Page No.
Date 13 02 23

Unit1: Fundamentals

- Strings: (Word).

Qun: Defing the strings with examples:
(word).

* String: A string an over alphabet Σ is define as, any finite sequence of symbol from Σ

Word length = No. of symbol in word
$= |x|$.

Conventionally, strings are denoted as small case letters. for e.g. x, y, z, u, v or w

(1) Prefix of string is any no. of leading symbol of string
for e.g. let x = abc
prefixes of x are ϵ, a, ab, abc
ϵ=Null
Where 'ε' (epilson) is an empty string with length zero

(2) Suffix of string is any no. of trailing symbol of string
e.g. x = abc
suffixes of x are ε, c, bc, abc
~~if w and x~~

(3) If w and x are string then 'wx' is called as the concatention of these two strings.

* Symbol:
It is an abstract or user defined entity it cannot be formally define (like points in Geometry)
Fore.g. letters, digits or any symbol that you want consider as part of language your are going to define.

* **Alphabet (Σ)**

It is Finite set of symbol.

e.g. $D = \{0, 1, 2, \ldots, 9\}$

$X = \{+, -, *, /, \%, ++, --\}$

$Y = \{a, b, D, \$, @\}$

The concepts of alphabet is similar to well-known alphabet set $\{a, b, \ldots, z\}$ for the english language.

In above ex. we may observe that, set of 'Y' is containing heterogenuous entities, though it is an alphabet.

* **language:**

language is defined as,

- set of string of symbols from some one alphabet
- ϕ (null set) and the set consisting of empty strings i.e $\{\epsilon\}$ are also languages
- Set of all strings over a fix alphabet.
- Σ is language denoted by (summation closure) Σ^*

e.g. $\Sigma = \{a\}$

$\Sigma^* = \{\epsilon, a, aa, aaa, \ldots\ldots\}$

* **Sets:**

It is collection of objects without repetition. These objects or entities in the set are called members of set.

Operation on set:

Usual operation on sets are :

1) $A \cup B = \{ x \mid x \subseteq A \text{ or } x \subseteq B \}$

2) $A \cap B = \{ x \mid x \subseteq A \text{ and } x \subseteq B \}$

3) $A - B = \{ x \mid x \subseteq A \text{ and } x \notin B \}$

4) $A \times B = \{ * (a,b) \mid a \subseteq A \text{ and } b \subseteq B \}$

~~Finite State Machine: (FSM)~~

- Basic Machine:

It is machine which recognizes an input state 'I' and produces output set 'O', where 'I' and 'O' are Finite.

This type of machine can be viewed as a function which ~~ma~~ maps input state 'I' to output state 'O'.

This function is called Machine function (MAF).

MAF: $I \rightarrow O$

e.g. All logic gates AND, OR, NOT etc.

For AND gate

Where $I = \{(0,0)\ (0,1)\ (1,0)\ (1,1)\}$

$O = \{(0,1)\}$

Machine function can be shown in form Foll^n table:

I	(0,0)	(0,1)	(1,0)	(1,1)
O	0	0	0	1

Features of Basic Machine:

1. It only interpets set input data and process output set which involves, may be combination of inputs. Hence

It is called as combinational Machine.
e.g. AND gate.

2. It performs only table look-up procedure from finite size table.
i.e Machine function Table (MF).

3. It has neither memory nor internal states.

• Finite State Machine:

The internal state of machine alters when the machine receives input and generates require output. It consists of pair of function namely:

1. Machine function:

$$MAF : I \times S \rightarrow O$$

2. State function:

$$STF : I \times S \rightarrow S$$

Where

S = Finite set of internal states of Machine.
I = Finite set of input symbol.
O = Finite set of output symbol.

Features of FSM:

1. The Behaviour of such machine can be completely determine provided its initial state and input which are known.

2. In this machine, finit set of final states is also specified denoted by 'F'.

Where $F \subseteq S$.

There may exist more than one Final States But for given input there will always be only one Final state. IF we change input, we may get some other states as Final states.

3. Tables for MAF and STF are called as Transition table and state table resp.

Problems:

1. Design FSM to check whether given decimal divide by 3.

Soln: Assume,

$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$S = \{S0, S1, S2\}$

$I = \{(0,3,6,9), (1,4,7), (2,5,8)\}$

where

$S0 \rightarrow$ for remainder '0'

$S1 \rightarrow$ for remainder '1'

