Theory of Computation

The theory of computation is a branch of computer science that deals with whether and how efficiently problems can be solved on a model of computation using ~~and~~ algorithm.

It can be divided into 3 major branches,

1) Comp~~at~~utability theory,
2) Complex~~ibility~~ theory
3) Autometa theory.

1) Computability theory:-

It deals primarily with the question of whether a problem is solvable at all on a computer.

2) Comple~~xibility~~ theory:-

Complexity theory considers not only whether a problem can be solved at all on a computer but also how efficiently the problem can be solved

3) Autometa theory:-

Autometa theory is the study of ~~the~~ abstract machines and the computational problems that can be solved using

these machines:
The word 'Automata' is greek word which means something is doing something by itself.

Formal languages

Formal language consist of words whose letters are taken from an alphabet and are well formed according to a specific set of rules.

Some basics

- **Data** - is represented by strings of symbols
- **Alphabets** - is a finite set of symbols.

Alphabets are represented by ~~letter Σ~~ symbol Σ

eg:- $\Sigma = \{A, B, C \ldots\}$

$\Sigma = \{1, 2, 3\}$.

- **Strings** - a string is a finite ordered sequence of symbols formed from an alphabet denoted by w.

→ Let $\Sigma = \{0, 1\}$

$W = 0, 1, 00, 01, 10, 11 \ldots$

Length of a string is denoted by $|w|$.

eg:- $w = 10110$

$|w| = 5$.

- Empty string / Null string :-

It is the string with 0 occurance of symbol or it is a string whose length is zero and can be indicated ~~&~~ ϵ / λ.

- Empty set / Null set

Indicated by symbol ϕ.

- Σ^k - set of all strings of ~~set~~ legth k. over Σ.

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

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

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

$\Sigma^3 = \{000, 001, 010, 011, \ldots\}$

If $k = 0$,

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

- Set of non-empty strings from alphabet Σ is denoted by Σ^+. (1 or more occurance)

Σ^+ - positive closure.

Σ^+ - $\Sigma^1 \cup \Sigma^2 \ldots$

- Set of strings including empty string is denoted by Σ^*. (0 or more occurance)

$\Sigma^* \rightarrow$ kleen closure.

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

$\Sigma^* = \Sigma^+ \cup \lambda$

~~Concatenation~~

Languages

Language is defined as a set of strings of symbols over an alphabet sigma. i.e, any subset of Σ^* will be called a Language.

Finite Language

Can be specified by listing all its strings.

~~Infinite language can be~~

Infinite language

Can be specified by,

$L = \{ w \in \Sigma^*, w \text{ has property } P \}$

eg: $L = \{ w \mid w \in (0,1)^*$, w has an equal no. of 0's & 1's $\}$

Operations on Languages.

1) Concatenation: If L_1 & L_2 are 2 languages, their concatenation $L = L_1 L_2$.

where $L = \{ w / w = xy$ for some $x \in L_1$ and $y \in L_2 \}$

formal

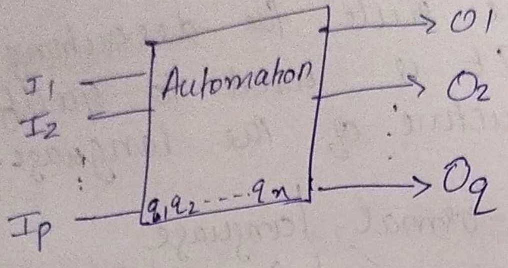
- Set of

- Gram langu synta

Typ

Ty

Model of discrete Automation



Automation is defined as a system whose energy materials & information are tranformed, transmitted and used for performing some functions without direct participation of man.

eg: washing machine.

Characteristics of automation are:- ~~output~~

1) Input : at each of discrete instance of time $T_1, T_2 \ldots T_m$, the input values $I_1, I_2 \ldots I_p$ each of which can take a finite no. of fixed values from the input alphabet sigma are applied to the input side the model.

2) Output : $O_1, O_2 \ldots O_q$ are the outputs of the model each of which can take a finite no. of fixed values from an output o

3) States : At any instant of time, the automaton can be in one of the states $q_1, q_2 .. q_n$

3) State relation :- The next state of an automaton at any instant of time is determined by the present state & present input.

4) Output relation : The output is related either to state only or to both the input and the state

5) An automaton in which the o/p depends only on the i/p is called an automaton without memory.

* An automaton in which the o/p depends on the stage as well i/p is called a automaton with finite memory.

* An automaton in which the o/p depends only on the state of the machine is called a ~~Moore~~ Moore machine

* An automaton in which the o/p depends on the stage as well as on the i/p at any instant of time is called Mealy m/c.

Formal Definition of Finite Automata

A finite automata DFA (Deterministic Finite Automata) can be represented by 5 tuples that are $(Q, \Sigma, \delta, q_0, F)$ ..

where $Q \rightarrow$ finite non-empty set of states.

$\Sigma \rightarrow$ ~~Alphabets~~ finite non-empty sets of i/ps (alphabets)

$q_0 \rightarrow$ Initial state or start state

$q_0 \in Q$

$F \rightarrow$ ~~final~~ set of final states or accepting states.

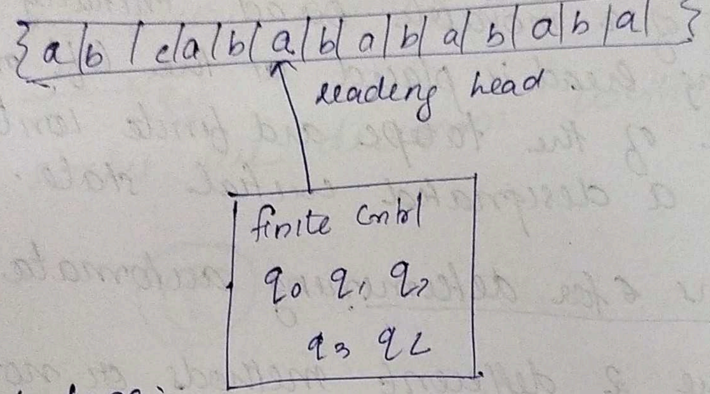
$F \subseteq Q$.

$\delta \rightarrow$ δ is a function which maps $Q \times \Sigma$ onto Q is usually called a direct transition function. This is a function which describes the change of stages during the transition. This mapping is usually represented by a transition table or a transition diagram.

* A transition function

takes 2 arguments a
state and i/p syst symbol
It returns a single state

Operations of finite Automata



1) Input tape:-
~~It consists of input tape~~. The i/p tape is divided into squares. Each squares containing a single symbol from the i/p alphabet sigma.
The left to right sequence of symbols b/w the 2 end markers is ~~op~~ the i/p string to be processed.

2) Reading head:-
The head examines only one square at a time. The movement of this head is only to the right side.

3) Finite Control:

At any specific moment, the finite control will be in any one of the state. Finite control can sense what symbol from sigma is written at any position on the i/p ~~tabe~~ tape by means of a movable head. Initially the reading head is placed at the left most square of the tatape and finite control is in a designated initial state.

Notations for determining automata

There are 2 different methods or notations for describing automata.

1) Transition diagram.

2) Transition table

1) Transition diagram (state digram/ transition graph)

Transition diagram is a directed graph.

vertices of this directed graph represents state

Edges represent transition

Labels on the vertices are the names of states, while labels on the edges are

current values of the i/p symbol.
In a finite set of states at least one of which is designated as the start state & some of which are designated a final states.
Initial stage is represented by a circle with an arrow pointing towards it.

$\longrightarrow q_0$

final states are ~~drawn~~ with double circle

q

Example :-
Consider the following DFA represented by M,

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

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

where δ is given by,

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

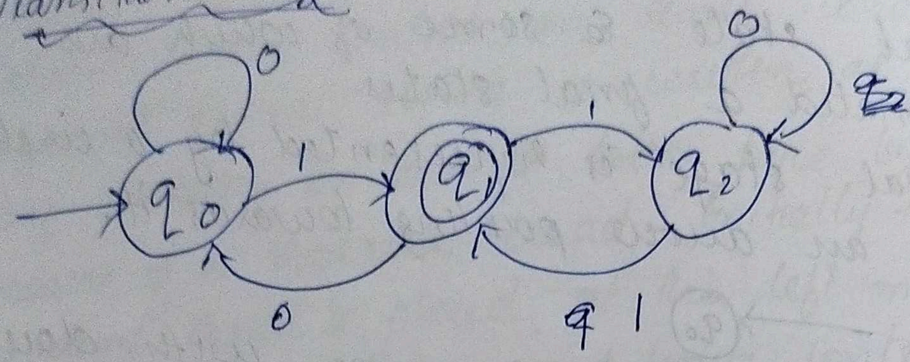
$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_1$

Transition diagram :-



Transition Table

Transition table is a tabular representation of function δ. It takes 2 arguments & returns a state. This table has a row for each state and column for each i/p symbol.

states / Σ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_1

Properties of Transition fn

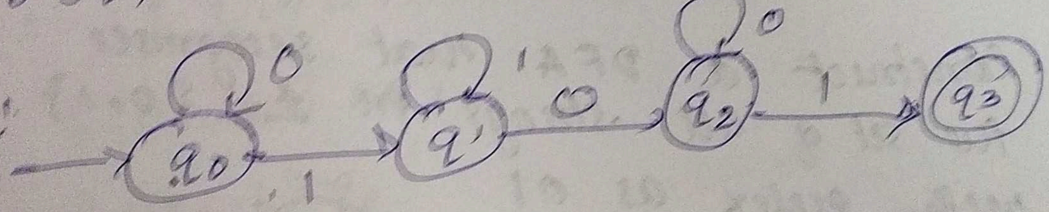
1) $\delta(q, \epsilon) = q$
ϵ = null string

This means that the state of the system can be changed only by an i/p symbol.

