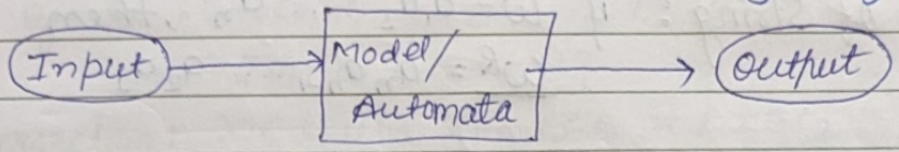
UNIT: 1 Dashrath Nandan TOC

Date :

branch that deals with what problems can be solved on a model of computation, using an algorithm, how efficiently they can be solved

★ **Theory of Computation**: TOC is a branch that deals with how efficiently problem can be solved in a model of computation using algorithm.



Eg:- Atm machine, Traffic light.

Automata theory is study of abstract computing device.

History:

1930s : Alan Turning studied Turing machines.

1940-50s : Finite automata, Chomsky Hierarchy.

1969 : Cook introduces NP-Hard problem.

1970- : Modern Computer Science : Compilers, ..

★ **Terminologies of TOC:**

I) **Symbols**: Symbols are an entity or individual object which can be any letter, alphabet or picture.
Eg:- 0, 1, a, b, α, Δ, ∀, #

II) **Alphabet (Σ)**: An alphabet is a finite, non empty set of symbols. Eg: $\Sigma = \{0,1\}$, $\Sigma = \{a,b\}$

III) **String**: Finite sequence of symbols choosen from alphabets, which is denoted by ω.
Eg: $\Sigma = \{0,1\}$, $\omega = \{0, 00, 10, 11, \ldots\}$

- **Length of String (|W|)**: The no. of symbols in the string.
Eg: $\omega = 011$, $|\omega| = 3$

- **Empty String (ε, Λ(Null), λ)**: String with zero occurrence of symbols, denoted by ε.
Eg. $|\varepsilon| = 1$ (length of empty set)

DOT

Date :

- Power of String : $w = 101$, $(101)^2 = 101101$
  $ba^2b = baab$

- Reverse of String : if $w = a_1 a_2 \ldots\ldots a_n$, then
  $w^R = a_n a_{n-1} \ldots\ldots a_2 a_1$

IV) Language : Language is set of strings with rules and denoted by L.

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

L_1 : { Set of strings starts with a }

$L_1 = \{a, aa, ab, abb, \ldots\ldots\}$

Language

Informal

↳ It concerned with rules and meanings.

Eg: English,

Formal

↳ It is concerned with rules or Syntax. (Syntatic Language.)

* Power of Alphabets :- Σ^K , $n^m \Rightarrow n = \{0,1\} = 2$ [No. of Strings]

$\Sigma^K = \{ w / w \text{ is string over } \Sigma \text{ and } |w| = k \}$, for $\Sigma = \{0,1\}$

$\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \{0,1\}$ [Set of all strings of length 1]

- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots$
- Kleen Closser (Σ^*) : Represent infinite no. of terms can be made including empty string.
  $a^* = \{\varepsilon, a, aa, aaa, \ldots\ldots\}$
- Kleen positive Clouser (Σ^+) : ..... Excluding empty string.

$\Sigma^+ = \Sigma^* - \{\varepsilon\}$

Date :

★ Operations on Formal language:

- Union: $x \in L_1 \cup L_2$ iff $x \in L_1$ or $x \in L_2$

  Eg: $L_1 = \{0, 1, 00, 11\}$, $L_2 = \{000, 11, 001\}$

  $L_1 \cup L_2 = \{0, 1, 00, 11, 000, 001\}$

- Intersection: $x \in L_1 \cap L_2$ iff $x \in L_1$ and $x \in L_2$

  Eg: $L_1 \cap L_2 = \{11\}$

- Complement : Complement of language L is $\overline{L} = \Sigma^* - L$

- Reverse : $\omega = 011$, $\omega^R = 110$

- Concatenation : L_1 and L_2.

