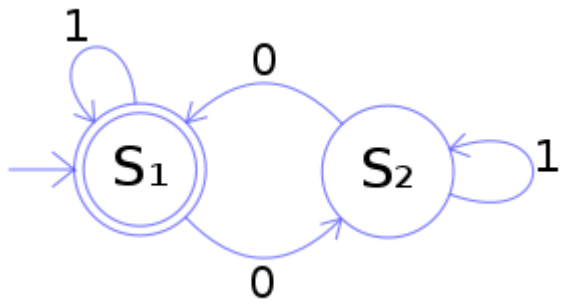
5Th Semester(COMPUTER SCIENCE AND ENGG .GCEKJR)

Formal Languages and Automata Theory.

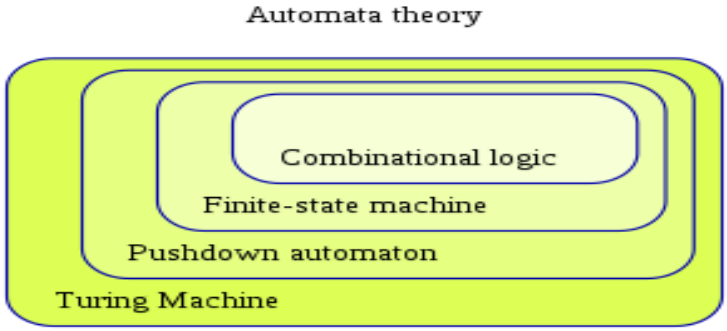
Introduction -:Automata theory is a study of abstract machine , automat and a theoretical way solve computational problem using this abstract machine .It is the theoretical computer science .The word automata plural of automaton comes from Greek word ...which means “self making”.

This automaton consists of states (represented in the figure by circles) and transitions (represented by arrows). As the automaton sees a symbol of input, it makes a transition (or jump) to another state, according to its transition function , which takes the current state and the recent symbol as its inputs.



Automata theory is closely related to formal language theory. A formal language consist of word whose latter are taken from an alphabet and are well formed according to specific set of rule . so we can say An automaton is a finite representation of a formal language that may be an infinite set.

Automata are often classified by the class of formal languages they can recognize, typically illustrated by the CHOMSKY HIERARCHY which describes the relations between various languages and kinds of formalized logics.



Following are the few automata over formal language.

Automaton	Recognizable Language.
Nondetermistic /Deterministic Finate state Machine(FSM)	Regular language.
Deterministic push down automaton(DPDA)	Deterministic context free language.
Pushdown automaton(PDA)	Context_free language.
Linear bounded automata(LBA)	Context _sensitive language.
Turing machine	Recursively enumerable language.

Why Study of automata theory is important?

Because Automata play a major role in theory of computation, compiler construction, artificial intelligence , parsing and formal verification.

Alphabets, Strings and Languages; Automata and Grammars-:

Symbols and Alphabet:

Symbol-: is an abstract user defined entity for example if we say pi(π) its value is 1.414..

A simple example could be -: letter , digits.

Where as

An **Alphabet** -: is a finite , nonempty set of symbols denoted by $\sum$ in theory of computation.

Example1-:$\sum$={0,1}

Example2-:$\sum$={a,b,c....z}

String and Languages-:

String-: Finite sequence of symbols (Symbols could be chosen from alphabet) and denoted by symbol w (depends on writer)

Example-: $\sum$={0,1}

Then 0101001 , 010, 00, 11, 111 ,0, 0011,..... etc are the string we can from by sequence of symbols.

Then a,b,ab,aa,bb,aab,abb,....etc are the string we can write by sequence of symbol.

Empty String-:Symbol 'Ɛ' Greek small letter epsilon is used to denote empty string and that is the string consisting of zero symbol. It is also called as null string .
Length of string-: is the total no of symbol present in the string .