2) $\delta(q, aw) = \cancel{\delta(q,a)}\ \delta(\delta(q,a), w)$
a → i/p alphabet, w → string

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

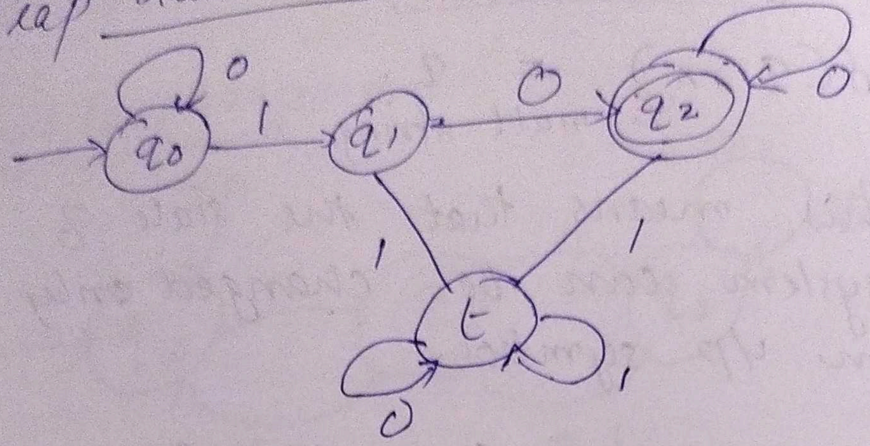
eg:



P.T. $\delta(q_0, 1001) = q_3$

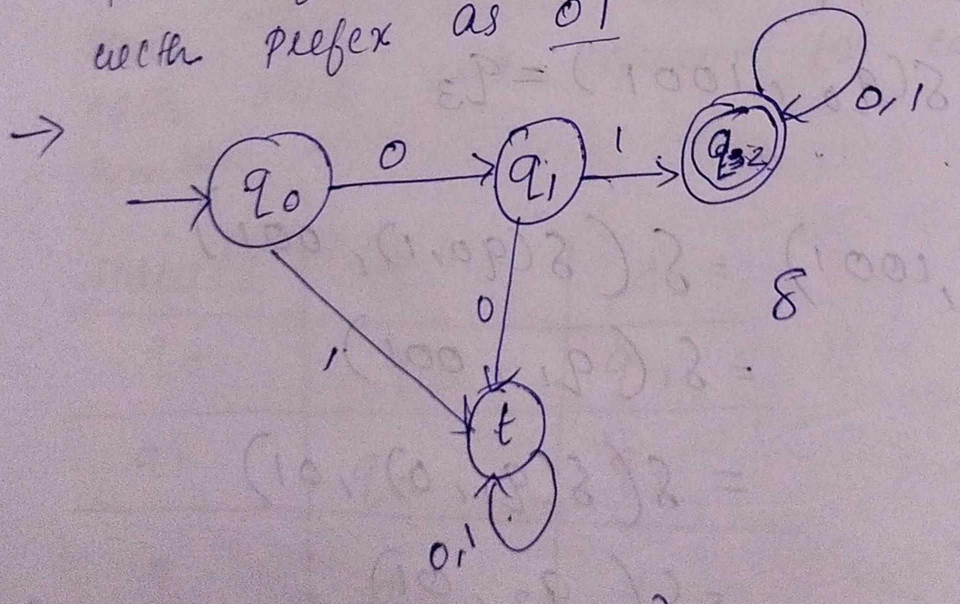
$$
\begin{aligned}
\delta(q_0, 1001) &= \delta(\delta(q_0, 1), 001) \\
&= \delta(q_1, 001) \\
&= \delta(\delta(q_1, 0), 01) \\
&= \delta(q_2, 01) \\
&= \delta(\delta(q_2, 0), 1) = \delta(q_2, 1) \\
&= q_3
\end{aligned}
$$

Trap state / Dead state



Trap state / Dead states are the non-final states from the trap state. we cannot make a move to the final state.

Q. Construct a DFA that recognises the set of all strings on $\Sigma = \{0,1\}$ with prefix as 01



δ

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

~~dB~~ $Q = \{q_0, q_1, q_2, t\}$

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

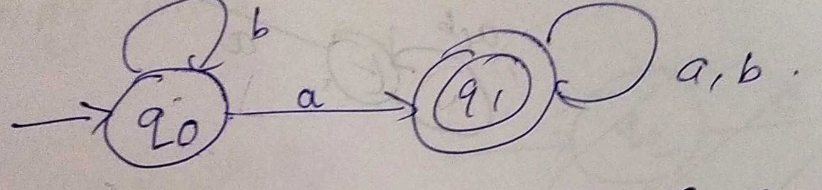
$q_0 = \{q_0\}$

$F = \{q_2\}$

Q. Construct a DFA that accepts all strings with atleast one a.

~~I/p a~~ $\Sigma = \{a, b\}$

→



$M = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$

$\delta(q_0, a) = q_1$

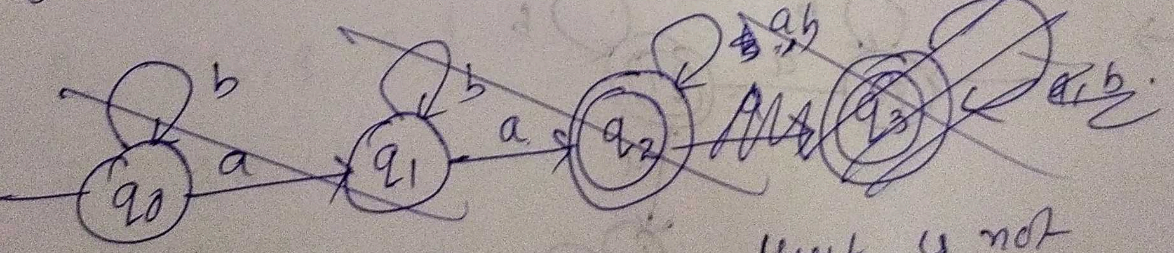
$\delta(q_0, b) = q_0$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_1$

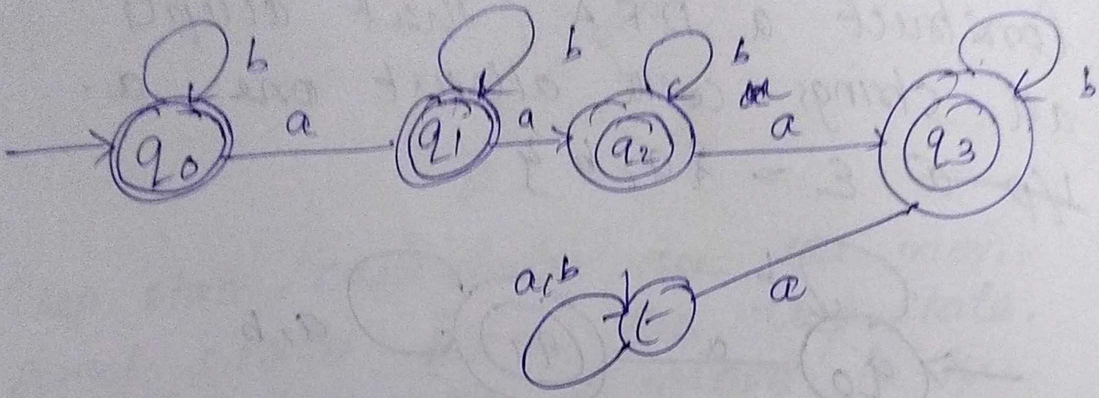
Q. Construct a DFA that accepts all strings ~~that accepts~~ not more that 3 a's. $\therefore \Sigma = \{a, b\}$

→



(Not more than 3 a's means that is not mandatory that it should include a i.e. it can be.

a
a a
a a a
ba, bba, abb
aabba . . . .



M = ({q0, q1, q2, q3, t}, {a, b}, δ ~~{q0, q1,~~
~~q2, q3}~~ q0, {q0, q1, q2, q3})

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

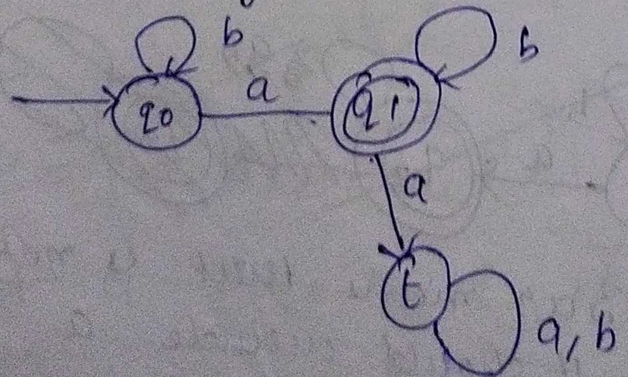
$F = \{q_0, q_1, q_2, q_3\}$

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

$q_0 = \{q_0\}$

Q Construct a DFA that accepts all strings with exactly one a.

→



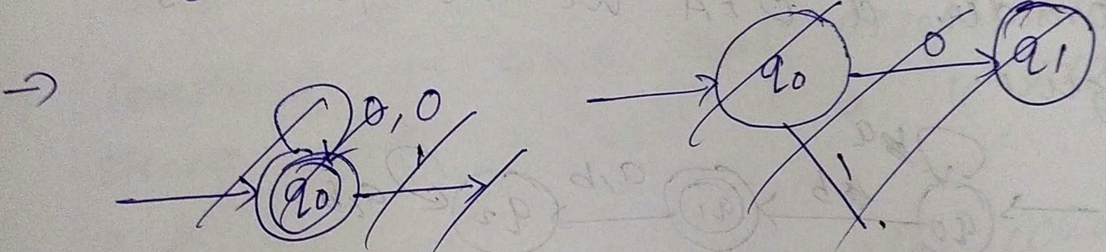
$M = \{ \{q_0, q_1, t\}, \{a, b\}, \delta, q_0, \{q_1\} \}$

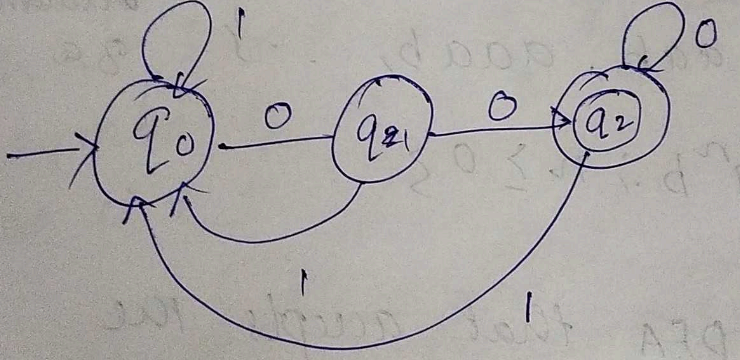
$\Sigma = \{a, b\}$ $\quad q_0 = \{q_0\}$

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

$F = \{q_1\}$

Q. Construct a DFA that accepts set of strings such that every string ends in 00. Over $\Sigma = \{0,1\}$





100
10100
0100
00100
100100

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

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

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

$q_0 = \{q_0\}$

$F = \{q_2\}$

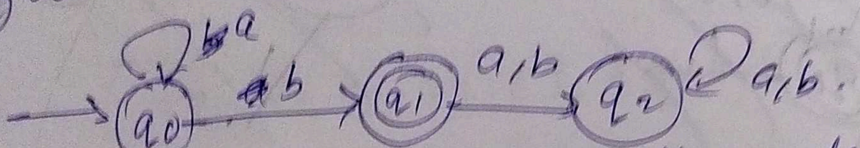
Language accepted by finite automata

A language accepted by a finite DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ is set of all strings on Σ accepted by M.

$L(M) = \{ x \mid \delta(q_0, x) \text{ is in } F \}$

$x \rightarrow$ i/p string.

Q. Consider a DFA as the following figure.



Find L(M), accepted by the above automata.

$\rightarrow$ $\{ b, ab, aab, aaab, \ldots \}$ 0 or more occurrence of a

$L(M) = \{ a^n b ; n \geq 0 \}$

Q.1 Construct a DFA that accepts the set of strings where no. of zeros in every string is a multiple of 3. over Σ {0, 1}

Q.2 Construct a DFA for all strings over set {a, b} that contain exactly ~~2 a~~ 2 a's.

Q.3 DFA for all strings over {a,b} that have length 3.

