www.BrainKart.com

Anna University

for Affilated Engineering College - 2021 Regulation

CSE Department

1st Semester | 2nd Semester | 3rd Semester | 4th Semester

5th Semester | 6th Semester | 7th Semester | 8th Semester

Click on Subject/Paper under Semester to enter.

1st Semester

- Professional English - I - HS3152
- Matrices and Calculus - MA3151
- Engineering Physics - PH3151
- Engineering Chemistry - CY3151
- Problem Solving and Python Programming - GE3151

2nd Semester

- Professional English - II - HS3252
- Statistics and Numerical Methods - MA3251
- Engineering Graphics - GE3251
- Physics for Information Science - PH3256
- Basic Electrical and Electronics Engineering - BE3251
- Programming in C - CS3251

3rd Semester

- Discrete Mathematics - MA3354
- Digital Principles and Computer Organization - CS3351
- Foundation of Data Science - CS3352
- Data Structure - CS3301
- Object Oriented Programming - CS3391

4th Semester

- Environmental Sciences and Sustainability - GE3451
- Theory of Computation - CS3452
- Artificial Intelligence and Machine Learning - CS3491
- Database Management System - CS3492
- Algorithms - CS3401
- Introduction to Operating Systems - CS3451

5th Semester

- Computer Networks - CS3591
- Compiler Design - CS3501
- Cryptography and Cyber Security - CB3491
- Distributed Computing - CS3551
- Elective 1
- Elective 2

6th Semester

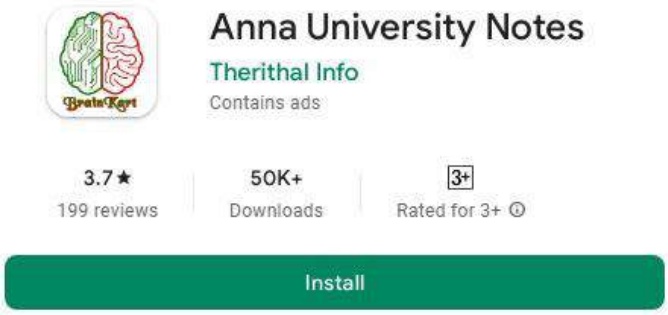
- Object Oriented Software Engineering - CCS356
- Embedded Systems and IoT - CS3691
- Open Elective-1
- Elective-3
- Elective-4
- Elective-5
- Elective-6

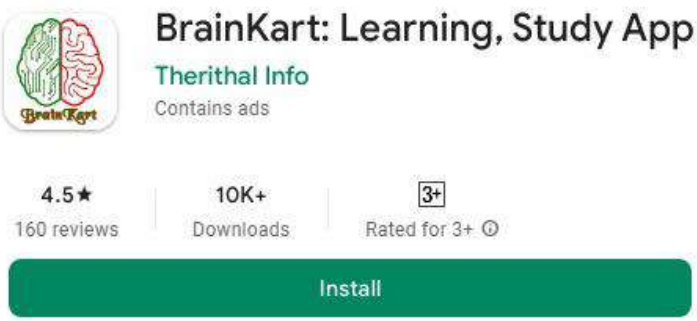
7th Semester

- Human Values and Ethics - GE3791
- Open Elective 2
- Open Elective 3
- Open Elective 4
- Management Elective

8th Semester

- Project Work / Intership





All Computer Engg Subjects - [B.E., M.E.,]		(Click on Subjects to enter)
Programming in C	Computer Networks	Operating Systems
Programming and Data Structures I	Programming and Data Structure II	Problem Solving and Python Programming
Database Management Systems	Computer Architecture	Analog and Digital Communication
Design and Analysis of Algorithms	Microprocessors and Microcontrollers	Object Oriented Analysis and Design
Software Engineering	Discrete Mathematics	Internet Programming
Theory of Computation	Computer Graphics	Distributed Systems
Mobile Computing	Compiler Design	Digital Signal Processing
Artificial Intelligence	Software Testing	Grid and Cloud Computing
Data Ware Housing and Data Mining	Cryptography and Network Security	Resource Management Techniques
Service Oriented Architecture	Embedded and Real Time Systems	Multi - Core Architectures and Programming
Probability and Queueing Theory	Physics for Information Science	Transforms and Partial Differential Equations
Technical English	Engineering Physics	Engineering Chemistry
Engineering Graphics	Total Quality Management	Professional Ethics in Engineering
Basic Electrical and Electronics and Measurement Engineering	Problem Solving and Python Programming	Environmental Science and Engineering

CS8501 THEORY OF COMPUTATION

UNIT- I

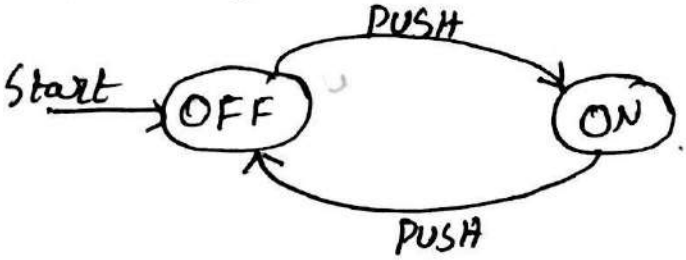
AUTOMATA FUNDAMENTALS

Introduction to formal proofs - Additional forms of proof - Inductive proofs - Finite Automata - Deterministic Finite Automata - Non-deterministic Finite Automata - Finite Automata with Epsilon Transitions.

I Introduction:

Finite Automata is a mathematical model that accepts a set of inputs, process through a set of states, generates on outputs.

Example: Finite state system - switch operation:



II BASIC MATHEMATICAL NOTATION AND TECHNIQUES:

1. Basic Mathematical objects:

SETS [A]:

A set is collection of elements of finite number.

Example:

$A = \{10, 01, 00, 11\}$

$B = \{ w \mid w$ is a set of prime numbers less than $100\}$

$\Rightarrow w \in B$

SUBSET [$\subseteq$]

Let A_1 and A_2 are two sets. then A_1 is a subset of A_2, if every element of A_1 is in A_2.

Example:

$A_1 = \{1, 2, 3, 4\}$ $A_2 = \{1, 2, 3, 4, 5\}$

$\Rightarrow A_1 \subseteq A_2$

COMPLEMENT OF A SET [A']:

Let A be a set of finite number of elements Then A' is a set of elements that are not the elements of set A [$A' \neq A$].

