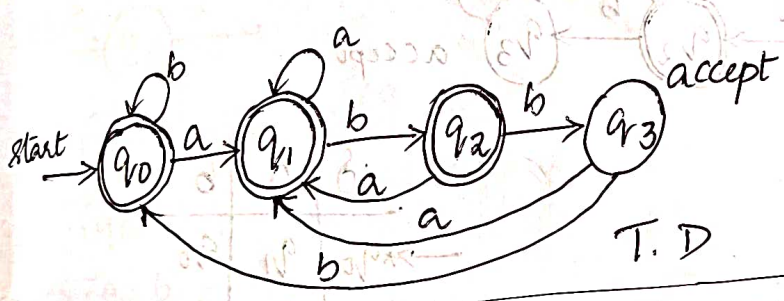
δ = shown in the transition diagram.

q_0 = start state

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

δ	a	b
$\rightarrow *q_0$	q_1	q_0
$*q_1$	q_1	q_2
$*q_2$	q_1	q_3
q_3	q_1	q_0

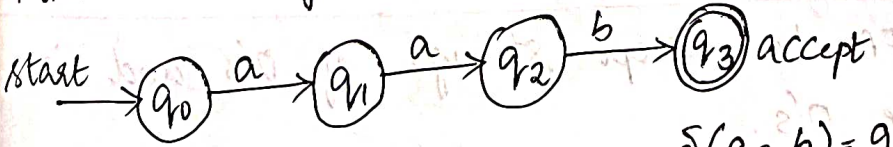
T. T



T. D

problem 7: Draw a DFA to accept the string of a's & b's having a substring aab.

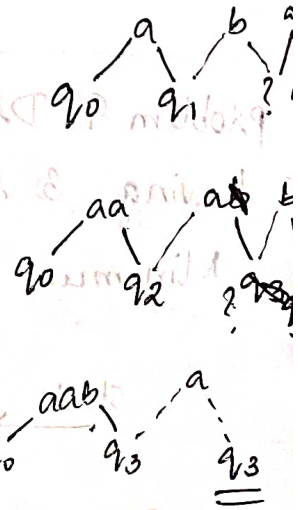
Minimum string aab $\Sigma = \{a, b\}$

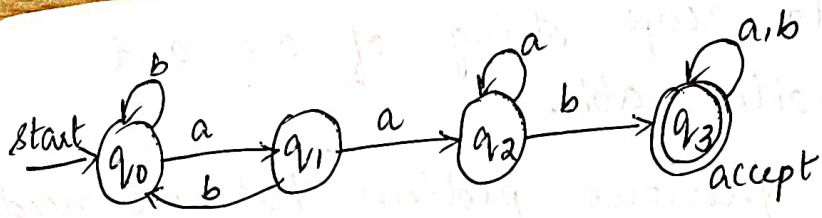


$\delta(q_0, a) = q_1$ $\quad \delta(q_1, a) = q_2$, $\quad \delta(q_2, b) = q_3$

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

$\delta(q_3, a) = q_3$ $\quad \delta(q_3, b) = q_3$





T.D

δ	a	b
→ q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
* q_3	q_3	q_3

T.T

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

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

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

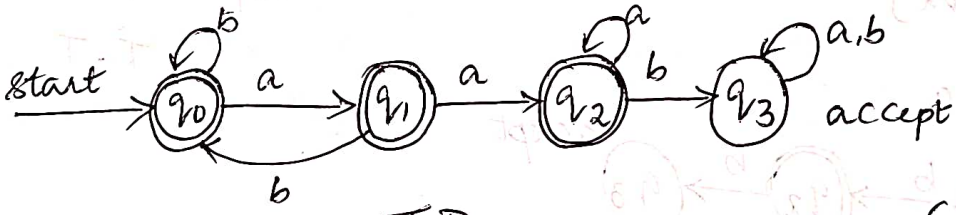
q_0 = start state

$F = \{q_3\}$

δ = shown above:

---

problem 8:- Draw a DFA to accept string of a's and b's except those having substring aab



T.D.

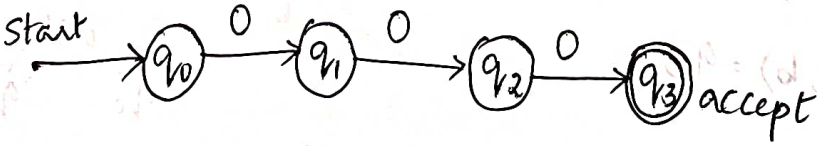
δ	a	b
→*q_0	q_1	q_0
*q_1	q_2	q_0
*q_2	q_2	q_3
q_3	q_3	q_3

T.T

---

problem 9: Draw a DFA to accept strings of 0's and 1's having 3 consecutive 0's.

Minimum string 000 $\Sigma = \{0, 1\}$



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

$\delta(q_0, 1) = q_0$

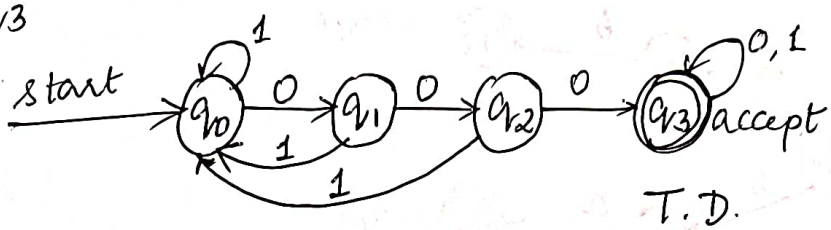
$\delta(q_1, 1) = q_0$

$\delta(q_2, 1) = q_0$

$\delta(q_3, 0) = q_3$

$\delta(q_3, 1) = q_3$

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
$*q_3$	q_3	q_3

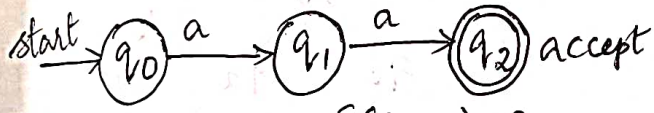


T.D.

Problem 10: Draw a DFA to accept string of a's and b's such that $L = \{awa \mid w \in (a+b)^n$ where $n \geq 0\}$

i.e., its a string of a's and b's starting with one 'a' & ending with one 'a'.

Minimum string = aa $\quad \Sigma = \{a, b\}$



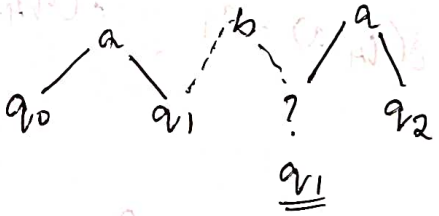
$\delta(q_0, a) = q_1 \quad \delta(q_1, a) = q_2$

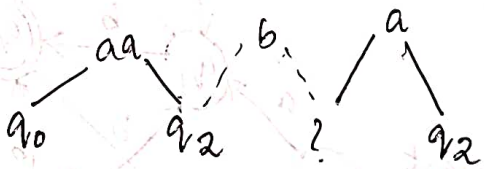
$\delta(q_0, b) = -$? Nt defined

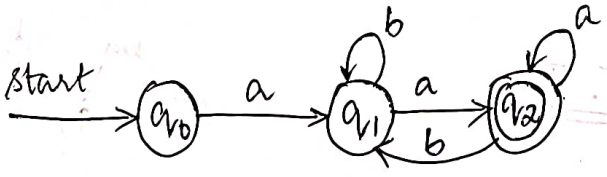
$\delta(q_1, b) = q_1$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_1$



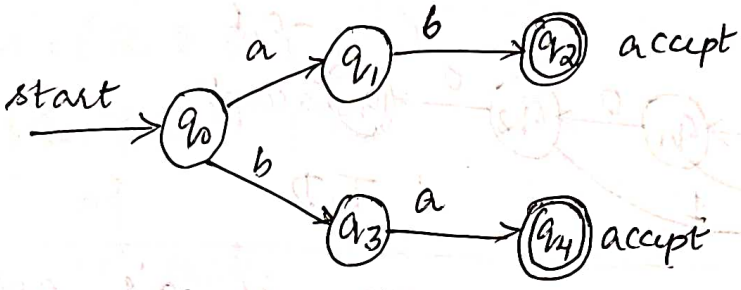




$M = \{Q, \Sigma, \delta, q_0, F\}$

Problem 11: Draw a DFA to accept string of a's and b's ending with ab or ba.

Minimum string ab or ba $\Sigma = \{a, b\}$



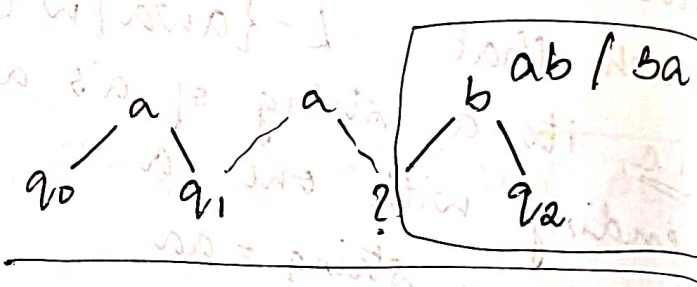
$\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_3$

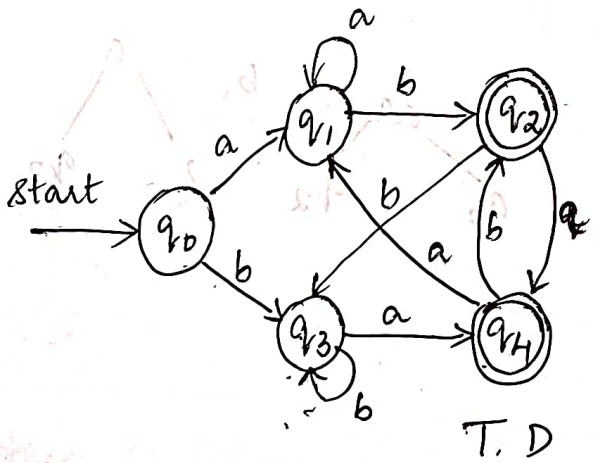
$\delta(q_1, b) = q_2$ $\delta(q_3, a) = q_4$

$\delta(q_1, a) = q_1$ $\delta(q_2, a) = q_4$

$\delta(q_2, b) = q_3$ $\delta(q_3, b) = q_3$

$\delta(q_4, a) = q_1$ $\delta(q_4, b) = q_2$



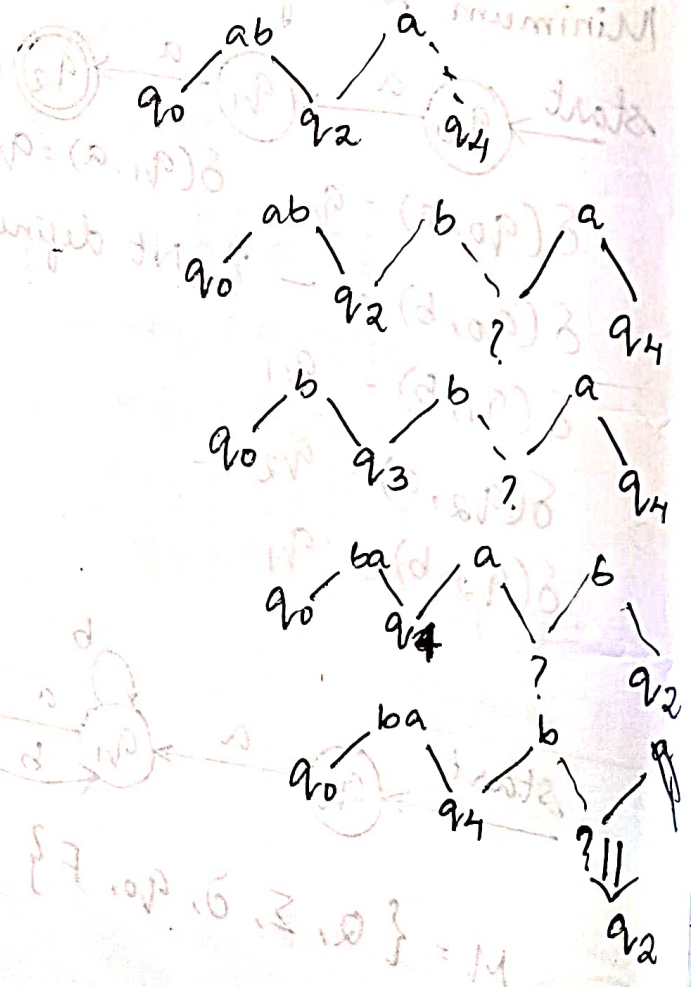


T. D

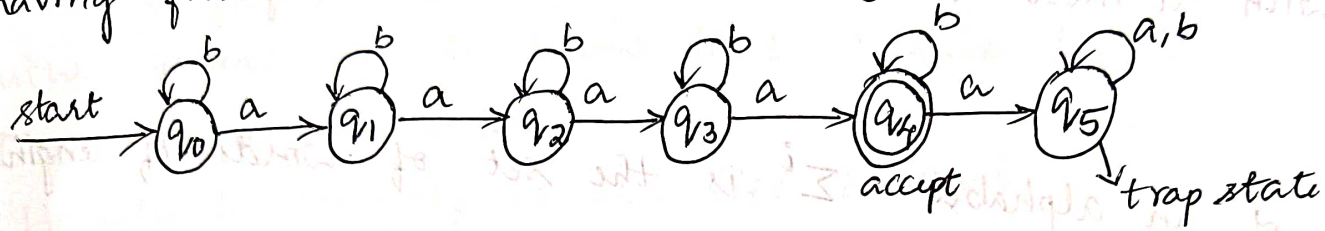
T. T

$M = \{Q, \Sigma, \delta, q_0, F\}$

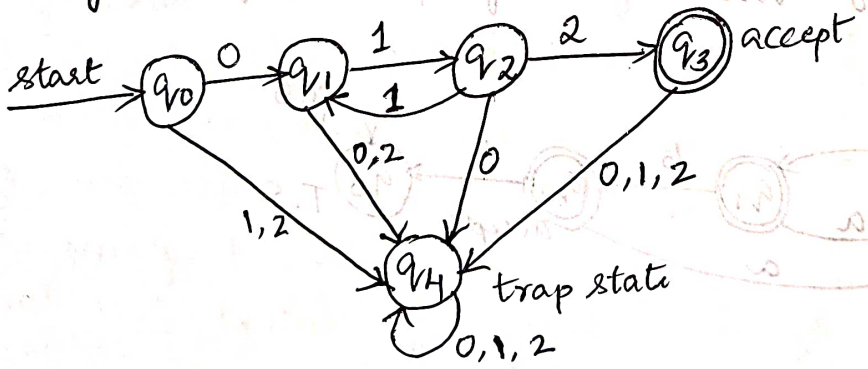
$L = \{w(ab + ba) \mid w \in \{a, b\}^*\}$



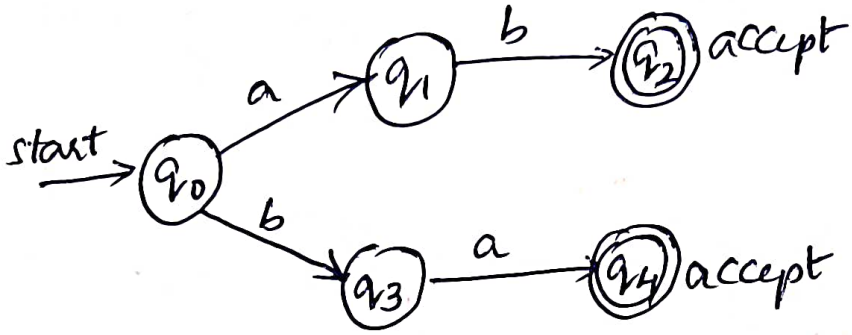
Problem 12 : Obtain a DFA to accept strings of a's and b's having fours a's where $\Sigma = \{a, b\}$



Problem 13 : Obtain a DFA to accept strings of 0's and 1's and 2's beginning with '0' followed by Odd no of 1's and ending with 2.