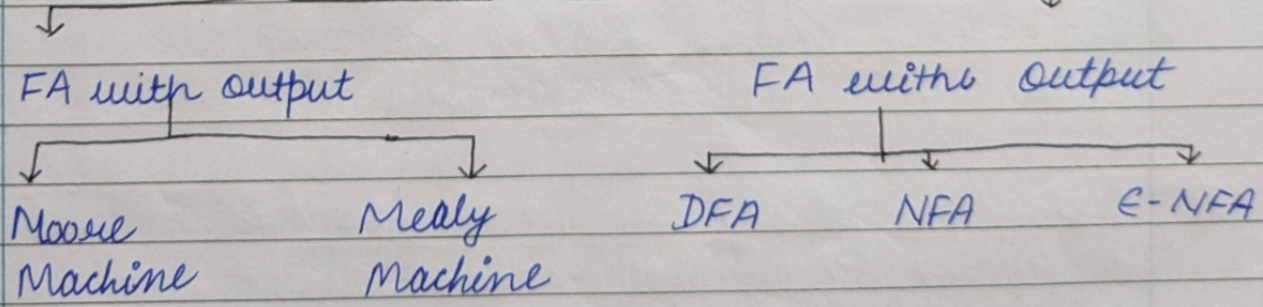
  $L_1, L_2 = \{xy : x \in L_1, y \in L_2\}$

  Concat $L_1 L_2 = \{0000, 011, 0001, 1000, 111, 1001 \dots\}$

- $L_1 * L_2 \neq L_2 * L_1$

★ Finite Automata

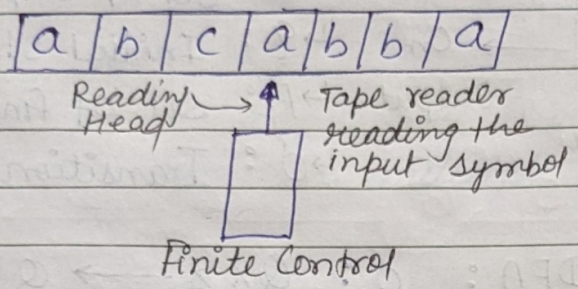
simplest machine to recognize patterns



Finite Automata (FA) is the simplest machine to recognize patterns.

Date : _____

* Block diagram of Finite Automata:

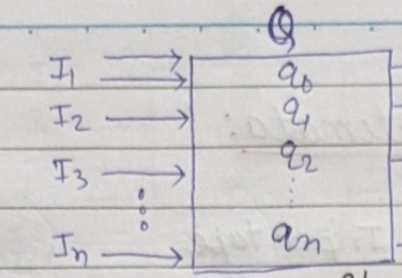


Input tape: It is a linear tape having some number of cells. Each input symbol is placed in each cell.

Finite Control: The finite control decides the next state on receiving particular input from input tape.

Reading head: The tape reader head reads the cells one by one from left to right, and at a time only one input symbol is read.

Date :



Finite Automata Consist of
↳ Q : Finite Set of states.
↳ Σ : Set of Input Symbols.
↳ q : Initial State.
↳ F : Set of final States.
↳ δ : Transition Function.

Note: In case of DFA : $\delta : Q \times \Sigma \longrightarrow Q$
In case of NFA : $\delta : Q \times \Sigma \longrightarrow 2^Q$

★ Finite Automat can be represented as —
- Transition diagram.
- Transition table.
- Transition function.

✱ Advantage of Finite State Machine :
- FSM are flexible.
- Easy to move from a significant abstract to code execution.
- low processor overhead.

✱ Disadvantage :
- Not applicable for all domain.
- The order of state conversion are inflexible.
- The implementation of huge system using FSM is hard.

★ Deterministic Finite Automata (DFA) :

In DFA, the machine can exists in only one state at any given time. Machine reads an input string one symbol at a time.

Date : _____

- In DFA, there is only one path for specific input from the current state to the next state.
- DFA doesn't accept the null move i.e. DFA cannot change state without any input character.
- DFA can contain multiple final states.

* **DFA consists of 5 tuples : $(Q, \Sigma, q_0, \delta, F)$**

Q: Set of all states.
Σ: Set of input symbol.
q_0: Initial State.
δ: transition Function
F: Final state.