Example:

$A_1 = \{a, e, i, o, u\}$

$A_1' = \{$all alphabets except vowels$\}$

OPERATIONS ON SETS:

Union [U]:

The union of two sets results in a set containing the elements of both the sets.

Example:

$A_1 = \{1, 3, 5, 7\}$ $A_2 = \{1, 2, 3, 4\}$

$A_1 \cup A_2 = \{1, 2, 3, 4, 5, 7\}$

Intersection [$\cap$]:

The intersection of two sets containing a set of elements that are available in both the sets.

Example:

$A_1 = \{1, 3, 5, 7\}$ $A_2 = \{1, 2, 3, 4\}$

$A_1 \cap A_2 = \{1, 3\}$

LAWS ON SETS:

commutative law:

$A \cup B = B \cup A$

$A \cap B = B \cap A$

Associative law:

$A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$

Distributive law:

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

RELATIONSHIP OF SETS:

Relation of two sets is the association of one set with other.

Reflexive : Every elements of the set is associated with it self.

Symmetric : Let $a, b \in A$, then a is related to b, and b is related to a [$a R b = b R a$]

Transitive : If $a \to b$ and $b \to c$, then $a \to c$. It is denoted as : $a R b$, $b R c$ then $a R c$.

2. PROOFS:

A Proof is single line / multiline derivation that provide a convincing arguments to make a statement true.

Forms of proofs:

There are basically two forms of proofs

1) Deductive Proofs

2) Inductive Proofs

Deductive Proofs:

These are sequence of statements which are derived from any assumption [hypothesis] or a given initial statement to a conclusion statement.

Example: If $x \geq 4$, $2^x \geq x^2$

The hypothesis statement $\Rightarrow x \geq 4$

conclusion statement $\Rightarrow 2^x \geq x^2$

This can be provided as subsition statement as:

when $x = 4$: $2^4 \geq 4^2 \Rightarrow 16 \geq 16 \Rightarrow$ True

when $x = 5$: $2^5 \geq 5^2 \Rightarrow 32 \geq 25 \Rightarrow$ True

when $x = 3$: $2^3 \geq 3^2 \Rightarrow 8 \geq 9 \Rightarrow$ False

Hence proved.

Inductive Proof:

It has a sequence of recursive / parametrical statements that handle the statements with lower value of its parameters.

There are two types of inductions,

* Mathematical Induction
* Structural Induction

Mathematical Induction:

This follows induction principle that follows two steps:

* Basis of Induction: we start with lowest possible value.
  Ex: To prove f[n]. we take n = 0 or 1 initially.
* Inductive step: Here we prove if f[n] is true, f[n+1] is also true.

Structural Induction:

Structural induction follows the mathematical induction concept but applies for trees and expressions.

Additional Forms of Proofs:

1. Proofs about sets
2. Proofs about contradiction
3. Proofs by counter example
4. Direct proofs.

Sample Problems : Inductive Proofs:

1) Prove that : $1+2+3+\cdots+n = (n+1)/2$ using method of induction for $[n \geq 0]$.

Proof:

Basis of Induction:

let $n = 1$

$$L.H.S = 1 \quad \text{---} (1)$$

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

$$= 2/2$$

$$= 1 \quad \text{---} (2)$$

since $L.H.S = R.H.S$

$(1) = (2)$

Inductive step:

we have $1+2+3+\cdots+n = \frac{n(n+1)}{2}$

let $n = n+1$

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

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

$$= \frac{n^2+n+2n+2}{2}$$

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

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

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

$$= \frac{n^2+2n+n+2}{2}$$

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

from (1) & (2) $\Rightarrow$ **L.H.S = R.H.S**

The hypothesis is proved.

— x —

2) For all $n \geq 0$ $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Proof:

Let $n=1$

$$L.H.S = \sum_{i=1}^{1} i^2 = 1^2 = 1$$

$$R.H.S = \frac{1(1+1)(2+1)}{6} = \frac{1(2)(3)}{6}$$

$$= 6/6 = 1$$

L.H.S = R.H.S $\Rightarrow$ Proved.

Inductive Step:

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

L.H.S : Sub $n = n+1$

$$= \sum_{i=1}^{n+1} i^2$$

$$= \sum_{i=1}^{n} i^2 + (n+1)^2$$

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

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

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

$$= \frac{2n^3 + n^2 + 2n^2 + n + 6n^2 + 12n + 6}{6}$$

$$= \frac{2n^3 + 9n^2 + 13n + 6}{6} \longrightarrow (1)$$

R.H.S : Sub $n = n+1$

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

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

$$= \frac{(n^2+2n+n+2)(2n+3)}{6}$$

$$= \frac{2n^3 + 3n^2 + 4n^2 + 6n + 2n^2 + 3n + 4n + 6}{6}$$

$$= \frac{2n^3 + 9n^2 + 13n + 6}{6} \longrightarrow (2)$$

from (1) & (2) $\boxed{\text{L.H.S} = \text{R.H.S}}$

The hypothesis is proved.

III FINITE AUTOMATA:

Basic definitions:

1) Alphabet (Σ):

An alphabet is a finite, non empty set of symbols.

Ex: $\Sigma = \{0,1\}$ → Binary alphabet

$\Sigma = \{a,b,c,\ldots z\}$ → lower case letters set

2) String (ω):

A string is a finite sequence of symbols chosen from some alphabet.

Ex:

01101 is a string from $\Sigma = \{0,1\}$

aa, bb, ab, ba are strings from $\Sigma = \{a,b\}$

3) Empty string (ε) | NULL string (λ | $\wedge$):

An empty string is the string with zero occurences of symbols.

4) Reverse string (ω^R):

The reverse of the string is obtained by writing the string in reverse order.

Ex: $\omega = \{abc\}$ $\omega^R = \{cba\}$

5) Kleene closure (Σ^*):

Let Σ be an alphabet. Then the Kleene closure, Σ^* denotes the set of all strings over the alphabet, Σ

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

Ex:

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

$\Sigma^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, \ldots\}$

6) Positive closure / Kleene plus (Σ^+):

Let Σ be an alphabet. Then the positive closure, Σ^+ denotes the set of all strings over the alphabet Σ except null string (ϵ).

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

Ex:

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

$\Sigma^+ = \{1, 0, 11, 00, 01, 10, 111, 000, \ldots\}$

7) Palindrome:

A palindrome is a string which is same when read in backward or forward direction.

$$w = w^R \Rightarrow w \text{ is a palindrome}$$

Ex:

$w = \{1001\}$

$\therefore$ $w^R = \{1001\}$ equal $\Rightarrow$ w is a Palindrome

8) Language (L):

Alphabet → finite set of symbols

Strings → collection of alphabets

language → collection of appropriate strings.

