UNIT - I

Introduction to formal proof

Deductive Proof

Consists of sequence of statements given logical reasoning in order to prove. The I^st (or) Initial Statement

Hypothesis

Inductive Proof

is recursive kind of proof which consists of sequence parameterized statements that use the statement itself with lower values of it parameter

In short, formal proof are the proof in which try to prove that Statement B is true because A is true. The Statement A is called Hypothesis and B is called conclusion. In other words "if A Then B" deduced from A.

Various Form of Proof

* Proofs above Set
* Proofs by contradictions
* Proofs by counter example.

A proof about sets.

The sets is a collection of elements (or) items.

this proof is the of the kind of "if and only if" that means an element x is in A if and only if it is in B

We will make use of Sequence of Statements along with logical Justification in order to prove this equivalence

Let as prove $P \cup Q = Q \cup P$

1.	x is in $P \cup Q$	1 given
2.	x is in P or x is in Q	2) And by definition of union
3.	x is in Q or x is in P	2 " " " "
4.	x is in $P \cup Q$	4. Rule(2) And by definiro of union

Proving RHS

S.No		
1.	x is in Q ∪ P	Given
2.	x is in Q or x is in P	1. And by definition of union
3.	x is in P (or) x is in Q	3. And by definition of union
4.	x is in P ∪ Q	3. Rule(2) And by definition of union

Hence P ∪ Q = Q ∪ P. Thus A = B is true elements x is in B if and only if x is in A.

Proof by contradiction

In this type of proof for the statement of the form if A Then B we start with statement A is not true and Thus by assuming false A we try to get the conclusion of statement B. When it becomes impossible to reach to Statement B we Contradict ourself and accept that A is true.

For example.

Prove that

$$P \cup Q = Q \cup P$$

Proof

* Initially we assume that $P \cup Q = Q \cup P$ is not true

That is $P \cup Q \neq Q \cup P$

Now consider that x is in Q, or x is in P. Hence we can say x is in $P \cup Q$. (according to definition of union)

* But this also implies that x is in $Q \cup P$ according to definition of union

* Hence the assumption which we made initially is false

* Thus $P \cup Q = Q \cup P$ is proved.

Prove that $\sqrt{2}$ is not rational

Sol

We will assume that, $\sqrt{2}$ is rational number

That means we can write

$$\sqrt{2} = \frac{a}{b} \quad \text{---} \rightarrow (1)$$

Where a and b are integer $\frac{a}{b}$ is irreducible and can be simplified to lowest term.

Squaring in both side eqn (1)

$$2 = \frac{a^2}{b^2} \quad (i.e) \quad 2b^2 = a^2$$

This show that LHS is even (i.e multiple of two). Hence RHS is also even

Now if we write $a = 2k$ then

$$2b^2 = (2k)^2 \Rightarrow 4k^2 \Rightarrow b^2 = 2k^2$$

this means even b is an even number

Hence our assumption that a/b is simplified to lowest term is correct

Proof by counter Example.

In order to prove certain statements, we need to see all possible conditions in which that statements remain true.

There are some situations in which the statements cannot be true

For example

There is no such pair of Integers such that

$$a \bmod b = b \bmod a$$

Proof

Consider $a=2$ & $b=3$ then clearly

$$2 \bmod 3 \neq 3 \bmod 2$$

Thus the given pair is true for any pair integers but if $a=b$ Then naturally

$$a \bmod b = b \bmod a$$

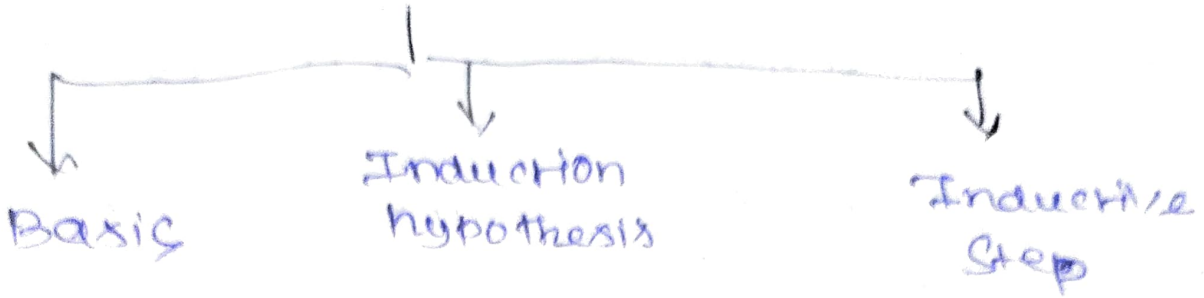
thus we need to change the Statement slightly. we can say

$$a \bmod b = b \bmod a \text{ when } a \neq b$$

This type of counter proof is called counter example Such proof is true only at some

Specific conditions.

Inductive proof



Inductive proof are special proofs based on some observations It is used to prove recursively defined objects. This type of proofs is also called as proofs by mathematical inductions

Basic

In this step we assume the lowest possible value. This is an initial step in the proof of mathematical induction.

for example, we can prove that that the result is true for n=0 (or) n=1

Induction hypothesis

In this step we assign value of n to some other value k. that mean we will check whether the result is true for n=k (or) not.

3. Inductive Steps

In this Step if $n=k$ is true then we check whether the result is true for $n=k+1$ (or) not. If we get the same result at $n=k+1$ then we can state that given proof is true by principles of mathematic Induction.

Example.

Prove by induction on n that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Sol

1) Basis of Induction

Assume that

$n=1$ Then

L.H.S $= n = 1$

R.H.S $= \frac{1(1+1)}{2}$

$= 2/2$

$= 1$

Now

Induction Hypothesis

Now we will assume $n=k$ and will obtain the result for it.

The equation then becomes.

$n=k$

$$1+2+3+\cdots+k = \frac{k(k+1)}{2}$$

Inductive Step

Now assume that equation is true for $n=k$ And we will then check if it is also true for $n=k+1$ or not

Consider the equation assumming

$n=k+1$

$$L.H.S = \underbrace{1+2+3+\cdots+k}+k+1$$

$$= \frac{k(k+1)}{2} + k+1$$

$$= \frac{k(k+1)+2(k+1)}{2}$$

$$= \frac{k(k+1)+2k+2}{2}$$

$$= \frac{k(k+1)+k(k+2)}{2}$$

$$c(k) = \frac{(k+1)\,(k+1+1)}{2}$$

$$= \frac{n(n+1)}{2}$$

Ex

Prove $n! > 2^{n-1}$

Sol

consider

Basis of Induction

Let $n=1$ then

$LHS = 1$, $RHS = 2^{1-1} \Rightarrow 2^0 = 1$

hence

$n! > 2^{n-1}$ proved

2. Induction hypothesis

$n=k$

$k! = 2^{k-1}$ where $k >= 1$

Then

$(k+1)! = (k+1)k!$ by definition of $n!$

$= (k+1)\,2^{k-1}$

$= 2 \times 2^{k-1} = 2^k$

Prove that $1+4+7+\cdots+(3n-2) = \frac{n(3n-1)}{2}$ for $n>0$

Sol Consider

1. Basis of Induction

Let $n=1$

$$= \frac{n(3n-1)}{2} = \frac{1(3\cdot1-1)}{2} = \frac{2}{2} = 1$$

As. LHS = RHS given expression is true for $n=1$

2. Induction hypothesis

We assume that the given expression is true for $n=k$

$$1+4+7+\cdots+(3k-2) = \frac{k(3k-1)}{2} \text{ is true.}$$

Now we will prove it for $n=n+1$

L.H.S $1+4+7+\cdots+3(k-2)+3(k+1)-2$

We will substitute RHS of equation (1)

$$= \frac{k(3k-1)}{2} + (3(k+1)-2)$$

$$= \frac{k(3k-1) + 2[3(k+1)-2]}{2}$$

$$= \frac{3k^2 - k + 2[3k+3-2]}{2}$$

$$= \frac{3k^2 - k + 2[3k+1]}{2}$$

$$= \frac{3k^2 - k + 6k + 2}{2}$$

$$= \frac{3k^2 + 5k + 2}{2} = \frac{3k^2 + 2k + 3k + 2}{2}$$

$$= \frac{(3k^2 + 3k) + (2k+2)}{2}$$

$$= \frac{2(k+1) + 3k(k+1)}{2}$$

$$= \frac{(k+1) + 3k^2 + 3k}{2}$$

$$= \frac{(k+1)(3k+2)}{2}$$

$$= \frac{(k+1)(3(k+1)-1)}{2}$$

If $4+7+\cdots+(3k-2)=\frac{k(3k-1)}{2}$

$n=k+1$

If $\underbrace{4+7+\cdots+(3k-2)}_{\downarrow}+(3(k+1)-2) = k+1(3(k+1)-1$

$\frac{k(3k-1)}{2}+\{3(k+1)$ (or) $(3k+3-2)$ $(3k+)\} = \frac{(k+1)(3k+3-1)}{2}$ $\frac{2}{2}$

$\frac{k(3k+1)+2(3k+1)}{2} = \frac{(k+1)(3k+2)}{2}$

$\frac{3k^2-k+6k+2}{2} = \frac{3k^2+2k+3k+2}{2}$

$\frac{3k^2+5k+2}{2} = \frac{3k^2+5k+2}{2}$

L.H.S = R.H.S

S(n) is true.

Prove that $1^2+2^2+3^2+\dots+n^2=\sum_{i=1}^{n} i^2=\frac{n(n+1)(2n+1)}{6}$

Sol

1. Basis of Induction

Let $n=1$

L.H.S $=1^2=1$

R.H.S $=\frac{1(1+1)(2\cdot1+1)}{6}=\frac{6}{6}=1$

R.H.S $=1$

L.H.S $=$ R.H.S It is proved for $n=1$

2. Induction Hypothesis

$$1^2+2^2+3^2+\dots+k^2=\sum_{i=}^{k}=\frac{k(k+1)(2k+1)}{6}$$

is true ----> (1)

Now we will try to prove IE for $n=k+1$. Hence we will sub n by $k+1$

$$1^2+2^2+3^2+\dots+(k+1)^2=\sum_{i=1}^{k+1} i^2=\frac{k+1(k+1+1)(2k+1+1)}{6}$$

(n → n^{k+1}, $2n+1$ → $2k+1+1$)

L.H.S $=1^2+2^2+3^2+\dots+(k+1)^2+k^2$

we will sub eqn 1 and we will get

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2 = (k+1)\left[\frac{k(2k+1)}{6} + (k+1)\right]$$

$$= (k+1)\left[\frac{k(2k+1)+6k+6}{6}\right]$$

$$= (k+1)\left[\frac{2k^2+k+6k+6}{6}\right] = (k+1)\left[\frac{2k^2+7k+6}{6}\right]$$

$$= \frac{(k+1)((2k^2+4k+3k+6))}{6}$$

$$= \frac{(k+1)(2k(k+2)+3(k+2)}{6} = \frac{(k+1)(2k+3)(k+2)}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

$$= \frac{(k+1)(k+1+1)\,2(k+1)+1}{6}$$

$$= \frac{n(n+1)(2n+1)}{6}$$

$= R.H.S$

$n = k+1$

$(2n+1)$

$(2 \cdot k+1+1)$

THEOREM

if $x \geq 4$, then $2^x \geq x^2$.

if H then C

$\therefore$ $H = x \geq 4$ & $C = 2^x \geq x^2$

parameter x = integer

$x = 4$
$x = 5$

True (or) false

For $x \geq 4$

$2^4 = 4^2 = 16$ $(2^x \geq x^2$

for $x = 5$

$2^5 = 32$, $5^2 = 25$

$32 \geq 25$

$\therefore$ True.