Q.4 All strings over {0,1} that contain substring. 1001

Q.5. All strings over {0,1} that end with 111

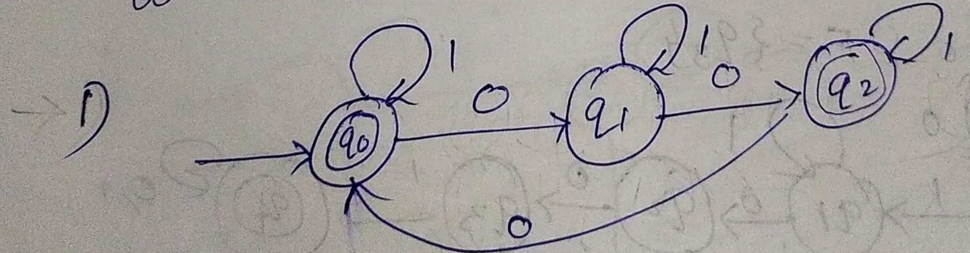
Q.6 DFA for all strings over {0,1} that begin with 111

Q.7 Construct a DFA that accepts all strings with atleast 1 a & exactly 2 b's.

Q.8 Design a finite automata which accepts the language ~~L=~~ $L = \{ w \in \{0,1\}^* \; / \;$ ~~where~~ second and 4th symbol of w is '0' ~~and 4th symbol of w is 1~~ $\}$

Q.8 Design a finite automata which accepts the language L = { w ∈ {0,1}* / second symbol of w is '0' and 4th symbol of w is 1 }

1)



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

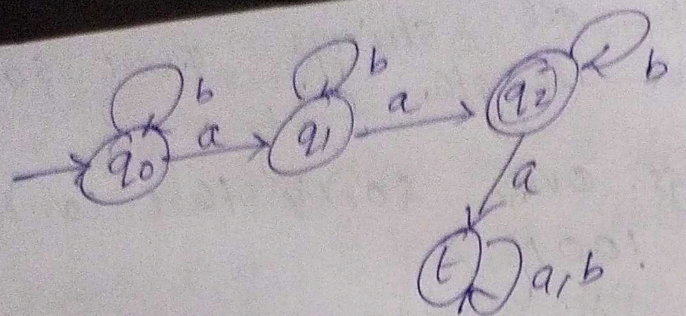
$Q = \{q_0, q_1, q_2\}$

$F = \{q_2\}$.

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

$q_0 - q_0$

2)

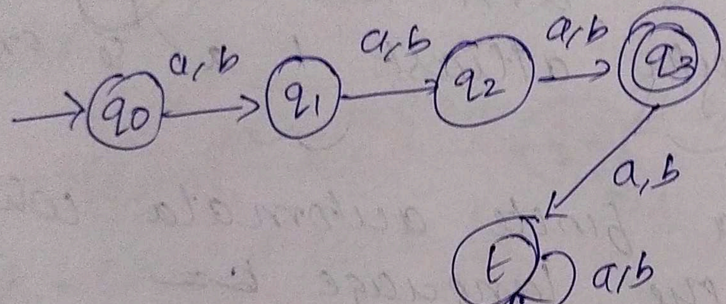


$M = \{\{q_0, q_1, q_2, t\}, \{a, b\}, \delta, q_0, \{q_2\}\}$

$Q = \{q_0, q_1, q_2, t\}$ $\Sigma = \{a, b\}$ $q_0 = q_0$

$F = \{q_2\}$.

3)

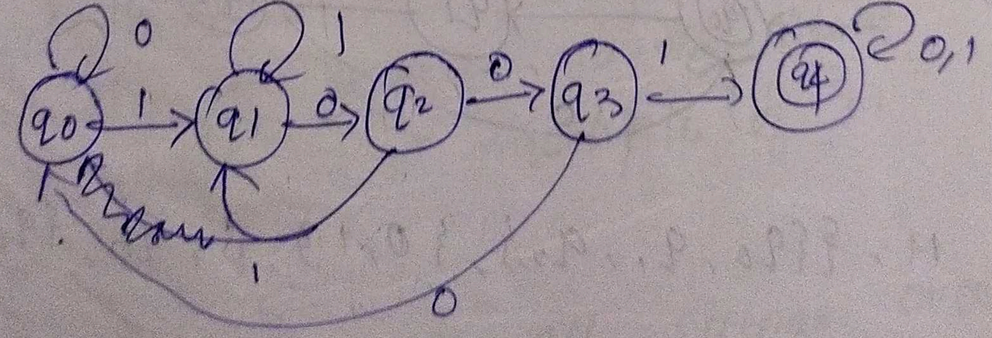


$M = \{\{q_0, q_1, q_2, q_3, t\}, \{a, b\}, \delta, q_0, \{q_3\}\}$

$Q = \{q_0, q_1, q_2, q_3, t\}$ $\Sigma = \{a, b\}$

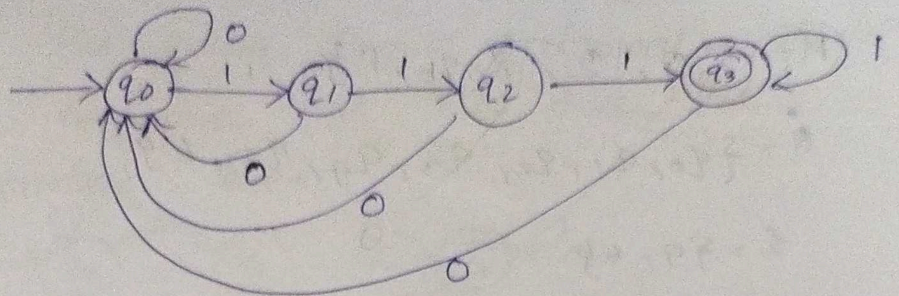
$q_0 = q_0$, $F = \{q_3\}$

4)



*

5)

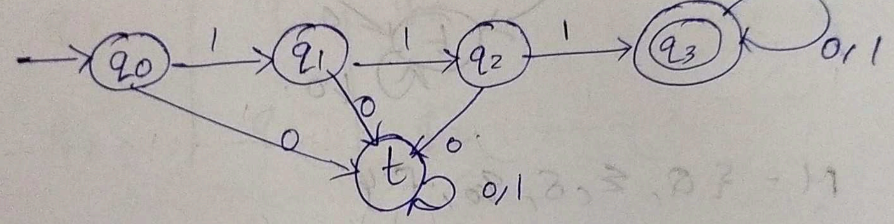


$M = \{Q, \Sigma, \delta, q_0, F\}$

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

$q_0 = q_0$ $F = \{q_3\}$

6)

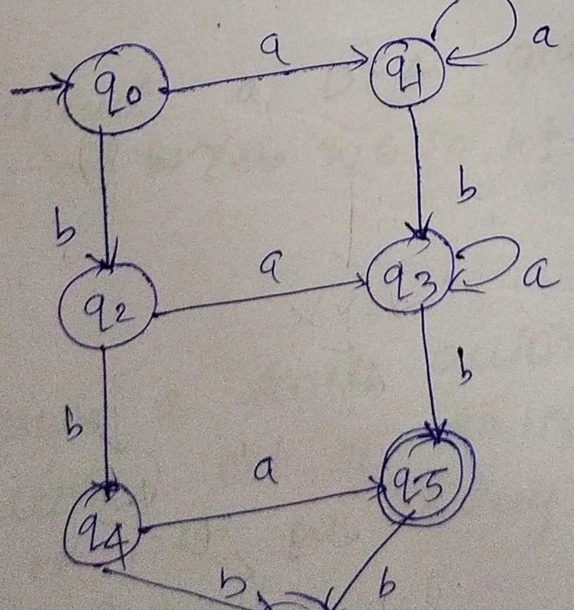


$M = \{Q, \Sigma, \delta, q_0, F\}$

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

$q_0 = q_0$ ~~q~~ $F = \{q_3\}$

7)



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

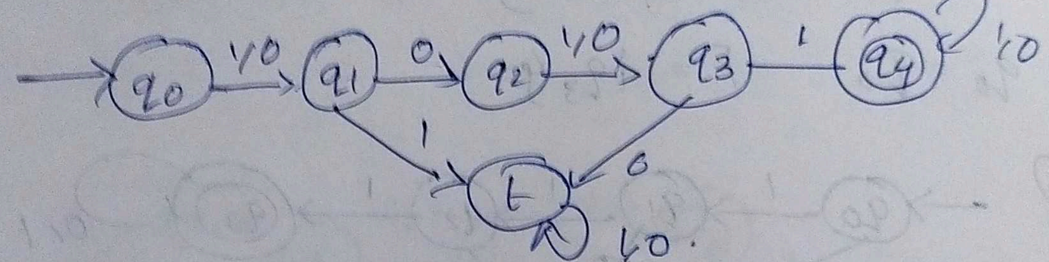
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, t\}$

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

$q_0 = q_0$

$F = \{q_5\}$.

8)



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

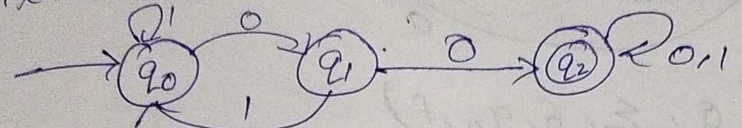
$Q = \{q_0, q_1, q_2, q_3, q_4, t\}$

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

$q_0 = q_0$

$F = \{q_4\}$.

Q. For the given finite automata, write the language and also give the transition table.



→

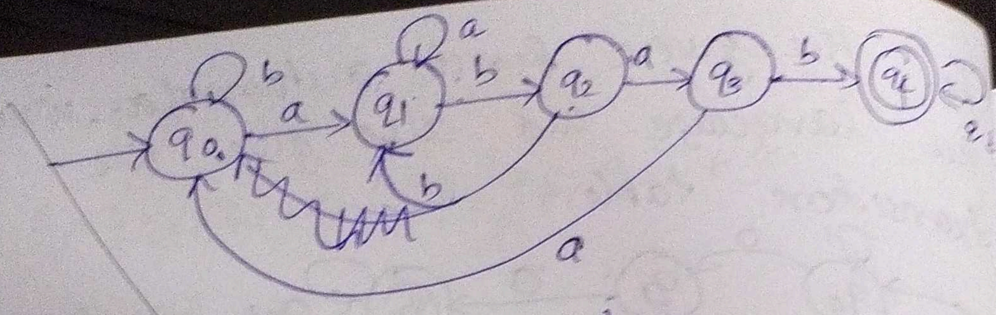
State/Σ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2

A- $L = \{ w \in \{0,1\}^* /$ every string w of the language containing 00 as substring. $\}$

Q.1) Construct a DFA accepting language $L = \{ \cancel{w \times a}\ w \in \{a,b\}^* / w$ has abab as substring $\}$.