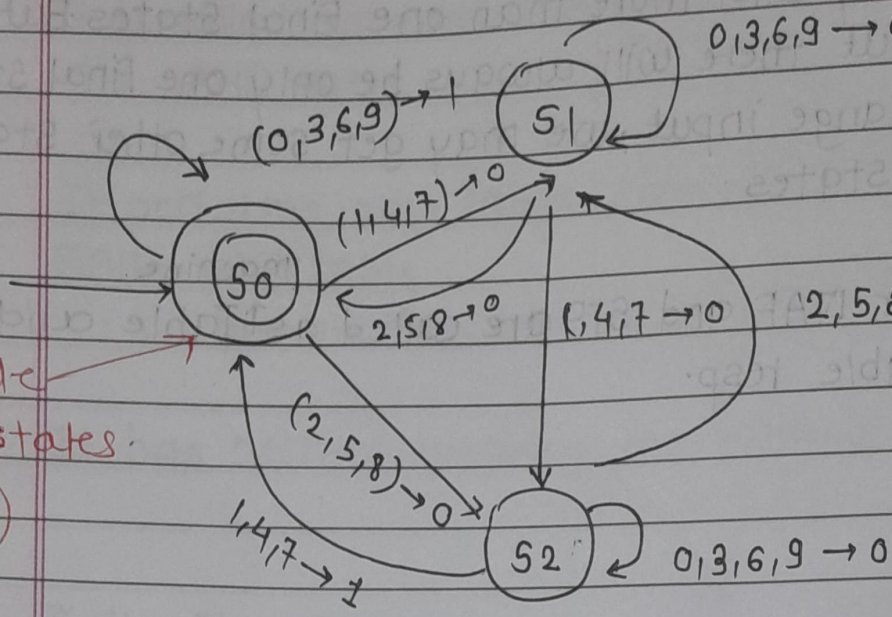
$S2 \rightarrow$ for remainder '2'

MAF: $I \times S \rightarrow O$ STF: $I \times S \rightarrow S$

S \ I	(0,3,6,9)	(1,4,7)	(2,5,8)
	ϕ		
S0	1	0	0
S1	0	0	1
S2	0	1	0

S \ I	(0,3,6,9)	(1,4,7)	(2,5,8)
S0	S0	S1	S2
S1	S1	S2	S0
S2	S2	S0	S1

Transition Diagram:



2. Design FSM to check whether given decimal no. divide by 4.

soln: Assume

$I = \{0,1,2,3,4,5,6,7,8,9\}$

$S = \{S0, S1, S2, S3\}$

$I = \{(0,4,8)\ (1,5,9)\ (2,6), (3,7)\}$

S0 = for remainder '0'

S1 = " '1'

S2 = " '2'

S3 = " '3'

MAF: I X S → O

S \ I	(0,4,8)	(1,5,9)	(2,6)	(3,7)
S0	1	0	0	0
S1	0	0	1	0
S2	1	0	0	0
S3	0	0	1	0

S \ I	(0,4,8)	(1,5,9)	(2,6)	(3,7)
S0	S0	S1	S2	S3
S1	S2	S3	S0	S1
S2	S0	S1	S2	S3
S3	S2	S3	S0	S1

- In the FSM, Final state is merge with only Final state
- Final state cannot merge with non-Final state.
- Non-Final state can be merge only with non-Final state.

By considering above logic,
Replace S1 and S3 ⇒ S13

So, the revise machine fun & state fun table are as below:

MAF:

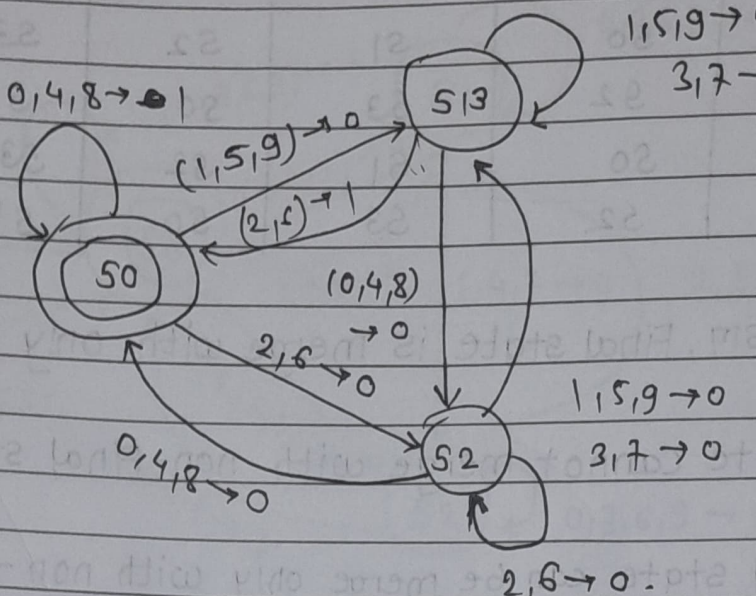
S \ I	(0,4,8)	(1,5,9)	(2,6)	(3,7)
S0	1	0	0	0
S13	0	0	1	0
S2	1	0	0	0

STF:

S \ I	(0,4,8)	(1,5,9)	(2,6)	(3,7)
S0	S0	S13	S2	S13
S13	S2	S13	S0	S13
S2	S0	S13	S2	S13

Page No.
Date

- Transition Diagram:



3. Design FSM to check binary divide by 5.

soln: Assume

$I = \{0, 1, 2, 3, 4, 5\}$

$S = \{S0, S1, S2, S3, S4\}$

S0 = Remainder for '0'
S1 = Remainder For '1'
S2 = Remainder For '2'
S3 = —"— '3'
S4 = —"— '4'