Example1 -: $\sum=\{0,1\}$

w=01010

Then length of string is denoted by |w| and here it is 5 so we can write

|w|=5

Example2-: w =ε then the length of w is given by
|w|=0
Prefix-: prefix of a string is any number of leading symbol of string.
Example-: let **w**=abc
Prefix of **w** are ε,a,ab,abc.
Suffix-: is the any number of trailing symbol of string.
Example **w=**abc
The suffix are ε,c,bc,abc
Substring-: Any sequence of symbol over the given string.
Example-: **w=abc**
The substring of **w** are ε,a,b,c,bc,ab,abc.
Power of an alphabet-The string that can be formed by taking number of symbol from given alphabet ∑ , no of symbol we can take will be given by the power.
Example-: $\sum=\{0,1\}$
Then $\sum^0=\{\varepsilon\}$
$\sum^1=\{0,1\}$
$\sum^2=\{00,01,10,11\}$
$\sum^3=\{000,001,011,....,111\}$
$\sum^*=\sum^0 \cup \sum^1 \cup \sum^2 \cup \sum^3.....$
So $\sum^*=\{\varepsilon,0,1,00,01,11,10,000,001.....\}$
$\sum^+$= All the string except non-empty string
So
$\sum^+= \sum^1 \cup \sum^2 \cup \sum^3.....$
And then
$\sum^+=\{0,1,00,01,11,10,000,001.....\}$

Concatenation of String-:
if w and x are two string then the concatenation of two string is denoted by wx.
Example-: w=1110 and x=0101
Then xy=11100101
Language-: It is the set of string over the alphabet ∑ which follows a set of rules called the grammar.L
Or otherwise we can say set of string generated by grammar taking symbol from ∑ .
Example 1-:All string start with 11 from ∑={0,1}
L={110,111,1100,1111,1101,1110,.....}
Example 2-:set of binary number whose value is prime.
L={10,11,101,111,1011,.....}
Note-: L={ϕ} is called empty language.
And L={ε} Language consisting of only the empty string.
Natural Language-: the language we use as medium of communication.
Formal Lanugage-: collection of string where format (given by grammar)is important and meaning is not important.

Language of machine -:Set of all accepted string of the machine
Grammar-: A set of rules used to generate the string for a particular language.

Question and Answers

Q1. Define Theory of computation-:
Ans-:Mathematical representation of computing machine and its capabilities.
Q2.Automata is a recognizing device or generating device?
Ans-: Recognizing device.
Q3. Grammar is a recognizing device of generating device?
Ans-: Generating .
Q4. Difference between language and grammar
Ans-: please follow the above notes.
Q5.Difference between natural language and formal language?
string governed by grammar and meaning is not important
.Recognized by Machine .
Q6.Find the number of prefix , suffix, and substring for a given string of length n.
Ans-: Number of prefix=n+1
Number of suffix=n+1
Number of substring =(n(n+1))/2 + 1.

Short answer type question.

Q1. What do you mean by power of alphabet.

Q2. Define formal language explain with example.

Deterministic finite Automata (DFA)

INTRODUCTION-:

Finite state machine (FSM) or Finite Automaton-: is a mathematical model of a machine ,which can only reach to finite number of states by transition after reading one symbol at a time .Finite automata are computing device that accept or recognize regular language (A formal language follows rules).It is used to model operation of many system encountered in practices.

Definition of finite automata-: A finite automata can be defined by a 5 tuple machine.

Machine $M= (Q,\sum, \delta,q_0,F)$

Where

$Q\rightarrow$ is the finite set of states.

$\sum\rightarrow$ is the set of input symbol / alphabet.

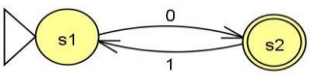
$\delta\rightarrow$is the mapping function which maps $Q \times \sum \rightarrow Q$

$q_0\rightarrow$is the initial state which must be element of $\sum$

$F\rightarrow$ Set of final states must be subset of Q

Transition system-:Is a directed labelled graph in which each vertex or node represent a state and directed edges represent the transition of a state and each edge labelled with input. A special kind state called final state are represented by double concentric circle and all other state are called non final state. One of the non final state is considered as starting state.