12) Draw a DFA to accept string of a's & b's ending with ab or ba. ab, ba



q_1 : $\delta(q_1, a) = q_1$

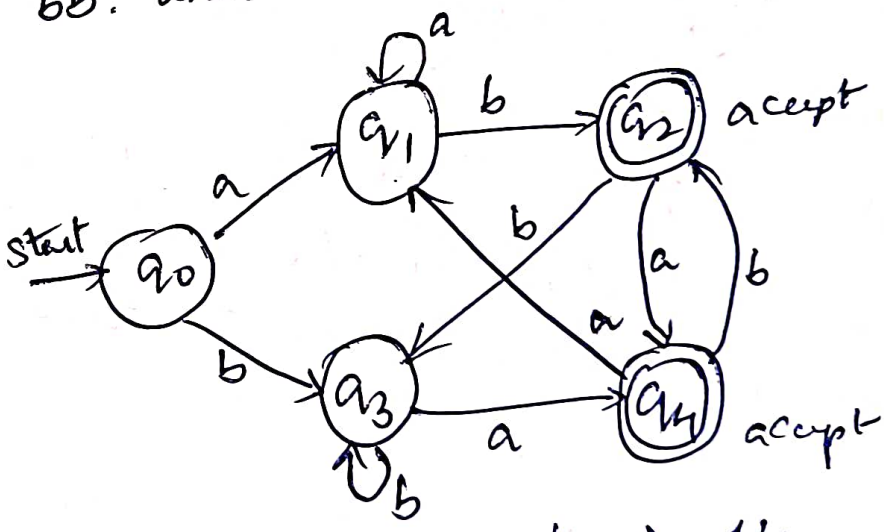
q_3 : $\delta(q_3, b) = q_3$

q_2 : $\delta(q_2, a) = q_4$ for i/p a, the string ends with ba and hence it should goto state q_4.

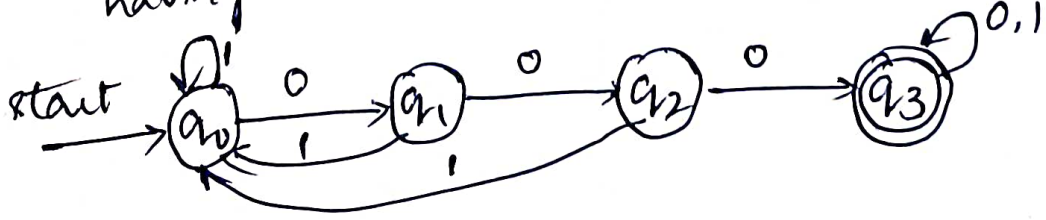
$\delta(q_2, b) = q_3$ for i/p b, the string ends with bb. and hence goto state q_3

q_4 : $\delta(q_4, a) = q_1$

$\delta(q_4, b) = q_2$

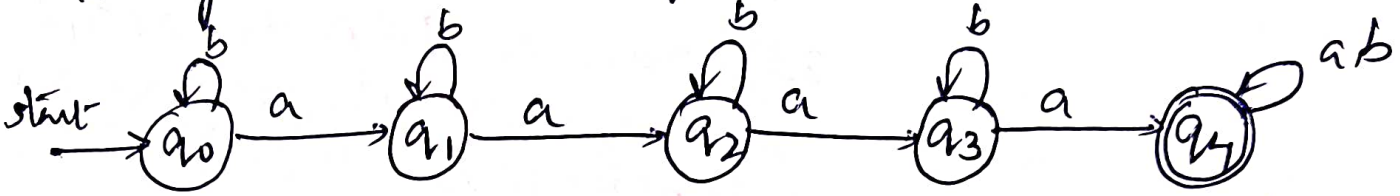


13) Draw a DFA to accept string of 0's & 1's having 3 consecutive 0's

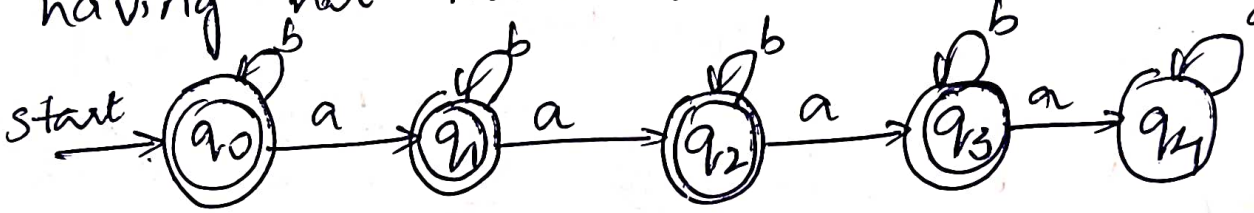


14) Draw a DFA to accept string of 0's & 1's having no 3 consecutive 0's.

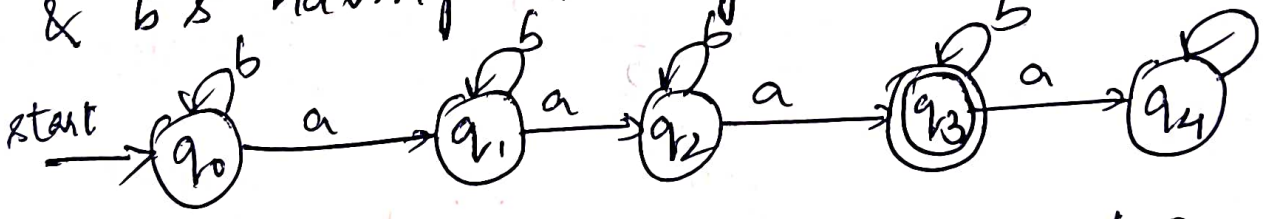
15) Draw a DFA to accept string of a's & b's having ~~to~~ atleast four a's.



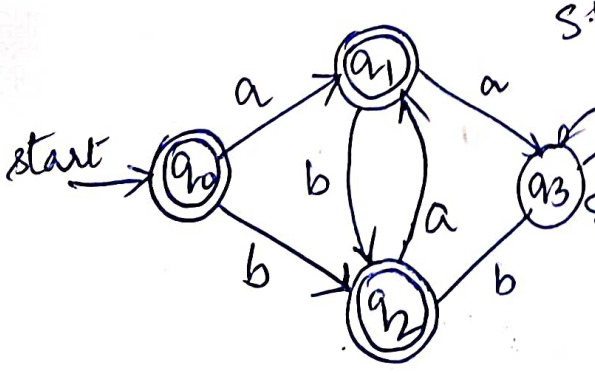
16) Draw a DFA to accept string of a's & b's having not more than 3 a's.



17. Draw a DFA to accept string of a's & b's having exactly three a's.



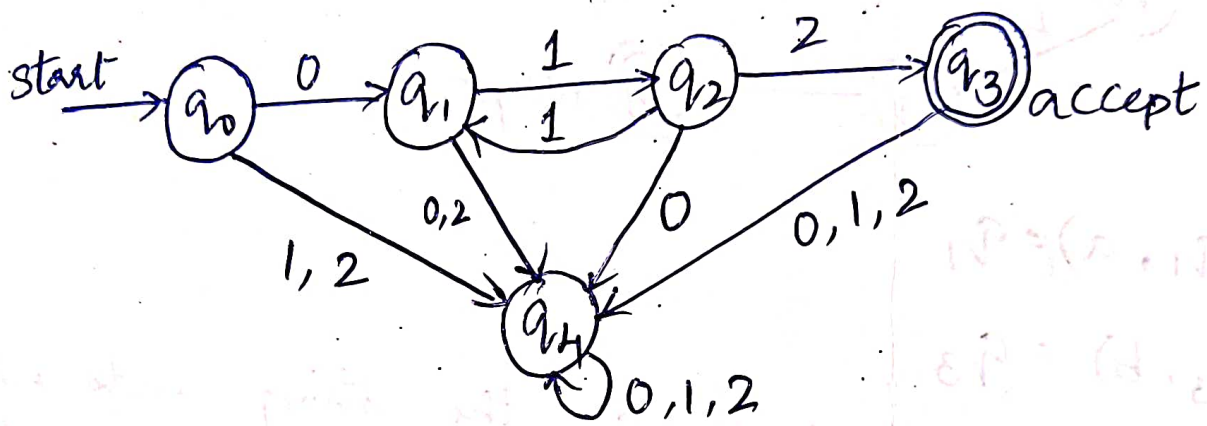
18. Draw a DFA to accept string of a's and b's such that no two consecutive characters are same.



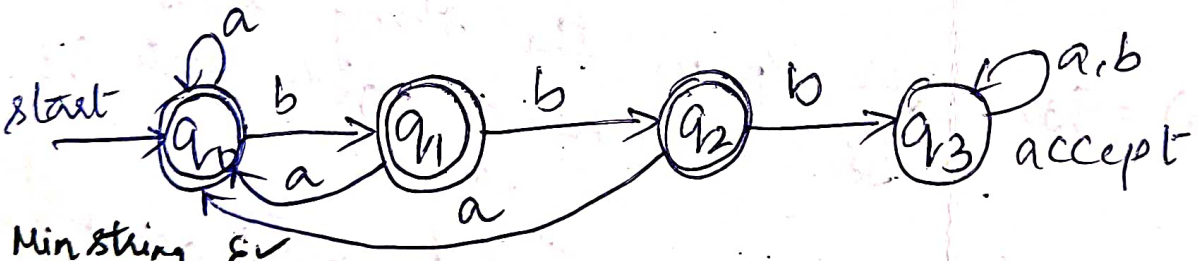
State q_1: for i/p 'b',
$\delta(q_1, b) = q_2$

State q_2: for i/p 'a',
$\delta(q_2, a) = q_1$

19) Obtain a DFA to accept strings of 0's, 1's and 2's beginning with a '0' followed by odd number of 1's and ending with a '2'

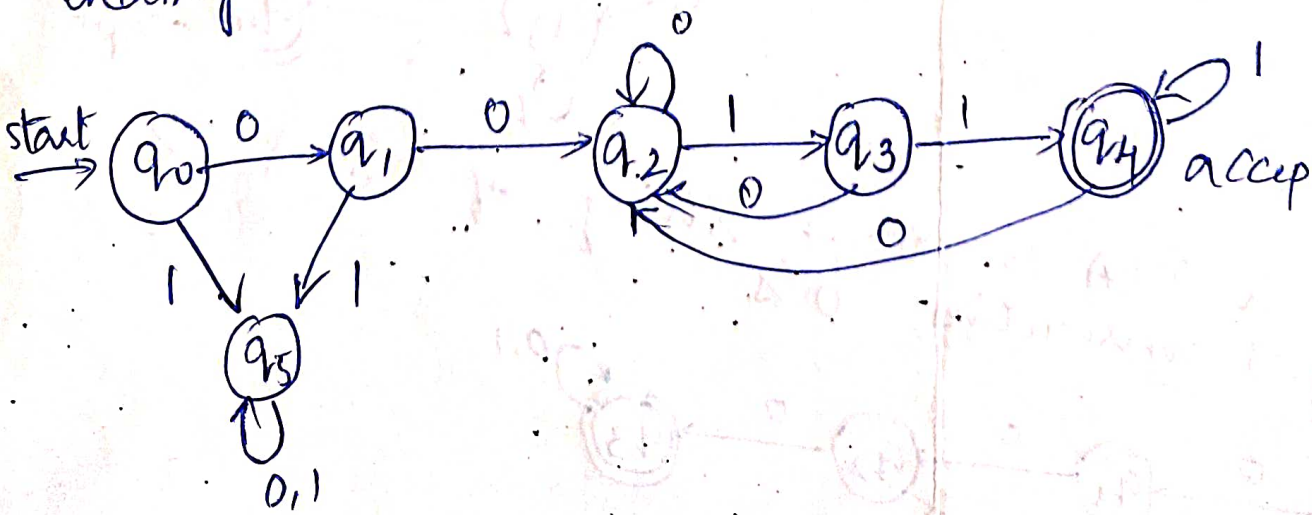


20) Obtain a DFA a accept strings of a's and b's with atmost two consecutive b's.



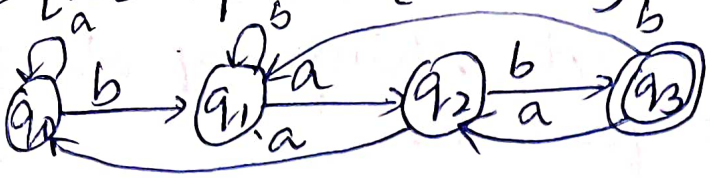
Min string ε
b
bb

21) Obtain a DFA to accept strings of 0's & 1's starting with atleast two 0's and ending with atleast two 1's.