- **State transition diagram:**

A state transition diagram is a directed graph which can be constructed as follows:

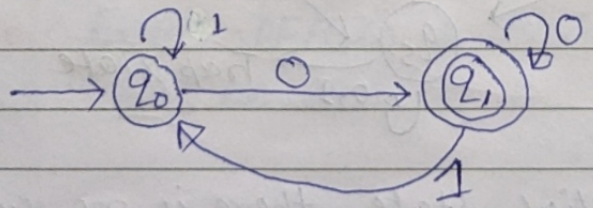
i. For each state Q there is a node.
ii. Directed edge from node q to p if $\delta(q,a)=p$.

→○ : Initial State
◎ : Final State
edge represent transition

iii. There is an arrow with no source into the start state.
iv. Accepting or final states are indicated by double circle.

Ex:- DFA with $\Sigma=\{0,1\}$, accept all string ending with 0.

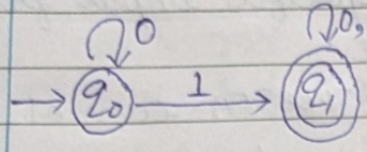
Sol:- $\{0, 00, 10, 110, \dots\}$



- **Transition table:** Tabular representation of transition function.
  - ↳ Row correspond to states.
  - ↳ Clounms correspond to input.
  - ↳ Entries represents next state.

Date :

Ex: DFA with $\Sigma = \{0,1\}$ having at least one 1.



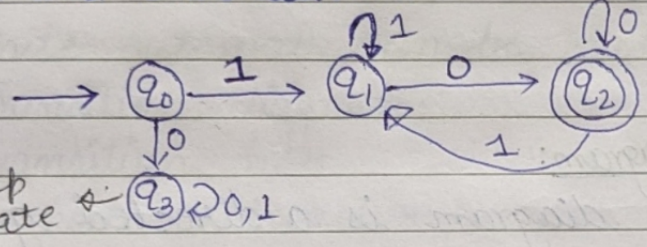
start state	States	I/P 0	I/P 1
→	q_0	q_0	q_1
final state	(q_1)	q_1	q_1

Transition Table

When input '1' is given on state 'q_0' then it moves to 'q_1' state as shown in diag.

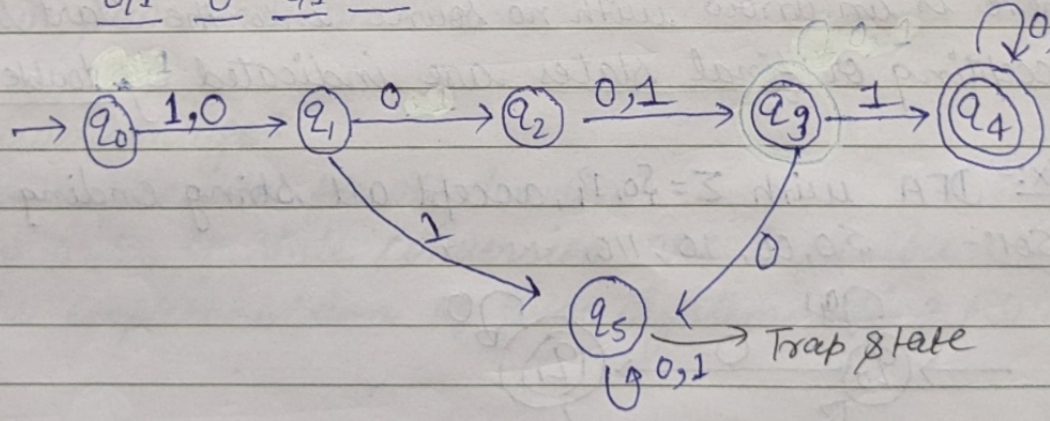
Ex: DFA with $\Sigma = \{0,1\}$ accepts those string starts with 1 and ends with 0.

Ans:-



Ex:→ Design a DFA which accept all string over (0,1) in which second symbol is '0' and fourth symbol is '1'.