MAS: I X S → O

I X S → S

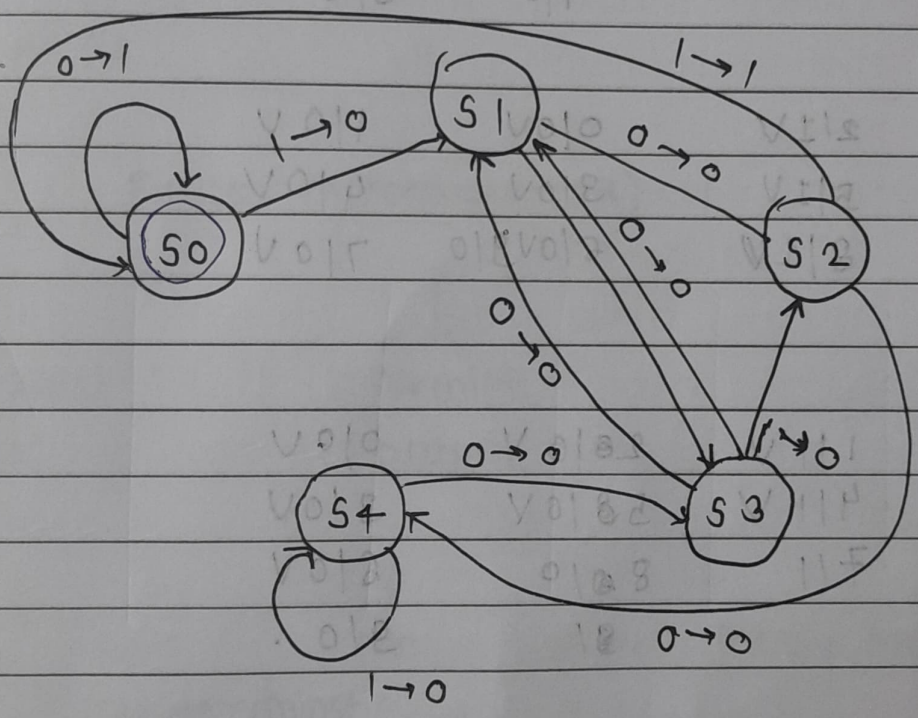
S \ I	0	1
S0	1	0
S1	0	0
S2	0	1
S3	0	0
S4	0	0

S \ I	0	1
S0	S0	S1
S1	S2	S3
S2	S4	S0
S3	S1	S2
S4	S3	S4

00	00	= 0
10	01	= 1
20	10	= 2
30	11	= 3
40	100	= 4
	101	= 5
	110	= 6
	111	= 7
	1000	= 8
	1001	= 9

~~MAP: IXS → 0~~

- Transition Diagram:



0
101

Theory Ques.

Date

* Transition / Adjacency Matrix.

STF: IXS → S

S \ I	(0,3,6,9)	(1,4,7)	(2,5,8)
S0	S0	S1	S2
S1	S1	S2	S0
S2	S2	S0	S1

MAF: IXS → O

S \ I	(0,3,6,9)	(1,4,7)	(2,5,8)
S0	1	0	0
S1	0	0	1
S2	0	1	0

CS \ NS	S0	S1	S2
S0	0|1 V 3|1 V 6|1 V 9|1	1|0 V 4|0 V 7|0	2|0 V 5|0 V 8|0
S1	2|1 V 5|1 V 8|1 V	0|0 V 3|0 V 6|0 V 9|0	1|0 V 4|0 V 7|0 V
S2	1|1 V 4|1 V 7|1	2|0 V 5|0 V 8|0 9|0	0|0 V 3|0 V 6|0 V 9|0 .

* Finite Automata: (FA)

1. It consider as, the mathematical model of, finite state machine.

2. It consist of finite set of states & set of transitio from state to state that occur on input symbol

chosen from an alphabet Σ.

* Formal Definition:

FA is denoted by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

where Q: Finite set of states

Σ: Finite input alphabet.