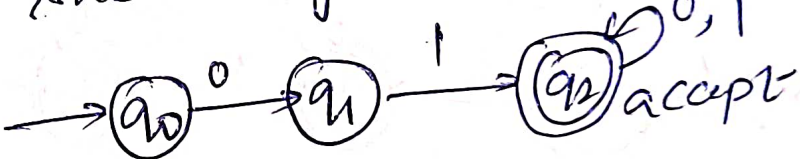


22) Obtain a DFA to accept the lang

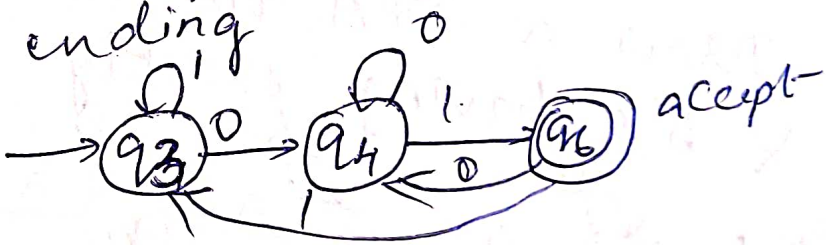
$L = \{wbab \mid w \in \{a, b\}^*\}$



23) Draw a DFA to accept set of all strings ~~of~~ on alphabets $\Sigma = \{0, 1\}$ that either begins or ends or both with the substring 01

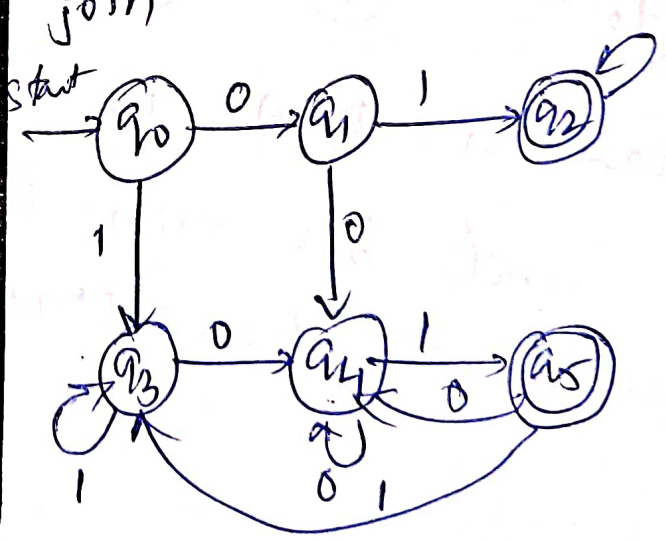


starting



ending

join



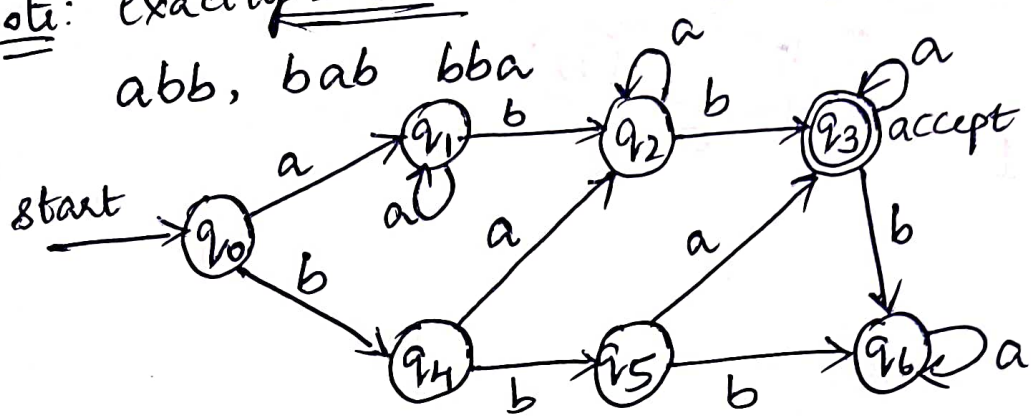
ODD SEMESTER

2018-19

24) Draw a DFA to accept the lang.

$L = \{ w : n_a(w) \geq 1, n_b(w) = 2 \}$

Note: exactly 2 b's and atleast one one, min string abb, bab bba



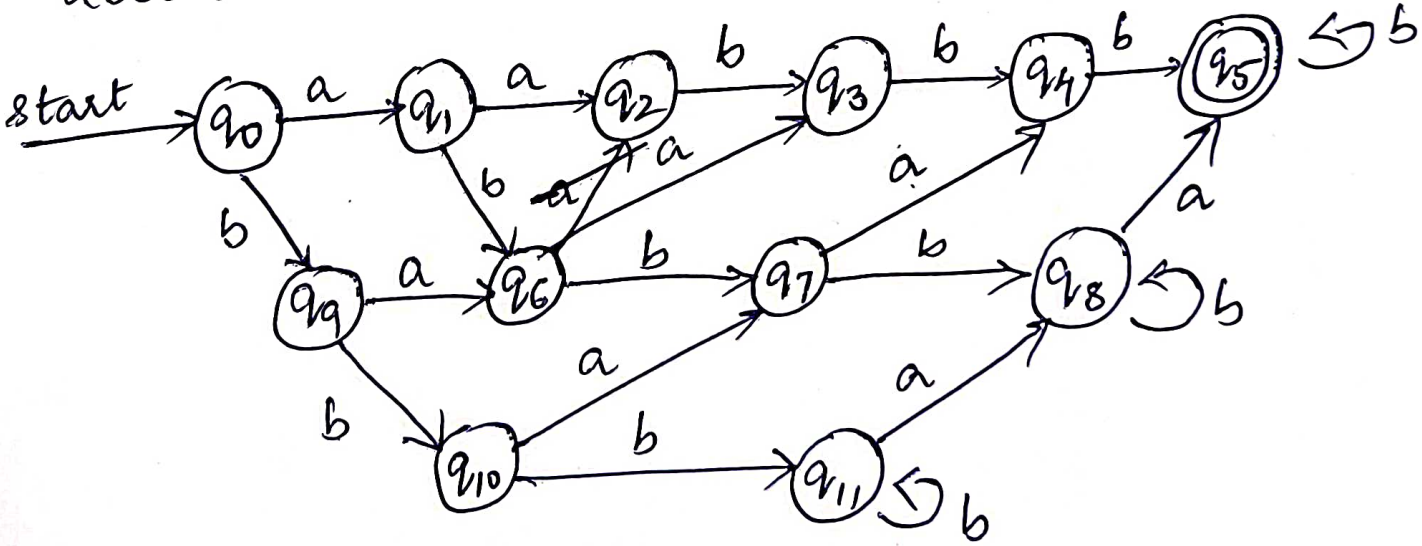
Trap state for more than 2 b's

25) Draw a DFA to accept the lang

$L = \{ w : n_a(w) = 2, n_b(w) \geq 3 \}$

Note: exactly 2 a's and atleast 3 b's

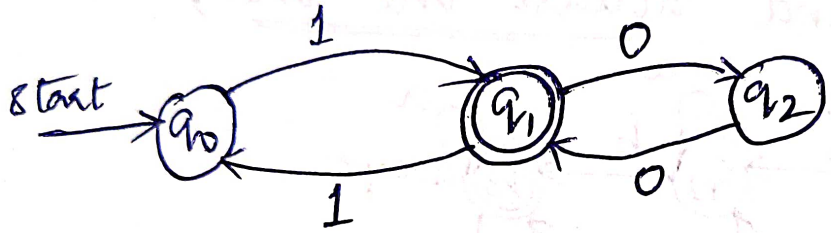
aabbb baabb bbaab bbbaa
ababb babab bbaba
abbab babba
abbba



From states q_2, q_3, q_4 & q_5 we can have transitions from i/p symbol 'a' to trap state

26) Draw a DFA to accept the lang

$L = \{w : w$ has odd no of 1's and followed by even number of 0's$\}$



software. (05 Marks)

IS. (05 Marks)

27) Obtain a DFA to accepts strings of a's & b's such that each block of 5 ~~se~~ consecutive symbols have atleast two a's.

Here DFA tracks no of symbols scanned & the number of a's.

Hence each state of DFA is shown as,

i j

↙ No. of symbols scanned ↘ No of a's scanned.

Divisible by k problems:

A method of constructing a DFSM that divide a number by k. For these problems, the transitions can be obtained using the following relation:

$$\delta(q_i, a) = q_j \text{ where } j = (r * i + d) \bmod k$$

* 'r' is the radix of input, for binary $r = 2$ ✓ 0, ... 9 ✓ decimal $r = 10$ ✓
* i is the remainder obtained after dividing by k.
* d represent digits for binary $d = \{0, 1\}$
* k is divisor

Steps followed to find FSM using these types of problem,

Step 1: Identify the radix, input alphabets & the divider

Step 2: Compute the possible remainders. These remainders represent the state of FSM.

Step 3: Find the transitions using $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$

step 4: Construct the FSM using the transitions obtained in Step 3.

Ex 1: Construct a DFA ~~using~~ which accepts strings of 0's & 1's where the value of each string is represented as a binary number. only the strings representing zero modulo five should be accepted. For:- 0000, 0101, 1010, 1111 etc should be accepted.

Step 1: $r = 2$, $d = \{0, 1\}$ $k = 5$

Step 2: Compute the possible remainders: After dividing by k. the remainders are,

$$i = 0, 1, 2, 3, 4$$

d 5)0

Step 3: find the transitions.

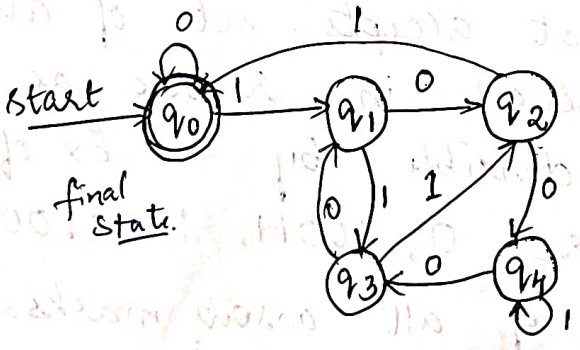
$\delta(q_i, a) = q_j$ where $j = (a * i + d) \bmod k$.

$a = 2, \quad k = 5$

$j = (2i + d) \bmod 5$

Remainder	d	$(2*i+d) \bmod 5 = j$	$\delta(q_i, d) = q_j$
$i = 0$	0	$(2*0+0) \bmod 5 = 0$	$\delta(q_0, 0) = q_0$
	1	$(2*0+1) \bmod 5 = 1$	$\delta(q_0, 1) = q_1$
$i = 1$	0	$(2*1+0) \bmod 5 = 2$	$\delta(q_1, 0) = q_2$
	1	$(2*1+1) \bmod 5 = 3$	$\delta(q_1, 1) = q_3$
$i = 2$	0	$(2*2+0) \bmod 5 = 4$	$\delta(q_2, 0) = q_4$
	1	$(2*2+1) \bmod 5 = 0$	$\delta(q_2, 1) = q_0$
$i = 3$	0	$(2*3+0) \bmod 5 = 1$	$\delta(q_3, 0) = q_1$
	1	$(2*3+1) \bmod 5 = 2$	$\delta(q_3, 1) = q_2$
$i = 4$	0	$(2*4+0) \bmod 5 = 3$	$\delta(q_4, 0) = q_3$
	1	$(2*4+1) \bmod 5 = 4$	$\delta(q_4, 1) = q_4$

Step 4: The DFSM can be defined as $M = \{Q, \Sigma, \delta, q_0, F\}$ is,



δ	0	1
$\rightarrow *q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

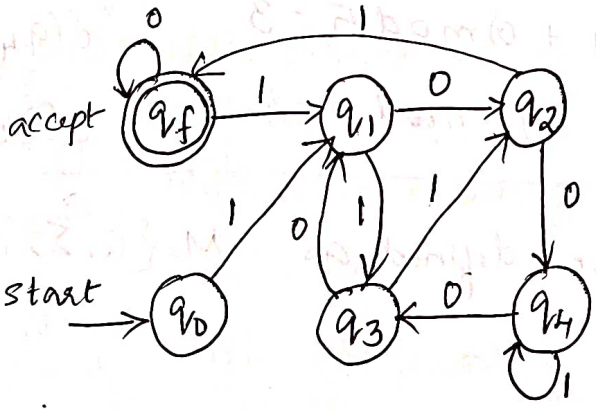
Note:- Its observed for modulo 5, no of states = 5

similarly for modulo k, no of states = k.

0 mod 5
1 mod 5
2 mod 5

Ex 2: Obtain a DFSM which accepts the set of all strings beginning with a 1 that when interpreted as a binary ~~ing~~ integer is a multiple of 5. For example 101, 1010, 1111 etc are multiples of 5. Note that 0101 is not beginning with 1 and should not be accepted.

Soln:- The solution to this problem is same as above problem. but, the number always should start with a 1. If a binary number starts with a 0, the number should never be accepted. So, let us rename the final state as q_f and have a new start state q_0 and from ~~w~~ this symbol on 1, enter into state q_1.



Ex 3: Obtain a DFSM that accepts set of all strings that, when interpreted in reverse as a binary integer, is divisible by 5, Ex of strings in the language are 0, 10011, 1001100, 0101

Soln:- Reverse the direction of all arrow marks. except for the one labelled start.

Ex.4: Draw a DFA to accept decimal strings divisible by 3.

Step 1: $r = 10$, $d = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $k = 3$.

Step 2: After dividing any decimal number by 3, results in following remainders,

$i = 0, 1, 2 \Rightarrow q_0, q_1$ & q_2 are states.

Step 3: $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$.