Ans:→ 0/1 0 0/1 1



Explanation:- q_0 is initial state, there is no restriction on first symbol therefore it will accept both 0 & 1 as an input and moves to next state 'q_1'. In q_1, there is a condition that second symbol must be 0, so, q_1 takes 0 as valid input and moves to next state, but when q_1 takes one '1' as a input it moves to dead or trap state.

Date : _____

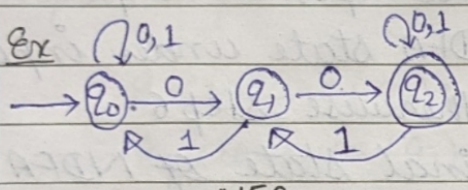
☆ Non-Deterministic Finite Automata (NFA):

- The finite automata are called NDFA when there exist many paths from specific input from current to next state.
- Every NFA is not DFA, but each NFA can be translated into DFA.

⇒ NFA also have 5 tuples as DFA: $\{Q, \Sigma, q_0, \delta, F\}$

$F \subseteq Q$

Ex

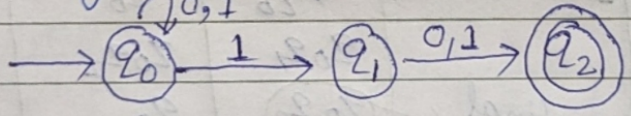


State	Next State for Input '0'	Next State for Input '1'
$\rightarrow q_0$	$q_0 q_1$	q_0
q_1	q_2	q_0
(q_2)	q_2	$q_1 q_2$

Ex: NFA of all binary strings in which 2nd last bit is 1.

Sol:- $L: \{01, 1, 01\}$

Regular Expression: $(0+1)^* 1 (0+1)$



Note:- Every DFA is a NDFA but every NDFA need not be DFA.

So, we can construct DFA also for a particular question of NDFA.

Date :

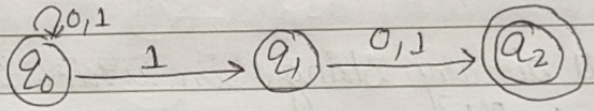
★ Equivalence of DFA and NDFA

Problem Statement :- let $X = (Q_x, \Sigma, q_0, \delta_x, F_x)$ be an NDFA which accept language $L(x)$. we have to design an equivalent DFA $Y = (Q_y, \Sigma, q_0, \delta_y, F_y)$ such that $L(Y) = L(X)$.

Procedure :

Step 1: Create state table from given NDFA.
Step 2: Create a blank state table for equivalent DFA.
Step 3: Mark the start state of DFA same as NDFA.
Step 4: Find out combination of states for each possible input.
Step 5: Each time we generate a new DFA state under input alphabet column, goto step 4 otherwise step 6.
Step 6: The state containing any final state of NDFA are finial state of the equivalent DFA.

Ex:



Sol :-

State	0	1
→ q_0	q_0	$q_0 q_1$
q_1	q_2	q_2
(q_2)	ϕ	ϕ

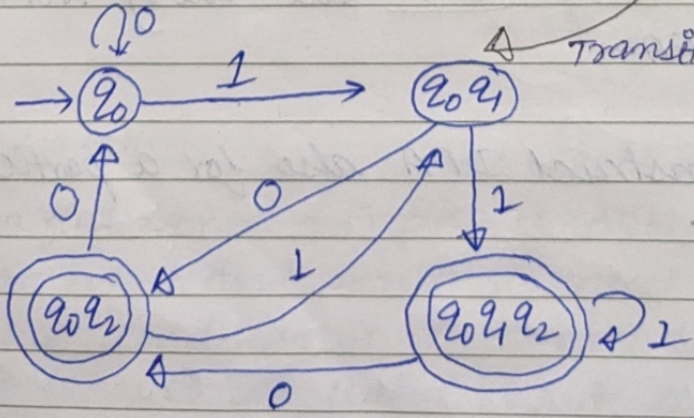
Transition table of NDFA.