Inductive Proof Example

if $x \geq 4$ then $2^x \geq x^2$

Basis $x = 4$

$2^4 \geq 4^2$

$16 \geq 16$

Induction Step

Assume $2^k \geq k^2$ is true ---> ①

Prove $2^{k+1} \geq (k+1)^2$

$2^{k+1} = 2 \times 2^k \geq 2k^2$

$2^{k+1} \geq 2k^2$

$2^{k+1} \geq 2k^2 \geq (k+1)^2$

Prove $2k^2 \geq (k+1)^2$

$2k^2 \geq (k^2 + 2k + 1)$

$k^2 \geq 2k + 1$

$\div k$ $k \geq 2 + \frac{1}{k}$ ----> ②

$k=4$

$4 \geq 2 + \frac{1}{4} = \frac{9}{4}$

$4 \geq 2.25$

$k=5$

$5 \geq 2 + \frac{1}{5} = \frac{11}{5}$

$5 \geq 2.2$

$k=6$

$6 \geq 2 + \frac{1}{6} \Rightarrow \frac{13}{6}$

$6 \geq 2.1667$

$2k^2 \geq (k+1)^2$ Prove

$2^{k+1} \geq (k+1)^2$ Prove

If $x \geq 4$ then $2^x \geq x^2$ prove by Mathematical Induction.

Basic Concept of Automata theory

The automata theory has a basic Fundamendal unit called Set.

Set

→ is defined as collection of objects. These object are called elements of the Set

Set is denoted by a capital letter.

- → Listing Method
- → Describing properties
- → Recursion Method.

Listing Method

The elements are listed in the set

For ex

A set of elements which are less then 5.

$A = \{0, 1, 2, 3, 4\}$

Describing properties

the properties of elements of a set define the set.

For: ex

A set of vowels

$A = \{ a, e, i, o, u \}$

Recursion Method

The Recursion occurs to define element of the set

For ex

$A = \{ x | x \text{ is square of } n \}$ where $n \geq 1$

$A = \{ 0, 1, 4, 9, 16 \ldots\ldots 100 \}$

Subset

The subset A is called subset of set B if every element of set A is present in set B but reverse is not true.

It is denoted by $A \subseteq B$

For ex

$A = \{ 1, 2, 3 \}$ and $B = \{ 1, 2, 3, 4, 5 \}$

Then $A \subseteq B$

Empty Set

The Set having no element An it is called empty set. It is denoted by $A = \{\}$ and it can be written as ϕ (phi)

Null String

The null element is denoted by ε or $\wedge$ character.

Null element means no value character But ε does mean ϕ

Power Set

The power set is a set is a set of all subsets of its elements

For ex $A = \{1, 2, 3\}$

The power set $Q = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{3,2\}, \{2,3\}, \{1,2,3\}$

the no. of elements are always equal to 2^n. where n is number of elements in original set

3 elements

$2^3 = 8$ elements.

Equality

The two set are said to be equal ($A=B$) if $A \subseteq B$ and $B \subseteq A$

For ex

$A = \{1,2,3\}$ and $B = \{1,2,3\}$

then $A=B$

$|A|$ denotes the Length of Set A. (i.e) number of elements in Set A

For ex : if

$A = \{1,2,3,4,5\}$ then

$|A| = 5$.

operation on Set

Union

$A \cup B$ is union operation - if

$A = \{1,2,3\}$ $B = \{1,2,4\}$

$A \cup B = \{1,2,3,4\}$

(i.e) combinations of both the sets.

Intersection operation

$A \cap B$

$A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$

$A \cap B = \{2, 3\}$ (i.e. collection of common elements from both sets)

$A - B$ difference operation if

$A = \{1, 2, 3\}$ $B = \{2, 3, 4\}$ then

$A - B = \{1\}$ the elements which are there in set A but not in set B

$\bar{A}$ is complement operation

$\bar{A} = U - A$ where U is universal set

For $U = \{10, 20, 30, 40, 50\}$

$A = \{10, 20\}$

$\bar{A} = \{30, 40, 50\}$

Cartesian product of two sets.

The Cartesian products of two sets A and B is a set of all possible ordered pairs whose first component is member of A and B whose second component is member of B.

The Cartesian product is denoted by

$A \times B$

$\therefore A \times B = \{\{a,b\} \mid a \in A \text{ and } b \in B$

For ex : Let

$A = \{a, b\}$ and $B = \{0, 1, 2\}$

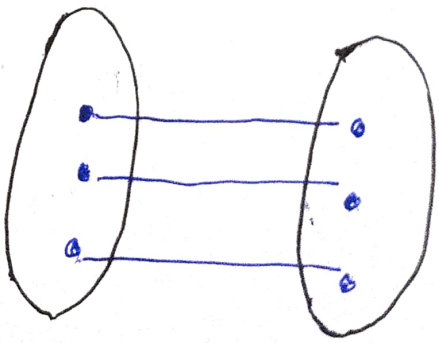
Then the Cartesian product of A and B is

$A \times B = \{(a,0)\ (a,1)\ (a,2)\ (b,0)\ (b,1)\ (b,2)\}$

Cardinality sets.

The number of members in the set of the set is nothing but

These sets A_1 and A_2 have the same cardinality as there is one to one mapping of the elements of A_1 and A_2. The different cardinalities of set can be one to-one one to many, many to-many



Relations

the Relation R is a collections for the sets S which Represents the pair of elements

For ex (a,b) is in R. We can Represents their relation as aRb. The first component of each pair is chosen from set called domain and second component of each pair is chosen from a set called range.

properties of relations

A relation R on set S

1. Reflexive if iRi for all i s
2. Irreflexive if iRi is false for all i in S

For example

Reflexive relation R can be = {(a,a),(b,b)}

irrefelxive relation R canbe = {(a,b)}

closure of Relations

Sometimes when the relations R is given, it may not be reflexive (or) transive By adding some pairs we make the relations as reflexive (or) transitive

for ex let {(a,b),(b,c),(a,a)(b,b)}

Kleen closure

The Kleen closure is also called Star closure.

This is set of string of any length (including null string ε).

Each string is obtained from input set Σ the Kleen star of the empty language ϕ is the empty string ε.

Ex Let $\Sigma = \{a, b\}$ obtain $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

Sol

$\Sigma^* = \{\varepsilon, a, b, aa, bb, aba, aaa, bbb, baba \ldots\}$

is a set of strings of any Length.

The positive closure Σ^+ can be defined as $\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$ that means it consists of all the string of any length expect a null string.

Ex Let $\Sigma = \{a, b\}$ obtain $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

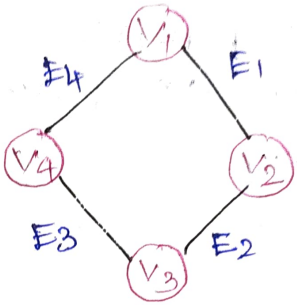
Sol $\Sigma^+ = \{a, b, aa, bb, ab, ba, aba, aaa, bbb, baba, abaab \ldots\}$ is a set of

strings of any length expect null string (ε epsilon).

Note that $\Sigma^* = \Sigma^+ + \varepsilon$

Graph

A graph denoted $G = (V, E)$ consists of finite set of (vertices or node) V and set of edges, the edges are nothing but pair of vertices.



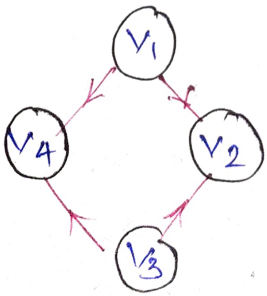
E_1 is a edges connecting the vertices V_1 & V_2

The Set $V = \{V_1, V_2, V_3, V_4\}$

& Edges $E = \{E_1, E_2, E_3, E_4\}$

Directed graph

The directed graph is also collection of vertices and edges in which the directions are mentioned along the edges



For exam

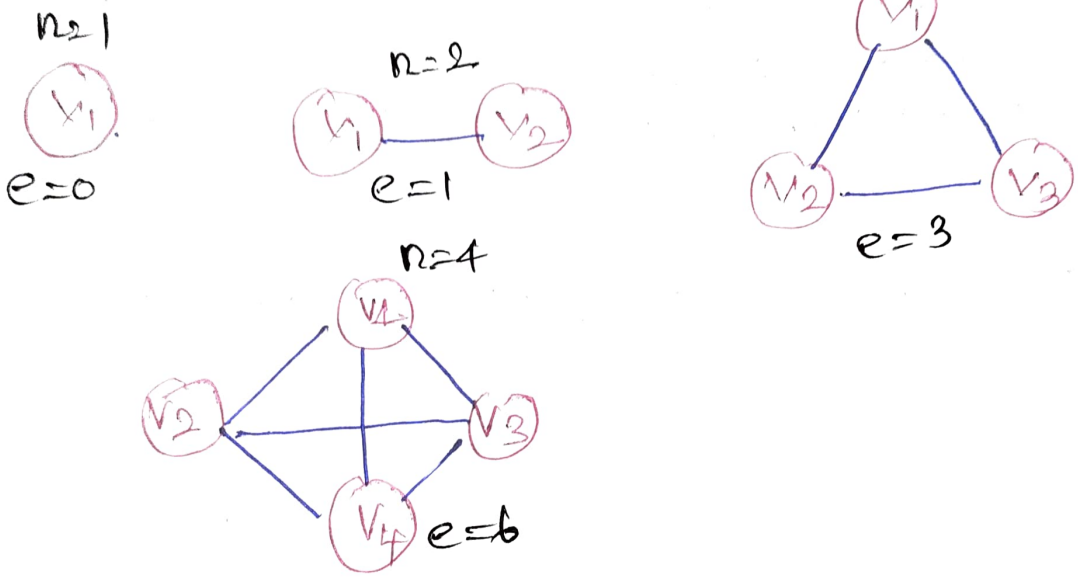
The Edges E_1 shows the directions to V_2 Vertex from Vertex V_1

We will see the directed graphs in further chapters and we will call those graph as transition graphs.