$r = 10, \quad k = 3$

$j = (2 * i + d) \bmod 3$

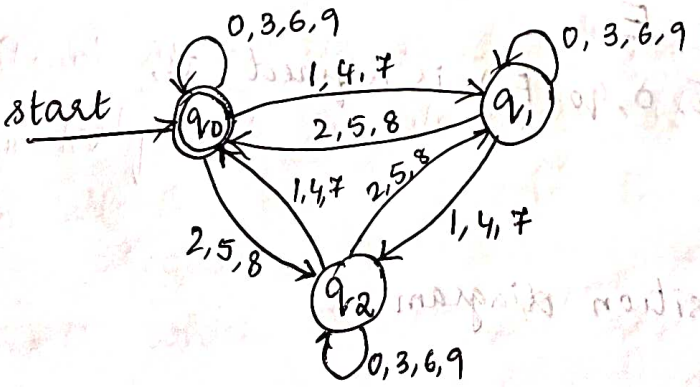
for convenience let us group the digits 0 to 9 based on the remainders after dividing by 3,

$\{0, 3, 6, 9\}$ with 0 as remainder δ from $\{0,3,6,9\} \Rightarrow \delta$ from $\{0\}$

$\{1, 4, 7\}$ with 1 as remainder, so δ from $\{1,4,7\} \Rightarrow \delta$ from $\{1\}$

$\{2, 5, 8\}$ with 2 as remainder, so δ from $\{2,5,8\} \Rightarrow \delta$ from $\{2\}$

remainder	d	$(10 * i + d) \bmod 3 = j$	$\delta(q_i, d) = q_j$	
$i = 0$	0	$(10 * 0 + 0) \bmod 3 = 0$	$\delta(q_0, 0) = q_0$	$\delta(q_0\{0,3,6,9\}) = q_0$
	1	$(10 * 0 + 1) \bmod 3 = 1$	$\delta(q_0, 1) = q_1$	$\delta(q_0\{1,4,7\}) = q_1$
	2	$(10 * 0 + 2) \bmod 3 = 2$	$\delta(q_0, 2) = q_2$	$\delta(q_0\{2,5,8\}) = q_2$
$i = 1$	0	$(10 * 1 + 0) \bmod 3 = 1$	$\delta(q_1, 0) = q_1$	$\delta(q_1\{0,3,6,9\}) = q_1$
	1	$(10 * 1 + 1) \bmod 3 = 2$	$\delta(q_1, 1) = q_2$	$\delta(q_1\{1,4,7\}) = q_2$
	2	$(10 * 1 + 2) \bmod 3 = 0$	$\delta(q_1, 2) = q_0$	$\delta(q_1\{2,5,8\}) = q_0$
$i = 2$	0	$(10 * 2 + 0) \bmod 3 = 2$	$\delta(q_2, 0) = q_2$	$\delta(q_2\{0,3,6,9\}) = q_2$
	1	$(10 * 2 + 1) \bmod 3 = 0$	$\delta(q_2, 1) = q_0$	$\delta(q_2\{1,4,7\}) = q_0$
	2	$(10 * 2 + 2) \bmod 3 = 1$	$\delta(q_2, 2) = q_1$	$\delta(q_2\{2,5,8\}) = q_1$



String length Mod k problems:

Step1: find remainders obtained by dividing string length by k.

Ex: divisor 2 $i = 0, 1$

3 $i = 0, 1, 2$

4 $i = 0, 1, 2, 3$

Step2: Identify the transitions

$\delta(q_i, a) = q_{(i+1 \bmod k)}$

Step3: Identify start & final state.

$L = \{w : |w| \bmod k = j\}$ q_j final state

Ex: $L = \{w : |w| \bmod k = 0\}$ then q_0 is the final state

Step4: Write the DFA.

1) Obtain a DFA to accept strings of even no of a's

$L = \{w : |w| \bmod 2 = 0\}$ q_0 final.

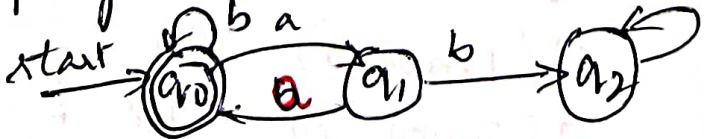
i $\delta(q_i, a) = q_{(i+1 \bmod k)}$

0 $\delta(q_0, a) = q_{(1 \bmod 2)} = q_1$

1 $\delta(q_1, a) = q_{(2 \bmod 2)} = q_0$

2) Obtain a DFA to accept $L = \{w \in \{a,b\} :$ every 'a' region in w is of even length$\}$

After accepting even a's we can accept any no of b's



Nxt q_1 of 'b' → trap state

$M = \{Q, \Sigma, \delta, q_0, F\}$

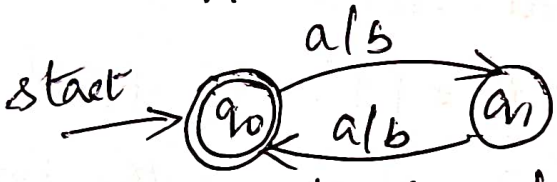
table, & State

3) Obtain a DFA to accept strings of a's & b's having even no of symbols.

$L = \{w : |w| \bmod 2 = 0$

~~$k=2$~~, $i = 0, 1$

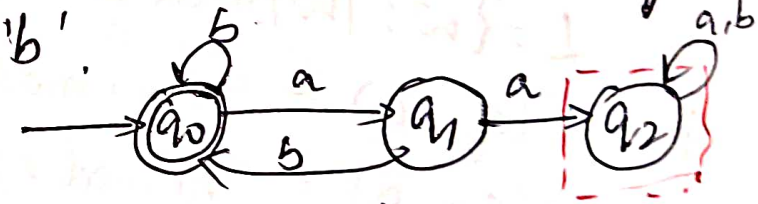
i	$\delta(q_i, a/b)$	$= q_{(i+1 \bmod 2)}$
0	$\delta(q_0, a/b)$	$= q_{(0+1 \bmod 2)} = q_1$
1	$\delta(q_1, a/b)$	$= q_{(1+1 \bmod 2)} = q_0$



SIGNATURE OF HOD

X: ODD no of symbols also changing status

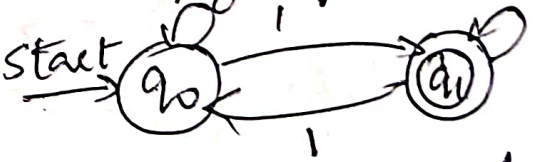
4) Obtain a DFA to accept the lang $L = \{w \in \{a,b\}$: every a is immediately followed by a 'b'.



5) Obtain a DFA to accept the lang

$L = \{w \in \{0,1\}$: 'w' has odd parity $\}$

$L = \{w : |w| \bmod 2 = 1\}$



6) Obtain a DFA to accept $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$ q_0 final, start

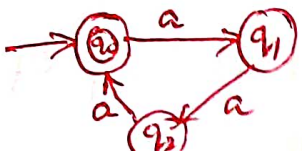
OR

w: $n_a(w)$ are divisible by 3

$k = 3$ $i = 0, 1, 2$

i	$\delta(q_i, a)$	$= q_{(i+1 \bmod k)}$
0	$\delta(q_0, a)$	$= q_{(1 \bmod 3)} = q_1$
1	$\delta(q_1, a)$	$= q_{(2 \bmod 2)} = q_2$
2	$\delta(q_2, a)$	$= q_{(2+1 \bmod 3)} = q_0$

$L = \{a^{3n} : n \geq 0\}$

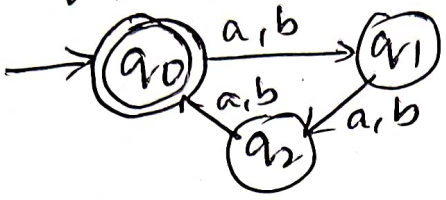


7) Obtain a DFA to accept the lang:

$L = \{w : (w) \bmod 3 = 0\}$ on $\Sigma = \{a, b\}$

$i = 0, 1, 2 \quad k = 3$

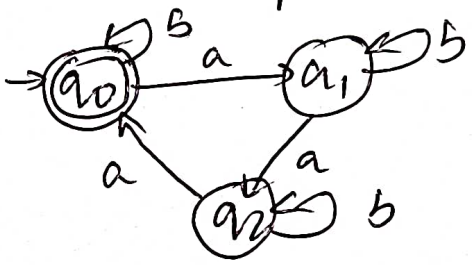
$\delta(q_i, (a, b)) = q_{(i+1 \bmod 3)}$



8) Obtain a DFA to accept $L = \{w : n_a(w) \bmod 3 = 0\}$

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

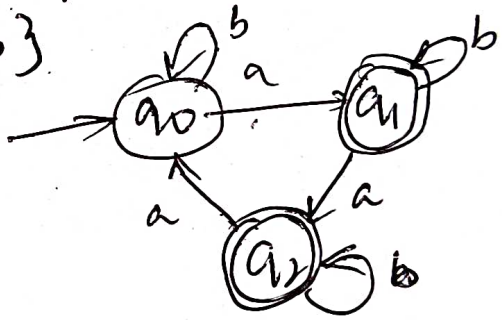
No. restriction of b's



9) obtain a DFA to accept $L = \{w : n_a(w) \bmod 3 \neq 0\}$ on $\Sigma = \{a, b\}$

$q_0 \rightarrow$ Not final

all others final.



10) Draw a DFA to accept lang

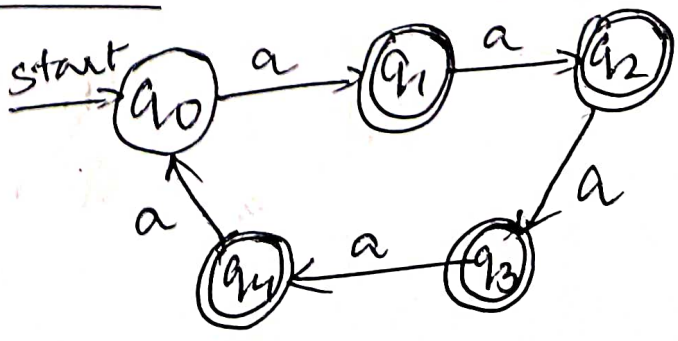
$L = \{w : |w| \bmod 5 \neq 0\}$ on $\Sigma = \{a\}$

$k = 5, \quad i = 0, 1, 2, 3, 4$

Note: $L = \{w : |w| \bmod 5 = 0\}$ 0 final now others final.

$\delta(q_i, a) = q_{(i+1 \bmod k)} \quad k = 5, \; i =$

i	$\delta(q_i, a)$
0	$\delta(q_0, a) = q_1$
1	$\delta(q_1, a) = q_2$
2	$\delta(q_2, a) = q_3$
3	$\delta(q_3, a) = q_4$
4	$\delta(q_4, a) = q_0$



String length mod k problems with relational operators:

Step 1: Solve the string length mod k problems connected by a relational operator as two independent problems.

Step 2: Now, the states Q of combined machine M that accepts a given string 'w' is obtained by taking cross product of Q1 & Q2,

$Q = Q1 \times Q2$ Q1 & Q2 are states of M1 & M2 resp

Step 3: Obtain the transitions of δ of combined machine using the transitions $\delta 1$ and $\delta 2$ of machines M1 & M2 resp.

$$\delta(\{p,q\}, a) = (\delta 1(p,a), \delta 2(q,a))$$

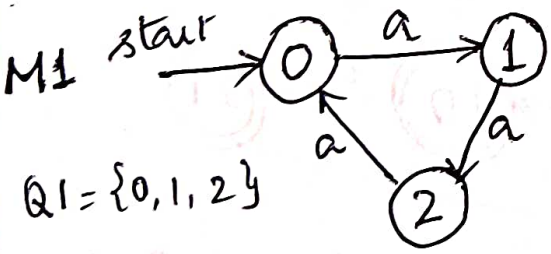
Step 4: Using relation identify final states.

Ex: Obtain a DFA to accept a string 'w' satisfying the following condition:

a) $|w| \bmod 3 \geq |w| \bmod 2$ where $w \in \Sigma = \{a\}$

Soln:- Construct to machines M1 & M2

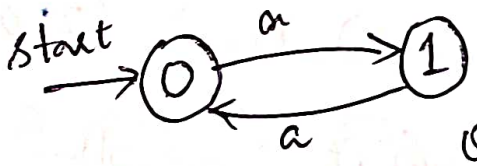
$|w| \bmod 3$



$Q1 = \{0,1,2\}$

δ	a
0	1
1	2
2	0

$|w| \bmod 2$



$Q2 = \{0,1\}$

δ	a
0	1
1	0

* Machine M1 accepts a string 'w' whose length is divided by 3 i.e., $|w| \bmod 3$

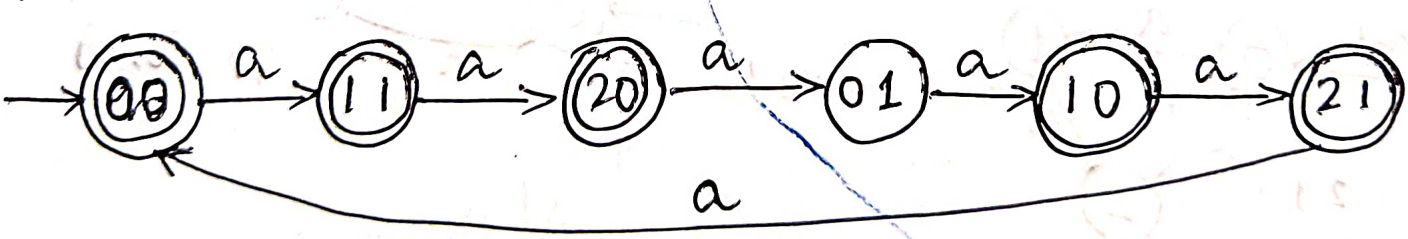
$Q = Q1 \times Q2$ $\{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\}$

So (0,0) is the start state

Transitions M $\delta(\{p,q\},a)$	M1 $\delta 1(p,a)$	M2 $\delta 2(q,a)$
$\delta((0,0),a)$	$\delta 1(0,a)$,	$\delta 2(0,a) = (1,1)$
$\delta((1,1),a)$	$\delta 1(1,a)$	$\delta 2(1,a) = (2,0)$
$\delta((2,0),a)$	$\delta 1(2,a)$	$\delta 2(0,a) = (0,1)$
$\delta((0,1),a)$	$\delta 1(0,a)$	$\delta 2(1,a) = (1,0)$
$\delta((1,0),a)$	$\delta 1(1,a)$	$\delta 2(0,a) = (2,1)$
$\delta((2,1),a)$	$\delta 1(2,a)$,	$\delta 2(1,a) = (0,0)$

To accept strings of 'w' such that $|w| \bmod 3 \geq |w| \bmod 2$ the pairs (x,y) such that $x \geq y$ are final states.

$F = \{(0,0), (1,1), (2,0), (1,0), (2,1)\}$



To accept strings of 'w' such that $|w| \bmod 3 \neq |w| \bmod 2$ the pairs (x,y) such that $x \neq y$ are final states.

$F = \{(2,0), (0,1), (1,0), (2,1)\}$

Obtain a DFA to accept the following lang.

$L = \{w$ such that

a) $|w| \bmod 3 \geq |w| \bmod 2$ where $w \in \Sigma^*$ & $\Sigma = \{a, b\}$

b) $|w| \bmod 3 \neq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a, b\}$

Number of specified symbols mod k problems:

$N_a(w)$ mod k problems:

1) Obtain a DFA to accept strings of even no of a's.