q_0: Initial state of FA,

$q_0 \subset Q$

F: Set of final state,

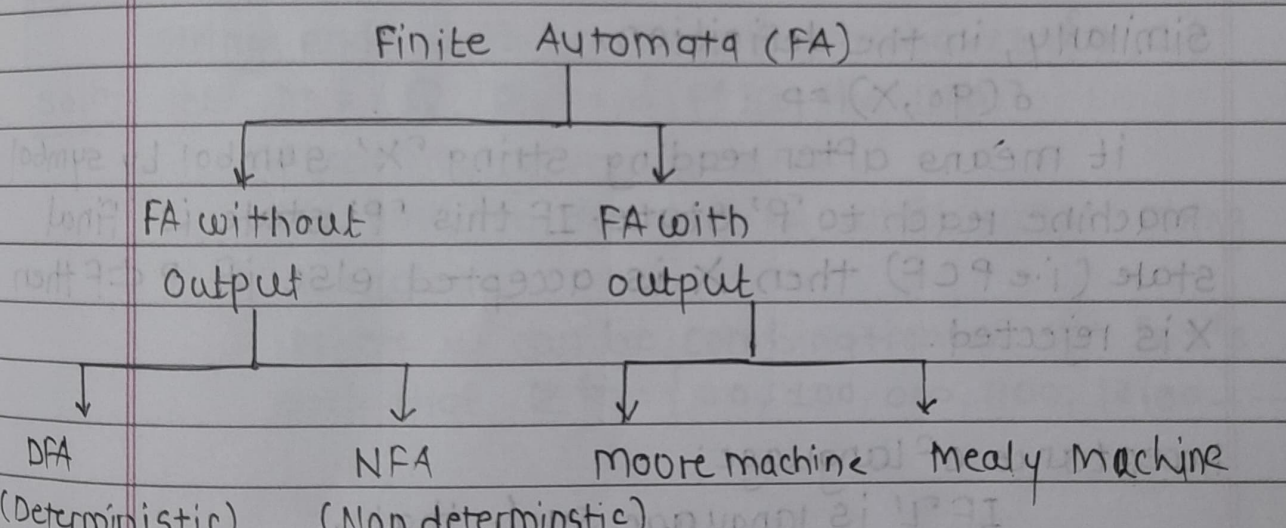
$F \subseteq Q$

δ: Transition ~~from~~ fun called as state fun mapping 'Q' $\times \Sigma$ to Q.

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

Theory Que

* Types of FA:



Acceptance of string:

A string 'x' is said to be accepted by a Finite automata $M = (Q, \Sigma, \delta, q_0, f)$

IF $\delta(q_0, x) = P_i$ for some $P \subset F$.

Explanation • As we know that δ is ~~bullet~~ state function does mapping from Q × Σ to Q. each individual transition can be denoted by

$\delta(q_0, a_1) = q_1$

Where q_0 is initial state / current state.

a_1 = current input / symbol.

q_1 = next state.

Similarly, $\delta(q_1, a_2) = q_2$

- If we combine above too as $\delta(q_0, a_1 a_2) = \delta(\delta(q_0, a_1), a_2)$
  $= \delta(q_1, a_2)$
  $= q_2$

- i.e IF current state q_0 then after reading the string $a_1 a_2$ the state reach to q_2.

- Similarly, in the definition,
  $\delta(q_0, X) = P$
  it means after reading string 'X' symbol by symbol machine reach to 'P' state. If this 'P' state is final state (i.e P ∈ F) then X is accepted else if P ∉ F then X is rejected.

Acceptance of language:

IF 'L' is language such that,

$L = \{X \mid \delta(q_0, X) \text{ is in } F\}$ then it is said to be accepted by 'FA' 'M' and denoted by L(M).

Explanation:

- Language is consisting of string or set of string

over some alphabets if all strings 'x' of the language are accepted by 'FA' then language is said to be accepted by 'FA'.

- For every string 'x', $\delta(q_0, x)$ gives the state in which 'FA' resides after reading all symbol in x.
- If it is final state for each & every string x ∈ L then 'L' is accepted by given 'FA'.
- Language is accepted by 'FA' is called as a Regular set.

Theory qut:

Deterministic FA (DFA) & Non-deterministic FA (NFA):

Qun: Design DFA that accept set of string such that every string ends with 00 over the alphabet {0, 1}

soln: let $M = (Q, \Sigma, q_0, \delta, F)$ be the DFA

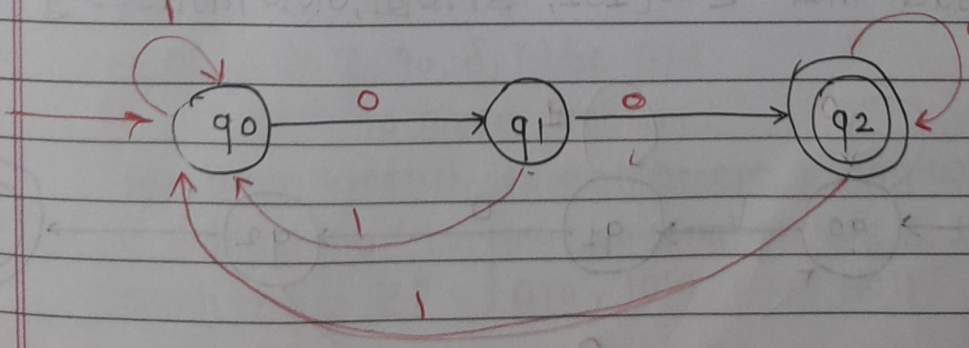
Accorn to given problem,

We have to design DFA which accept string

$\omega = \omega'00$

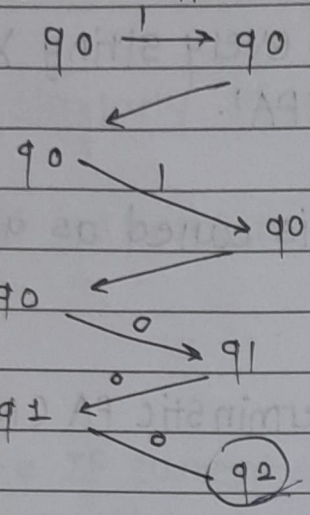
where ω' can be combination of 0's & 1's.

such that $\Sigma^* = \{00, 100, 000, 1100, 10100 \ldots\}$

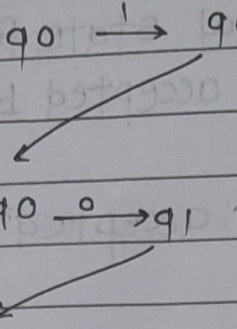


Σ / S	0	1
q0	q1	q0
q1	q2	q0
q2	q2	q0

For ex. 1100.