A set of strings taken from an alphabet is called a language

Ex:

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

$L = \{a^n b \; n \geq 0\} \Rightarrow \{b, ab, aab, aaab, \ldots\}$

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

$L = \{$set of strings ending with $11\}$

$\Rightarrow \{11, 011, 0011, 1011, 0111, \ldots\}$

Definition of FINITE AUTOMATA:

A finite Automata is a quintuple (5 tuple)

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

where,

Q – Finite set of states

Σ – Finite set of symbols called i/p alphabet

$\delta : Q \times \Sigma \to Q$ – Transition function

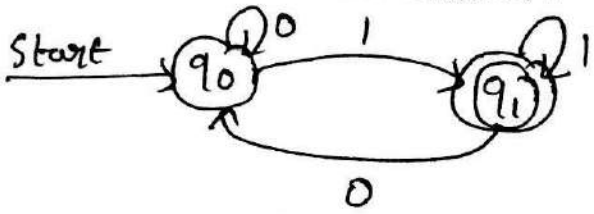
$q_0 \in Q$ – Initial state

$F \subseteq Q$ – set of finial state

Transition Diagram:

A transition diagram is a directed graph associated with the vertices of the graph corresponding to the state of finite automata.

Ex:



Transition Table:

A transition table is a conventional, tabular representation of function like δ, that takes 2 arguments and returns a value.

Ex:

	δ	0	1
→	q_0	q_2	q_0
	q_1	q_1	q_1
*	q_2	q_2	q_1

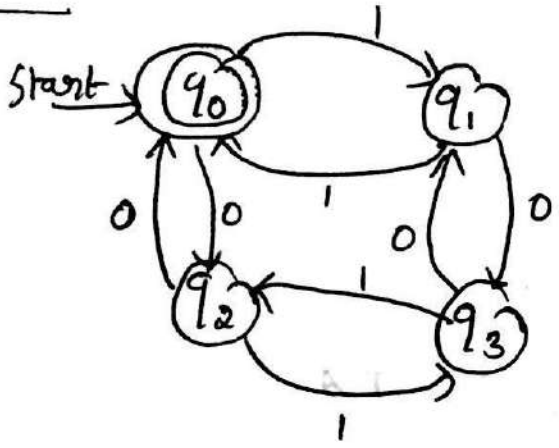
Language Acceptance by FA:

A string 'x' is accepted by finite automata $M=(Q, \Sigma, \delta, q_0, F)$ only if $\delta[q_0, x] = p$ for some p in F

The language accepted by M which is denoted by L(M).

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

Ex1:



Check whether the i/p string 110101 is accepted by FA or not.

Soln:

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

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

δ:

State / i/p	0	1
$\rightarrow$ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$\delta[q_0,110101] = \delta[\delta(q_0,1),10101]$

$= \delta[\delta(q_1,1),0101]$

$= \delta(\delta[q_0,0),101]$

$= \delta[\delta[q_2,1),01]$

$= \delta[\delta[q_3,0),1]$

$= \delta[q_1,1]$

$= q_0 \Longrightarrow$ accepted state.

$\therefore$ 110101 is in L(M)

Types of Finite Automata (FA):

There are two types of FA

1. Deterministic Finite Automata (DFA)
2. Non Deterministic Finite Automata (NFA or NDFA)

Deterministic Finite Automata (DFA):

The term deterministic refers to the fact that on each i/p there is one and only state to which the automaton can have transition from its current state.

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

Where,

Q – Finite set of states

Σ – Finite set of symbols

$\delta: Q \times \Sigma \to Q$ – Transition Function

$q_0 \in Q$ – Initial state

$F \subseteq Q$ – set of final state

Examples:

1) Design a DFA for accepting all strings of $L = \{0^m 1^n \mid m \geq 0 \text{ and } n \geq 1\}$.

Soln:

DFA:

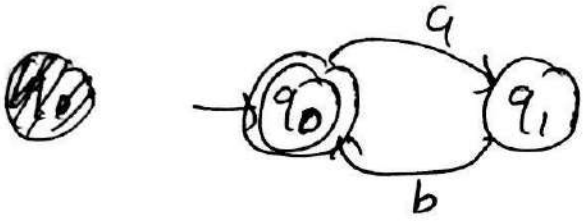
2) Design DFA over $\Sigma = \{a, b\}$ for

i) $(ab)^n$ with $n \geq 0$

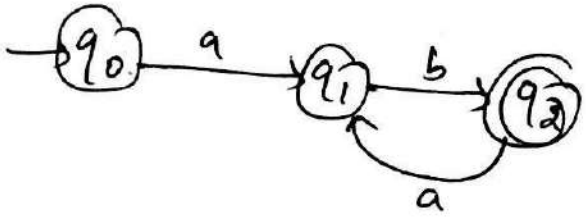
ii) $(ab)^n$ with $n \geq 1$

Soln:

i)



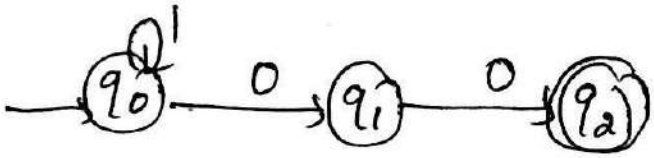
ii)



3) Give DFA's accepting the following languages over $\Sigma = \{0, 1\}$

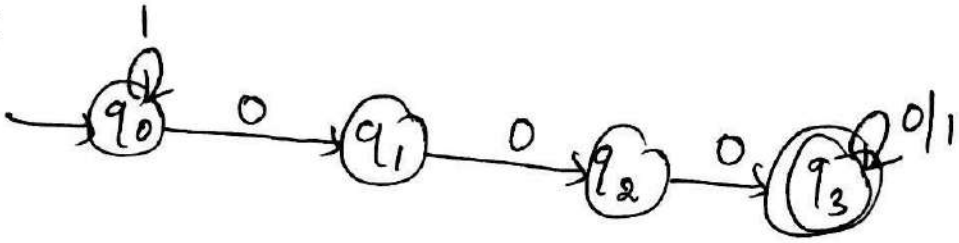
a) The set of all strings ending in 00

Soln:



b) The set of all strings with 3 consecutive 0's (not necessarily at the end)

Soln:



Non Deterministic Finite Automata (NFA / NDFA):

A NFA has power to be in several states at once.