Complete graph

A Complete graph is a simple undirected graph in which every pair of distinct vertices is connected by unique edges

The complete graph with n vertices has n(n-1)/2 edges. This can be shown as follows.

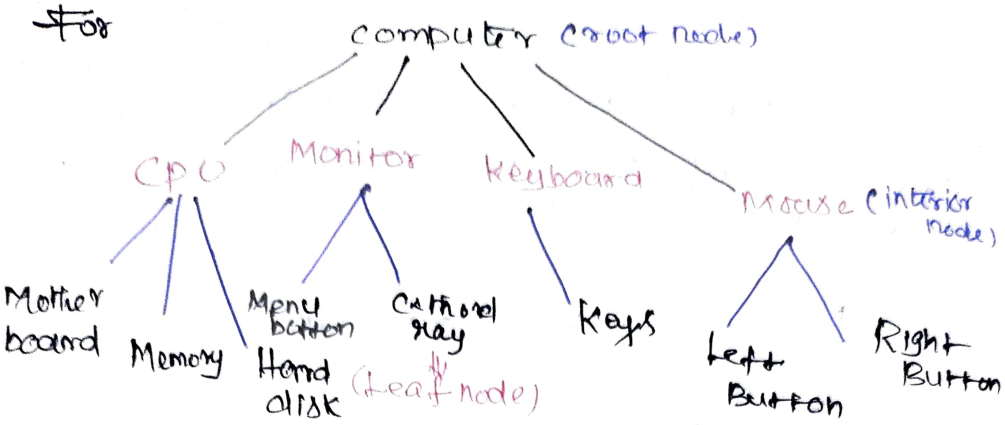


Tree

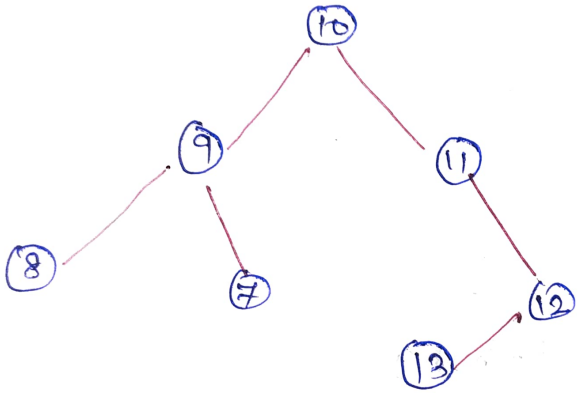
Tree is a collection of vertices and edges with following properties

(i) There is one vertices, called root which has no predecessors

(ii) From this root node all the successors are ordered



Binary tree.



8, 7, 13 ⟶ Leaf node (having no. children)

The height of tree the maximum depth of node. The above given tree is having height

A Complete binary tree in which there are 2^k nodes at every depth k where $k \leq n$.

Alphabet

An Alphabet is a Finite collection of symbols. We cannot define symbol formally

The example of alphabet is

$$\delta = \{a, b, c \ldots z\}$$

The elements a .... z are the alphabets.

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

Here 0 & 1 are alphabets. denoted by ω.

String

String is Finite Collection of symbols From alphabet.

For ex

$\Sigma = \{a, b\}$ then Various String that can be formed from Σ are $\{ab, aa, aaa, bb, ba, aba \ldots\}$ An Infinite number of strings can be generated from the set

* The empty string can be denoted by ε

* The prefix of any string is any number of leading symbol of that string can be generated.

Symbols

For example in string "011" the prefixes can be 0, 00, 001.
suffixes can be 1, 11, 011.

in string "Mango" prefix :- Man
suffix :- go

Language

The language is a collection of appropriate string

The language is defined using an input set. The input set is typically denoted by ~~letter~~ Σ

$$\Sigma = \{ \varepsilon, 0, 00, 000 \ldots \}$$

Here the Language L is defined as any number of zeros

Note that the language is formed by appropriate string and strings are formed by alphabets.

* The empty string can be denoted by ε

* The prefix of any string is any number of leading symbol of that string can be generated.

Symbols

For example in string "011" the prefixes can be 0, 00, 001.
suffixes can be 1, 11, 011.

in string "Mango" prefix :- Man
suffix :- go

Language

The language is a collection of appropriate string

The language is defined using an input set. The input set is typically denoted by letter Σ

$$\Sigma = \{ \varepsilon, 0, 00, 000 \ldots \}$$

Here the Language L is defined as any number of zeros

Note that the language is formed by appropriate string and strings are formed by alphabets.

operation on string

Concatenation:

In this operation two string combined together to form a single string.

For exam

$x = \{white\}$ $y = \{house\}$

xy = whitehouse.

T-Transpose: The Transpose of operation is also called Reverse operation.

For example:

$(Xa)^T = a(x)^T$

if $(ababbb)^T = (bbbaba)$

Palindrome

↳ of string is a property of a string in which string can be read same From left to Right as well as From right to left.

example APPA

AMMA

operations on language.

Language is Collection of string.

if L_1 & L_2 two language

union ---> $L_1 \cup L_2$

intersection ---> $L_1 \cap L_2$

concatenation ---> $L_1 L_2$

Difference ---> $L_1 - L_2$.

Ex:.

Q:- Describe the following over the input set $A = \{a, b\}$

(i) $L_1 = \{a, ab, ab^2\}$

(ii) $L_2 = \{a^n b^n \mid n \geq 1\}$

(iii) $L_3 = \{a^n b^m \mid n > 0\}$.

Ans

1. L_1 consists of all the string with Letter a followed by any number of b's

2. L_2 consists of all the string having equal number of a's and equal number b's such that all all the a's are followed by all the b's

3. L_3 consists of all the strings with any numbers of a's followed by atleast one b.

Automata.

↳ is a kind of Machine which takes some string as input. this I/p goes through a finite number of states and may enter in the final state.

Applications

* Automata Theory is a base for the formal language and these formal languages are useful of the programming language.

* Automata Theory plays in important role in compiler design

* Automata theory deals with the design FSM (Finite State Machine)

Finite State System.

Represent a mathematical model of a system. with certain I/p. the Model finally certain output.

The I/p when is given to the Machine it processed by various states. these states are called as intermediate state.

Definition

A finite Automata is a collection of 5 tuple $(Q, \Sigma, \delta, q_0, F)$

Q is a finite set of states, which is non-empty

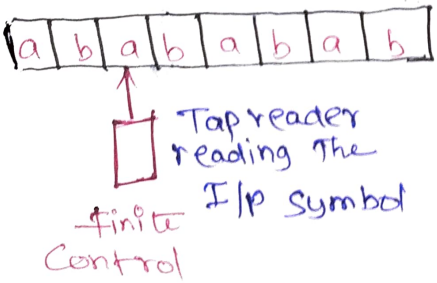
Σ is input alphabet, Indicates input set

q_0 is an initial state and q_0 is in Q (i.e) $q_0 \in Q$.

F is a set of final states

δ is a transition function (or) mapping function, using this function the next state can be determined.

Finite Automata Model



Input tape :- It is a linear tape having some number of cells. Each input symbol is placed in each cell.

Finite control :- Finite control decides the next state on receiving particular Input from input table.

The tape reader reads the cells one by one from Left to right at a time only one input symbol read.

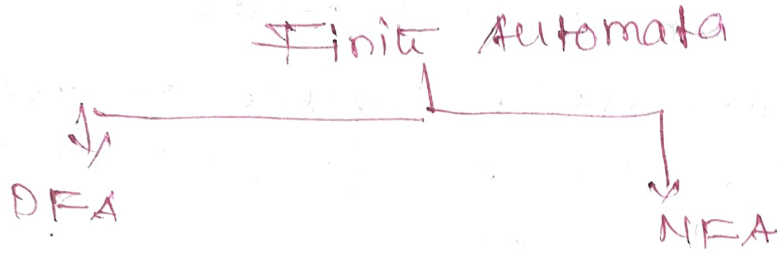
$(q, w) \vdash (q', w')$

M is accepted by Language L. The Acceptance of M by some language L is denoted by L(M)

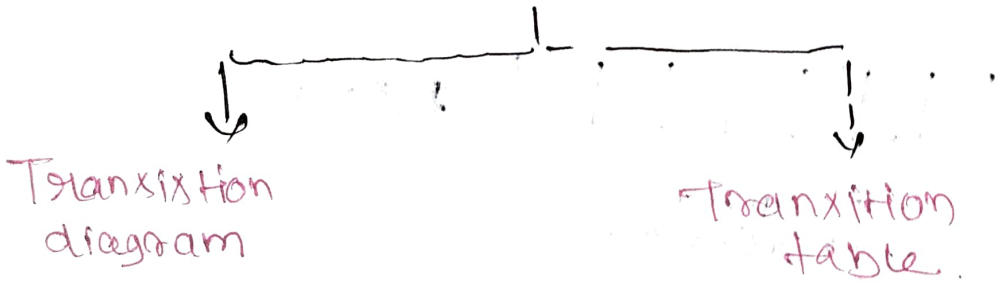
A Machine M accepts a Language L iff

$$L = L(M)$$

Types of Automata

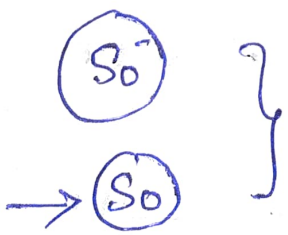


Acceptance of string language



Represents States

" transition from one State to another



Start States

Final States.

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

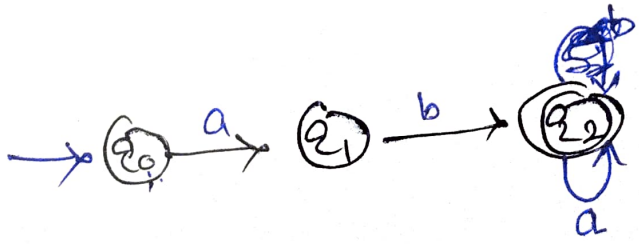
S_0 --> Initial State

S_4 --> Final States

S_1, S_2, S_3 all the states called as intermediate states.