States	0	1
→ q_0	q_0	$q_0 q_1$
$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
($q_0 q_2$)	q_0	$q_0 q_1$
($q_0 q_1 q_2$)	$q_0 q_2$	$q_0 q_1 q_2$

final states

T.T. Equivalent DFA

Transition table to diagram.

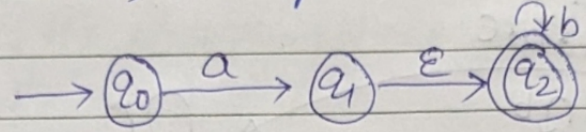


Dashrath Nandan

Date :

☆ NFA with ε - Transition

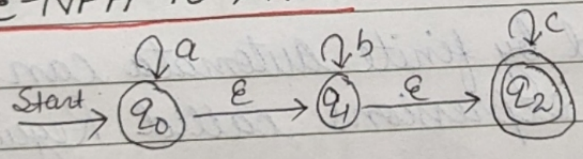
A transition on reading ε means that the NFA-ε makes the transition without reading any symbol in the input. Thus, the tape head does not move when ε is read.

Ex: 

ε-closure of a state q, Eclose(q), is the set of all states (including itself) that can be reached from q by repeatedly making an arbitary no. of ε- transitions.

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

✱ ε-NFA to NFA :



ε-NFA

δ	a	b	c	ε
→q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
(q_2)	ϕ	ϕ	$\{q_2\}$	ϕ

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

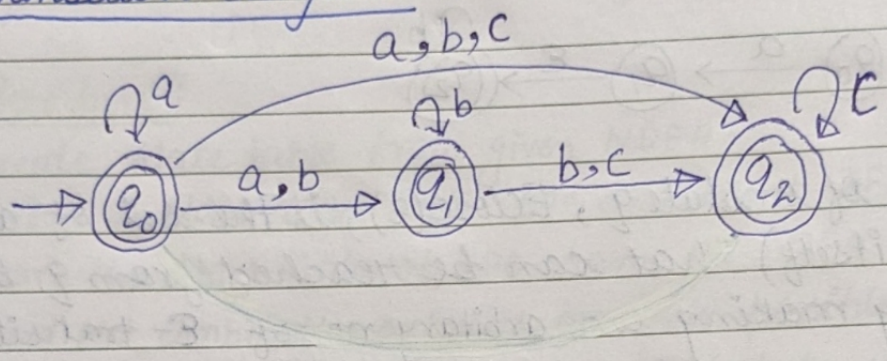
NFA δ'	a	b	c
→(q_0)	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
(q_1)	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
(q_2)	ϕ	ϕ	$\{q_2\}$

$\delta'(q_0, a)$ = ε-closure(δ(ε-closure(q_0), a))
= ε-closure($\delta(\{q_0, q_1, q_2\}, a)$)
= ε-closure($\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$)
= ε-closure($q_0 \cup \phi \cup \phi$)
= ε-closure(q_0)
= $\{q_0, q_1, q_2\}$

Date :

⇒ State q_0, q_1 and q_2 becomes the final state as ε-closure q_1 and q_2 contains the final state q_2.

Transition diagram:



☆ Regular language: language represented by a regular expression is defined as a regular language.

☆ Regular Expression:
The language accepted by finite automata can easily described by simple expressions called Regular Expression.

* Operations on Regular language:

i. Union: $L_1 \cup L_2 = \{S \mid S \text{ in } L_1 \text{ or } S \text{ in } L_2\}$

ii. Intersection: $L_1 \cap L_2 = \{S \mid S \text{ in } L_1 \text{ and } S \text{ in } L_2\}$

iii. Kleen closure: L_1^* = Zero or more occurence of L_1 language.

Language	R.E.
$\{\varepsilon\}$	ε
$\{0\}$	0
$\{001\}$	001
$\{0, 1\}$	0+1
$\{0, 10\}$	0+10
$\{1\}^* \{10\}$	$(1^* 10)$
$\{1, \wedge\}\{001\}$	$(1+\wedge)001$

Date :

Ex: Find R.E. for language accepting all strings containing any number of a's and b's. $\Sigma = \{a, b\}$.