Definition:

A NFA is defined by a 5 tuple

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

where,

Q - Finite set of states.

Σ - Finite set of symbols

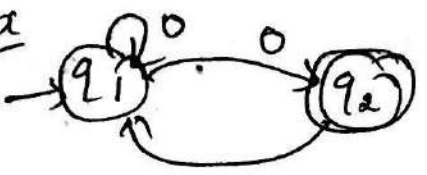
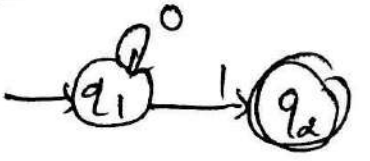
$q_0 \in Q$ - Start state

$F \subseteq Q$ - Finite set of final state

δ :- transition function mapping

$$Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$$

Difference b/w NFA & DFA:

NFA	DFA
* δ is a transition function that takes a state and i/p symbol as arguments but returns a zero, one or more state. $Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$	* δ is a transition function that takes a state and i/p as arguments but returns exactly one state. $Q \times \Sigma \rightarrow Q$
* Ex  $\delta[q_1, 0] = \{q_1, q_2\}$	* Ex  $\delta(q_1, 0) = \{q_2\}$

Extended Transition function ($\hat{\delta}$):

This is used to represent transition functions with a string of i/p symbol 'w' and returns a set of state. It is represented by $\hat{\delta}$. suppose $w = xa$

$$\delta[q, x] = \{p_1, p_2, \ldots p_k\}$$

Then

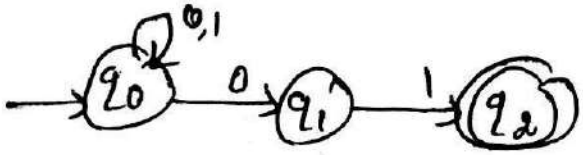
$$\bigcup_{i=1}^{k} \hat{\delta}(p_i, a) = \{r_1, r_2, \ldots r_m\}$$

$$\therefore \hat{\delta}[q, w] = \{r_1, r_2, \ldots r_m\}$$

(or)

$$\hat{\delta}(q, w) = \hat{\delta}[\delta(q, x), a]$$

Ex:



Process the i/p 001.

Soln:

$$\hat{\delta}(q_0, 001) = \hat{\delta}[\delta(q_0, 0), 01)$$

$$= \hat{\delta}[\delta(\{q_0, q_1\}), 01)$$

$$= \delta([\delta(q_0, 0) \cup \delta(q_1, 0)), 1]$$

$$= \delta[\{q_0, q_1\} \cup \phi\}, 1)$$

$$= \delta(q_0, 1) \cup \delta(q_1, 1)$$

$$= \{q_0\} \cup \{q_2\}$$

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

Language of an NFA:

If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA then

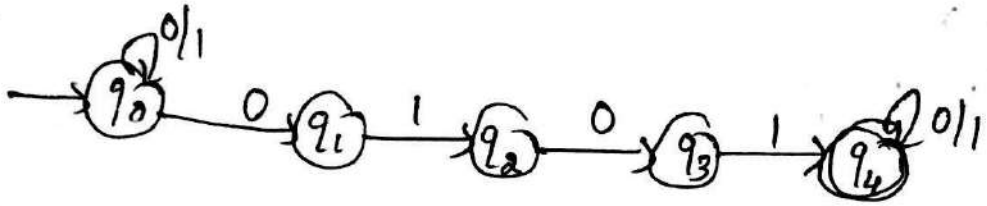
$$L(A) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$$

$L(A)$ is the set of strings w in Σ^* such that $\hat{\delta}(q_0, w)$ contains atleast 1 accepting state.

Problems:

1) Design a NFA to accept strings containing the substring "0101".

Soln:



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

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

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

$q_0 = \{q_0\}$

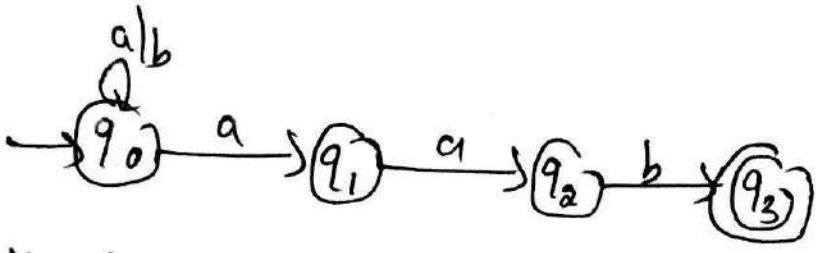
$F = \{q_4\}$

, δ :

Q / Σ	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	$\{q_3\}$	ϕ
q_3	ϕ	$\{q_4\}$
* q_4	$\{q_4\}$	$\{q_4\}$

2) Construct a NFA that accepts $L = \{x \in \{a,b\} / x$ ends with 'aab$\}$

Soln:



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

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

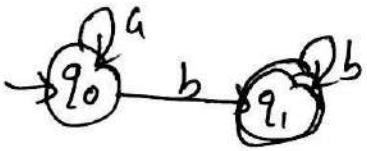
δ is

Q | Σ	a	b
→q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	ϕ
q_2	ϕ	$\{q_3\}$
*q_3	ϕ	ϕ

—×—

3) Design a NFA for $L = \{x \in \{a,b\}^* / x$ contains any number of a's followed by atleast one b$\}$

Soln:



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

$Q = \{q_0, q_1\}$

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

$q_0 = \{q_0\}$

$F = \{q_1\}$

δ is:

Q | Σ	a	b
→q_0	$\{q_0\}$	$\{q_1\}$
*q_1	ϕ	$\{q_1\}$

Equivalence of NFA and DFA:

Theorem: Let L be a set accepted by NFA. Then there exists a DFA that accept L.

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA for language L. Then define DFA M' such that $M' = (Q', \Sigma, \delta', q_0', F')$.

The states of M' are all the subset of M'. The $Q' = 2^Q$. F' be the set of all the final states in M.

The elements in Q' will be denoted by $[q_1, q_2, \ldots q$ and the elements in Q are denoted by $\{q_1, q_2, \ldots q_n\}$. The $[q_1, q_2, \ldots q_i]$ will be assumed as one state in Q' if in the NFA q_0 is initial state it is denoted in DFA as $q_0' = [q_0]$. we define,

$$\delta'([q_1, q_2, \ldots q_i], a) = [p_1, p_2, \ldots p_j]$$

if and only if,

$$\delta[\{q_1, q_2, \ldots q_i\}, a] = \{p_1, p_2, \ldots p_j\}$$

This means that whenever in NFA, at the current states $\{q_1, q_2, \ldots q_i\}$ if we get i/p 'a' and it goes to the next states $\{p_1, p_2, \ldots p_j\}$ then while constructing DFA for it the current state is assumed to be $[q_1, q_2, \ldots q_i]$ at this state, the i/p is 'a' and the next is assumed to be $[p_1, p_2, \ldots p_j]$.