101



Qun: Design DFA for string with accept a string connecting 101 as substring over Σ^* such that $\Sigma = \{0,1\}$.

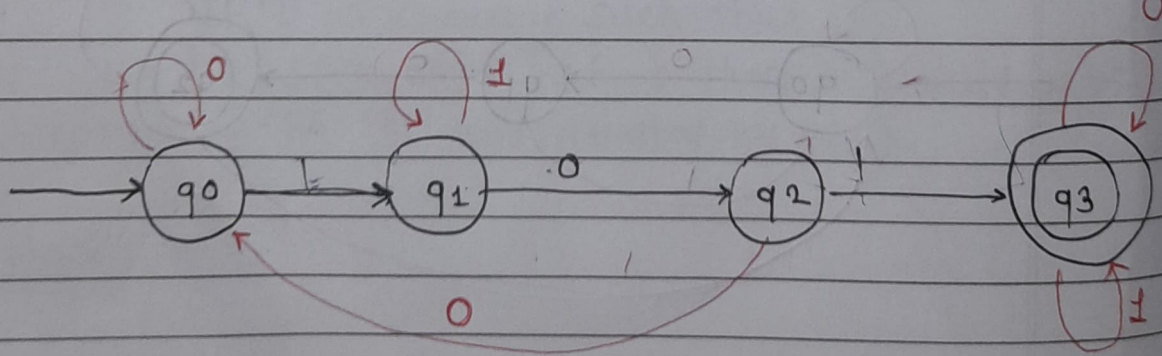
soln:

Let $M = (Q, \Sigma, q_0, \delta, f)$ be the DFA.

Accorn to given problem,

we have design DFA which accept string $\omega = \omega' \boxed{101} \omega'$ where ω' is combination of 0's & 1's.

such that $\Sigma^* = \{101, 1101, 0101, 01010, 11011 \ldots\}$



Transition Diagram.

Binary Adder qun.

Transition Table:

Σ / S	0	1
q0	q0	q1
q1	q2	q1
q2	q0	q3
q3	q3	q3

For 1101.

$\delta(q0, 1101) = \delta(\delta(q0,1), 101)$

$= \delta(q, 101)$

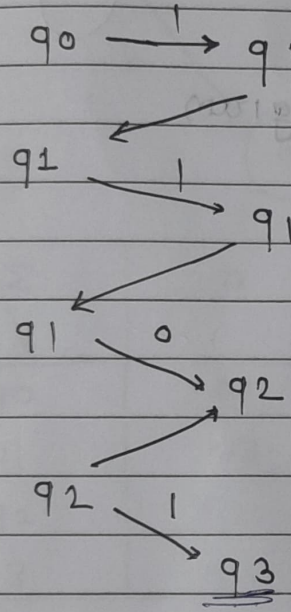
$= \delta(\delta(q,1), 01)$

$= \delta(q, 01)$

$= \delta(\delta(q,0), 1)$

$= \delta(q2, 1)$

$= q3$



Qun: Design DFA for string which do not accept 001 over Σ^* such that $\Sigma = \{0, 1\}$