Example-:



$M=(\ \{s_1,s_2\}\ ,\ \{0,1\}\ ,\ \delta\ ,\ s_1\ ,\ \{s_2\}\ )$

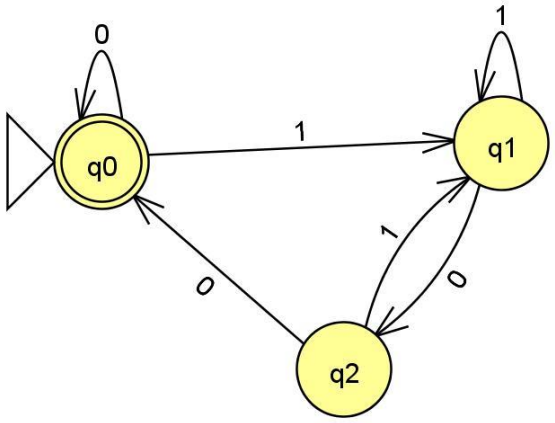
Transition function δ defined as follows

$\delta(s_1,0)=\{\ s_2\}$

$\delta(s_2,0)=\{ s_1\}$
it can also be represented by transition table. Note in the transition table arrow mark is used to indicate starting state and * symbol is used to indicate final state and each row indicate the transition of state by reading particular symbol which is given by transition function. As shown below.

States ↓ / input→	0	1
→s_1	s_2	ϕ
*s_2	ϕ	s_1

Example 2-: Give the 5 tuple representation and draw the transition table for the following fig.



DFA

Finite automata is devided in to two categories.
1→DFA (Deterministic Finite Automata.
2→NFA (Non Deterministic finite Automata.
Formal Definition-: DFA-:Formally defined using 5 tuple notation with following restriction
Each and every state for every input symbol , there should be exactly one transition.
DFA is the special case of NFA but the reverse is not true.
Simplified notation.
Machine $M= (Q,\sum, \delta,q_0,F)$
Where
Q→ is the finite set of states.
∑→ is the set of input symbol / alphabet.
δ→is the mapping function which maps $Q \times \sum \rightarrow Q$
q_0→is the initial state which must be element of ∑
F→ Set of final states must be subset of Q

Design of DFA-: In order to design the DFA first of all we need to understand the language(Set of string) accepted by DFA.

→Language accepted by DFA can be of the following type

1. Starting point restriction.

Ex-:All string start with '01'

2. End point restriction .

Ex-:All string end with '01'

3. Sub string restriction

Ex-:All string where 101 is the substring.

4. Other

Ex-: String divisible by 4.

→The second step towards solution is to identify the minimum length string that is accepted by language and accordingly identify the minimum state required and draw the diagram . the diagram will indicate initial and final state.

→ The third step is ,for each state , decide the transition to be made for each symbol as input.

→If transition is not valid then insert a new state called dead state or trap state and transit to this state.

Example-:Design a DFA which accepts the language of all string start with 01 over input symbol {0,1}

Sol-:

Step1-:First step is to identify the language and that is

L={01,011,010,0100,0110,0111,0101,01111....}

It is easily observable that all string in the language start with ' 01'

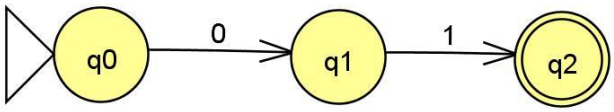
Sept-2

It is also clear from the above statement that minimum string accepted by the DFA is '01' the length of the string is 2

Note-: if minimum length of string with starting point restriction is 'n' Then we need n+2 number of state including trap state.

Since length of minimum string length is 2 we can conclude that minimum no of state required is 2+2(four).Then draw the basic diagram to accept

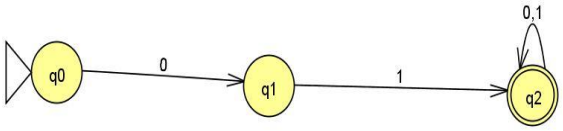
minimum string as shown below.