The theorem can be proved with the induction method by assuming length of i/p string x.

$$\delta'(q_0', x) = [q_1, q_2, \cdots q_i]$$

if and only if,

$$\delta[q_0, x) = \{q_1, q_2, \cdots q_i\}$$

Basis: If length of i/p string is 0 (ie) $|x| = 0$, that means x is ε then $q_0' = [q_0]$.

Induction: If we assume that the hypothesis is true for the string of length m or less than m. Then if x is a string of length m+1. Then function δ' could be written as

$$\delta'(q_0', xa) = \delta'(\delta'(q_0, x), a)$$

By the induction hypothesis,

$$\delta'(q_0', x) = [P_1, P_2, \cdots P_j]$$

if and only if,

$$\delta[q_0, x) = \{P_1, P_2, \cdots P_j\}$$

By defn of δ'

$$\delta'([P_1, P_2, \cdots P_j], a) = [r_1, r_2, \cdots r_k]$$

if and only if,

$$\delta[\{P_1, P_2, \cdots P_j\}, a) = \{r_1, r_2, \cdots r_k\}$$

Thus

$$\delta'[q_0', xa] = [r_1, r_2, \cdots r_k]$$

if and only if

$$\delta[q_0, xa) = \{r_1, r_2, \cdots r_k\}$$

is shown by inductive hypothesis.

Thus $\boxed{L(M) = L(M1)}$. Hence proved.

Ex:

1) Let $M = [\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\}]$ be NFA where $\delta[q_0,0] = \{q_0, q_1\}$, $\delta[q_0,1] = \{q_1\}$, $\delta[q_1,0] = \phi$, $\delta[q_1,1] = \{q_0 q_1\}$ construct its equivalent DFA.

Soln:

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

Q': $Q' = 2^Q$ states $\Rightarrow 2^2 \Rightarrow 4$

$Q' = \{[q_0], [q_1], [q_0 q_1], \phi\}$, $\Sigma = \{0,1\}$

$q_0' = [q_0]$, $F' = \{[q_1], [q_0 q_1]\}$

δ':

$\delta'([q_0], 0) = [q_0 q_1]$

$\delta'([q_0], 1) = [q_1]$

$\delta'([q_1], 0) = \phi$

$\delta'([q_1], 1) = [q_0 q_1]$

$\delta'([q_0 q_1], 0) = \delta[[q_0],0] \cup \delta[[q_1],0]$

$= [q_0 q_1] \cup \phi$

$= [q_0 q_1]$

$\delta[[q_0 q_1], 1] = \delta(q_0,1) \cup \delta(q_1,1)$

$= [q_1] \cup [q_0 q_1]$

$= [q_0 q_1]$

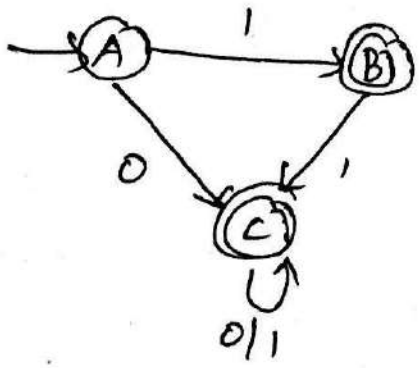
$\therefore A = q_0$, $B = q_1$, $C = q_0 q_1$

Table:

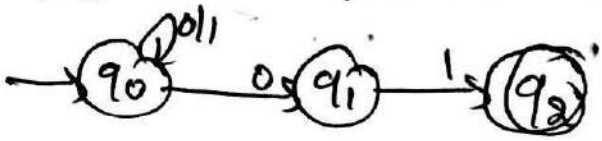
δ	0	1
→*A	C	B
*B	ϕ	C
*C	C	C

δ	0	1
→*A	C	B
*B	A	C
*C	C	C

Transition diagram:



2) Obtain the DFA equivalent to the following NFA



Soln:

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

$Q' = \{[q_0], [q_1], [q_2], [q_0q_1], [q_0q_2], [q_1q_2], [q_0q_1q_2], \phi\}$

$\Sigma = \{0, 1\}$, $q_0' = [q_0]$, $F' = \{[q_2], [q_0q_2], [q_1q_2], [q_0q_1q_2]\}$

δ':

$\delta'([q_0], 0) = [q_0q_1]$

$\delta'([q_0], 1) = [q_0]$

$\delta'([q_1], 0) = \phi$

$\delta'([q_1], 1) = [q_2]$

$\delta'([q_2], 0) = \phi$

$\delta'([q_2], 1) = \phi$

$\delta'([q_0q_1], 0) = \delta[q_0, 0] \cup \delta(q_1, 0)$
$= [q_0q_1] \cup \phi$
$= [q_0q_1]$

$\delta'([q_0q_1], 1) = \delta[q_0, 1] \cup \delta[q_1, 1]$
$= [q_0] \cup [q_2]$
$= [q_0q_2]$

$\delta'([q_0q_2], 0) = \delta[q_0, 0] \cup \delta[q_2, 0]$
$= [q_0q_1] \cup \phi$
$= [q_0q_1]$

$\delta'([q_0q_2], 1) = \delta[q_0, 1] \cup \delta[q_2, 1]$
$= [q_0] \cup \phi$
$= [q_0]$

$\delta'([q_1q_2], 0) = \delta[q_1, 0] \cup \delta[q_2, 0]$
$= \phi \cup \phi$
$= \phi$

$\delta'([q_1q_2], 1) = \delta(q_1, 1) \cup \delta(q_2, 1)$
$= [q_2] \cup \phi$
$= [q_2]$

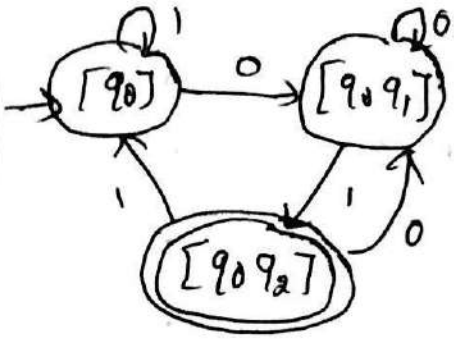
$\delta'([q_0q_1q_2], 0) = \delta[q_0, 0] \cup \delta[q_2, 0] \cup \delta[q_2, 0]$
$= [q_0q_1] \cup \phi \cup \phi$
$= [q_0q_1]$