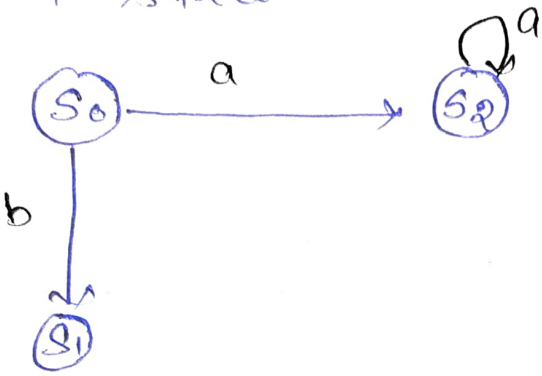
Transition diagram

I/P / States	a	b
q_0	q_1	-
q_1	-	q_2
q_2		q_2



DFA

if there is only one path for specific I/p from current state to next state.



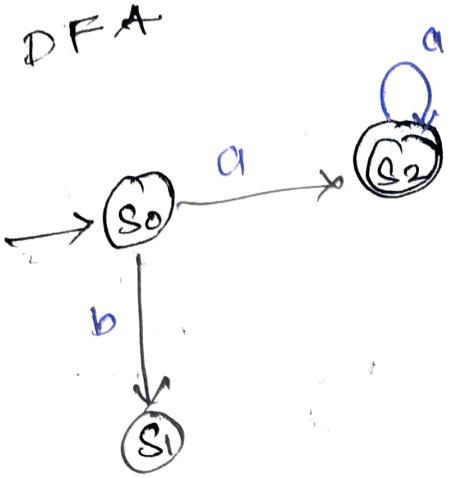
Definition DFA.

DFA is a collection of following things.

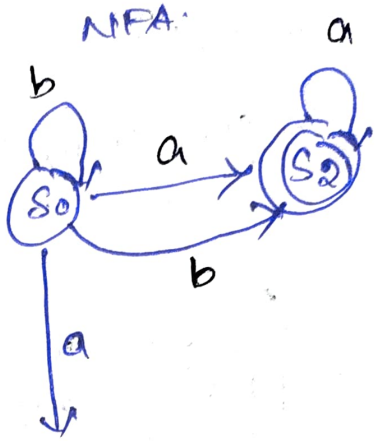
* The Finite set of states which can be denoted by Q.
* the Finite set of input symbol Σ
* the State state q_0 such that $q_0 \in Q$
* A set of Final States F such that $F \in Q$.
* the mapping function(or) transition Function denoted by δ

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

DFA



NFA



DFA Examples.

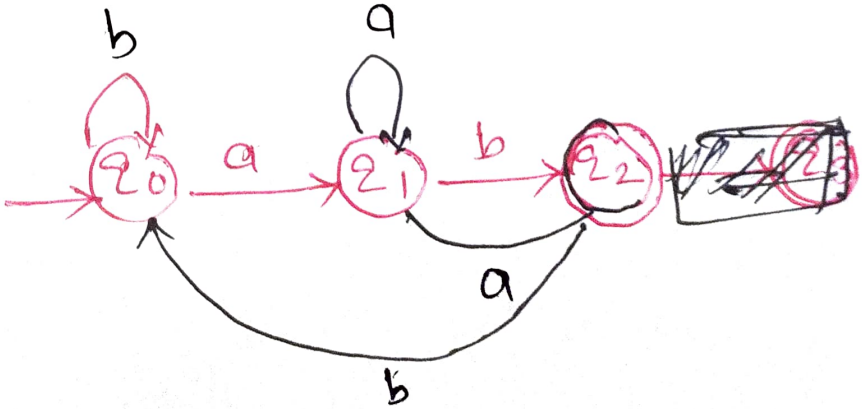
construct DFA that Accepts set of all Strings over $\{a,b\}$ of strings ends with "ab"

ends with "ou"

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

$L = \{ab, aab, bab, ababab\}$

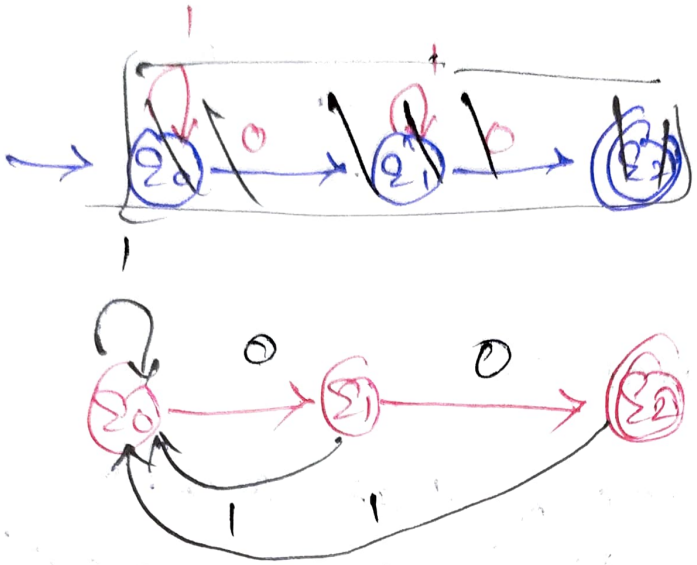
$\{bbabb\}$



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

$L = \{00, 000, 100, 0100, \}$

Why because we have to DFA means use all symbol present present in the alphapet on every State.



Ep: 10100

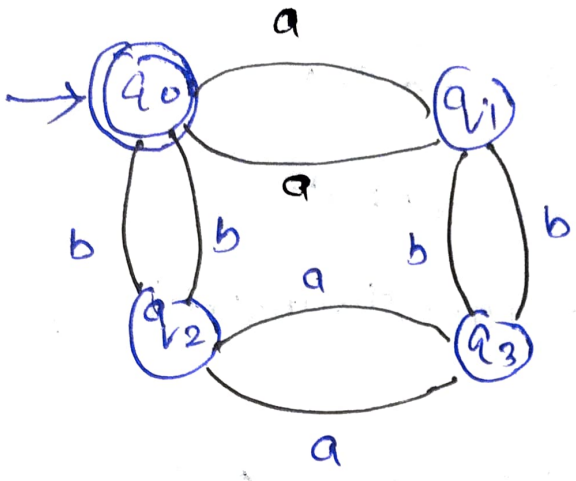
Q Construct DFA which Accepts set of String over $\{a, b\}$ with.

a) Even no. of a's and even 'b's

b) Even " " and odd of b's

c) odd " " and even of b's

d) odd " " and odd of b's



--> aabb ---> q_0

--> abab --> q_0

--> aab --> q_2

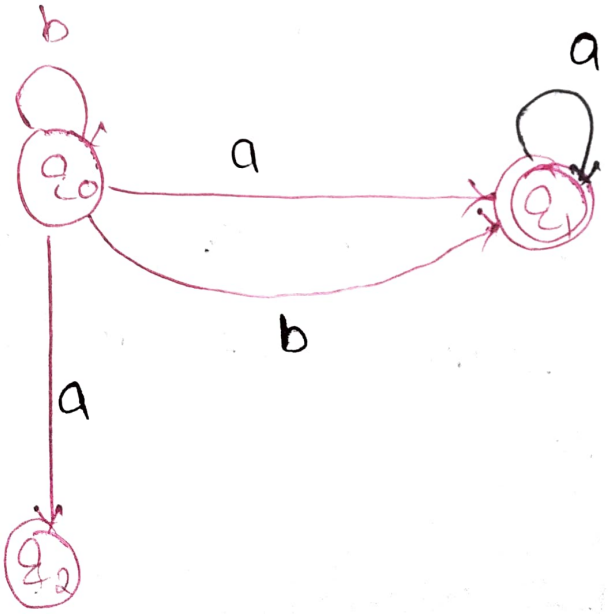
--> abb --> q_1

--> ab ---> q_3

NFA

--> NFA is Exactly reverse of DFA

L-> The FA is called NFA when there exits many path for a specific input from current state to next state.



Definition of NFA

Q is a Finite Set of States

Σ is a Finite set of Inputs

δ is a called next to (or) transition function

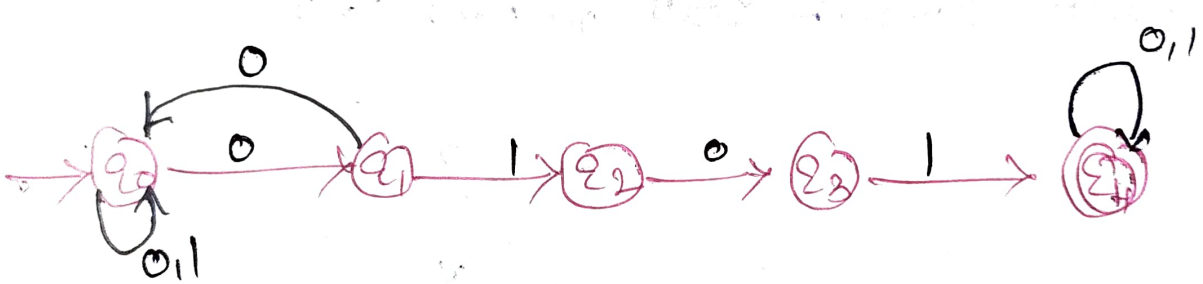
q_0 Initial state

F is a Final state where $F \subseteq Q$.

Examples. Construct a NFA for the language.

L_1 = { consisting a substring 0101 }

L_2 = $\{a^n b^n\}$



The string 00010101 is acceptable by above given NFA and it is shown below.

00010101

$\vdash q_0$0010101

$\vdash$ 0$q_0$010101

$\vdash$ 00$q_0$10101

$\vdash$ 000$q_1$0101

$\vdash$ 0001$q_2$101

$\vdash$ 00010$q_3$01

$\vdash$ 000101$q_4$1

$\vdash$ 00010101q_4

$0001q_3 \vdash 101$

$00101q_4 \vdash 01$

$000101q_4 \vdash 1$

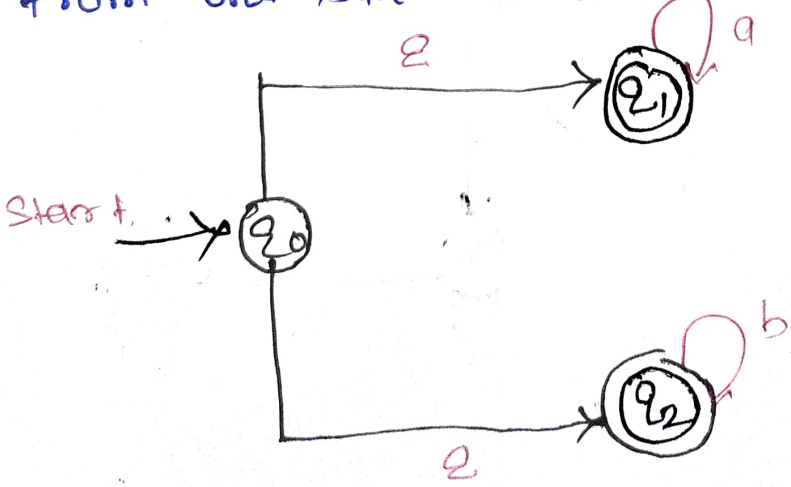
$0001010q_4 \vdash$ it's a Accepted as q_4 is final state.