2) Draw a DFA to accept lang: $L = \{N_a(w) \bmod 5 \neq 0\}$

3) Draw a DFA to accept lang: $L = \{N_a(w) \bmod 3 = 0\}$

Number of specified symbols mod k problems with logical operators $N_a(w)$ mod k.

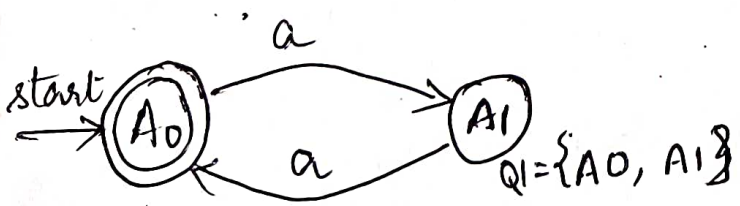
General procedure remains the same as string length mod 'k' problems.

i) Obtain a DFA to accept strings of a's and b's having even number of a's and even no of b's.

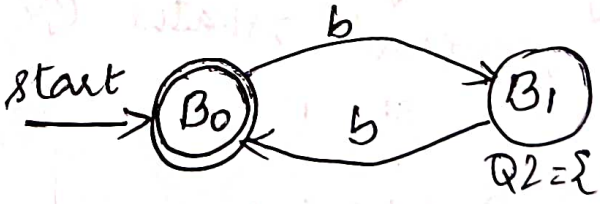
$L = \{N_a(w) \bmod 2 = 0$ and $N_b(w) \bmod 2 = 0\}$

$L = \{N_a(w) \bmod 2 = 0\}$ — M1

$\{N_b(w) \bmod 2 = 0\}$ — M2



$Q_1 = \{A0, A1\}$

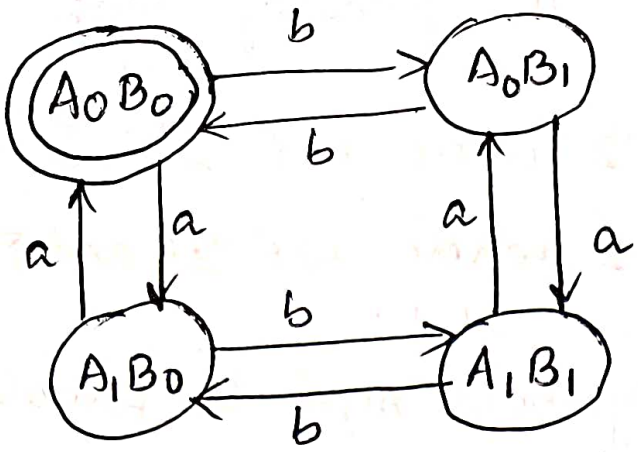


$Q_2 = \{B0\ B1\}$

$Q = Q_1 \times Q_2 = \{(A_0B_0), (A_0B_1), (A_1B0), (A_1B_1)\}$

// horizontal transition for i/p symbol b.

// vertical transition for i/p symbol a.



DFA to accept even no of a's and even no of b's.

$L = \{w : w$ has even no of a's & even no of b's$\}$

$L = \{w :$ Both $N_a(w)$ & $N_b(w)$ are divisible by $2\}$

$L = \{w :$ Both $N_a(w)$ & $N_b(w)$ are multiples of $2\}$

$L = \{w / N_a(w) \bmod 2 = 0$ and $N_b(w) \bmod 2 = 0\}$

(A_0B_1) – even a's & odd b's

(A_1B_0) – odd a's & even b's

(A_1B_1) – odd a's & odd b's.

Disadvantages of DFA:

1) Constructing some of the DFA's very difficult
2) DFA cannot guess about its i/p.
3) DFA is not very powerful.
4) At any point of time, the DFA is in ~~only~~ only one state. not powerful to be in several states @ a time

Obtain a DFA to accept the language

$L = \{ w : |w| \bmod 5 \neq 0 \}$ on $\Sigma = \{a\}$

Not multiples of 5 a's

no. of a's the string is not divisible by 5,

Identify states $|w| \bmod 5$

0, 1, 2, 3 & 4 results in remainders so five ca

case 0 : q_0 q_2 q_3 q_4

1 q_1

Identify start & final.

0 a's — so q_0 is start

so $|w| \bmod 5 \neq 0$ remainder is not zero

q_1 q_2 q_3 & q_4 are final state

Design-

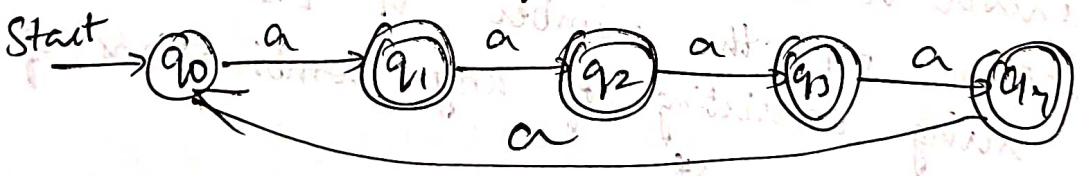
$\delta(q_0, a) = q_1$ q_2

$(q_1, a) = q_2$

$(q_2, a) = q_3$

$(q_3, a) = q_4$

$(q_4, a) = q_0$



$L = \{ w : |w| \bmod 5 \neq 0$ on $\Sigma = \{a, b\}$

Obtain a DFA to accept strings of a's & b's having even no of a's & even no of b's

Identify the no of states

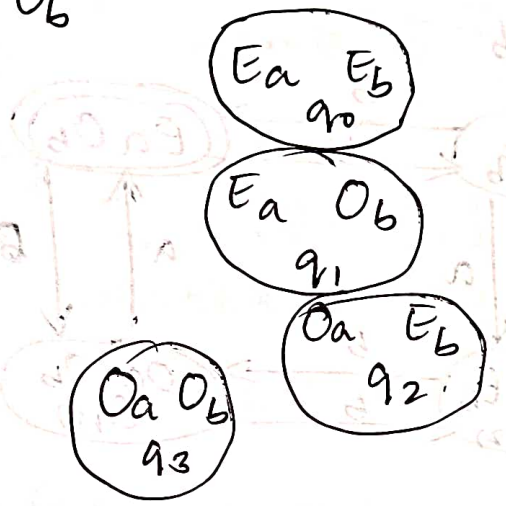
Even no of		
E	a's	E_a
O	b's	E_b
O	a's	O_a
O	b's	O_b

Case 0 E_a E_b

E_a O_b

O_a E_b

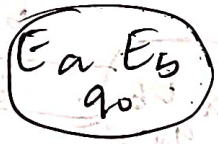
O_a O_b



Identify start & final state

before reading any a is 0 b is 0 $E_a q_0 E_b$

Even a & b so final



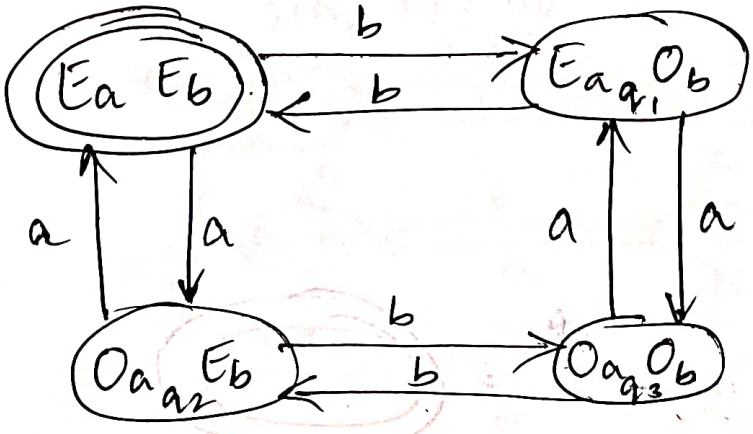
Design:

$\delta(E_a, a) = O_a$

$\delta(O_a, a) = E_a$

$\delta(E_b, b) = O_b$

$\delta(O_b, b) = E_b$



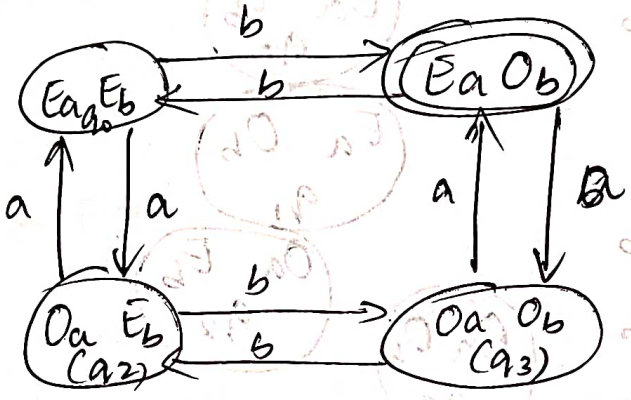
even no of a's & even b's

$L = \{w : \text{both } N_a(w) \text{ \& } N_b(w) \text{ are divisible by } 2\}$

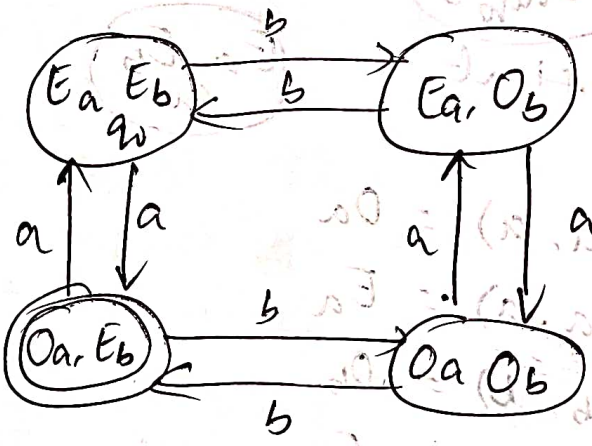
$L = \{w : \text{both } N_a(w) \text{ \& } N_b(w) \text{ are multiples of } 2\}$

$L = \{w \mid N_a(w) \bmod 2 = 0 \text{ \& } N_b(w) \bmod 2 = 0\}$

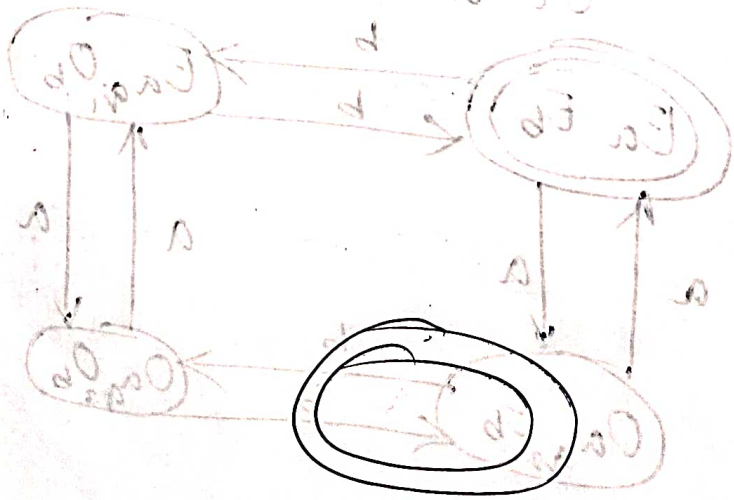
Next, E_a, O_b



Nxt, O_a, E_b



$O_a O_b$



Obtain a DFA to accept strings of a's & b's such that

$L = \{w \mid w \in (a+b)^*$ such that $N_a(w) \bmod 3 = 0$ & $N_b(w) \bmod 2 = 0\}$

Soln:- The $N_a(w) \bmod 3$ gives the remainder after dividing number of a's by 3 $\{0, 1, 2\}$

$Q_1 = \{A_0, A_1, A_2\}$ — $N_a(w) \bmod 3 = 2$

$N_a(w) \bmod 3 = 0$ $\quad$ $N_a(w) \bmod 3 = 1$

$N_b(w) \bmod 2$

number of b's by 2 $\{0, 1\}$

$Q_2 = \{B_0, B_1\}$

Identify the states

So possible states obtained by cross product of Q_1 & Q_2

$Q \times Q_2 = \{(A0, B0), (A0, B1), (A1, B0), (A1, B1), (A2, B0), (A2, B1)\}$

Start state: before reading any input.

So, $N_a(w) \bmod 3 = 0$ $\quad$ $N_b(w) \bmod 2 = 0$..

$(A0, B0)$ — Start state

final state $L = \{w: N_a(w) \bmod 3 = 0$ (↓ A0) & $N_b(w) \bmod 2 = 0\}$ (↓ B0)

$(A0, B0)$ — final state

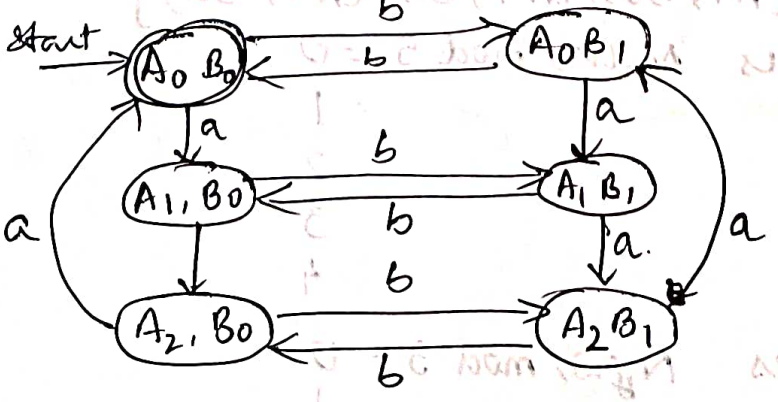
Design:

A0 on reading input symbol 'a', A1

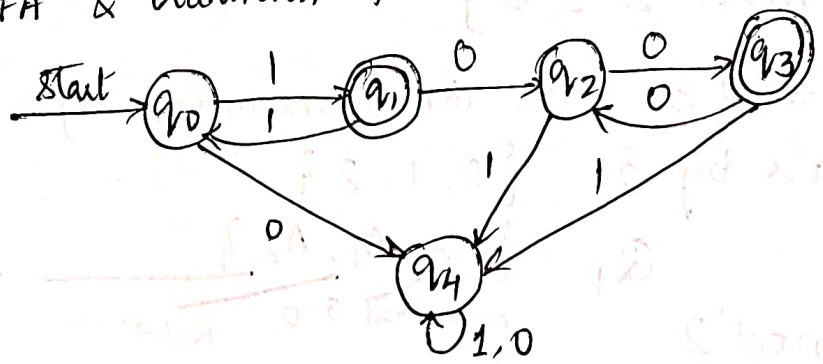
$\delta(A_0, a) = A_1$ $\quad$ $\delta(B_0, b) = B_1$

$\delta(A_1, a) = A_2$ $\quad$ $\delta(B_1, b) = B_0$

$\delta(A_2, a) = A_0$



Obtain a DFA to accept language $L = \{w : w$ has odd number of 1's and followed by even number of 0's$\}$ Completely define DFA & transition function.