Q.2) Design a finite automata which accepts set of strings containing 4 to 1's in every string over $\Sigma = \{0,1\}$

HW 1)



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

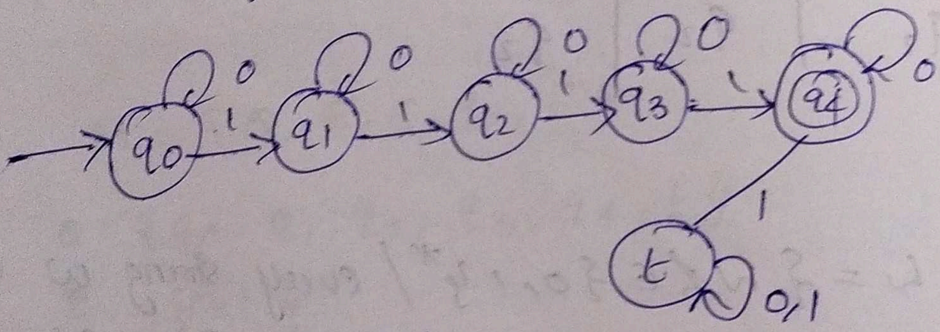
$\Sigma = \{a, b\}$

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

$q_0 = q_0$

$F = \{q_4\}$

2)



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

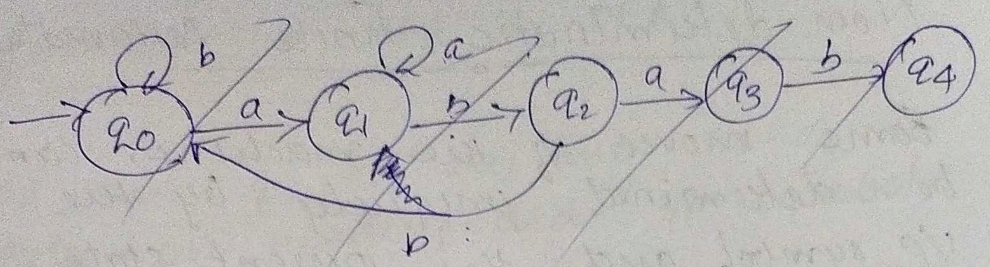
$\Sigma = \{a, b\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, t\}$

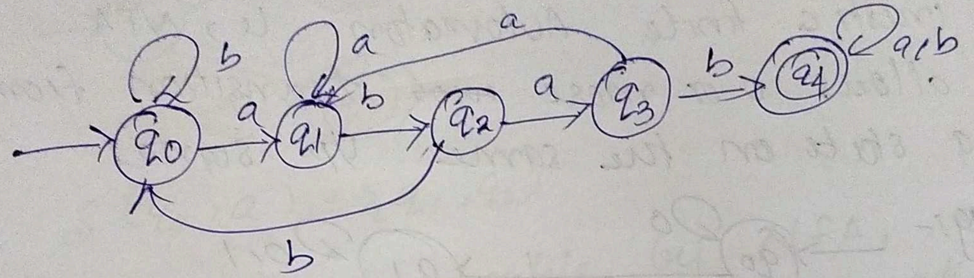
$q_0 = q_0$

$F = \{q_4\}$

1)



1)



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

$\Sigma = \{a, b\}$ $\quad Q = \{q_0, q_1, q_2, q_3, q_4\}$

$q_0 = q_0$ $\quad F = \{q_4\}$

~~Non Deterministic Finite Automata~~

Properties of DFA

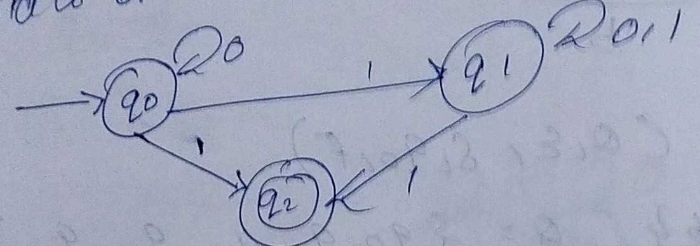
1) for each i/p symbol there is a unique transition i.e. a single symbol on a state cannot move to more than one state.

2) Each and every state should have transitions for ~~transitions~~ for all i/p ~~signals~~ symbols in Σ.

Non-deterministic Finite Automata:

Some moves of the machines cannot be determined uniquely by the i/p symbol and the present state. Such machines are called Non-deterministic finite Automaton. i.e., NFA allows 0 or more ~~not~~ transition from a state on the same i/p signal.

eg:-



Formal defenition of NFA

NFA $(Q, \Sigma, \delta, q_0, F)$ can be defined as 5 ~~tuple~~ tuple i.e.,

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

Q = finite non-empty set of states

Σ = finite " set of i/p alphabets

δ = Transition function which maps to a set of states

$Q \times \Sigma \rightarrow 2^Q$.

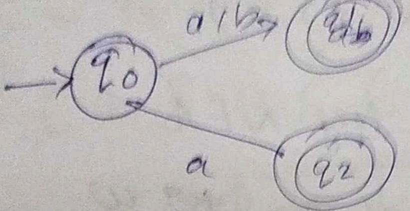
q_0 = Initial state $q_0 \in Q$

F = set of finite states.

Difference b/w NFA & DFA

1) NFA can go to ~~any~~ any no. of states on reading and i/p. symbol

eg:-



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

2) ε transition is possible by NFA, this means that NFA can make a transition without consuming an i/p signal.

3) In NFA, the set of ~~s~~ $\delta(q, a)$ may be empty.

i.e, $\delta(q, a) = \phi$.

This means that there is not transition defined for this specific situation. but in DFA there is a next state for each state it with all symbols of Σ.

Extended transition function / conduct transition fn.

→ ~~δ^* or $\hat{\delta}$~~

→ Represented as δ^* or $\hat{\delta}$

→ $\hat{\delta}(q, w) = P$, where P = set of states ~~no. string~~.

ω = string , where $\omega \in \Sigma^*$

w is an $\in \Sigma^*$ which is accepted by NFA by $\delta(q,\omega)$

$\delta(q,\omega)$ = any of final state

$\hat{\delta}(q,xa) = \delta(\hat{\delta}(q,x),a)$

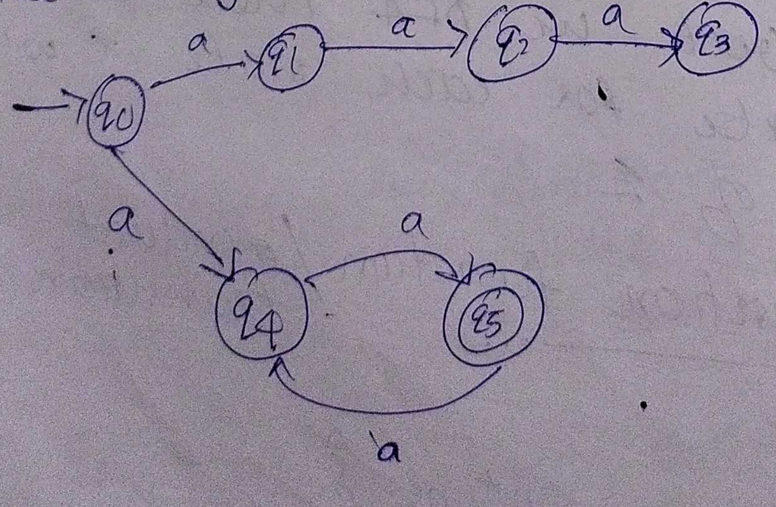
Language accepted by NFA

An NFA accepts a string ~~to~~ ω if it is possible to make any sequence of choice of next state while reading the character of ω and move from initial state to any accepting state.

Language accepted by

$$L(M) = \{ \omega \mid \hat{\delta}(q_0,\omega) \cap F \neq \phi \}$$

eg:- Check whether a^5 is accepted by the NFA given below,



$\delta(q_0, a^5)$

$= \hat{\delta}(q_1, a^4) \cup \hat{\delta}(q_4, a^4)$

$= \hat{\delta}(q_2, a^3) \cup \hat{\delta}(q_5, a^3)$

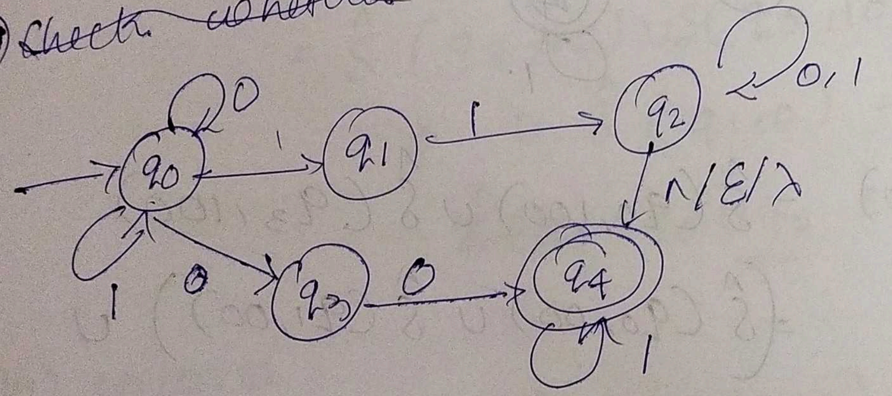
$= \hat{\delta}(q_3, a^2) \cup \hat{\delta}(q_4, a^2)$

$= \{\phi\} \cup \hat{\delta}(q_5, a)$

$= \phi \cup q_4 = \underline{\underline{q_4 \notin f}}$

String not accepted.

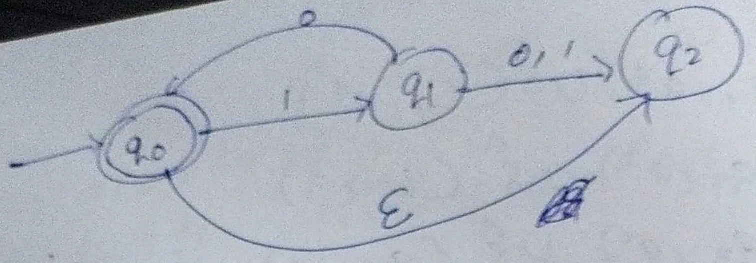
Q.1) ~~Check whether~~



find the set of ~~all~~ states which accept the string 0100 and also draw the transition table.

Q.2) Check whether the following strings are accepted by NFA

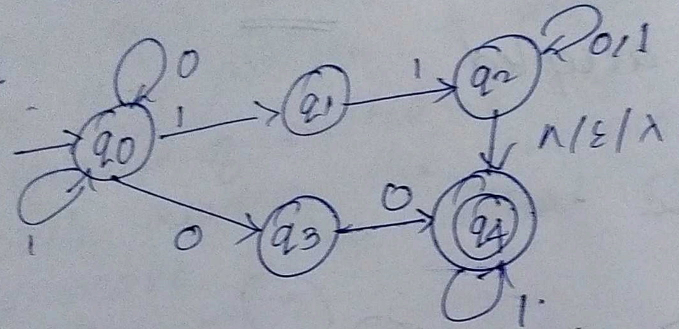
∧, 1010, 101010, 110, 10100



~~$\delta(q_0, 0100)$~~ ~~$\hat{\delta}(q_3, 100)$~~

$= \hat{\delta}(q_0, 100)$

HW

1. 