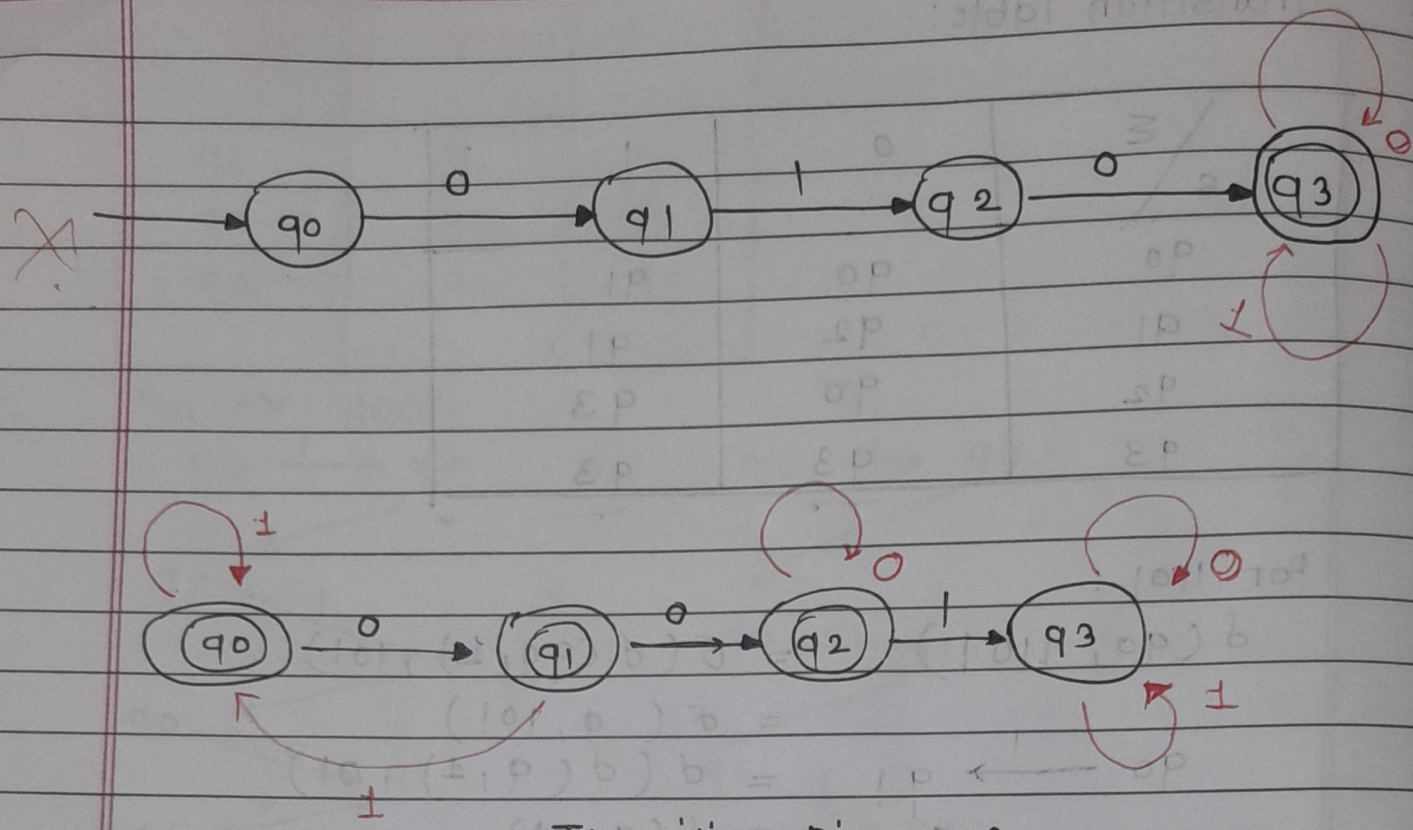
Soln: let $M = (Q, \Sigma, q_0, \delta, F)$ be DFA

Accorn to given problem,

We have design do not accept string $w = w'001w'$

Where w' is any combination 0's & 1's.

such that $\Sigma^* = \{010, 1010, 1100, 11011$



Transition Diagram

Transition Table

Σ / S	0	1
q0	q1	q0
q1	q2	q0
q2	q2	q2
q3	q3	q3

e.g. 1111

$\delta(q0, 1111) = \delta(\delta(q0, 1), 111)$

$q0 \xrightarrow{1} q0$ $\quad \delta(\delta(q0,1),1) = \delta(q0, 11)$

$\delta(q0,1) = \delta(\delta(q0,1), 11)$

$q0 \xrightarrow{1} q0$ $\quad q0 = \delta(q0, 111)$

$q0 \xrightarrow{1} q0$

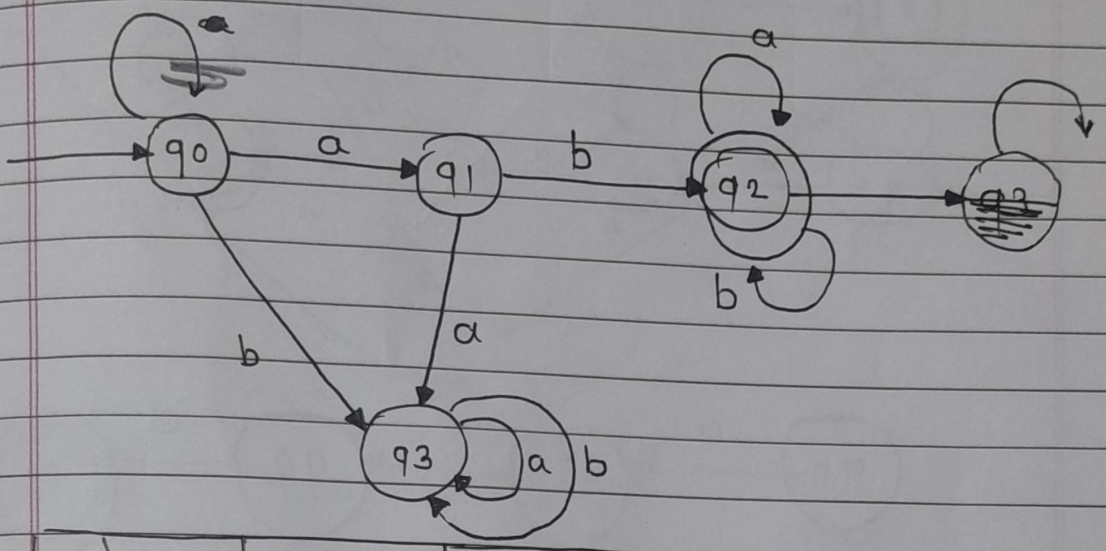
$q0 \rightarrow q0$

Qun: Design DFA for string which accept set of string on $\Sigma = \{a, b\}$ starting with prefix ab.

Soln: let $M = (Q, \Sigma, q_0, \delta, F)$ be DFA.