(ii) $L_2 = \{a^n \cup b^n\}$

Now we will build a NFA for L_2. The Language L_2 it's a language in which there be any number of a's or any number of b's

it Accepts $\{a, b, aa, bb, aaa, bbb \ldots\}$

The NFA shows two different states q_1 and q_2 for the Σ/p ε from q_0 state (ε- pronounced as epsilon) it's a basically null move (ie) a moving carrying no symbol from Σ/p set Σ. But a state change occurs from one state to other



Ex: Design the NFA transition diagram for the transition table as given below.

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_3\}$	–
q_2	$\{q_2, q_3\}$	$\{q_3\}$
(q3)	$\{q_3\}$	$\{q_3\}$

Here the NFA is $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$

Sol

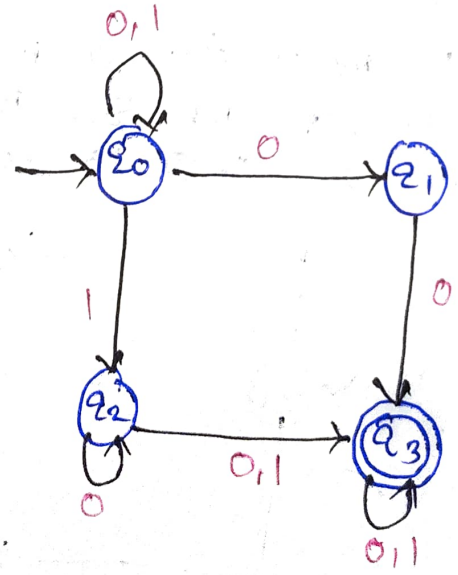
The transition diagram can be drawn by using the mapping function as given in table.

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

Then
$\delta(q_1, 0) = \{q_3\}$
$(q_1, 1) = \phi$

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

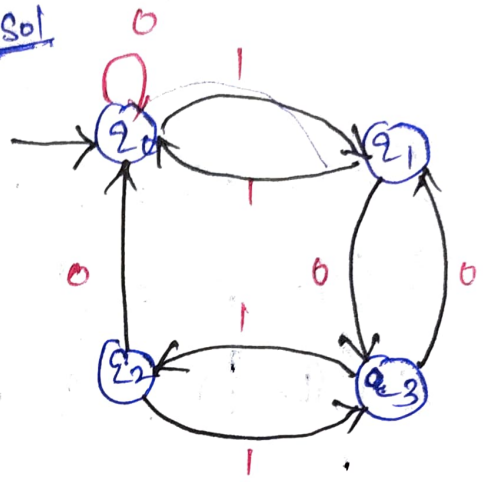
Then
$\delta(q_3, 0) = \{q_3\}$
$\delta(q_3, 1) = \{q_3\}$



Ex For FSM M given in the following table, test whether the strings 10110 1 11111 are accepted by M.

State	Input 0	Input 1
q_0	q_0	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Sol



(i) Let us string 10110

$q_0 \vdash 10110$

$1q_1 \vdash 0110$

$10 \vdash q_3 1101$

$101 \vdash q_2 101$

$1011 \vdash q_3 01$

$10110 \vdash q_1 1$

$10110 \vdash q_0$

is a final state.

(ii) Now consider string 11111

$q_0 \vdash 11111$

$1q_1 \vdash 1111$

$11q_0 \vdash 111$

$111q_1 \vdash 11$

$1111q_2 \vdash 1$

$11111q_3$ which is not final state

Q

Construct a transition diagram for the NDFA $M = (\{q_1, q_2, q_3\}, \delta, q_1, \{q_3\})$ where δ is given by

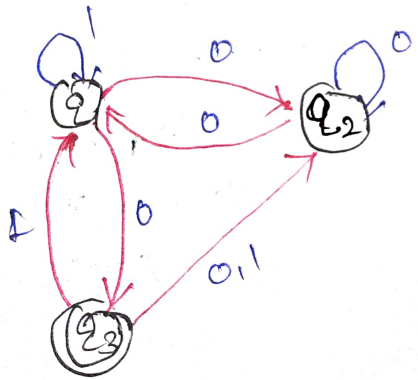
$\delta(q_1, 0) = \{q_2, q_3\}$ $\quad \delta(q_1, 1) = \{q_1\}$

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

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

Sol Construct Transition table.

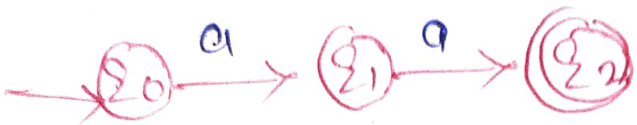
State	Input 0	Input 1
q_1	$\{q_2, q_3\}$	$\{q_1\}$
q_2	$\{q_1, q_2\}$	ϕ
(*) q_3	q_2	$\{q_1, q_2\}$



Design NFA to Accept Strings with a's and b's such that the strings end with aa

Sol

The simple FA with accepts a Strings with aa i's

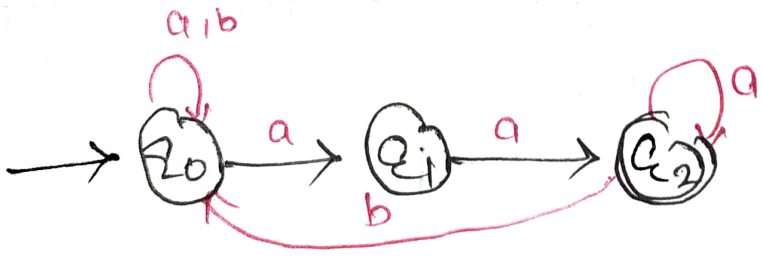


Now there can be a situation where in

Any thing either a or b

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

Initial, Final, Σ



consider

$\delta(q_0, aa) \vdash \delta(q_0, aa)$

$\vdash \delta(q_1, a)$

$\vdash \delta(q_2, \varepsilon)$

We reach the to Accept State

$\delta(q_0, aaa) \vdash \delta(q_0, aa)$

$\vdash \delta(q_0, a)$

$\vdash \delta(q_1, \varepsilon)$

are not Final in Final State.

Q: Construct a NFA in which double '1' is followed by double 0's over $\Sigma = \{0, 1\}$

Sol

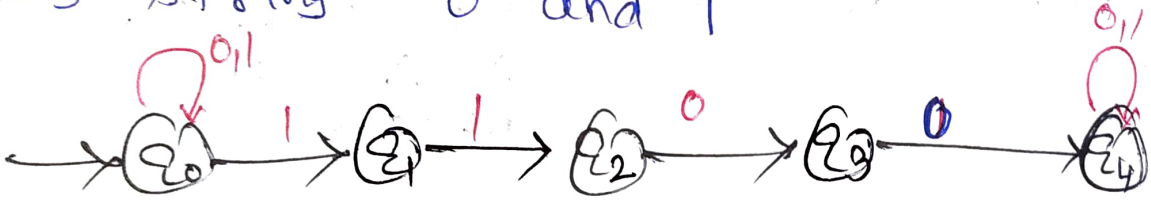
The NFA with double 1 is as drawn below:

It should be immediately followed by double 0

Then

Now before double 1 there can be any ~~string~~ 0 and 1.

Also after double 0 there can be any ~~string~~ 0 and 1



The transition table.

State	I/P 1	0
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	-
q_2	-	$\{q_3\}$
q_3	~~$\{q_4\}$~~	$\{q_4\}$
q_4	$\{q_4\}$	$\{q_4\}$

consider string 11100

$\delta(q_0, 11100) \vdash \delta(q_0, 1100)$

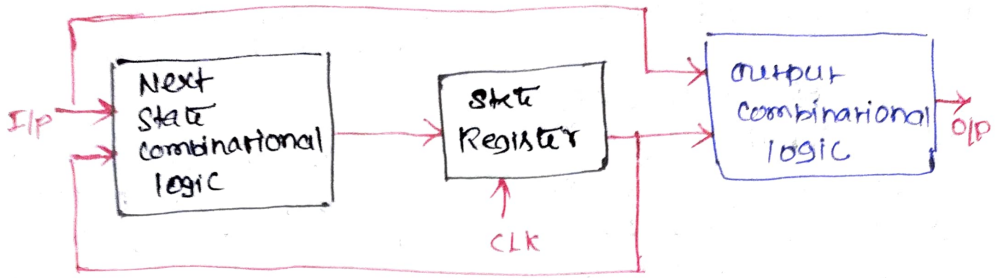
$\vdash \delta(q_1, 100)$

$\vdash \delta(q_2, 00)$

$\vdash \delta(q_3, 0)$

$\vdash \delta(q_4, \varepsilon)$ is a final state.