Obtain a DFA to accept strings of a's & b's such that the no of a's is divisible by 5 & no of b's is divisible by 3

$$L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 5 = 0 \;\&\; N_b(w) \bmod 3 = 0\}$$

The $N_a(w) \bmod 5$ gives the remainder after dividing no. of a's by 5, remainder are $\{0, 1, 2, 3, 4\}$

$$Q_1 = \{A0, A1, A2, A3, A4\} \quad \text{---} \quad (1)$$

$N_b(w) \bmod 3$, $\{0, 1, 2\}$

$$Q_2 = \{B0, B1, B2\}$$

Identify the states:

$$Q_1 \times Q_2 = \{(A0, B0), (A0, B1), (A0, B2), (A1, B0), (A1, B1), (A1, B2), (A2, B0), (A2, B1), (A2, B2), (A3, B0), (A3, B1), (A3, B2), (A4, B0), (A4, B1), (A4, B2)\}$$

A0 indicates $N_a(w) \bmod 5 = 0$

A1 = 1

A2 2

A3 3

A4 4

B0 indicates $N_b(w) \bmod 3 = 0$

B1 1

B2 2

Identify start state (A0, B0) start.

final $L = \{w : N_a(w) \bmod 5 = 0 \;\&\; N_b(w) \bmod 3 = 0\}$

$N_a(w) \bmod 5 = 0$ → A0 ; $N_b(w) \bmod 3 = 0$ → B0

(A0, B0)

Design:

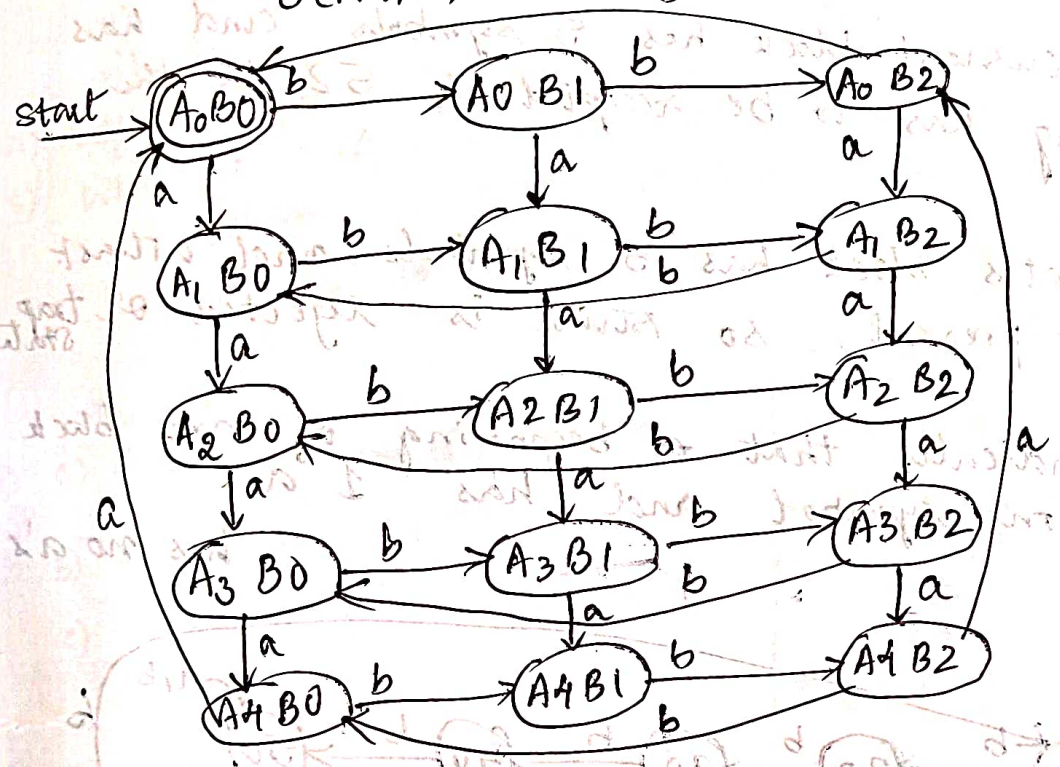
$\delta(A_0, a) = A1$ $\quad$ $\delta(B0, b) = B1$

$\delta(A1, a) = A2$ $\quad$ $\delta(B1, b) = B2$

$\delta(A2, a) = A3$ $\quad$ $\delta(B2, b) = B0$

$\delta(A3, a) = A4$

$\delta(A4, a) = A0$



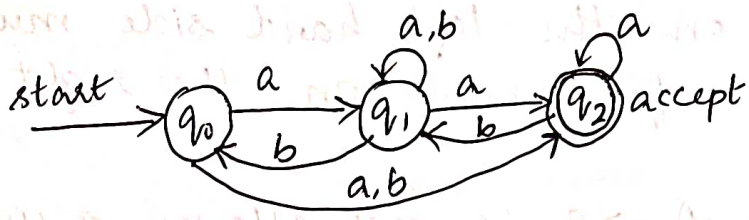
Nxt $L = \{w : n_a(w) \bmod 5 \neq n_b(w) \bmod 3\}$

Except states (A0, B0) (A1, B1) (A2, B2) the rest of the states are final states.

Nxt $L = \{w : n_a(w) \bmod > n_b(w) \bmod 3\}$

(A_i, B_j) if here $i > j$ those are the final states

Non-deterministic finite Automaton:



Advantages:
* Constructing most of the NFA is very easy.
* NFA has ability to guess something about its i/p.
* It is more powerful than DFA.
* It has the power to be in several states at once.
* An NFA is an efficient mechanism to describe some complicated lang concisely.

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

Power set of Q: 2^Q

$2^Q = \{\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

Input alphabets:

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

Transitions:

Change of state after consuming an i/p symbol.

q_i to q_j on 'a'

$$\delta(q_i, a) = q_j$$

Change of state

q_i to q_j & q_i to q_k on 'a'

$$\delta(q_i, a) = \{q_j, q_k\}$$

Current state	I/p	Next state	Rep.
q_0	a	q_1, q_2	$\delta(q_0, a) = \{q_1, q_2\}$
q_0	b	q_2	$\delta(q_0, b) = q_2$
q_1	a	q_1, q_2	$\delta(q_1, a) = \{q_1, q_2\}$
q_1	b	q_0, q_1	$\delta(q_1, b) = \{q_0, q_1\}$
q_2	a	q_2	$\delta(q_2, a) = \{q_2\}$
q_2	b	q_1	$\delta(q_2, b) = \{q_1\}$

$$\delta : Q \times \Sigma \text{ to } 2^Q$$

So, the transition function δ is defined as,

$$\delta : Q \times \Sigma \text{ to } 2^Q$$

So from the discussion it is observed that NFA has 5 components,

$\{Q, \Sigma, \delta, q_0, F\}$

Def:- The non-deterministic finite Automaton is a 5-tuple or quintuple indicated 5 components,

M = name of the machine

Q = is non-empty, finite set of states.

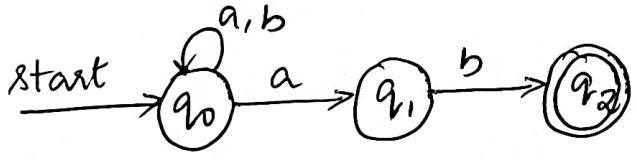
Σ = is non-empty, finite set of input alphabets.

δ = $Q \times \Sigma$ to 2^Q is transition function which is a mapping from $Q \times \Sigma$ to 2^Q.

$q_0 \in Q$ - is the start state.

$F \subseteq Q$ - is set of accepting or final states.

Ex: NFA to accept strings of a's & b's ending with ab is shown below,



δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$*q_2$	ϕ	ϕ

Sequence of moves made by NFA for the string aaab and aba

$(q_0, aaab) \vdash (\{q_0, q_1\}, aab)$
$\vdash (\{q_0, q_1\}, ab)$
$\vdash (\{q_0, q_1\}, b)$
$\vdash (\{q_0, q_2\}, \varepsilon)$

final configuration

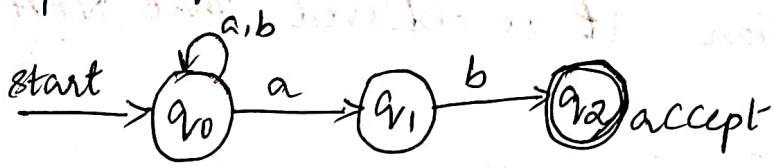
Set $\{q_0, q_2\}$ has a final state and so the string is accepted

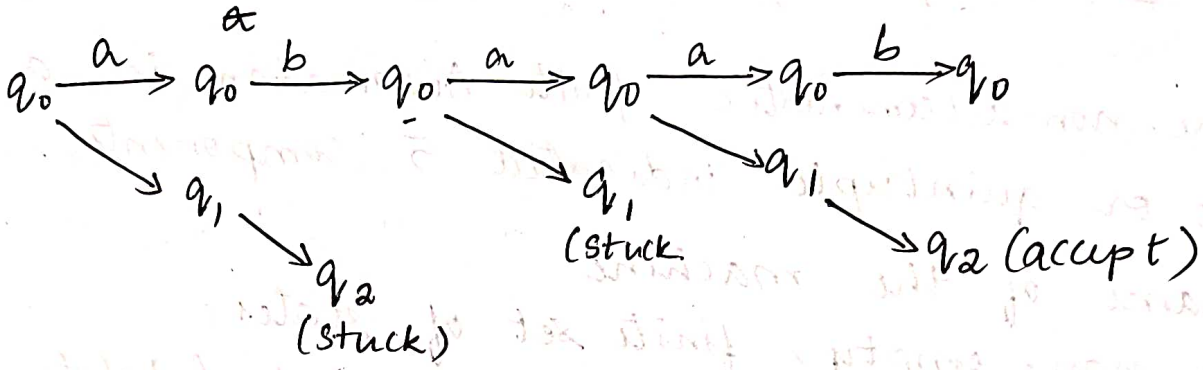
$(q_0, aba) \vdash (\{q_0, q_1\}, aa)$
$\vdash (\{q_0, q_1\}, a)$
$\vdash (\{q_0, q_1\}, \varepsilon)$

final config

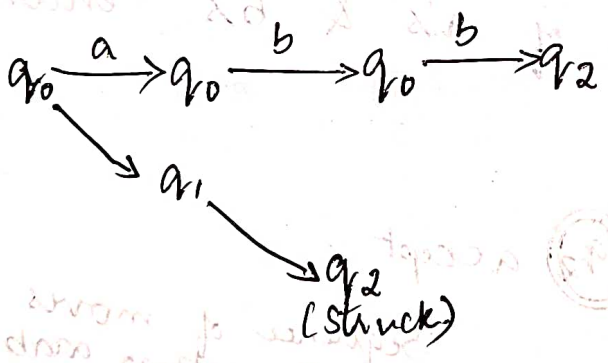
No final state, string is rejected.

What are the states NFA is in during processing of input seq. abaab & abb?





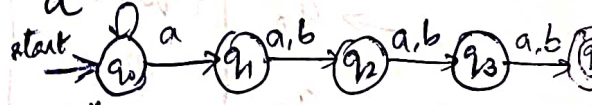
* q_2 on a there is no transition hence NFA is stu
* q_1 on b NFA goes to q_2. So at the end, NFA is in q_2 which is a final state and hence the input string abaab is accepted by the machine.



Hence the string abb, is rejected by the machine

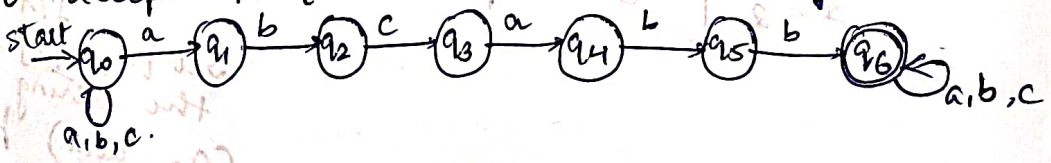
Obtain an NFA to accept strings of a's and b's such that fourth symbol from the right side is a

R.E ~~(a+b)~~ a(a|b)(a|b)(a|b)(a|b)



Obtain an NFA to accept $L=\{w \in \{a,b,c\}^* : x \text{ and } y \in \{a,b,c\}^*$

$w = x\,abcabb\,y$

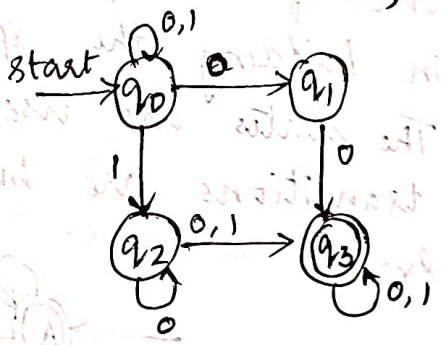


Difference b/w NFA & DFA.

DFA	NFA
1) Every input string leads to the unique state of finite automata.	For the same i/p there can be more than one nxt state.
2) Conversion of regular expression to DFA is complex.	Regular exp can be easily converted to NFA using Thompson's construction.
3) The DFA requires more memory for storing the state info.	The NFA requires more computations to match RE with i/p.
4) The DFA it is not possible to move to nxt state without reading any symbol.	In NFA we can move to nxt state without reading any symbols.

Ex: Design the NFA transition diagram for the t.t below,

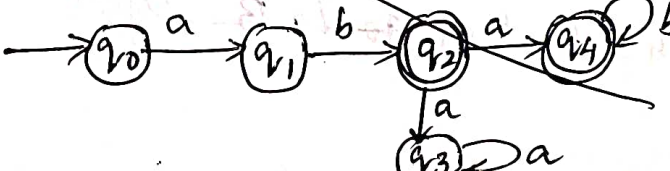
	0	1
→* q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_3\}$	
q_2	$\{q_2, q_3\}$	$\{q_3\}$
* q_3	$\{q_3\}$	$\{q_3\}$



Ex 2: Design an NFA with no more than 5 states for foll lang.