$\delta(q_0, 0100) = \hat{\delta}(q_0, 100) \cup \hat{\delta}(q_3, 100)$

$= \Big(\hat{\delta}(q_0, 00) \cup \hat{\delta}(q_1, 00) \Big) \cup \big(\phi \big)$

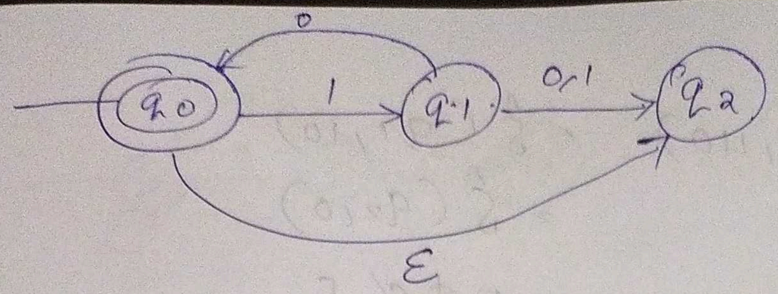
$= \Big(\hat{\delta}(q_0, 0) \Big) \cup \Big(\hat{\delta} \Big) \hat{\delta}(q_3, 0)$

$= q_0 \cup q_3 \cup q_4 \in F$

String accepted

	0	1	ε
$\to q_0$	q_3, q_0	q_0, q_1	ϕ
q_1	ϕ	q_2	ϕ
q_2	q_2	q_2	q_4
q_3	q_4	ϕ	ϕ
q_4	ϕ	q_4	ϕ

2.



(i) Λ

$\hat{\delta}(q_0, \Lambda) = q_2 \notin F$

String not accepted

(ii) 1010

$\hat{\delta}(q_0, 1010) = \hat{\delta}(q_1, 010)$

$= \hat{\delta}(q_2, 10) \cup \hat{\delta}(q_0, 10)$

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

$= \cancel{\hat{\delta}(q_0} \; q_0 \cup q_2 \in F$ string accepted

(iii) 101010

$\hat{\delta}(q_0, 101010) = \hat{\delta}(q_1, 01010) = \hat{\delta}(q_2, \cancel{0}1010)$

$= \hat{\delta}(q_0, 1010) \cup \hat{\delta}(q_2, 1010)$

$= \hat{\delta}(q_1, 010) \cup \phi$

$= \hat{\delta}(q_0, 10) \cup \hat{\delta}(q_2, 10)$

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

$= q_0 \cup q_2$

string accepted.

iv) 110

$\hat{\delta}(q_0, 110) = \hat{\delta}(q_1, 10)$
$= \hat{\delta}(q_2, 0)$
$= \phi \notin F$

String not accepted

v) 10100

$\delta(q_0, 10100)$
$= \hat{\delta}(q_1, 0100) = \hat{\delta}(q_0, 100) \cup \hat{\delta}(q_2, 100)$
$= \hat{\delta}(q_1, 00) \cup \phi$
$= \hat{\delta}(q_1, 100) =$
$= \hat{\delta}(q_0, 0) \cup \hat{\delta}(q_2, 0)$

$= \phi \cup \phi$
$= \phi \notin F$

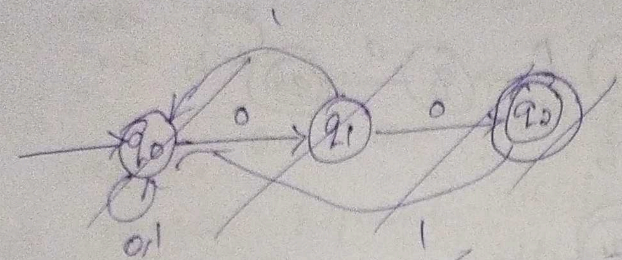
Q.1 D

Q.2

Q.3

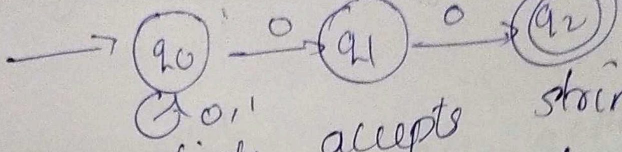
1.

Q) Design an NFA to accept set of strings over alphabet ~~set~~ $\{0, 1\}$ and ending with 2 consecutive 0s



~~M = {Q, Σ, δ, q0, F}~~

~~Q =~~



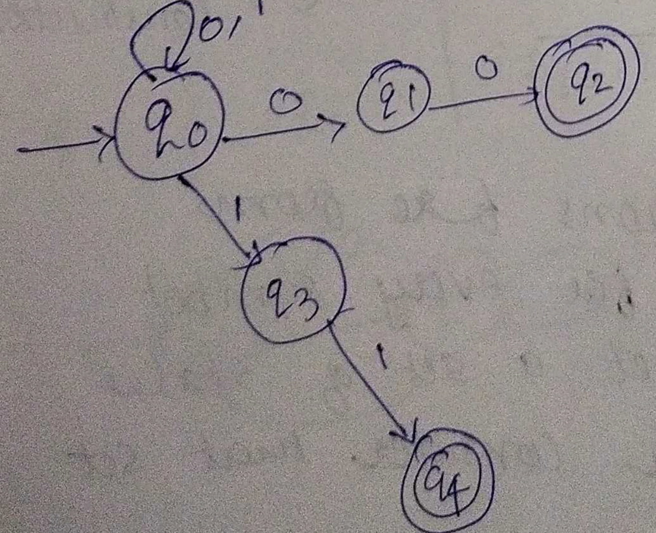
Q.1 Design NFA which accepts strings with 0s and 1's such that string contain either 2 consecutive 0s or 2 consecutive 1's

Q.2 Design NFA with no more than 5 states for the set $L = \{abab^n, n \geq 0\}$ $\cup \{aba^n, n \geq 0\}$.

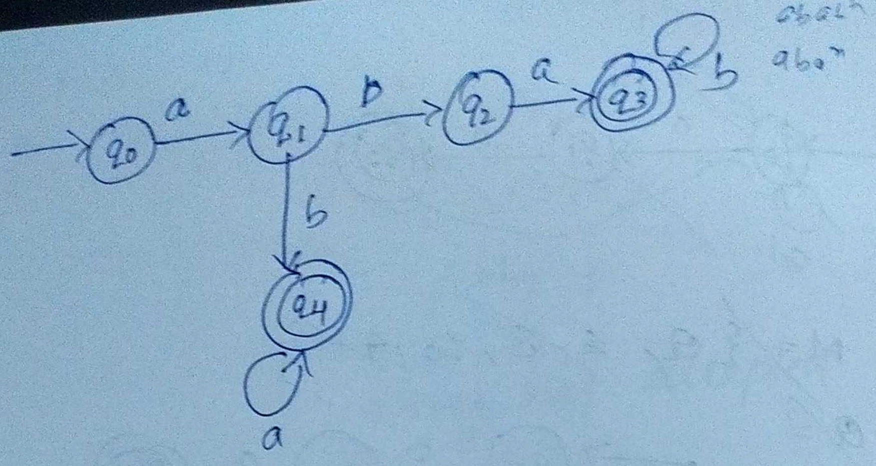
Q.3. Construct NFA with 3 states that accepts $L = \{ab, aba\}^*$ or $L = \{ab \cup aba\}^*$ (ie any combination of ab, aba including ^)

ababa
ababaab
abab ab...

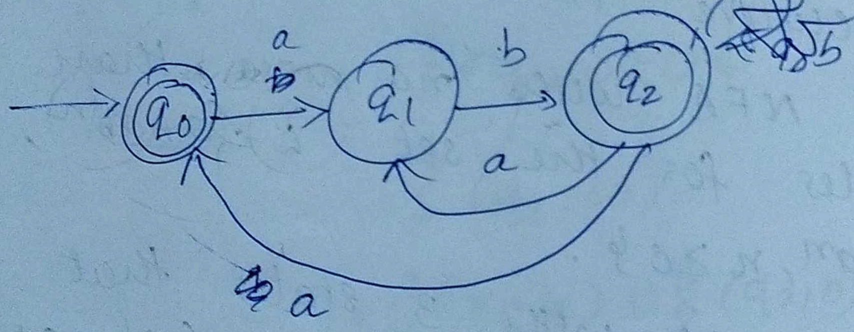
1.



2.



3.



Conversion of NFA to DFA (By subset construction)

Step 1 :-

Select all transitions ~~are~~ from starting state q_0 for every symbol in Σ. If you get a set of states for some i/p then consider that set a new single state

step 2 :-

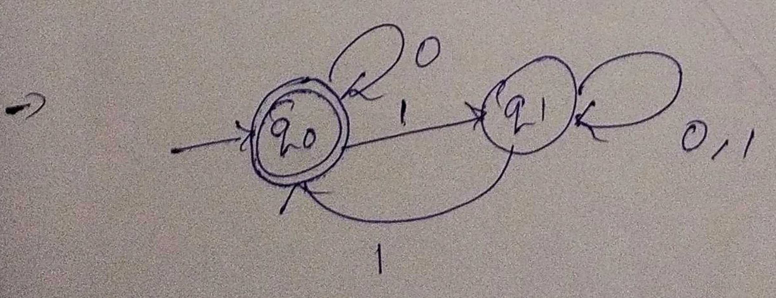
In step 1, we are getting a new state. now repeat step1 for this new state. ie, ~~set all~~ a check all transitions of ε from the new state.

step-3 :-

Repeat step 2 till you are getting a new state. All these states which consist of atleast one accepting state of given NFA as member state will be considered as final state.

Q. Construct a DFA equalent to $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$ is given by state table,

Q/Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	$\{q_0, q_1\}$

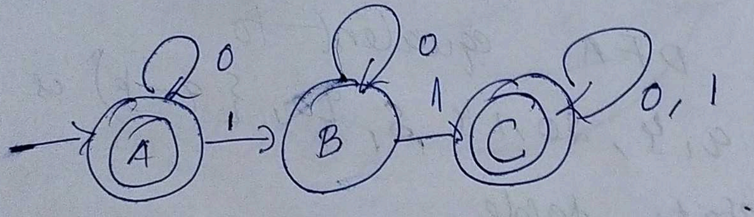


The convert NFA → DFA by subset construction.

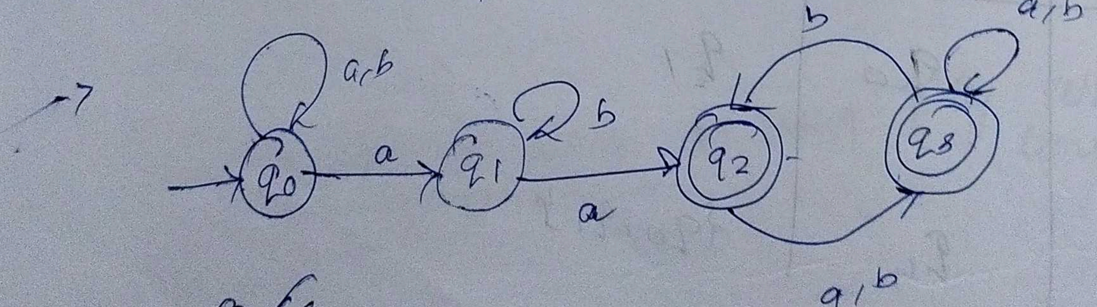
Q/Σ	0	1
A $\to q_0$	q_0	q_1 B
B q_1	q_1	$\{q_0, q_1\}$ C
C $\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_0\}$

considered as a state

stop when no new state is obtained



Q Convert the following NFA to DFA :



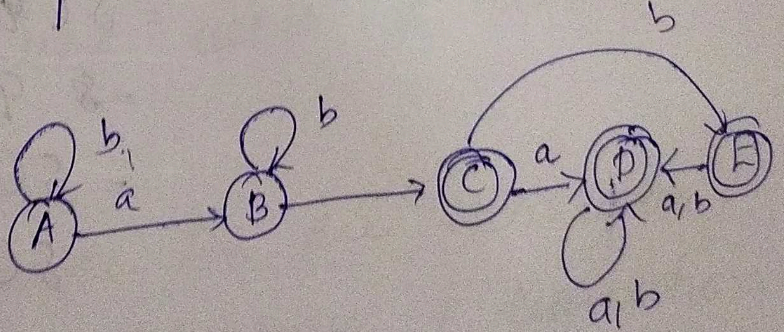
Q/Σ

NFA

Q/Σ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3	q_3	$\{q_3, q_2\}$

DFA :-

	Q/Σ	a	b
A	$\rightarrow q_0$	$\{q_0, q_1\}$ B	q_0
B	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$ C	$\{q_0, q_1\}$ B
C	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$ D	$\{q_0, q_1, q_3\}$ E
D	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_3, q_2\}$
E	$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_2\}$.