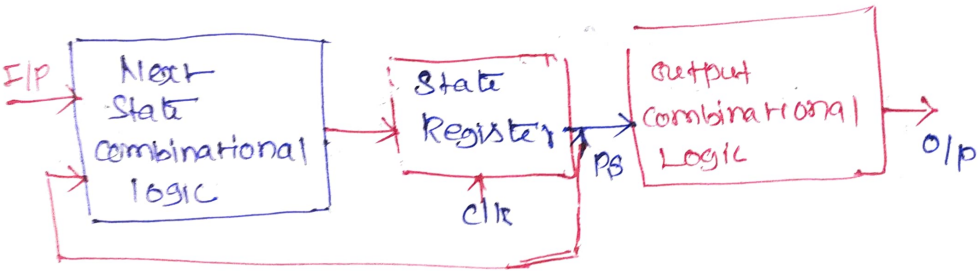
Mealy Model



Mealy Model

Depends upon the present state I/p state

Moore Model



Depends upon the present state Only

Mealy Model

1.) the output is a function of both present state and Input

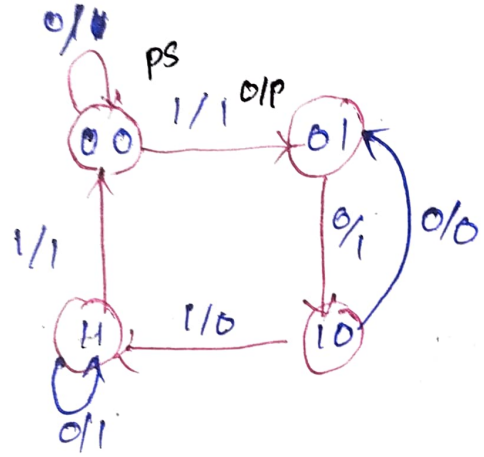
eg: $y(A+1B)\bar{x}$ (present state flip flop, $\bar{x}$ I/p)

1) the output is a function of present state only.

eg: $y = AB$

2. Both input and output value separated by a slash along the direct line

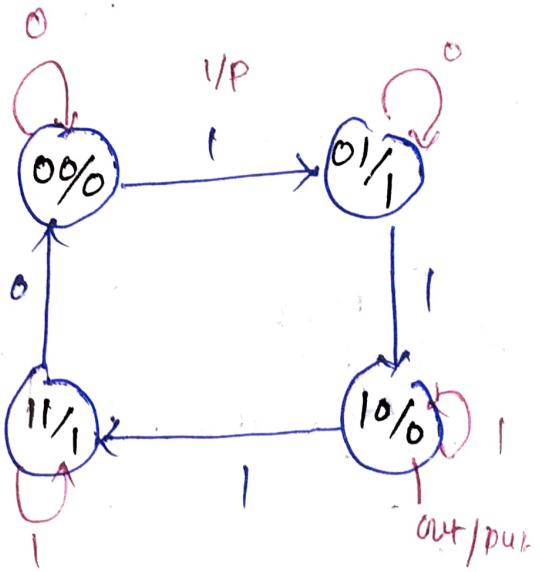
3. The output may change if the input change during the clock pulse Hence the output may have momentry false value because of time delay.



I/p change means output also chages possibliy high.

2. only inputs are marked along the directed lines the outputs are marked inside the circle.

3. The output are Synchroni -zed with clock pulse because they depend on only flip flop outputs.



I/p change but output change poxiblity low

Moore Machine

6 tuples ($Q, \Sigma, \Delta, \delta, \lambda, q_0$)

Q → Finite set of states

Σ → Input alphabet

Δ → output alphabet (delta big)

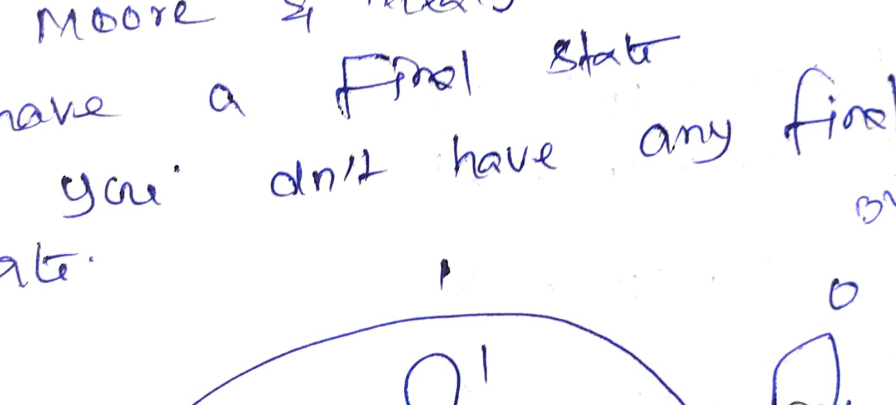
δ → transition function (or mapping function) $Q \times \Sigma \to Q$

λ → output functions $Q \to \Delta$

q_0 → Initial state

(A). The outputs always depends upon the present state.

* Moore & Mealy. doesn't have a final state you don't have any final state.



Current State	Next State		Output
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

Input string :- 0110 n

O/P " : 11111 n+1

$\delta(q_0, 0110) \rightarrow 1$

$\delta(q_0, 0110) \rightarrow 1$

$\delta(q_1, 110) \rightarrow 1$

$\delta(q_1, 10) \rightarrow 1$

$\delta(q_1, 0) \rightarrow 1$

$\delta \; q_2$

Mealy Machine

5 tuple $(Q, \Sigma, \Delta, \delta, q_0)$

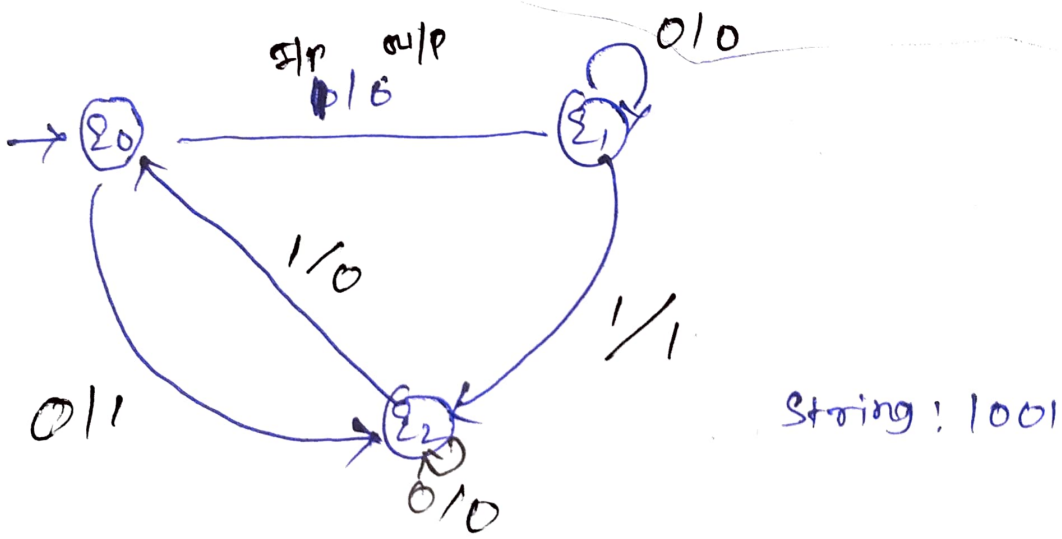
Q → Finite set of States

Σ → Input set Alphabet

Δ → output Alphabet

q_0 → Initial State

δ → Mapping function.



Current State	0		1	
	n.s	o/p	n.s	o/p
q_0	q_2	1	q_1	0
q_1	q_1	0	q_2	1
q_2	q_2	0	q_0	0

$\delta(q_0, 1001) \rightarrow 0$

$\delta(q_1, 001) \rightarrow 0$

$\delta(q_4, 01) \rightarrow 0$

$\delta(q_1, 1) \rightarrow 1$

$q_2 \rightarrow 0001$

Mealy Machine

1's complement

2's complement

Increment Binary number

1's complement

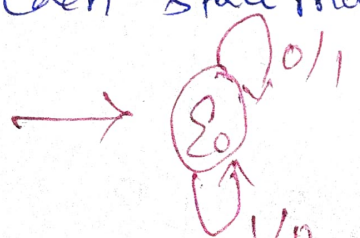
Ex:- 1001 --> I/p

0110 --> o/p

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

Mealy Machine only DFA.

This is DFA only ~~you~~ why because each state must use all the present in the alphabet.



In KT. in DFA have final state mealy we have any final state.

1001

0 1 1 0 ===> O/P

2.) 2's complement.

= 1s complement +1

Read the data from LSB

MSB 0 100100 LSB

1011011

1

1011100

10100

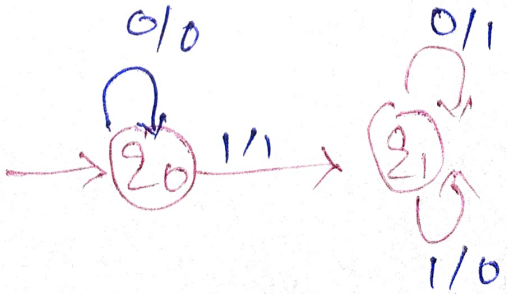
01011

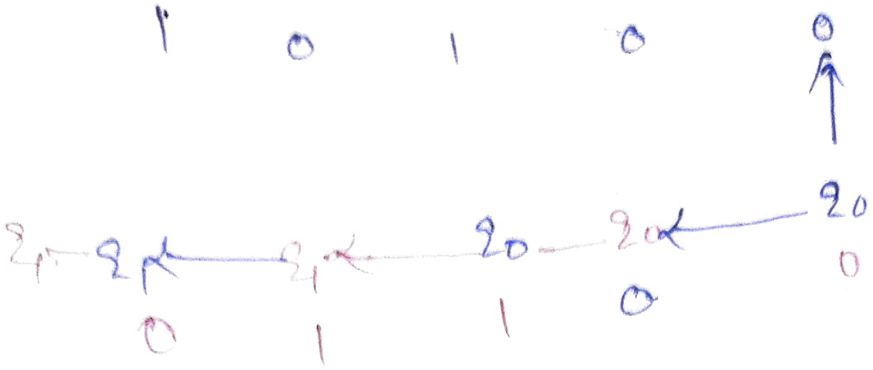
1

01100

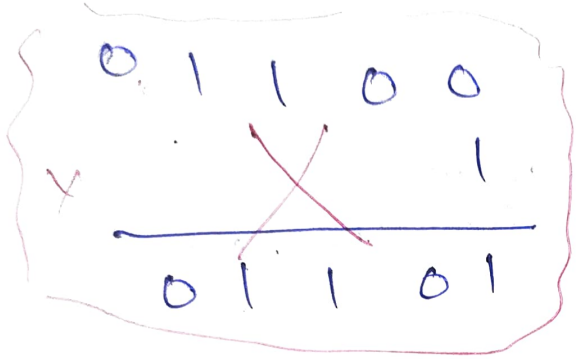
(i) upto one upto the first one keep the bits as its

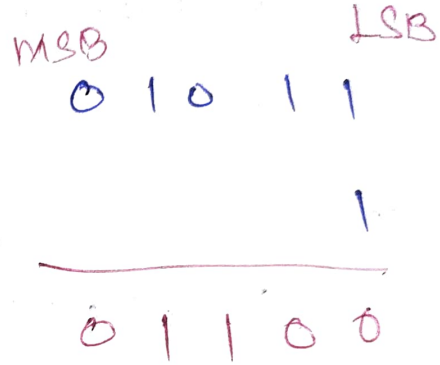
(ii) After the first one complement each in every one.