$\delta'([q_0q_1q_2], 1) = \delta[q_0, 1] \cup \delta[q_1, 1] \cup \delta[q_2, 1]$
$= [q_0] \cup [q_2] \cup \phi$
$= [q_0q_2]$

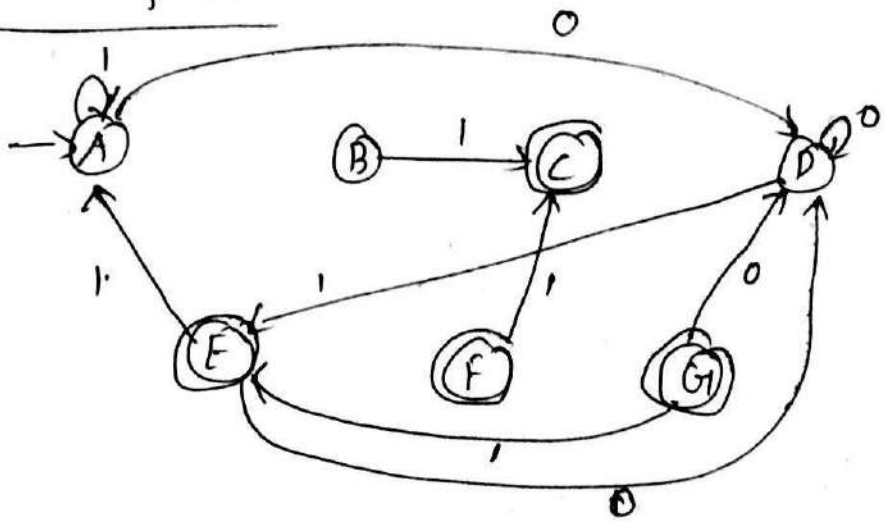
Transition Table:

δ	0	1
→[q0] A	[q0 q1] D	[q0] A
[q1] B	ϕ	[q2] C
*[q2] C	ϕ	ϕ
[q0 q1] D	[q0 q1] D	[q0 q2] E
*[q0 q2] E	[q0 q1] D	[q0] A
*[q1 q2] F	ϕ	[q2] C
*[q0 q1 q2] G	[q0 q1] D	[q0 q2] E

δ	0	1
→[q0]	[q0 q1]	[q0]
[q0 q1]	[q0 q1]	[q0 q2]
*[q0 q2]	[q0 q1]	[q0]



Transition diagram:



FINITE AUTOMATA WITH ε MOVES:

It is possible in NFA that an NFA is allowed to make transition spontaneously, without receiving an i/p symbol. This move is called ε-moves. This ε represents "any number of times".

States	input			
	ε	0	1	2
→ q_0	q_1	q_0	ϕ	ϕ
q_1	q_2	ϕ	q_1	ϕ
* q_2	ϕ	ϕ	ϕ	q_2

EPSlion(ε) closure:

If state p is in ε-closure(q), and there is a transition from state p to state r labled ε, then r is in ε-closure(q). More precisely, if δ is the function of the ε-NFA involved, and p is in ε closure(q), then ε-closure(q) also contains all the states in δ(p, ε).

Naturally let ε-closure, where P is a set of states. then

$$\bigcup_{q \text{ in } P} \varepsilon\text{-closure}(q)$$

Ex: Find ϵ-closure

Find $\hat{\delta}(q_0, 01)$.

Soln:

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

$\hat{\delta}(q_0, 01) = \epsilon$-closure $(\delta(\hat{\delta}(q_0, 0), 1))$

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

$= \epsilon$-closure $(\delta[q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)], 1)$

$= \epsilon$-closure $(\delta(\{q_0 \cup \phi \cup \phi\}), 1)$

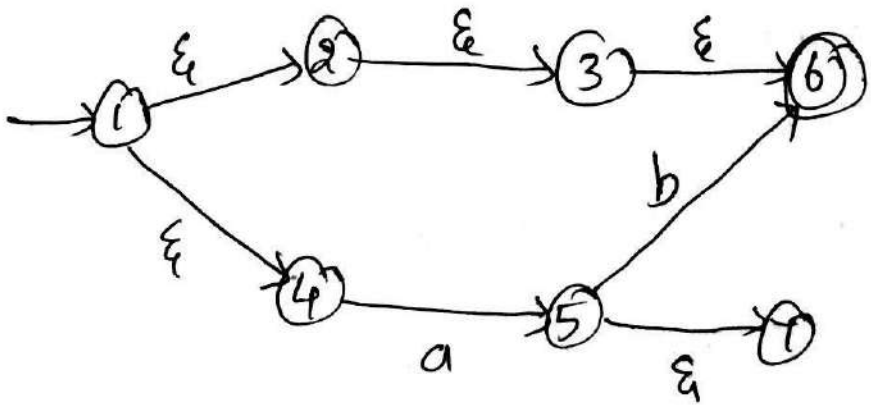
$= \epsilon$-closure $(\delta(q_0, 1))$

$= \epsilon$-closure (ϕ)

$= \phi$

— x —

Consider the NFA given below and find $\delta(1, ab)$



Language of ϵ-NFA:

The language of an ϵ-NFA, $M = (Q, \Sigma, \delta, q_0, F)$ is

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

(or) the language of M is the set of strings w that take the start state to atleast one accepting state

Equivalence of NFA's with and without ϵ-Moves:

Theorem:

If L is accepted by NFA with ϵ-transitions, then L is accepted by an NFA without ϵ-transitions.

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ϵ-transitions

Construct M' which is NFA without ϵ-transitions

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

where

$$F' = \begin{cases} F \cup \{q_0\} & \text{if } \epsilon\text{-closure}(q_0) \text{ contains a state of } F \\ F & \text{otherwise} \end{cases}$$

By induction: δ' and $\hat{\delta}$ are same

δ' and $\hat{\delta}$ are different

Let x be any string

$$\delta'(q_0, x) = \hat{\delta}(q_0, x)$$

This statement is not true if $x = \epsilon$ because

$\delta'(q_0, \epsilon) = \{q_0\}$ and $\hat{\delta}(q_0, \epsilon) = \epsilon\text{-closure}(q_0)$.