NFA with ε transition

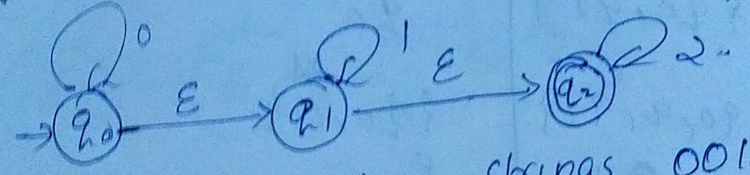
If the finite automata is modified to permit a transition without i/p symbol, then we get an NFA with ε transition.

ε means null.

A string x in Σ will be accepted by NFA with ε moves, if there exist atleast one path corresponding to x which starts in an initial state and ends in one of the final state

eg: consider the following NFA

~~q_0, q_1, q_2~~.



Check whether the strings 0012 is accepted by the NFA.

$$\delta(q_0, 0012) = \delta(q_0, 012)$$
$$= \delta(q_0, 12)$$
$$= \delta(q_0, \varepsilon)$$
$$= q_1$$
$$= \delta(q_1, 12)$$
$$= \delta(q_1, 2) = \delta(q_1, \varepsilon)$$
$$= q_2 = \delta(q_2, 2)$$
$$= q_2 \in F$$

so the string 0012 is accepted.

Epsilon Closure (ε-closure)

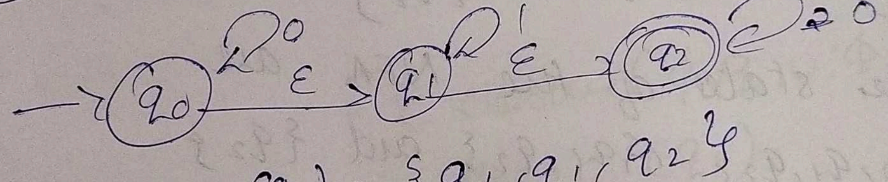
Epsilon closure of a state q ie ε-closure(q) is a set that contains the state q itself and all other states that can be reached from state q by following ε transition.

ε-closure will never be an empty set. ~~Consider the NFA~~

eg: Consider the NFA,

~~ε-closure(q_0) =~~

ε-closure(q) = { }



ε-closure(q_0) = {q_0, q_1, q_2}

ε-closure(q_1) = {q_1, q_2}

ε-closure(q_2) = {q_2}

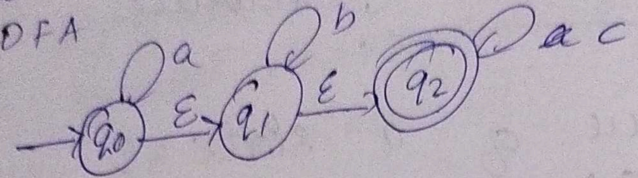
Conversion of NFA with ε transition into DFA

There are 2 steps for the conversion.

1) Transform the NFA with ε transition to NFA without ε transition.

2.) Convert the resulting NFA to DFA.

Q. Convert the following NFA into DFA



The above NFA has states q_0, q_1 & q_2. The start state is q_0 and final state is q_2.

Step 1: Find the ε closure of all the states.

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

The states of the NFA are $\{q_0, q_1, q_2\}$, $\{q_1, q_2\}$ and $\{q_2\}$

The start state of the New NFA will be the ε closure of (start state q_0).

The final state of the new NFA will be those states that contain the final state of old NFA

The final states $= \{q_0, q_1, q_2\}$

1) (b) ~~Step 2~~:- find transitions of these new states on i/p symbols a, b, c.

consider the 1st state $\{q_0, q_1, q_2\}$

$$\delta'(\{q_0, q_1, q_2\}, a) = \varepsilon.\text{closure}(\delta(\{q_0, q_1, q_2\}, a))$$

$$= \varepsilon.\text{closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$$

$$= \varepsilon.\text{closure}(q_0 \cup \phi \cup \phi)$$

$$= \varepsilon.\text{closure}(q_0)$$

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

$$\delta'(\{q_0, q_1, q_2\}, b) = \varepsilon.\text{closure}(\delta(\{q_0, q_1, q_2\}, b))$$

$$= \varepsilon.\text{closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$$

$$= \varepsilon.\text{closure}(\phi \cup q_1 \cup \phi)$$

$$= \varepsilon.\text{closure}(q_1)$$

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

$$\delta'(\{q_0, q_1, q_2\}, c) = \varepsilon.\text{closure}(\delta(\{q_0, q_1, q_2\}, c))$$

$$= \varepsilon.\text{closure}(\delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c))$$

$$= \varepsilon.\text{closure}(\phi \cup \phi \cup q_2)$$

$= \varepsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\delta(\{q_1, q_2\}, a) = \varepsilon\text{-closure}(\delta(\{q_1, q_2\}, a))$

$= \varepsilon\text{-closure}(\delta(q_1, c) \cup \delta(q_2, a))$

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

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

$= \phi$

$\delta(\{q_1, q_2\}, b) = \varepsilon\text{-closure}(\delta(\{q_1, q_2\}, b))$

$= \varepsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b))$

$= \varepsilon\text{-closure}(q_1 \cup \phi)$

$= \varepsilon\text{-closure}(q_1) = \{q_1, q_2\}$

$\delta(\{q_1, q_2\}, c) = \varepsilon\text{-closure}(\delta(\{q_1, q_2\}, c))$

$= \varepsilon\text{-closure}(\delta(q_1, c) \cup \delta(q_2, c))$

$= \varepsilon\text{-closure}(\phi \cup q_2)$

$= \{q_1, q_2\}$

$\delta(\{q_2\}, a) = \varepsilon\text{-closure}(\delta(q_2, a))$

$= \phi$

$\delta(\{q_2\}, b) = \varepsilon\text{-closure}(\delta(q_2, b))$

$= \phi$

$\delta(\{q_2$

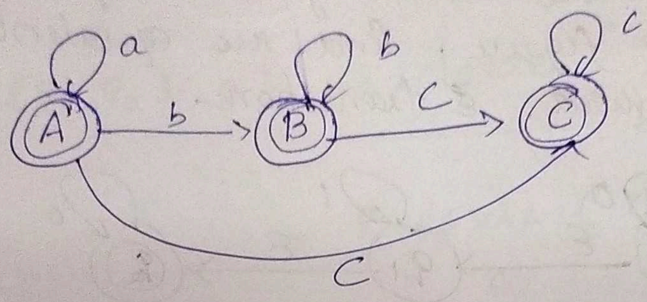
Step-2

Q

NF

$\delta(\{q_2\}, c) = \varepsilon\text{-closure}(\delta(q_2, c))$
$= \{q_2\}$

Step-2

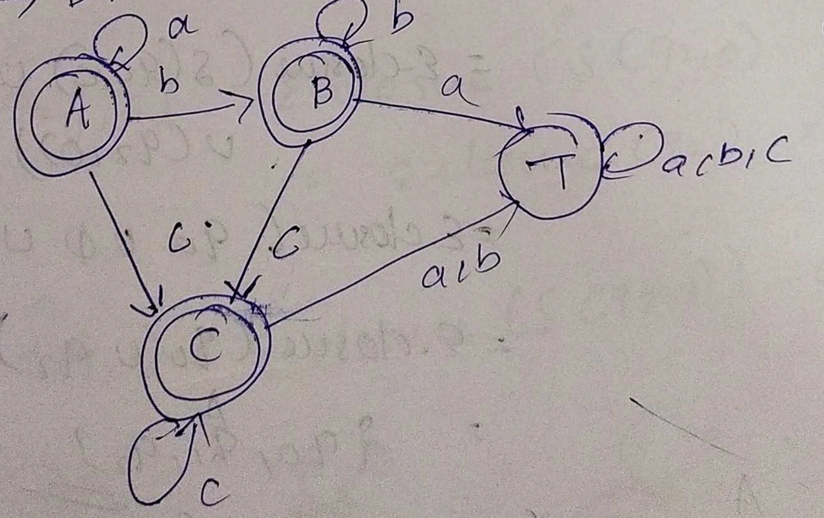


$A = \{q_0, q_1, q_2\}$
$B = \{q_1, q_2\}$
$C = \{q_2\}$

Q/Σ	a	b	c
A	A	B	C
B	ϕ	B	C
C	ϕ	ϕ	C

NFA ⟶ DFA

(DFA should have transition for each symbol)

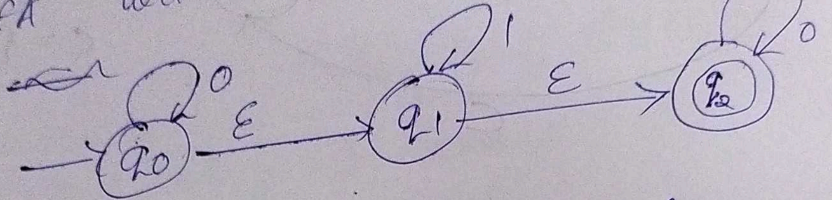


Step 2- explanation

NFA obtained is also a ~~DFA~~ DFA
ie, ~~only one~~ there only one transition

for an i/p. symbol from a state.
So, the DFA obtained is above diagram.