$\Rightarrow$ $L = \{\varepsilon, a, b, ab, ba, aba, \ldots\}$

$R.E. = (a+b)^*$

Ex R.E., the first symbol should be 1 and last should be 0.

$R.E. = 1(0+1)^*0$

Ex L = Exactly 3b's. $\Sigma = \{a, b\}$

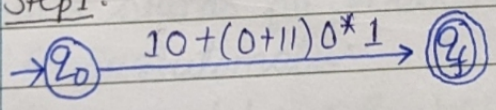
$R.E. = a^*b\ a^*\ b\ a^*\ b\ a^*$

Ex: L = at least 3b's: $RE = (a+b)^*b\ (a+b)^*b\ (a+b)^*b\ (a+b)^*$

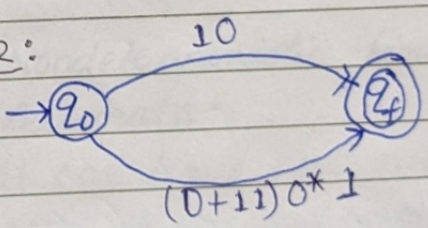
* Conversion of RE to FA:

$R.E. = 10 + (0+11)0^*1$

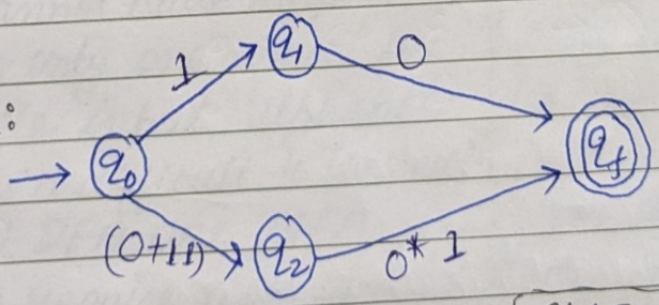
Step1:



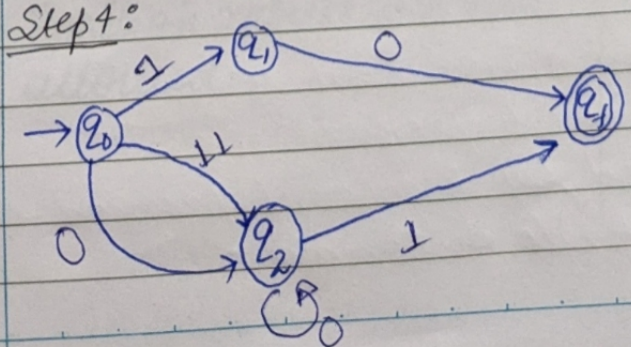
Step2:



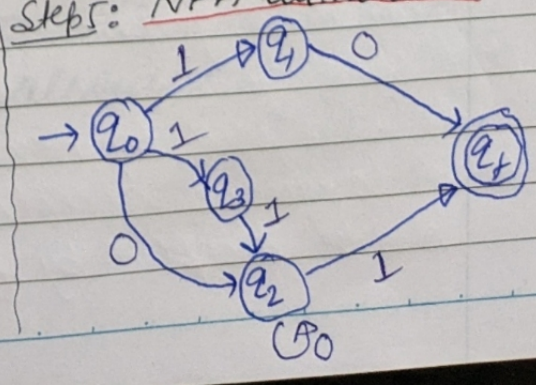
Step3:



Step4:



Step5: NFA without ε



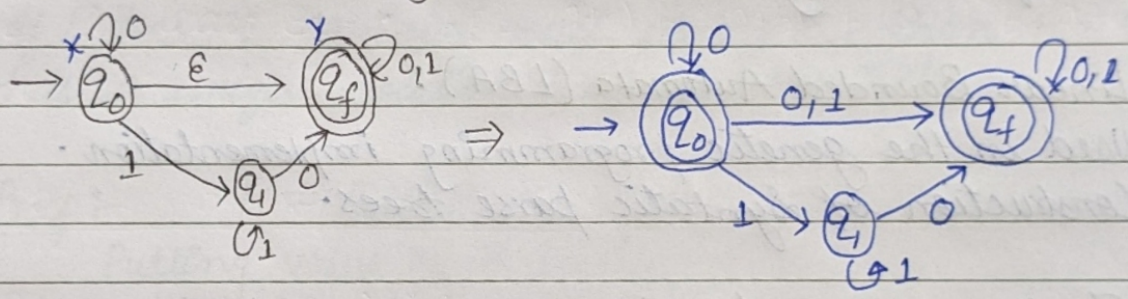
Date :