Accorn to given problem.

$\Sigma^* = \{ab, aba, abb, abaa, abbb, \ldots\}$



Σ / s	a	b
q_0	q_1	q_3
q_1	q_3	q_2
q_2	q_2	q_2
q_3	q_3	q_3

e.g. abb

$q_0 \xrightarrow{a} q_1$

$q_1 \xrightarrow{b} q_2$

$q_2 \xrightarrow{b} q_2$

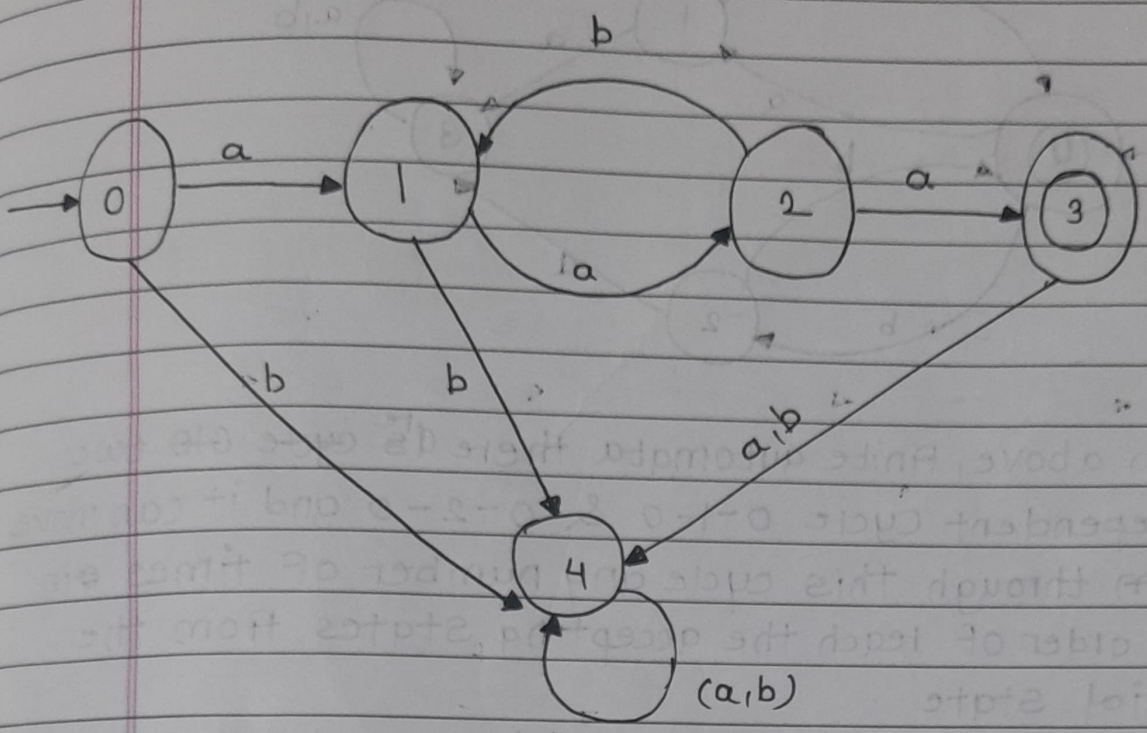
$\delta(q_0, abb) = \delta(\delta(q_1, a), bb)$

$\delta(\delta(q_0, b), b) = \delta(q_1, ab)$

$\delta(q_0, b) = \delta(\delta q_1, b), bb)$

$= q_2$

Qun: Find set of string which is accepted by given finite automata.

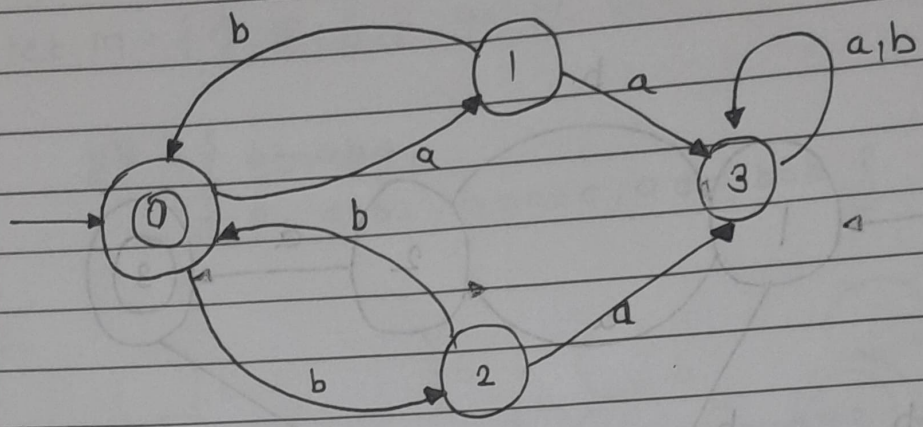


In above, finite automata there is cycle ~~1-0-1~~ 1-2-1 & can go through this cycle any number of times, By reading substring repeatedly.