Q. Consider the NFA given by the following figure. Find the equalent NFA without ε transition.



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

ε.closure $(q_1) = \{q_1, q_2\}$

ε.closure $(q_2) = \{q_2\}$

The new states are : $\{q_0, q_1, q_2\}$, $\{q_1, q_2\}$ and $\{q_2\}$.

$$\delta'(\overset{A}{\{q_0, q_1, q_2\}}, 0) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0))$$

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

$$= \varepsilon\text{.closure}(q_0 \cup \phi \cup q_2)$$

$$= \varepsilon\text{.closure}(q_0 \cup q_2)$$

$$= \overset{A}{\{q_0, q_1, q_2\}}$$

$$\delta'(\overset{A}{\{q_0, q_1, q_2\}}, 1) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 1))$$

$$= \varepsilon\text{.closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$$

F ε.closure $(\phi \cup q_1 \cup \phi)$

$= \overset{B}{\{q_1, q_2\}}$

$\delta'(\overset{B}{\{q_1, q_2\}}, 0) = \varepsilon\text{.closure}\ (\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \varepsilon\text{.closure}\ (\phi \cup q_2)$

$= \overset{C}{\{q_2\}}$

$\delta'(\overset{B}{\{q_1, q_2\}}, 1) = \varepsilon\text{-closure}\ (\delta(q_1, 1) \cup \delta(q_2, 1))$

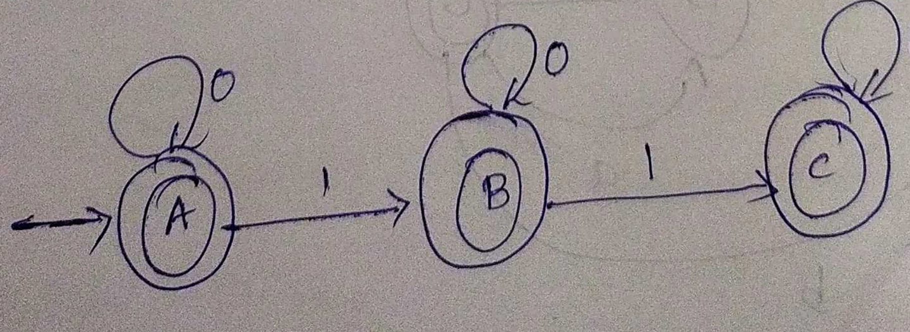
$= \varepsilon\text{-closure}\ (q_1 \cup \phi)$

$= \overset{B}{\{q_1, q_2\}}$

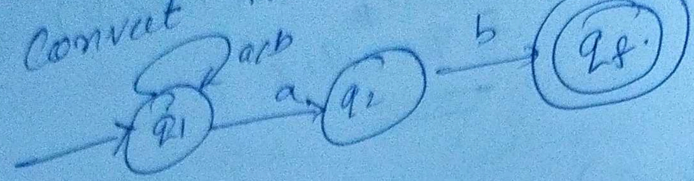
$\delta'(\overset{C}{\{q_2\}}, 0) = \varepsilon\text{-closure}\ (\delta(q_2, 0))$

$= \varepsilon\text{-closure}\ (q_2) = \overset{C}{\{q_2\}}$

$\delta'(\overset{C}{\{q_2\}}, 1) = \varepsilon\text{-closure}\ (\delta(q_2, 1)) = \phi$



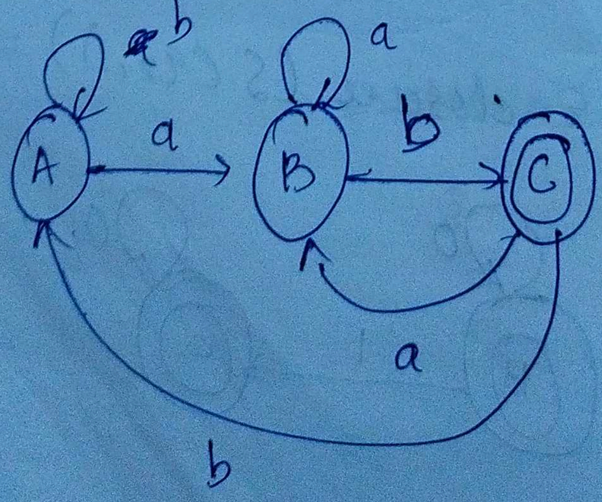
Q. Convert NFA to DFA



→ NFA

Q/Σ	a	b
q_1	$\{q_1, q_2\}$	q_1
q_2	ϕ	q_f
q_f	ϕ	ϕ

Σ/Q	a	b
A q_1	B $\{q_1, q_2\}$	A q_1
B $\{q_1, q_2\}$	B $\{q_1, q_2\}$	C $\{q_1, q_f\}$
C $\{q_1, q_f\}$	B $\{q_1, q_2\}$	A $\{q_1\}$



Regular Grammar

A Grammer G can be formally written as a 4 tuple (V, T, P, S) where V or N is a finite Non-empty set of non-terminal symbols.

T or Σ is a finite non-empty set terminal symbols.

S - speacial non-terminal called the start symbol. ~~S G~~. S∈V

~~se~~

P - Production rule for terminals and non terminals of the form $\alpha \to \beta$ where α & β are terminals or non-terminals.
There should be atleast one non-terminal in α.

eg: Grammer $G_1 \to (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

Here S, a and b are non-terminals.
a and ~~b~~ b are terminal symbol.
S - start symbol and the production
$S \to AB$, $A \to a$ and $B \to b$.

Derivations.

Strings may be derived using productions in a Grammer.

If the ~~a~~ Grammer G has a production $\alpha \rightarrow \beta$, we can say that ~~a $\alpha x y$~~ ~~a~~ $x\alpha y$ derives $x\beta y$ in G.

The derivation is written as

$x\alpha y \Rightarrow x\beta y$.

$\Rightarrow$: derives

eg: Let us the following grammer

$G = (\{S, A\}, \{a, b\}, S, \{S \rightarrow aAb,\ aA \rightarrow aaAb,\ A \rightarrow \varepsilon\})$

Some of the strings that can be ~~dt~~ derived are :-

1) $S \Rightarrow aAb$ -- using production $S \rightarrow aAb$

$\Rightarrow ab$ -- using prodn $A \rightarrow \varepsilon$

2) $S \Rightarrow aAb$ - using pdn $S \rightarrow aAb$

$\Rightarrow aaAbb$ - using pdn $aA \rightarrow aaAb$

$\Rightarrow aaaAbbb$ - "

$\Rightarrow aaabbb$ using pdn $A \rightarrow \varepsilon$

Q. Following is grammer for ~~expression~~

Expr → Expr Op Num | Num   (operator)   (i.e expr is expr operator num)

Op → + | - | * | /

Num → 0/1/2/3/4/5/6/7/8/9
check whether 5+3 is a valid expression.

→ Expr ⟹ Expr op Num
⟹ Num op Num (using expr→Num)
⟹ 5 op Num (Num → 5)
⟹ 5 op 3 (Num → 3)
⟹ 5 + 3 (op → +)

5+3 is a valid expression.

Language generated by Grammer

The set of all strings that can be derived from a grammer is said to be the language generated from that grammer.

A language generated by a grammer G is a subset formally defined by $L(G) = \{w \mid w \in \Sigma^*, S \Rightarrow w\}$

If $L(G_1) = L(G_2)$, the grammer G_1 is equivalent to grammer G_2

eg:- If there is a grammer

G: $N = \{S, A, B\}$, $T = \{a, b\}$

$P = \{S \to AB, A \to a, B \to b\}$

Find $L(G)$?

→ 1) $S \Rightarrow AB$

$\Rightarrow aB \quad (A \rightarrow a)$

$\Rightarrow ab \quad (B \rightarrow b)$

$L = \{ab\}$

Q. Given $G = (\{S, A, B\}, \{a, b\}, S,$
$\{S \rightarrow AB, A \rightarrow Aa/a, B \rightarrow bB/b\})$

→ $S \Rightarrow AB$

$\Rightarrow \cancel{aB}\ \cancel{AB}\ aAB \quad (A \rightarrow aA)$

$\Rightarrow aAb \quad (B \rightarrow b)$

$\Rightarrow aAbB \quad (B \rightarrow bB)$

$\Rightarrow a\cancel{A}\ CA$

1) $S \Rightarrow AB$

$\Rightarrow aAB \quad (A \rightarrow aA)$

$\Rightarrow aAbB \quad (B \rightarrow bB)$

$\Rightarrow aaAbB \quad (A \rightarrow aA)$

$\Rightarrow aaAbbB \quad (B \rightarrow bB)$

2) $S \Rightarrow AB$

$\Rightarrow aB \quad (A \rightarrow a)$

$\Rightarrow ab \quad (B \rightarrow b)$

$a^n b^m$

3) $S \Rightarrow AB$

$\cancel{S} \Rightarrow aAB$

$\Rightarrow aAb$

$\Rightarrow aaAb$

4) $S \Rightarrow AB$

$\Rightarrow bB$

$\Rightarrow abB$

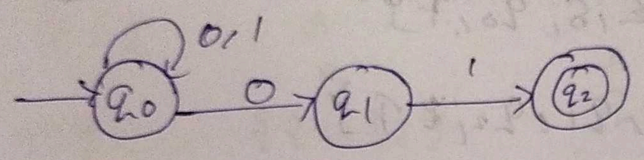
$\Rightarrow abB^{b}B$

$L(G) = \{a^n b^n, n \geq 0\}$

Tutorial 1

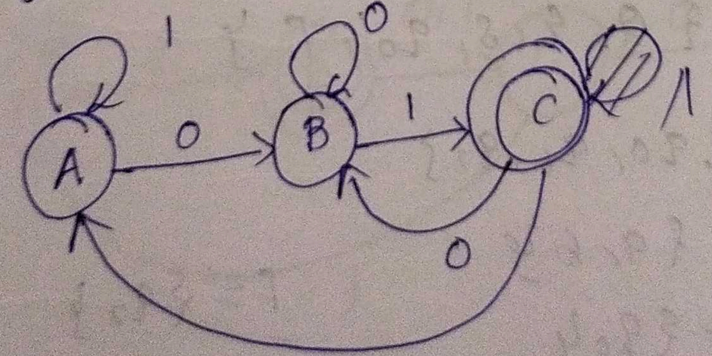
Abhinav M Aji
S5 CSE A
Roll No: 2

1. Design a DFA from the following NFA that accepts all the strings of 0's and 1's that end with 01

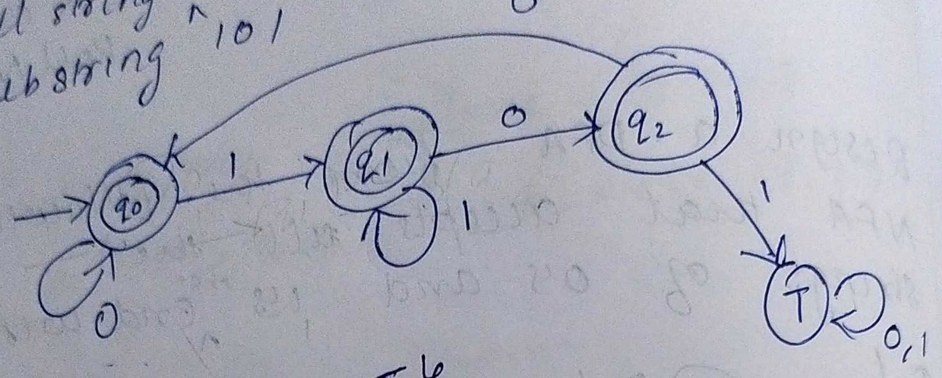


→ Q/Σ	0	1
A: → q_0	B: $\{q_0, q_1\}$	A: q_0
B: $\{q_0, q_1\}$	B: $\{q_0, q_1, \cancel{q_2}\}$	C: $\{q_0, q_2\}$
C: $\{q_0, \cancel{q_1}, q_2\}$	B: $\{q_0, q_1\}$	A: $\{q_0, \cancel{q_2}\}$

States of DFA : q_0 (A), $\{q_0, q_1\}$ (B), $\{q_0, q_2\}$ (C)