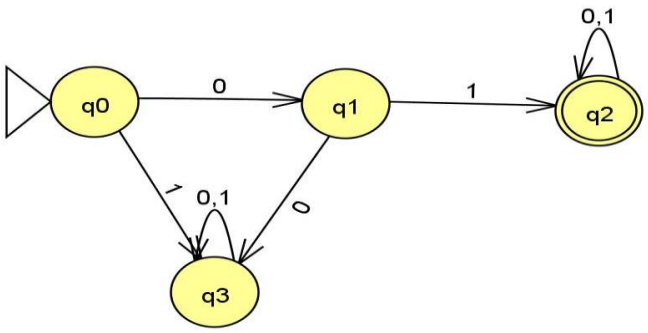
Initial state /starting state here is q_0 and final state is q_2

Third step -:complete the figure by completing other transition over input symbol.
For example q_0 has one remaining transition over input symbol '1'
q_1 has one remaining transition over input symbol '0'
q_2 has two remaining transition over input symbol '0' and '1'

we cannot draw the transition for q_0 and q_1 because the language has starting point restriction so the initial state will not able to accept any other symbol other than '0' .



step4-: q_0 has one remaining transition over input symbol '1' which violets the language rule that is if it accepts input symbol '1' the pattern '01' no longer valid .
q_1 has one remaining transition over input symbol '0'wich violets the language rule. As discuss above
so we have to introduce new state called trap state. And the final figure is -:



Draw the transition table related to transition function δ for above fig.

States ↓ / input→	0	1
→q_0	q_1	q_3
q_1	q_3	q_2
*q_2	q_2	q_2
q_3	q_3	q_3

Try to solve the following question .(The solution will be provided later)

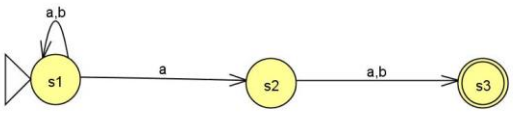
1. Construct a DFA that accept all string end with '01' over input alphabet {0,1}.
2. Construct a DFA that accept all string which has a substring '101'
3. Construct a DFA that accept all string which has a substring '001'
4. Construct a DFA that accept all string end with 'ab' over input alphabet {a,b}.
5. Construct a minimum DFA that accept all 'a's and 'b's where each start with 'a'
6. Construct a minimum DFA that accept all 'a's and 'b's where each start with 'b' and end with 'a'
7. Construct a minimum DFA that accept all 'a's and 'b's where length of string is exactly Three(3).
8. Construct a minimum DFA that accept all 'a's and 'b's where length of string is at least Three(3).
9. Construct a minimum DFA that accept all 'a's and 'b's where length of string is divisible by 2.

10. Construct a minimum DFA that accept all ‘a’s and ‘b’s where length of string is divisible by 5
11. Construct a minmum DFA the accept all ‘a’s and ‘b’s where length of the string is congruent to 2 mod 3.
12. Construct a minimum DFA that accept all binary number which are divisible by 2
13. Construct a minimum DFA that accept all binary number which are divisible by 5
14. Construct a minimum DFA that accept all string of ‘a’s and ‘b’s where 1st and last symbol are same
15. Construct a minimum DFA that accept all string of ‘a’s and ‘b’s where 1st and last symbol are different.
16. Construct a minimum DFA that accept all string of ‘a’s and ‘b’s where Second symbol from right hand side is ‘a’.

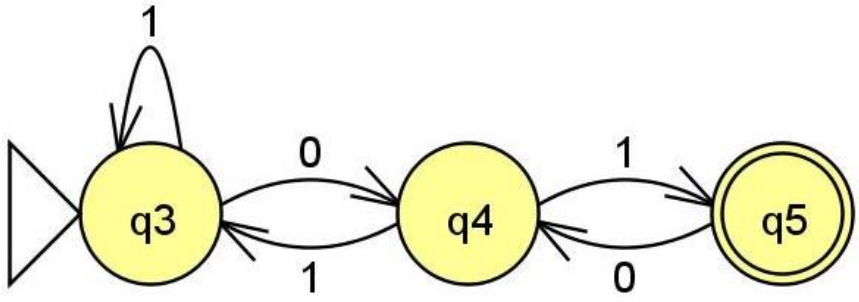
Nondeterministic finite Automata (NFA)

It is know as nondeterministic because several choice exist for a particular state for transit to different state after consuming single input symbol.
In simple one input symbol , more than one transition can reach to more than one state.
It is not compulsory that all the state have to consume all symbol in $\sum$.