Basic:

$|x| = 1$ x is a symbol whose value is a

$$\delta'(q_0, a) = \hat{\delta}(q_0, a)$$

Induction:

Let $x = wa$ where a is in Σ

$$\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a)$$

$$= \delta'(\hat{\delta}(q_0, w), a)$$

$$= \delta'(P, a) \quad [\text{because by inductive hypothesis } \delta(q_0, w) = \hat{\delta}(q_0, w) = P]$$

Now we must show that

$$\delta'(P, a) = \hat{\delta}(q_0, wa)$$

But

$$\delta'[P, a] = \bigcup_{p \text{ in } q} \delta'(q, a) = \bigcup_{q \text{ in } P} \hat{\delta}(q, a)$$

$$= \hat{\delta}(\hat{\delta}(q_0, w), a)$$

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

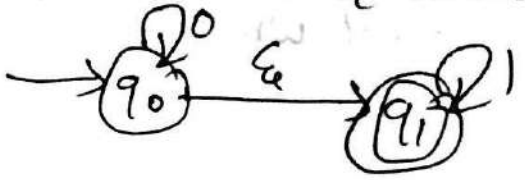
$$= \hat{\delta}(q_0, x)$$

Hence

$$\boxed{\delta'(q_0, x) = \hat{\delta}(q_0, x)}$$ Proved

Ex1:

Construct NFA without ϵ-moves from NFA with ϵ-move



Soln:

ϵ-closure $(q_0) = \{q_0, q_1\}$

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

$F' = \{q_0, q_1\}$

$\hat{\delta}(q_0, \epsilon) = \epsilon$-closure $(q_0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 0) = \epsilon$-closure $(\delta(\hat{\delta}(q_0, \epsilon), 0))$

$= \epsilon$-closure $(\delta(\{q_0, q_1\}), 0)$

$= \epsilon$-closure $(\delta(q_0, 0) \cup \delta(q_1, 0))$

$= \epsilon$-closure $(q_0 \cup \phi)$

$= \epsilon$-closure (q_0)

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 1) = \epsilon$-closure $(\delta(\hat{\delta}(q_0, \epsilon), 1))$

$= \epsilon$-closure $(\delta(\{q_0, q_1\}, 1))$

$= \epsilon$-closure $(\delta(q_0, 1) \cup \delta(q_1, 1))$

$= \epsilon$-closure $(\phi \cup q_1)$

$= \epsilon$-closure (q_1)

$= \{q_1\}$

$\hat{\delta}[q_1, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1, \varepsilon), 0))$

$= \varepsilon\text{-closure}(\delta[\{q_1\}, 0)]$

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

$= \phi$

$\hat{\delta}[q_1, 1] = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1, \varepsilon), 1))$

$= \varepsilon\text{-closure}(\delta[q_1, 1))$

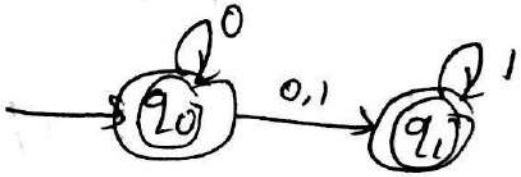
$= \varepsilon\text{-closure}(q_1)$

$= \{q_1\}$

Transition Table

State	input	
	0	1
→*q_0	$[q_0, q_1]$	$\{q_1\}$
*q_1	ϕ	$\{q_1\}$

Transition diagram



—×—

2) Convert the ε NFA to NFA (UQ)

δ	ε	a	b
→P	{r}	{q}	{P, r}
q	ϕ	{P}	ϕ
*r	{P, q}	{r}	{P}

3) Obtain an NFA without ε-transition to the following NFA with ε-transition (UQ)

—×—

REGULAR EXPRESSIONS and REGULAR LANGUAGE:

The language accepted by finite automata are easily described by simple expressions called regular expressions.

OPERATIONS OF RE:

There are 3 operations on languages that the operations of RE represent.

i) UNION
ii) CONCATENATION
iii) CLOSURE

i) UNION:

The union of 2 languages L_1 and L_2 is $L_1 \cup L_2$ is the set of strings that are in either L_1 or L_2 or both.

$L_1 \cup L_2 = \{ x \text{ or } y \mid x \text{ is in } L_1 \text{ or } y \text{ is in } L_2 \}$

Ex:

$L_1 = \{0, 11, 110\}$

$L_2 = \{\epsilon, 0, 01\}$

$L_1 \cup L_2 = \{\epsilon, 0, 110, 1101, 1100, 011, 0110, \ldots\}$

ii) CONCATENATION:

The concatenation of 2 languages L_1 and L_2 is $L_1 \cdot L_2$, which is formed by choosing a string L_1 and following it by a string in L_2, in all possible combinati

$L_1 \cdot L_2 = \{ xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2 \}$

$L_1 = \{10, 1\}$ $L_2 = \{011, 11\}$

$L_1 \cdot L_2 = \{10011, 111, 1011\}$

iii) CLOSURE:

* Kleene closure:

The Kleene closure of a language L is denoted L^* is defined as the set of strings that can be formed by taking any number of string from L, defined as,

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

* Positive closure:

The positive closure of L, denoted by L^+, is the set of string that can be formed by taking any number of string from L excluding ε. defined as

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^* - L^0 = L^* - \{\varepsilon\}$$

Ex1: $L_1 = \{10, 1\}$

$L^* = \{\varepsilon, 10, 1, 1010, 1010, 110, 11, \ldots\}$

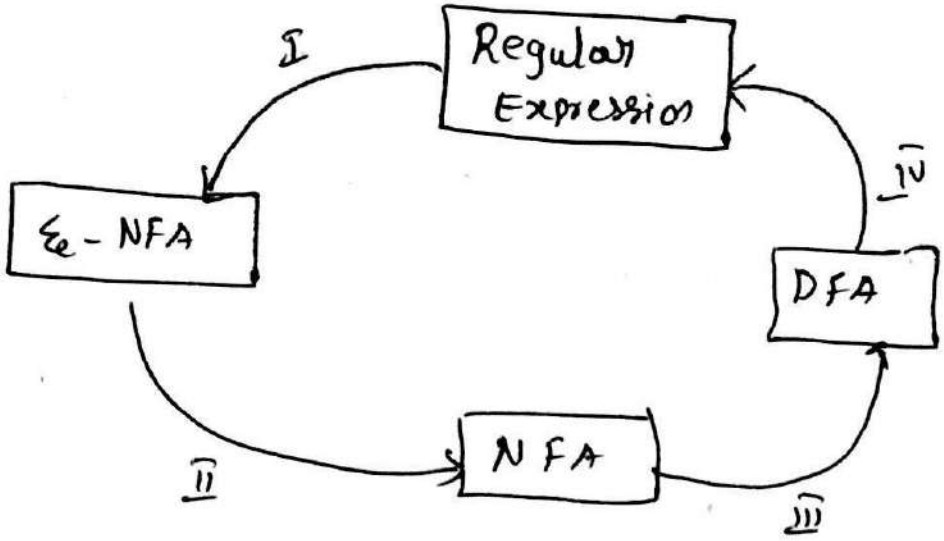
$L^+ = \{10, 1, 1010, 101, 110, 11, \ldots\}$