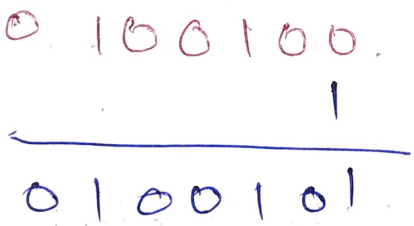


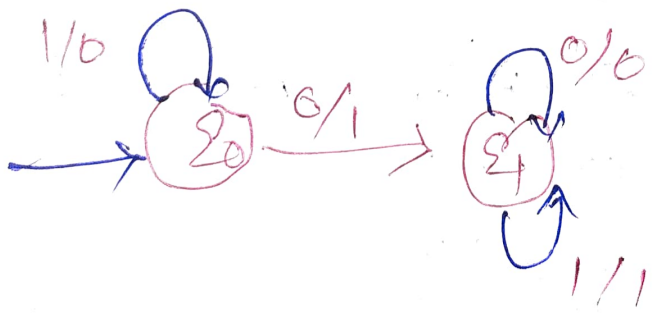


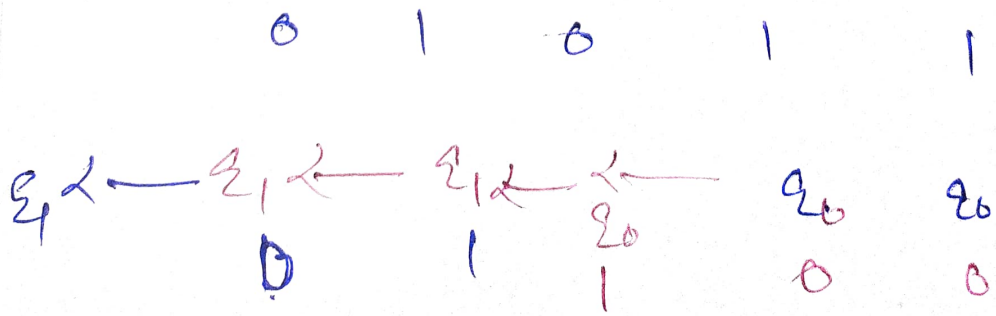
3. Increment given binary number by 1









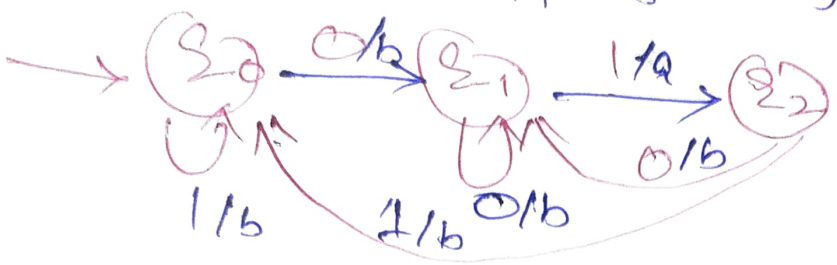


Mealy Machine

Construct a mealy machine that print 'a' whenever one sequence '01' encountered in any I/p string

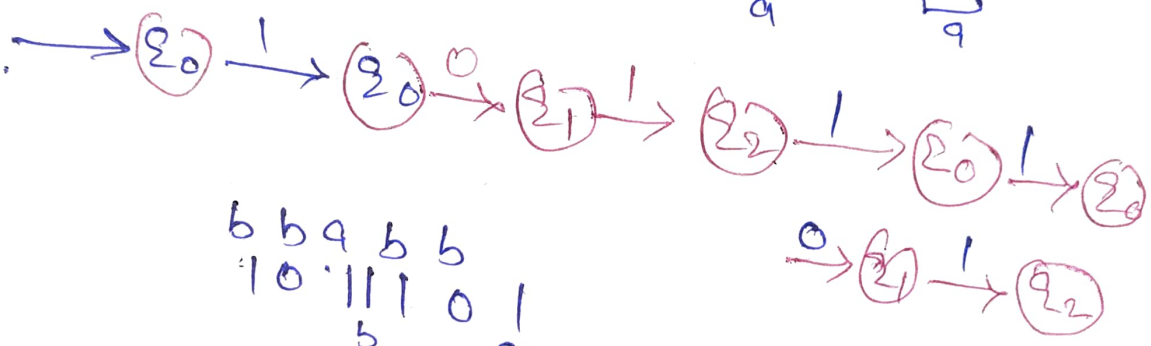
$\Sigma = \{0, 1\}$

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



1 0 1 1 0 1

a a



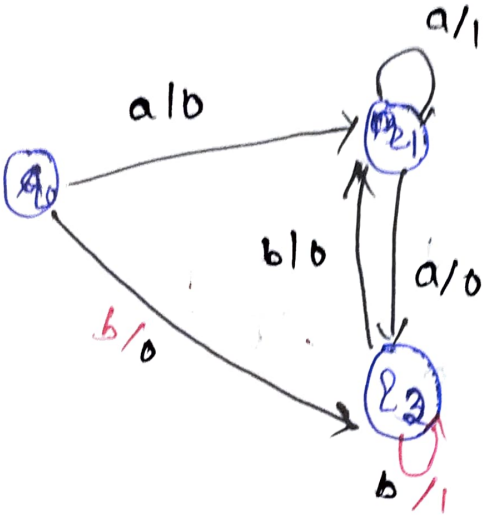
b b a b b

1 0 1 1 1 0 1

b a

Design a Mealy machine accepting the language consisting of strings from Σ^* where $\Sigma = \{a, b\}$ and the strings should end with either aa or bb.

aa - 1
bb - 1



a b b

b a a

b a	a a
0 0	0 1

2's complement

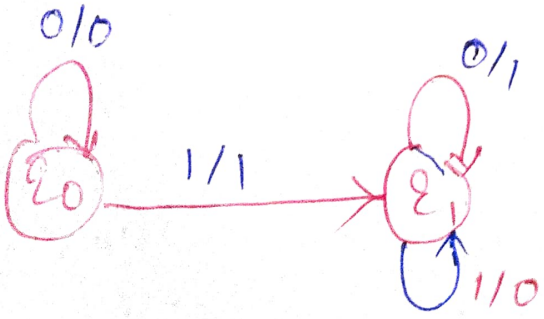
MSB LSB

1 0 1 0 0
0 1 0 1 1
+ 1
0 1 0 0 0

MSB LSB

1 1 1 0 0
0 0 0 1 1
+ 1
0 0 1 0 0

1 1 1 1
0 0 0 0
+ 1
0 0 0 1



1 0 1 0 0

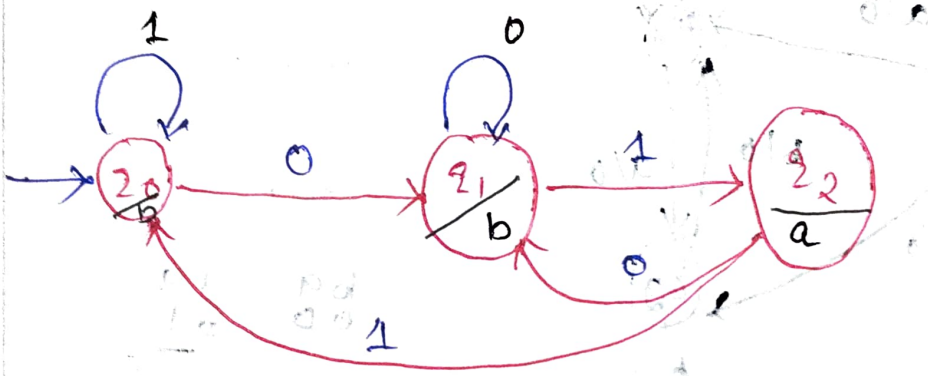
$\leftarrow q_1 \leftarrow q_1 \leftarrow q_1 \leftarrow q_0 \leftarrow q_0$

0 1 1 0 0

Moore Machine.

Q:- Construct a moore Machine that prints a whenever the sequence '01' is encountered in any I/p binary string.

$$\Sigma = \{0, 1\} \qquad \Delta = \{a, b\}$$



0	1	1	0
A q_0	q_0	q_2	q_0	q_1
↓				
q_0 b	a	b	b

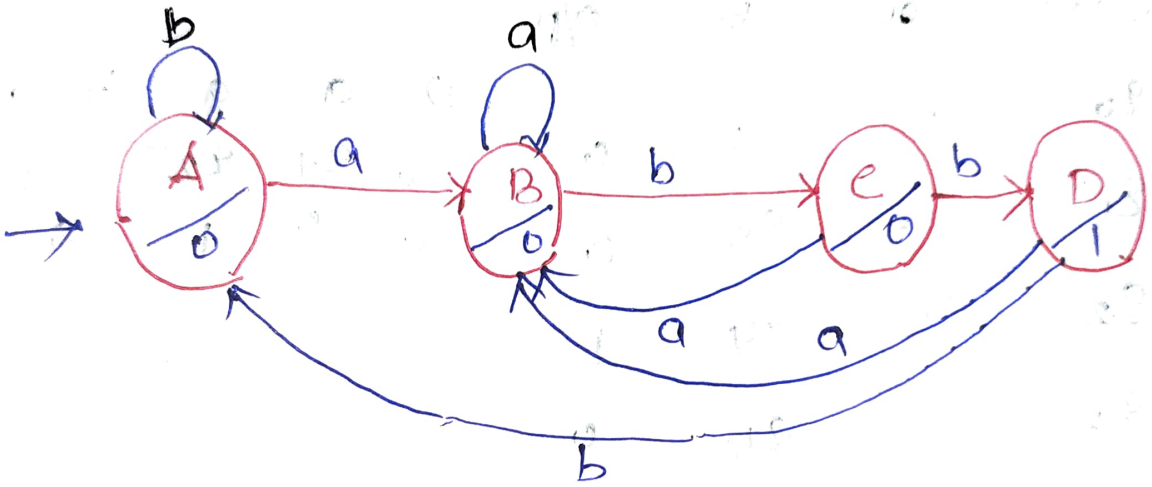
A	0	1	0	1
q_0	q_1	q_2	q_1	q_2
b	b	a	b	a

Moore Machine

Construct a moore Machine that count the occurences of the sequence 'abb' in any input string over {a,b}

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

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



	a	b	b
A	B	C	D
0	0	0	1

	a	b	b	a	b	b
A	B	e	D	B	C	D
0	0	0	1	0	0	1

Construction of moore Machine

For the following moore Machine the I/p alphabet is $\Sigma = \{a, b\}$ and the o/p alphabet is $\Delta = \{0, 1\}$

Run the following I/p sequence and find the respective outputs.