Formal definition of NFA-:-: NFA-:Formally defined using 5 tuple notation.
Simplified notation.
Machine M= $(Q,\sum, \delta, q_0, F)$
Where
Q→ is the finite set of states.
$\sum$→ is the set of input symbol / alphabet.
δ→is the mapping function which maps $Q \times \sum \rightarrow 2^Q$
q_0→is the initial state which must be element of $\sum$
F→ Set of final states must be subset of Q(More than one final state is possible)
Example-:

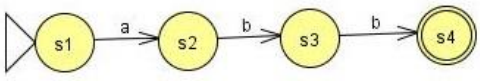


From the above figure it is clear that accepting single symbol ‘a’ state s_1 can move to two different state that is s_1 and s_2.
State s_2. can accept two symbol and there is no transition for state s_3.
Design of NFA-:
Basic design strategy of NFA is as follows.
1→Identify the language for the given problem.
2→Determine the alphabet and no of state required .
3→Draw the transition diagram which have initial, accepting state for the minimum string.
→Draw the transition table for NFA.
Example-:Design an NFA that accept a set of all string ending in 01 over the alphabet set {0,1}
Setp1--: L(M)={01,101,001,1101,1001,0101,0011,.....}
Setp2→∑={0,1} and minimum string is 01 so at least we need 3 state
(length of minimum string is 2 and in NFA we need n+1 states where n is the length of minimum string.
So we need 3 state.
Setp3→Draw the diagram.

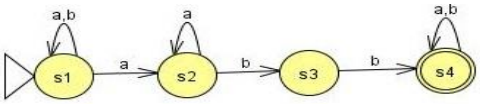


Note-:When we construct NFA , only valid combination are considered over ∑.
Example 2-: construct a minimum NFA that accepts all string of ‘a’ s and ‘b’ s where each string contains ‘abb’ as substring.
Sol-: It is clear that minimum string that will be accepted by machine is abb

So first of all we construct machine to accept ‘abb’ and then we will complete the diagram by all other transition.



Then complete the fig but remember that in NFA for a given input we can go to any no of states .Max state is 2^Q. The rule for the language should not be violated.



The transition Table for the above NFA is as follows.

States ↓ / input→	a	b
→S_1	S_1,S_2	S_1
S_2	S_2	S_3

S_3	-----	S_4
$*S_4$	S_4	S_4

Equivalence Between NFA and DFA.

We know that every DFA is a NFA but vice versa is not true.
Which means it is possible to draw the DFA which is equivalent to NFA.
The process is called subset construction.

Example1-: Construct a deterministic automaton equivalent to NFA
M=($\{q_0,q_1\}$, {0,1} , δ , q_0 , $\{q_0\}$)
δ transition function is given as follows.

State ↓ / ∑→	**0**	**1**
→*q_0	**q_0**	**q_1**
q_1	**q_1**	**q_0,q_1**

Sol-:

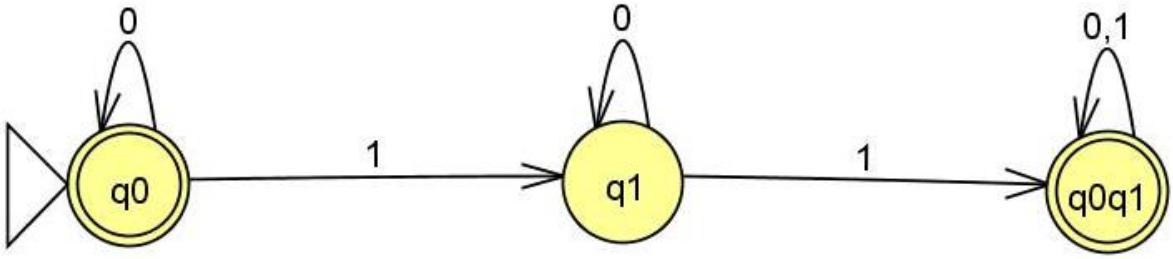
i. Initial state remains same for DFA and NFA so **q_0** is the initial state.