To find language, it accept first from initial state '0' go to state 1 by reading 'a' then from state 1 then go to cycle 1-2-1 any no. of times, by reading substring ab. Any no. of times to come back to state 1, this represented by (ab)*

then from state 1 it goes to state 2 by reading 'a' again then from state 2 to state 3 again reads 'a'. Thus the string i.e accepted by above FA can be represented by a(ab)*aa.

Qun: Find set of string i.e accepted by foll^n automata.

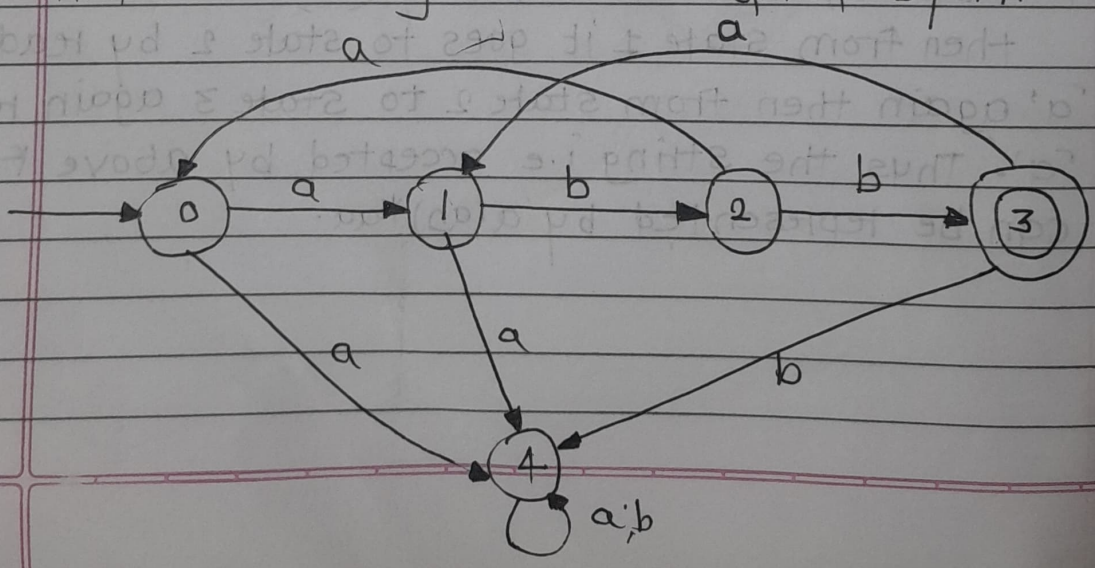


In above, Finite automata there is ~~cycle~~ two Independent cycle 0-1-0 & 0-2-0 and it can move ~~from~~ through this cycle any number of times in any order of reach the accepting states from the initial state

for the 1st cycle 0-1-0 it have substring (ab)* and 2nd cycle 0-2-0 it has substring (bb)*. Thus the string accept these FA. can be Represented as, $\boxed{(ab)^* + (bb)^*}$

i.e $\boxed{(ab + bb)^*}$

Q. Find set of string which is accepted by FA.



cycle1: 1-2-0-1 $a(baab)^*$

cycle 2: 1-2-3,-1 $a(bba)^*$

$\therefore$ Thus string accepted by FA can be represented as $a(baa)^* + a(bba)^*$

i.e $a[(baa+(bba)^*]bb$

Q. Design FSA For Binary Adder. OR. Design FSA to add two binary no. of equal length

Soln: (1) Assume,

$I = \{(0,0), (0,1)(1,0), (1,1)\}$

$S = \{\text{No carry, Carry}\}$ ←

(2) STF: $I \times S \to S$

(3) MAF: $I \times S \to O$.

STF: $I \times S \to S$

I \ S	No carry	carry
(0,0)	0	0
(0,1)	0	0
(1,0)	0	0
(1,1)	0	1

MAF: $I \times S \to O$.

I \ S	No carry	Carry
(0,0)		
(0,1)		
(1,0)		
(1,1)		

I \ S	No carry	carry
(0,0)	0	1
(0,1)	1	2
(1,0)	1	2
(1,1)	2	3

No carry:

STF: IXS → S.

I1 \ I2	0	1
0	Nocarry	Nocarry
1	Nocarry	carry.

MAF: IXS → O

I1 \ I2	0	1
0	0	1
1	1	0

I1 I2

0+0+0 = No-carry

0+1+0 = No carry.

1+0+0 = No carry.

1+1+0 = Carry.

I1 I2

0+0+0 = 0

0+1+0 = 1

1+0+0 = 1

1+1+0 = 0

0 (1) carry.

Carry:

STF: IXS → S

MAF: IXS → O

I1 \ I2	0	1
0	Nocarry	carry
1	carry	carry

I1 \ I2	0	1
0	1	0
1	0	1

I1 I2

0+0+1 = No carry

0+1+1 = = carry

STF : I X S → S NSF : I X S → O

I \ S	Nocarry	carry
(0,0)	No carry	No carry
(0,1)	〃	Carry
(1,0)	〃	〃
(1,1)	Carry	〃

I \ S	Nocarry	carry.
(0,0)	0	1
(0,1)	1	0
(1,0)	1	0
(1,1)	0	1