2. Design a DFA that accepts all strings except those containing the substring 101



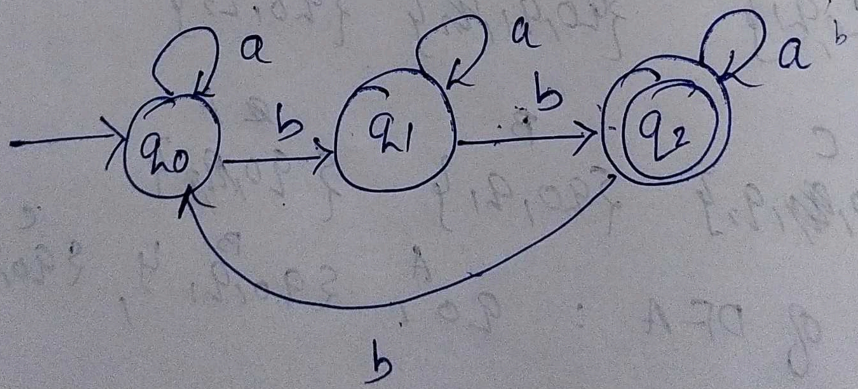
$M = \{Q, \Sigma, \delta, q_0, F\}$

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

$\Sigma = \{1, 0\}$ $\qquad$ $F = \{q_0, q_1, q_2\}$

$q_0 = \{q_0\}$

3 Design a DFA that accepts $L = \{w \in (a,b)^* \,/\, n_b(w) \bmod 3 > 1\}$



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

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

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

$q_0 = \{q_0\}$ $\qquad$ $F = \{q_2\}$

4 Convert

$M = ($

where

$\delta(q$

$\delta($

$\delta($

4 Convert the given NFA to DFA

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

where δ is given by,

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

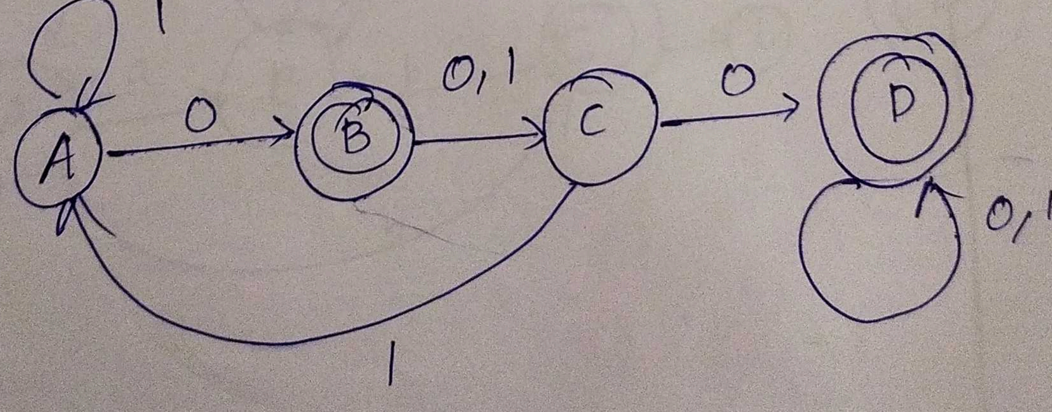
$\delta(q_2, 0) = \{q_1, q_2\}$ $\quad \delta(q_2, 1) = \phi$

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

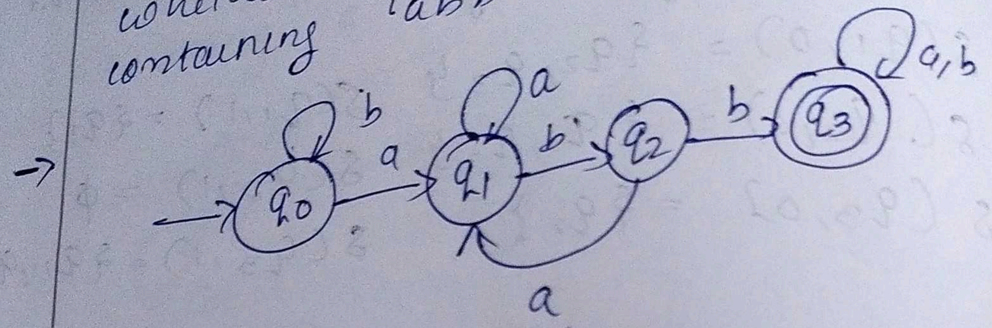
→

δ / Σ	0	1
A $\rightarrow q_1$	B $\{q_2, q_3\}$	A q_1
B $\{q_2 q_3\}$	C $\{q_1, q_2 \cancel{q_3}\}$	C $\{q_1, q_2\}$
C $\{q_1, q_2\}$	D $\{q_1, q_2, q_3\}$	A $\{q_1\}$
D $\{q_1, q_2, q_3\}$	D $\{q_1, q_2, q_3\}$	D $\{q_1, q_2, q_3\}$

New states :- A q_1, B $\{q_2, q_3\}$, C $\{q_1, q_2\}$, D $\{q_1, q_2, q_3\}$



5 Design a DFA to check whether a string over $\Sigma = \{a, b\}$ containing 'abb' is accepted.



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

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

1.) Find

$M = \{$

1.) Find a DFA equivalent to NFA

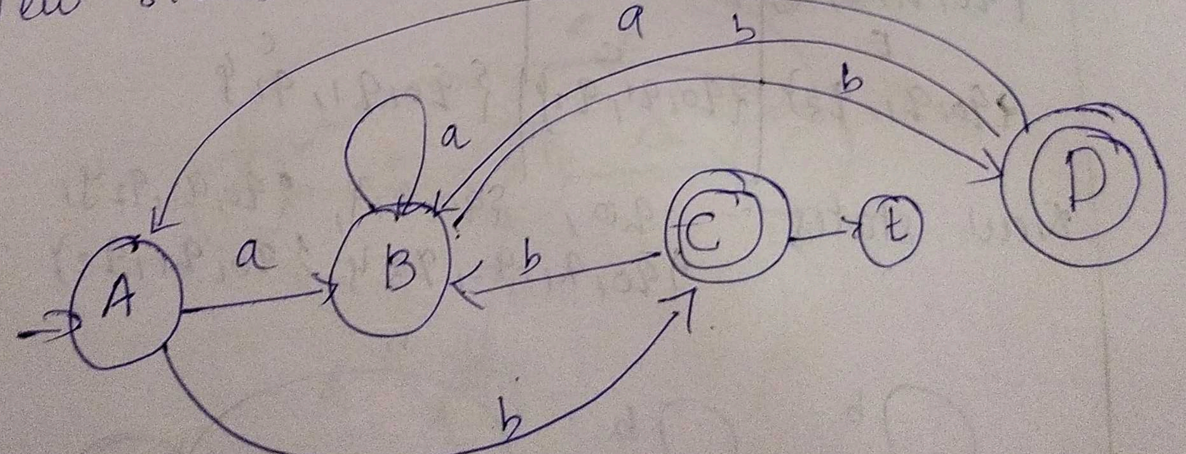
$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$

Q/Σ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_2
q_1	q_0	q_1
q_2	ϕ	$\{q_0, q_1\}$

→

Q/Σ	a	b
A $\rightarrow q_0$	B $\{q_0, q_1\}$	C q_2
B $\{q_0, q_1\}$	B $\{q_0, q_1\}$	D $\{q_2, q_1\}$
C q_2	ϕ	B $\{q_0, q_1\}$
D $\{q_2, q_1\}$	A $\{q_0\}$	B $\{q_0, q_1\}$

New states are :- q_0, $\{q_0, q_1\}$, q_2, $\{q_1, q_2\}$

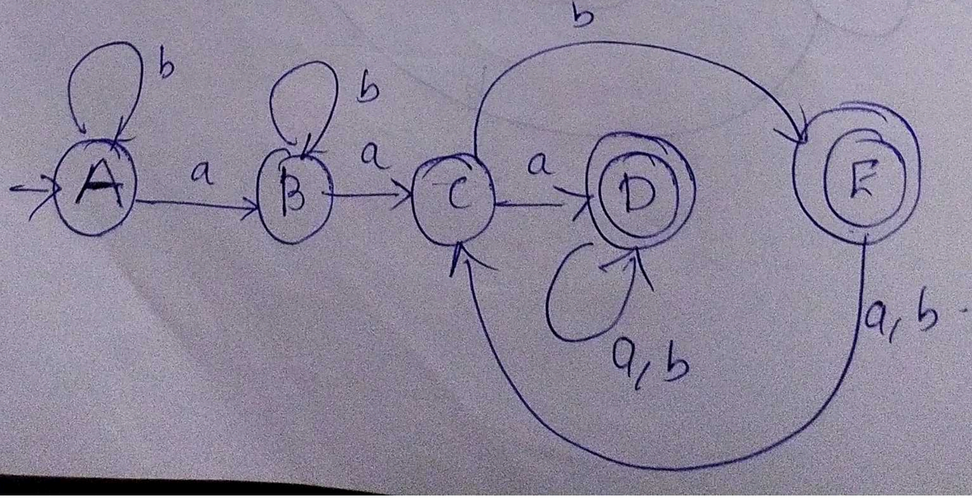


2. Construct a DFA equivalent to NFA
$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\})$

Q/Σ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3	ϕ	q_2

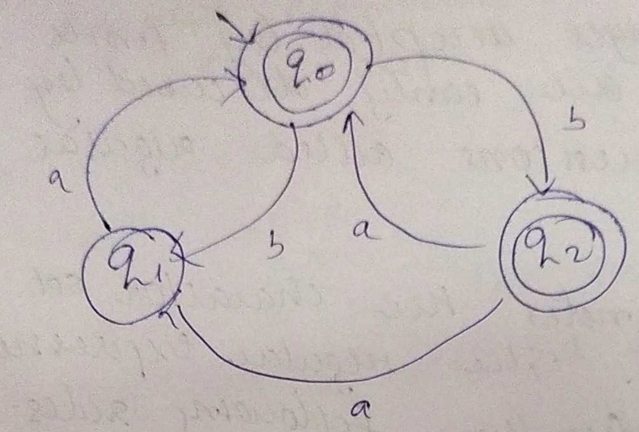
Q/Σ	a	b
$\rightarrow q_0$ A	$\{q_0, q_1\}$ A	q_0 B
$\{q_0, q_1\}$ B	$\{q_0, q_1, q_2\}$ C	$\{q_0, q_1\}$ B
$\{q_0, q_1, q_2\}$ C	$\{q_0, q_1, q_2, q_3\}$ D	$\{q_0, q_1, q_3\}$ E
$\{q_0, q_1, q_2, q_3\}$ D	$\{q_0, q_1, q_2, q_3\}$ D	$\{q_0, q_1, q_2, q_3\}$ D
$\{q_0, q_1, q_3\}$ E	$\{q_0, q_1, q_2\}$ C	$\{q_0, q_1, q_2\}$ C

New states :- q_0, $\{q_0, q_1\}$, $\{q_0, q_1, q_2\}$, $\{q_0, q_1, q_2, q_3\}$, $\{q_0, q_1, q_3\}$



3. Construc

3. Construct DFA to NFA



Q/Σ	a	b
$\to q_0$ A	ϕ A	$\{q_1, q_2\}$ B
B $\{q_1, q_2\}$	$\{q_0, q_1\}$ C	ϕ
C $\{q_0, q_1\}$	q_0 A	$\{q_1, q_2\}$ B

New states :- q_0, $\{q_1, q_2\}$, $\{q_0, q_1\}$