ii. In DFA we cannot transit to two different state accepting input symbol so we have to consider all possible state that is subset of **$\{q_0,q_1\}$**
That is -: ϕ,[q_0],[q_1],[q_0q_1].
Note-: **[q_0q_1] is a single state in case of DFA.**

iii. In case of NFA **q_0** is the final state hence in DFA all the state which has **q_0** will be the final state . so [**q_0**] and **[q_0q_1]** are the two final state for equivalent DFA .

iv. δ transition function for old state remains the same but for new state it is calculate as follows
δ([**q_0,q_1**],0)=δ(**q_0,0)U** δ(**q_1,0)= q_0q_1**
δ([**q_0,q_1**],1)=δ(**q_0,1)U** δ(**q_1,1)= q_1 U q_0q_1= q_0q_1**
Now construct the transition table and diagram.

State ↓ / ∑→	**0**	**1**
→*q_0	**q_0**	**q_1**
q_1	**q_1**	**q_0,q_1**
q_0,q_1	**q_0,q_1**	**q_0,q_1**



NFA with ε Transition.

In some situation , it would be useful to enhance the NFA by allowing transition that are not triggered by any input symbol , such transition are called ε-Transition.

Formal Definition-:NFA with ε move is denoted using 5 tuple

Machine M= $(Q,\sum, \delta,q_0,F)$

Where

Q→ is the set of states.

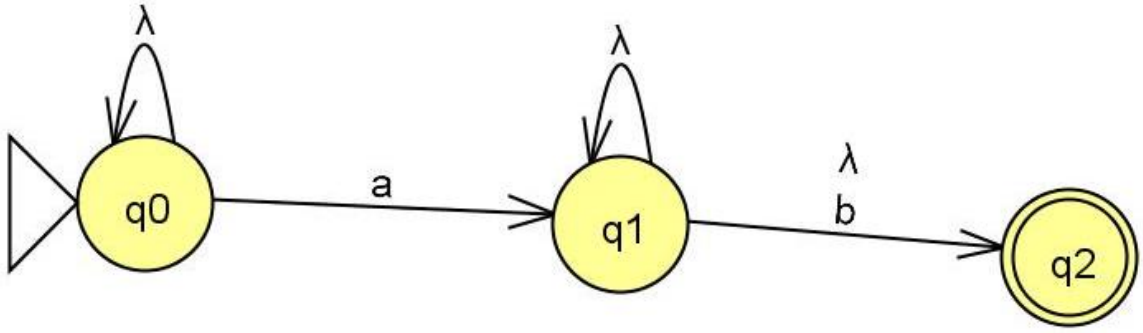
∑→ is the set of input symbol / alphabet.

δ→is the mapping function which maps $Q \times (\sum U\{\varepsilon\}) \rightarrow 2^Q$

q_0→is the initial state

F→ Set of final states must be subset of 2^Q(More than one final state is possible)

Example-: ε are replaced with λ in the figure.



Equivalence of NFA and DFA.

Date :- 22/9/2020.

Minimization of finite Automata.

Minimization of DFA is required to find a Minimal version of DFA which Consist of minimum no of states.

DFA given | n states → Minimization → Resultant DFA with minimum no states But Same function

Inefficient

Many DFA possible to a given problem

Minimization →

Efficient

Unique and only one with minimum states.

In order to find Minimum DFA we first we have to understand the concept of equivalent Two states ~~A and B~~ q_i and q_j are said to be equivalent if.

$\hat{\delta}(q_i, X) \rightarrow F$ Then $\hat{\delta}(q_j, X) \rightarrow F$

Note :- Since DFA Contains more than one final states Then Final states reached by q_i and q_j (By processing string) may be same or different

O/R.