$L = \{abab^n \mid n \geq 0\} \cup \{aba^n \mid n \geq 0\}$

* The strings belonging to $abab^n$ are $A = \{aba, abab, ababb, abab bb..\}$
* The strings belonging to aba^n are $B = \{ab, aba, abaa, abaaa..\}$
* The union of A & B sets give us $\{ab, aba, abab, ababb, abaaa..\}$



From FSM to OS:

FSM is an abstract machine that can be used for solving various problems without worrying about implementation details:

Uses of FSM in OS are:

1) The FSM can be translated into circuit design & implemented directly in hardware.
2) The FSM is useful as general purpose interpreter or compiler. [Simulation - to check a design before it is translated into hardware]
3) For describing the behavior of the complex systems the FSM can be used. The specification of FSM can be used in implementing or writing the software solution for such complex systems.

Simulators for FSM:

Once created, we need to simulate its execution.

* One of the application of FSM is that it is useful in building the specification.
* The states are used for designing the routines while transitions are build to i/p to nxt state.

Ex: Consider an ex DFSM that accepts the language: $L = \{w \in \{a, b\}^* : w$ contains no more than one $b\}$

Program for FSM, pseudo code is,

```
A:  Ch = get-nxt-symbol
    If ch = end-of-file then accept
    else if ch = a then go to A
    else if ch = b then go to B
B:  Ch = get-nxt-symbol
    if ch = end-of-file then accept
    else if ch = a then go to B
    else if ch = b then reject
END
```

* The problem with above method is that - we need to generate new code for every FSM we create.
* To overcome this problem, an interpreter that builds the FSM is created.
* The interpreter is specified for deterministic FSM

$$M = (K, \Sigma, \delta, s, A)$$

K: states, s: start state, A: accepting states

Algorithm:

dfs-simulate (M : DFSM, w : string) =

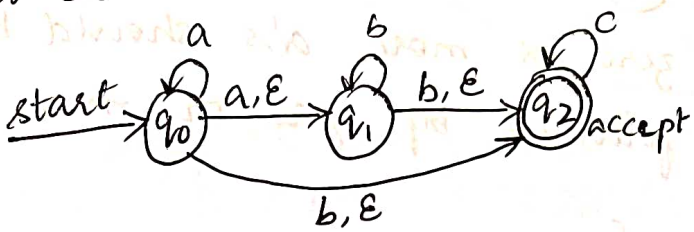
1. st = s.
2. Repeat:
   2.1. ch = get-nxt-symbol (w)
   2.2 If ch ≠ end-of-file then:
   2.2.1 st = δ(st, ch)
   until ch = end-of-file
3. If st ∈ A then accept else reject.

ε-NFA :-

A transition with an empty input string is called an ε-transition.

Transition from one state to another state without any input is called ε-transition.

Ex:



States:

Q

2^Q =

$q_0 \in Q$ is the start state.

$F \subseteq Q$ which is the accepting or final state.

Input alphabets:

Σ =

Transitions:

$\delta(v_i, a) = q_j$

$\delta(v_i, a) = \{q_j, q_k\}$

$\delta(v_i, \varepsilon) = \{q_j\}$

$\boxed{\delta : (Q \times (\Sigma \cup \varepsilon)) \text{ to } 2^Q}$

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

$\delta = Q \times (\Sigma \cup \varepsilon)$ to 2^Q

ε-CLOSURE :- The ε-CLOSURE of 'q' denoted by ECLOSE (q) is the set of all states which are reachable from 'q' on ε-transitions only.

The states q_0, q_1 & q_2 are reachable from q_0 without giving any i/p, so,

ECLOSE $(q_0) = \{q_0, q_1, q_2\}$

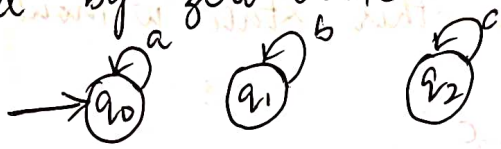
states q_1 & q_2 are reachable from q_1 so

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

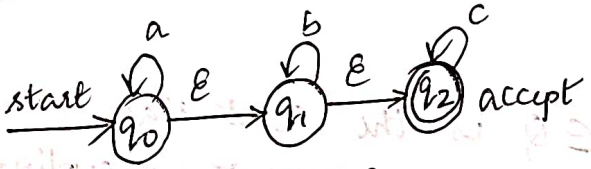
states q_2 is reachable from q_2 without giving an i/p so

ECLOSE $(q_2) = \{q_2\}$

Ex:- "Obtain an ε-NFA which accepts strings consisting of zero or more a's followed by zero or more b's followed by zero or more c's"



But it is given that zero or more a's should be followed by zero or more b's followed by zero or more c's.



δ	a	b	c	ε
→ q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
* q_2	ϕ	ϕ	q_2	ϕ

Conversion from ε-NFA to DFA (Algorithm to convert ε-NFA to DFA)

Procedure: $M_E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ε-NFA where q_{0E} is the start state & F_E final states. The equivalent DFA, $M_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

Step 1: If q_0 is the start ~~start~~ state of NFA, then ECLOSE(q_0) is the start state of DFA

$$q_{0D} = \text{ECLOSE}(q_0)$$

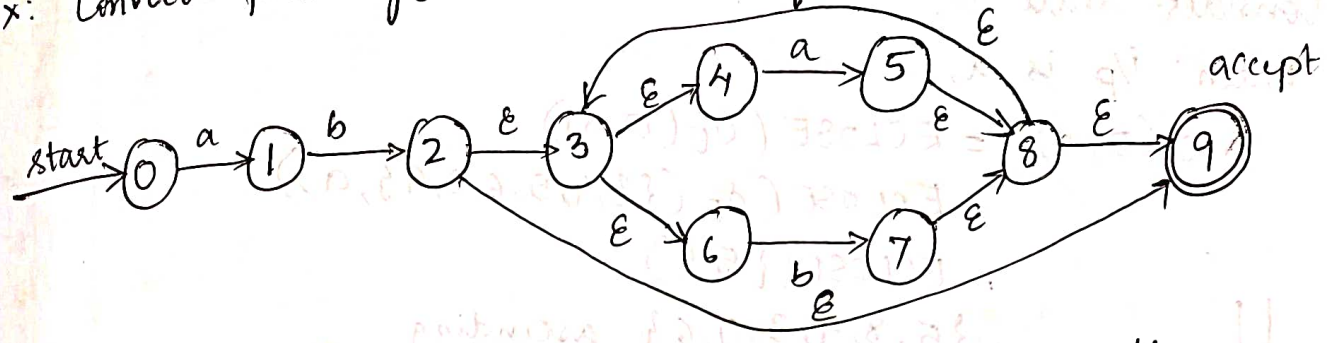
Step 2: Compute the transitions from DFA, Let $\{q_i, q_j \ldots q_k\}$ is a state in DFA then, the transitions from this state on a i.e., $\delta_D(\{q_i, q_j \ldots q_k\}, a)$

$$\delta_E(\{q_i, q_j \ldots q_k\}, a) = \{p_1, p_2 \ldots p_m\}$$

Then take ECLOSE $\{p_1, p_2 \ldots p_m\}$

Step 3: If $\{q_i, q_j \ldots q_k\}$ is a state in DFA and if this set contains atleast one final state of ε-NFA then $\{q_i, q_j \ldots q_k\}$ is a final state of DFA.

Ex: Convert following ε-NFA to its equivalent DFA:



Step 1: Identify the start state of DFA, 0 is the start state of ε-NFA, ECLOSE(0) is the start state of DFA. ECLOSE(0) = {0} — A

Consider state A,
when i/p is a

$\delta(A, a)$ = ECLOSE($\delta_E(A, a)$)
= ECLOSE($\delta_E(0, a)$)
= {1} — B

i/p is b

$\delta(A, b)$ = ECLOSE($\delta_E(A, b)$)
= ECLOSE($\delta_E(0, b)$)
= {ϕ}

Consider state B,
when i/p is a,

$\delta(B, a)$ = ECLOSE($\delta_E(1, a)$)
= ϕ

when i/p is b,

$\delta(B, b)$ = ECLOSE($\delta_E(B, b)$)
= ECLOSE($\delta_E(1, b)$)
= ECLOSE({2})
= {2, 3, 4, 6, 9} — C

Consider state C,
when i/p is a,

$\delta(C, a)$ = ECLOSE($\delta_E(C, a)$)
= ECLOSE(δ_E {2, 3, 4, 6, 9}, a)
= ECLOSE{5} = {5, 8, 9, 3, 4, 6} — D

{3, 4, 5, 6, 8, 9}

Consider state D,

when i/p is a,

$\delta(D, a)$ = ECLOSE ($\delta_E(D, a)$)

= ECLOSE (δ_E ({3,4,5,6,8,9}, a))

= ECLOSE({5})

= {5, 8, 9, 3, 4, 6} ascending

{3, 4, 5, 6, 8, 9} — D

when i/p is b,

$\delta(C, b)$ = ECLOSE ($\delta_E(C, b)$)

= ECLOSE (δ_E {2, 3, 4, 6, 9}, b)

= ECLOSE {7}

= {3, 4, 6, 7, 8, 9} — E

when i/p is b

$\delta(D, b)$ = ECLOSE (δ_E (D, b))

= ECLOSE (δ_E {3, 4, 5, 6, 8, 9}, a))

= ECLOSE {7}

= {3, 4, 5, 6, 7, 8, 9} — E

Consider state E

when i/p is a

$\delta(E, a)$ = ECLOSE ($\delta(E, a)$)

= ECLOSE (δ_E (3, 4, 6, 7, 8, 9), a))

= ECLOSE {5}

= {3, 4, 5, 6, 8, 9} — D

when i/p is b,

$\delta(E, b)$ = ECLOSE ($\delta_E(E, b)$)

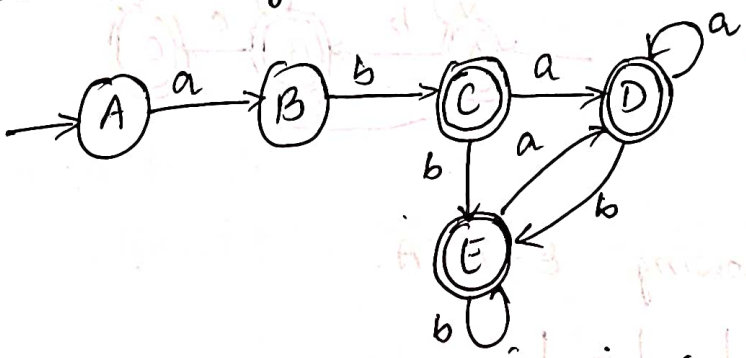
= ECLOSE (δ_E {3, 4, 6, 7, 8, 9}, b)

= ECLOSE {7}

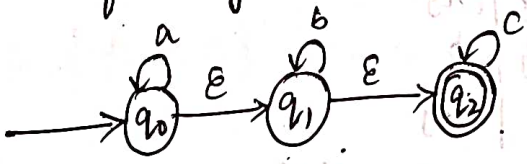
= {3, 4, 6, 7, 8, 9} — E

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

The states C, D, E are final states since q which is final state of ε-NFA is present in C, D & E. The final transition diagram of DFA



Convert the following NFA to its equivalent DFA:



Identify start state, $ECLOSE(q_0) = \{q_0, q_1, q_2\}$ — A

Consider state $\{q_0, q_1, q_2\}$

on i/p a, $\delta\{(q_0, q_1, q_2), a\} = ECLOSE\{q_0\}$
$= \{q_0, q_1, q_2\}$ — A

on i/p b, $\delta((q_0, q_1, q_2), b) = ECLOSE\{q_1\}$
$= \{q_1, q_2\}$ — B

on i/p c, $\delta((q_0, q_1, q_2), c) = ECLOSE\{q_2\}$
$= q_2$ — C

Consider state $\{q_1, q_2\}$

on i/p a, $\delta\{(q_1, q_2), a\} = ECLOSE\{\phi\} = \phi$

on i/p b, $\delta\{(q_1, q_2), b\} = ECLOSE\{q_1\}$
$= \{q_1, q_2\}$ — B

on i/p c $\delta\{(q_1, q_2), c\} = ECLOSE\{q_2\}$
— C

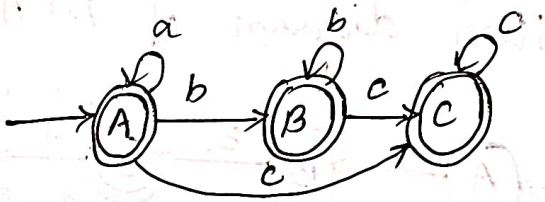
Consider state $\{q_2\}$

on i/p a, $\delta(q_2, a) = \phi$

on i/p b, $\delta(q_1, b) = \phi$

on i/p c, $\delta(q_2, c) = ECLOSE\{q_2\}$
$= q_2$ — C

δ	a	b	c
→*A	A	B	C
*B	ϕ	B	C
*C	ϕ	ϕ	C



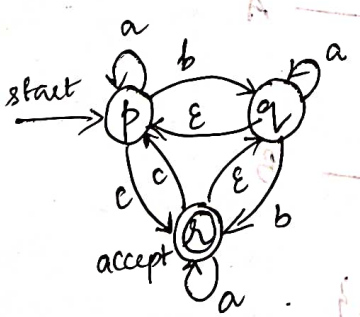
Consider the following ε-NFA:

δ	ε	a	b	c
→p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
*r	{q}	{r}	ϕ	{p}

i) Compute ε-closure of each state

ii) ~~Give all the strings of length 3 or less accepted by the machine~~

iii) Convert the machine to DFA



ε-CLOSURE of each state,

ECLOSE(p) = {p}

ECLOSE(q) = {p, q}

ECLOSE(r) = {p, q, r}

Identify the start state, 'p' is the state,

ECLOSE(p) = {p} —— A

Consider 'p' on i/p a

$\delta(p, a) = \delta_E\{p, a\}$

ECLOSE{p} = {p} —— A

on i/p b $\delta(p, b) = \delta_E\{p, b\}$

ECLOSE{p, q} = {p, q} — B

on i/p c,

$\delta(\{p\}, c) = ECLOSE(r) = \{p, q, r\}$ — C

consider state {p,q} on i/p a,

$\delta((p,q), a) = ECLOSE(\delta(p,a) \cup \delta(q,a))$

$ECLOSE(p, q)$

$= (p, q)$ — B

state {p,q} on i/p b,

$\delta((p,q), b) = ECLOSE(\delta(p,b) \cup \delta(p,b))$

$= ECLOSE(q, r)$

$= (p, q, r)$ — C

state {p,q} on i/p c,

$\delta((p,q), c) = ECLOSE(\delta(p,c) \cup \delta(q,c))$

$= ECLOSE(r, \phi)$

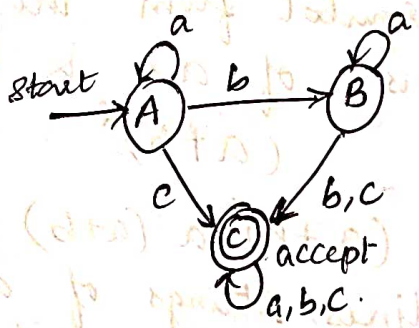
$= ECLOSE(r)$

$= \{p, q, r\}$ — C

* Consider state {p, q, r} i.e, C,

— C — C — C

δ	a	b	c
→ A	A	B	C
B	B	C	C
*C	C	C	C



Minimization of DFA:

Step 1: Divide Q into 2 groups: 1) final states
2) Nonfinal states π_0

~~Step 2:~~

Obtain the distinguishable table for the machine & then minimize the states of the following DFA:

δ	a	b
→A	B	F
B	G	C
*C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

partition:

{A B C D E F G H}

$\pi_0 = \{[A\ B\ D\ E\ F\ G\ H]_{G1}\ [C]_{G2}\}$

find π_1 on i/p symbol a, b:

$\pi_0 = \{[A\ B\ D\ E\ F\ G\ H]_{G1}, [C]_{G2}\}$ — a

D & F on i/p 'a' are entering G2 so remove [D, F] from G1 and create a n/w group.