(i) aabaab (ii) aabb (iii) ababb.

States	a	b	o/p
→ q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

i) a a b a

q_0 q_1 q_2 q_4 q_2

0 0 1 0 1

(ii)

a b b b

q_0 q_1 q_3 q_4 q_0

0 0 0 0 0

(iii)

a b a b b

q_0 q_1 q_3 q_4 q_0 q_2

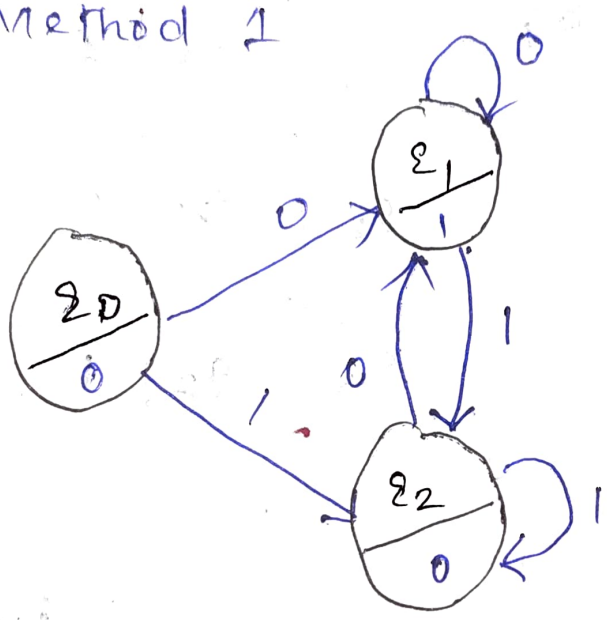
0 0 0 0 0 1

Design a moore Machine to find the 1's complement of a given binary number

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

1 0 1 1
0 1 0 0

Method 1



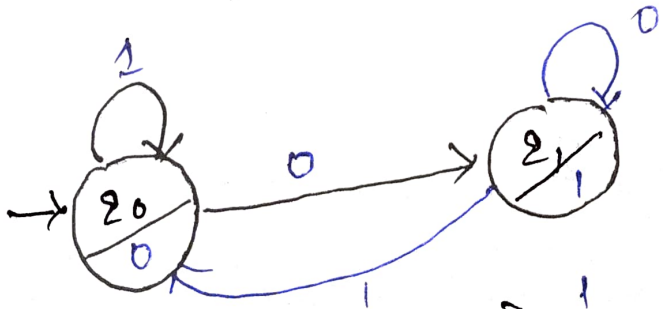
q_2
1

1 0 1 1 (n)

q_0 q_2 q_1 q_2 q_2

0 0 1 0 0 (n+1)

Method 2



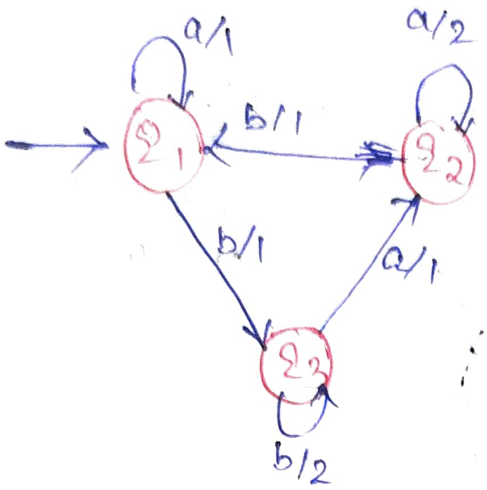
1 1 0 1 1

(n) ---->

q_0 q_0 q_1 q_0 q_0

negbit n+1 (0) 0 1 0 0

Q. Conversion of Mealy Machine to moore Machine



Current State	a		b	
	n.s	o/p	n.s	op
q_1	q_1	1	q_3	1
q_2	q_2	2	q_1	1
q_3	q_2	1	q_3	2

Moore Machine transition table

C.S	NS a	NS b	o/p
q_1	q_1	q_{31}	1
q_{21}	q_{22}	q_{1}	1
q_{22}	q_{22}	q_1	2
q_{31}	q_{21}	q_{22}	1
q_{32}	q_{2}	q_{32}	2

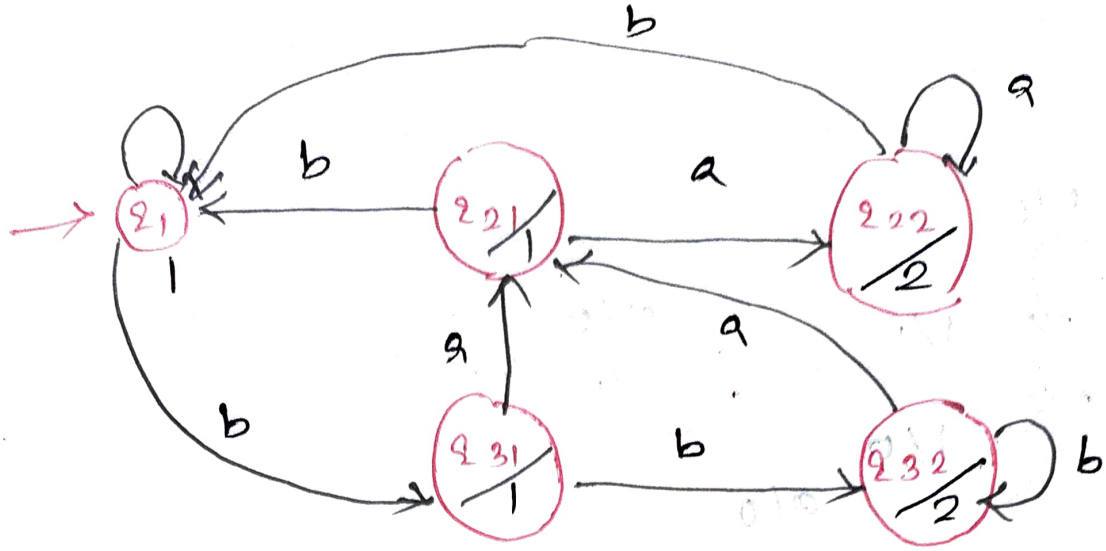
q_1 ---> 1

q_2 ---> 2 ---> 22

q_2 ---> 1 ---> 21

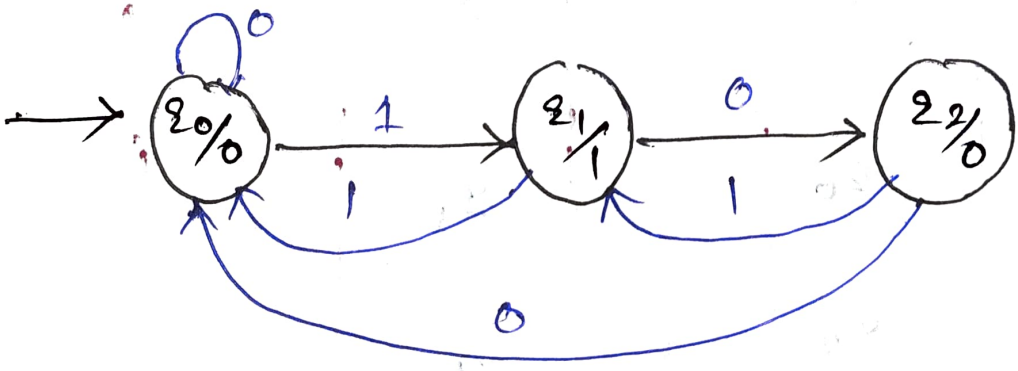
q_3 ---> 1 ---> 31

q_3 ---> 2 ---> 32

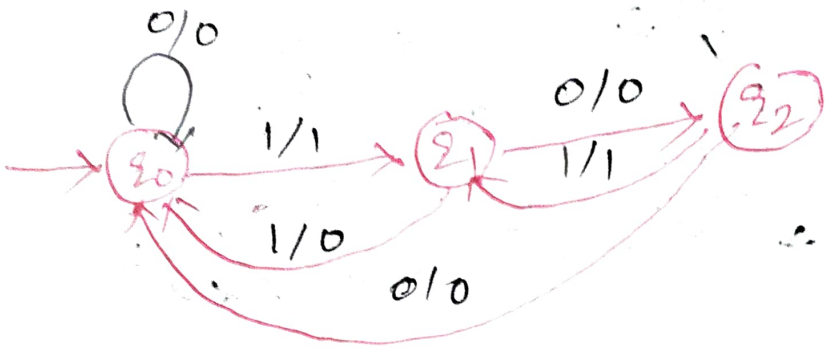


Moore to Mealy.

Conversion of moore Machine to mealy Machine



current State	Next State		output
	0	1	
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

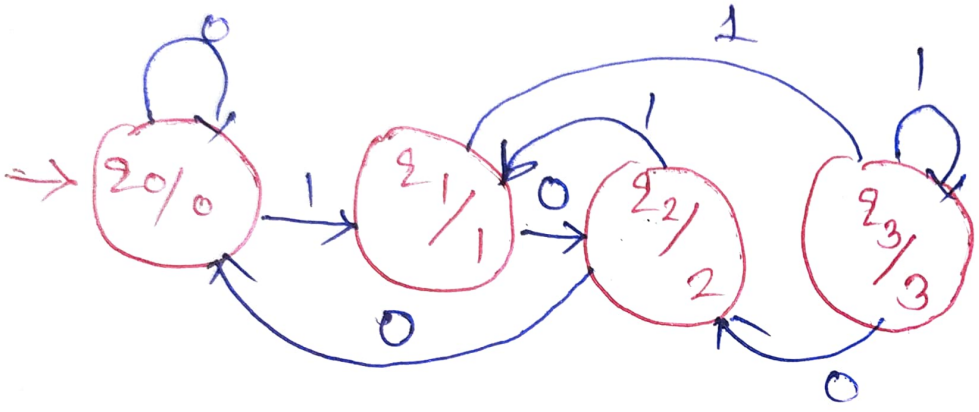


Mealy Machine transition table.

Current State	NS	O	NS	1
q_0	q_0	0	q_1	1
q_1	q_2	0	q_0	0
q_2	q_0	0	q_1	1

Design a moore Machine to determine residue module 4 For Binary Number

$\Sigma = \{0,1\}$ $\Delta = \{0,1,2,3\}$



8	4	2	1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10

15 1 1 1 1

q_0 q_1 q_3 q_3 .

$q_3/3$

1 0 1 0

q_0 q_1 q_2 q_1 $q_2/2$