★ Removal of Null moves from finite automata :

If in an NDFA, there is ε-moves between Vertex X and Vertex Y, we can remove by following steps

- Find all the outgoing edges from Y.
- Copy all the edges starting from X without changing the edge label.
- If X is initial state, make Y also an initial state.
- If Y is final state, make X also a final state.

Ex: Convert following NFA-ε to NFA without null moves.



★

DFA	NFA
i. DFA stands for Deterministic Finite Automata.	→ Nondeterministic Finite Automata.
ii. It cannot have empty move.	→ It can have null moves.
iii. Allow only one move for single input alphabet.	→ More than one move for single input alphabet.
iv. DFA is difficult to construct.	→ NFA is easier to construct.
v. All DFA are NFA.	→ Not all NFA are DFA
vi. DFA require more space.	→ NFA require less space.
vii. Epsilion moves are not allowed.	→ Epsilion moves are allowed.

Date : _____

★ Application Of Different Automata:

1. Finite Automata –
   - Design of lexical analysis of a compiler.
   - Recognize the pattern by using regular expression.
   - Helpful in text edition.
   - Used for spell checkers.

2) Push Down Automata (PDA):
   - Used in Syntax Analysis phase.
   - Used for Solving Tower of Hanoi Problem.

3) Linear Bounded Automata (LBA):
   - Used in the genetic programming implementation.
   - Construction of Syntatic parse trees.

4) Turing Machine (TM):
   - Used to solve the recursively enumerable problem.
   - Used for knowing complexity theory.

★ Kleen's Theorem

Kleene's Theorem states the equivalence of the following three statements –

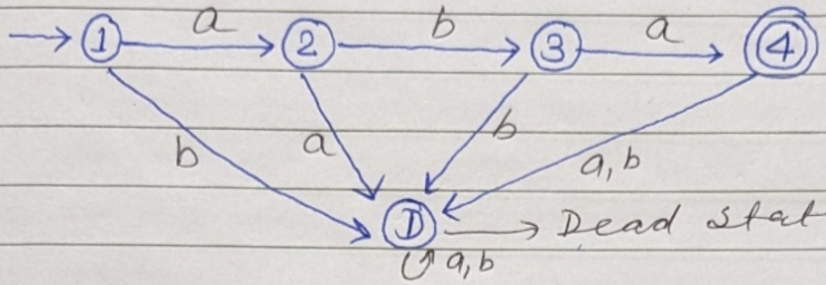
i. A language accepted by Finite Automata can also be a Transition graph.

ii. A language accepted by a Transition graph can also be accepted by Regular Expression.

iii. A language accepted by Regular Expression can also be accepted by finite Automata.

Date : _____

Proof of (i):- Let L = aba over an alphabet {a, b}.



★ Arden's Theorem:

Let P and Q be two regular expression over alphabet Σ. If P doesn't contain null string ε, then R = Q + RP has a unique solution that is R = QP*.

Proof :- $R = Q + RP$ ..... (i)

Putting value of R in (i) –

$$R = Q + (Q + RP)P = Q + QP + RP^2 \quad \text{.... (ii)}$$

again putting value of R in (ii)

$$R = Q + QP + (Q + RP)P^2 = Q + QP + QP^2 + RP^3 + \cdots$$

Similary –

$$R = Q + QP + QP^2 + QP^3 + \cdots$$
$$= Q(1 + P + P^2 + P^3 + \cdots)$$
$$= Q(\varepsilon + P + P^2 + P^3 + \cdots)$$

$$\boxed{R = QP^*}$$ (By the defination of closure operation for regular expression.)