$\hat{\delta}(q_i, X) \not\rightarrow F$ Then $\hat{\delta}(q_j, X) \not\rightarrow F$

Nonfinal states belongs to Particular class or group.

$X \rightarrow$ is input string may Consist of one more Symbols.

if $|x|=0$, Then q_i and q_j are said to be zero, '0' equivalent.

if $|x|=1$, Then 1 equivalent.
if $|x|=2$, " 2 " "

where $|x|$ = Length of given string.

Zero equivalent is denoted by π_0
one " " " π_1
and so on.

Steps to minimize the DFA:

→ Form a given DFA if a state is not reachable from starting state then remove it from DFA.
Removal includes the states and its transition.

→ Draw the Transition table related to the DFA.

→ Find the π_0 equivalent that is devide the states in to two groups consisting of final states (Productive States) and non final states (Non productive states)
Remember:-
Non productive states Such as unreachable one already removed.
Non productive states Such as Dead states more then one need to Combined.

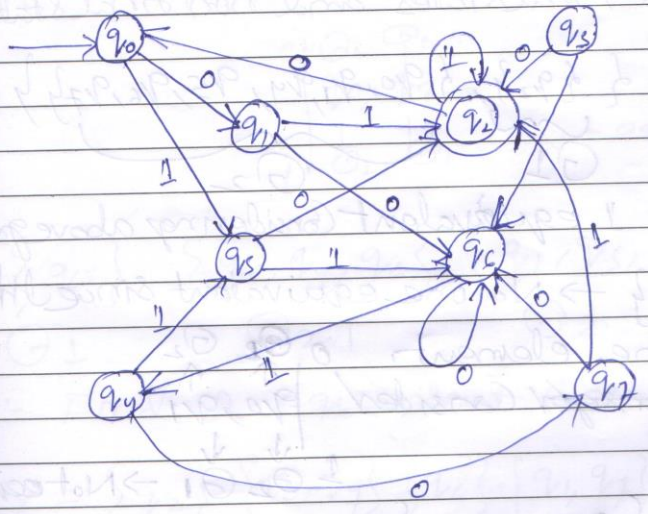
→ Subsequently find Π_4 equivalent and by the process find more equivalent groups or class to which states belongs.

→ Finally we will reach to a point where no more class or group can be formed.

→ Stop the process and Draw the DFA based on same transition table but with eq different no of states.

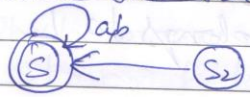
Rem:- A group or class contains more than one states can be is to be replaced by single states (Any one among the states but if starting state is there then it will be ~~kept~~ given first priority)

Ex:- Minimize the following DFA.



Myhill-Anilenerode Theorem:

Consider the following example;



Transition Table.

	a	b
→*S1	S1	S1
S2	S1	S1

From the Partitioning method point of view we have To groups which can't be further subdivided.

$\pi_0 = (S_1)\ (S_2)$

Final state. Nonfinal.

Myhill says.

(1) if L is regular the no of equivalence classes over L should be finite.

(2) Sum(union) of equivalent classes should be equivalent to L. (3) → Equivalance Relation R_L

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

Language L denoted by above fig/ is

$L = (a+b)^*$

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

and $L = \{\varepsilon, a, b, aa, bb, ab, aab, abb \ldots\}$

Let's take Two pair of string belong to L

$x = \varepsilon$ and $y = a$

Let's take another string from Σ^* that's z

if $xz \in L$ Then x is logically equivalent to y.

and $yz \in L$

Now Let $z = a$

Then $xz = \varepsilon a = a \in L$

$yz = aa = aa \in L$

Minimization of Finite Automata.

① → The minimal finite automata for any finite automata contains.

a) Always less no of states.
b) May contain less no of states.
c) Always more no of states.
d) None of above.

→ (F.A) Minimized → (MFA)

n states ≤ states ≤ n

Sol:-

Minimize finite automata is same

So,

Minimized FA may contains less no of states from a given FA.

There are Three Methods to minimize finite automata.

① → State-equivalence method [O log n]
② → Table-filling [$O(n^2)$]
③ → Myhill- Theorem.