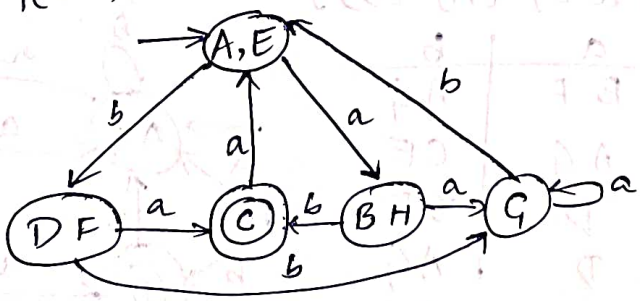
$\pi_1 \{[A\ B\ E\ G\ H]_{G1}\ [D, F]_{G2}\ [C]_{G3}\}$ — b

$\pi_2 = \{[A\ E\ G]_{G1}\ [B\ H]_{G2}\ [D, F]_{G3}\ [C]_{G4}\}$ → on i/p a & b,

$\pi_3 = \{[A\ E]_{G1}\ [B\ H]_{G2}\ [D\ F]_{G3}\ [C]_{G4}\ [G]_{G5}\}$ → on i/p a & b,

δ	a	b
→(AE)	(BH)	(DF)
(BH)	G	C
(DF)	C	G
*C	AE	C
G	G	(AE)

$\pi_0 \xrightarrow{a} \pi_1 \xrightarrow{b} \pi_2 \xrightarrow{a} \pi_3 \xrightarrow{b}$ No splitting



$\pi_0 = [\{A\ B\ D\ E\ F\ G\ H\}\ \{C\}]$ (G1, G2)

a

Find π_1 on i/p a

$\pi_1 = \{[A\ B\ E\ G\ H]\ [D\ F]\ [C]\}$ (G1, G2, G3)

b b

π_1 on i/p b

π_2 [A E] [B H] [D F] [C] [G]

π_2 on i/p a

π_3 [A E] [B H] [D F] [C] [G]

π_3 on i/p b

$\pi_4 = \{[A\ E]\ [B\ H]\ [D\ F]\ [C]\ [G]\}$

Find the minimized DFA for the following:

δ	0	1
→ A	B	A
B	A	C
C	D	B
* D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

$\pi_0 = \{[A\ B\ C\ E\ F\ G\ H]\ [D]\}$ (G1, G2)

on i/p '0',

$\pi_1 = \{[A\ B\ F\ G\ H]\ [C\ E]\ [D]\}$

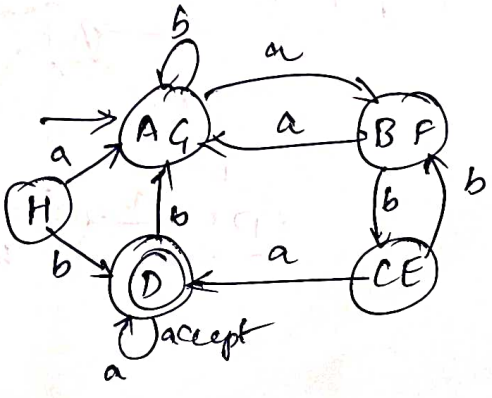
on i/p 1;

$\pi_2 = [A\ G]\ [B\ F]\ [C\ E]\ [D]\ [H]$ (0, 0)

on i/p 0 0

$\pi_3 \{[A\ G]\ [B\ F]\ [C\ E]\ [H]\ [D]\}$ on i/p 1

	a	b
A G	B F	A G
B F	A G	C E
C E	D	B F
D	D	A G
H	A G	D



Minimize the following DFA using table-filling algorithm.

δ	0	1
$\rightarrow$ A	B	E
B	C	F
* C	D	H
D	E	H
E	F	I
* F	G	B
G	H	B
H	I	C
* I	A	E

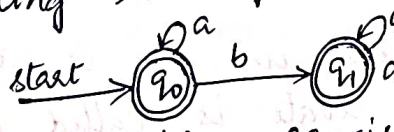
δ	a	b
ADH	BEH	BEH
BEH	CFI	CFI
* CFI	ADH	BEH

From FSM to operational Systems [Applications of FA]

Some of the applications where automata play an important role are:

* Design of digital circuits:- FSM can to translated into a circuit design and can be directly implemented in hardware.
* Compiler construction:- Used in the design of lexical analyzer (I phase of CD) which breaks the i/p text into units like identifiers, keywords, punctuation etc.
* String matching: In designing S/W for identifying words, phrases and other patterns in large bodies of text.
* String processing: To design S/W for processing the natural lang. Ex: Spelling checkers, multi-lang dictionaries, indenting does etc.
* S/W design: In building the S/W to verify the systems having finite no of states. Ex: CP in CN.
* Other applns: AI & Knowledge Engineering
  Game theory & games
  CG, linguistics etc. [lang & structure]

Simulating DFA: / Simulators of FSMs.



DFA accepts string consisting of a's & b's such that the string contains atmost one 'b'.

$L = \{w : w \in \{a, b\}^*$ and w contains no more than one $b\}$

State q_0:
- $a \rightarrow q_0$
- $b \rightarrow q_1$
- EOF $\rightarrow$ accept

State q_1:
- $a \rightarrow q_1$
- $b \rightarrow$ reject
- EOF $\rightarrow$ accept

Program segment can be written as:

```
repeat
    q0 : ch = get_next_symbol
         if (ch == a) then goto q0;
         if (ch == b) then goto q1;
         if (ch == EOF) then accept;
```

```
q1: ch = get_next_symbol;
    if (ch == a) then goto state q1
    if (ch == b) then reject
    if (ch == EOF) then accept
until (accept or reject)
```

Simple interpreter :-

```
DFA_Simulate (M: DFA, w: string)
{
    cs = q0
    repeat
        ch = get_next_symbol()
        if (ch != EOF) cs = δ(cs, ch)
    until (ch = EOF)
    if (cs ∈ F) then
        accept
    else
        reject
}
```

Finite State Transducers:

The FSM are used as lang recognizers. → a lang → accepts or reject

The process of converting the i/p data from one form to other form as the o/p is called transducers.

FST: A finite state machine where an o/p is associated with either a transition or a state is called FST.

FST are ~~calls~~ classified as,

1) Mealy machine.
2) Moore machine.

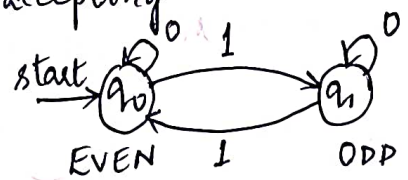
Mealy machine:- A finite state machine where an o/p is associated with each transition is called Mealy machine.

$M = (Q, \Sigma, \Gamma, \delta, \lambda, q_0)$

Q : finite set of states
Σ : set of i/p alphabets.
Γ : set of o/p alphabets.
δ : $Q \times \Sigma$ to Q
λ : o/p function which is a mapping from $Q \times \Sigma$ to Γ
q_0 : start state.

Construct a Mealy machine which accepts strings of 0's and 1's output EVEN when the i/p string has even no of 1's and o/p ODD when the i/p string has odd no of 1's.

FSM which accepts strings of 0's & 1's as the i/p accepting even no of 1's.



$q_0 \rightarrow (0,1)$ having even no of 1's
So, whenever there is a transition to state q_0, we associate the o/p E

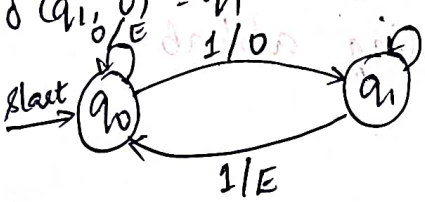
$\delta(q_0, 0) = q_0$ $\quad \lambda(q_0, 0) = E$

$\delta(q_1, 1) = q_0$ $\quad \lambda(q_1, 1) = E$

$q_1 \rightarrow (0,1)$ having ODD no of 1's
So, whenever there is a transition to state q_1, we associate the o/p 0.

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

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



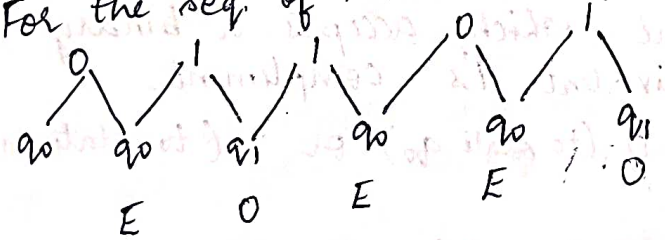
$M = (Q, \Sigma, \Gamma, \delta, \lambda, q_0)$

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_0

$\rightarrow$

λ	0	1
q_0	E	0
q_1	0	E

For the seq of moves made 01101,



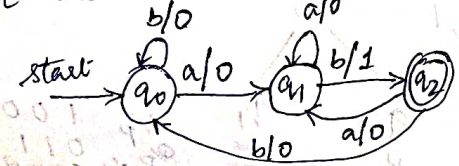
Construct a Mealy machine which accepts strings of a's and b's and count the no of times the pattern "ab" is present in the string.

Construct FSM $\rightarrow$ string ending with ab.

Change the FSM to Mealy machine.

entering q_2 $\rightarrow$ o/p 1

Other transition o/p 0



$M = (Q, \Sigma, \Gamma, \delta, \lambda, q_0)$

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

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

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

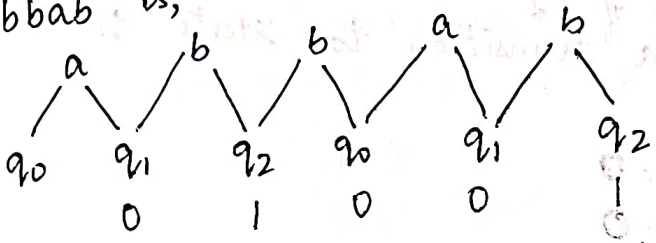
δ and λ are shown using transition table & o/p func

δ	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

λ	a	b
q_0	0	0
q_1	0	1
q_2	0	0

q_0 is the start state.

Seq of moves made by Mealy machine for i/p string abbab is,



In the o/p two 1's are present. So, the pattern 'ab' is present 2 times in the given string abbab

Construct a Mealy machine which accepts a binary no and produces its equivalent 1's complement.

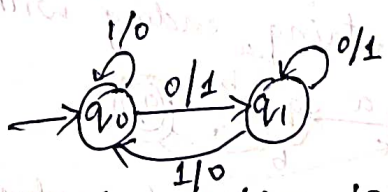
* FSM which produces either 0 (to state q_0) or 1 (to state q_1) for i/p symbols 1 & 0.

$(q_0, 1) = q_0$ for all transitions entering to state q_0, associate 0

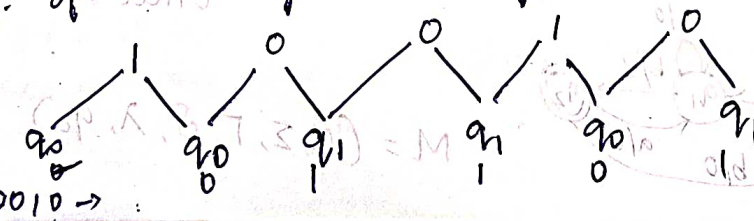
$q_0, 0) = q_1$ ——— q_1 associate 1

$(q_1, 0) = q_1$

$q_1, 1) = q_0$



Seq. of moves by the Mealy machine for i/p string 10010;



i/p 10010

o/p 01101 1's complement.

Moore machine:-

A finite state machine where an o/p is associated with each state is called Moore machine.

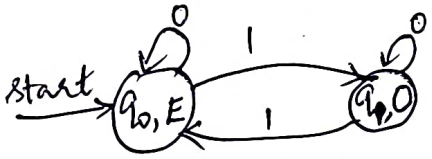
$M = (Q, \Sigma, \Gamma, \delta, \lambda, q_0)$

Construct a Moore machine which accepts strings of 0's & 1's o/p even when the i/p string has even no of 1's and o/p odd when the i/p string has odd no of 1's.

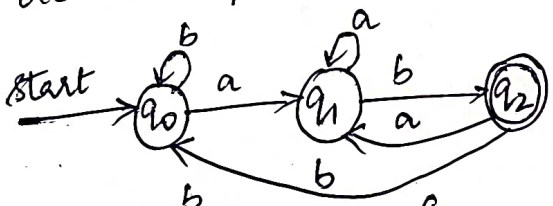
In state q_0, machine accepts strings of 0's & 1's having EVEN no of 1's.

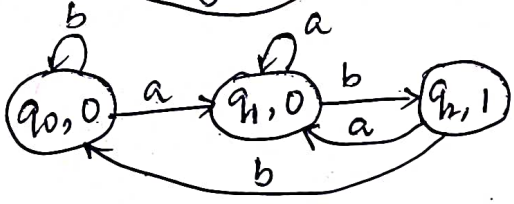
So, associate state q_0 with o/p E

——"—— 0



Construct a Moore machine which accepts strings of a's and b's and count the no of times the pattern "ab" is present.





Construct a Moore machine which accepts a binary no and produces its equivalent 1's complement.