Ex2:

$0^* = \{\varepsilon, 0, 00, 000, \ldots\}$

$0^+ = \{0, 00, 000, \ldots\}$

FINITE AUTOMATA and REGULAR EXPRESSION:



Converting Regular Expressions to Automata:

Theorem:

Let L be a regular expression, then there exists an NFA with ε-transitions that accepts L(r).

Or

Every language defined by a regular expression is also defined by a finite automaton.

Proof:

To prove $L(M) = L(r)$ for some ε-NFA M.

Basis

i) Exactly one accepting state

$r = \varepsilon$

ii) No arcs in to the initial state

$r = \phi$

iii) No arcs out of the accepting state

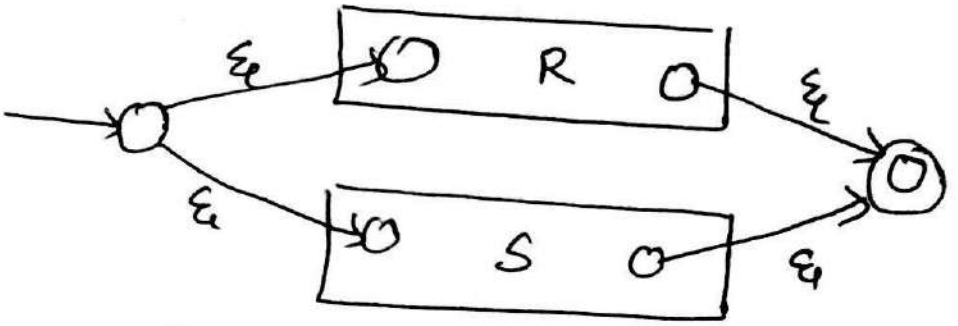
$r = a$

From the above figure it is clear that the expression r must be ε, ϕ or a for some a in ε.

Induction:

Assume that the theorem is true for the immediate sub expressions of a given regular expressions.

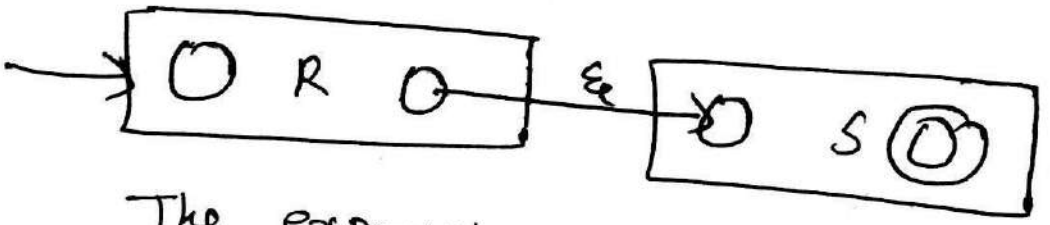
case 1:



r = R+S

The expression is R+S for some smaller expressions R and S. Right from the start state, we can go to the start state of either the automaton for R or automaton for S. Finally reach the accepting state of automata. Thus the language of automaton is L(R) ∪ L(S

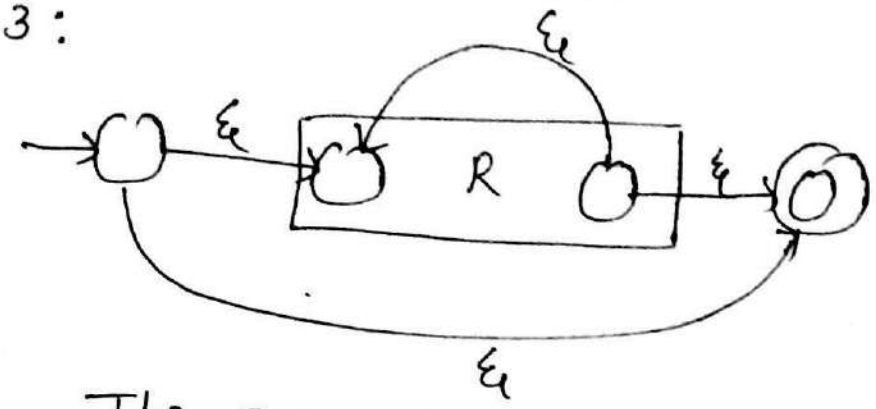
case 2:



r = R·S

The expression RS, the start state of the first automaton becomes the start state. and the accepting state of the second automata becomes the accepting state of the whole. Thus the language of automaton is L(R). L(S)

Case 3:



The expression is given by R^*, directly from the start state to accepting state along a path labled ε. That ε also belongs to R^*.

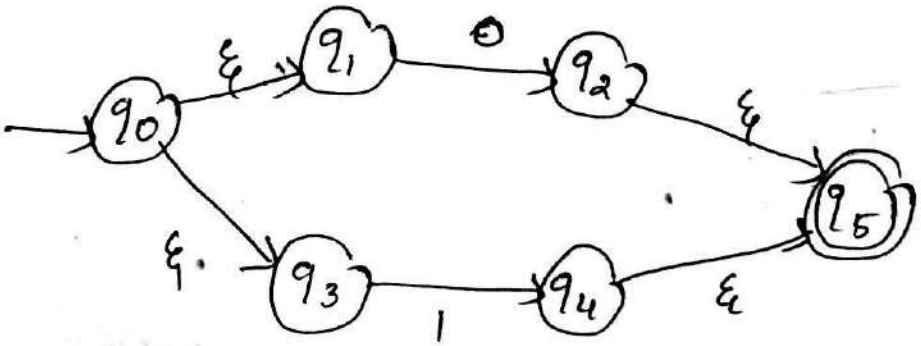
Thus the automaton satisfy the three conditions given in the inductive hypothesis - one accepting state, with no arcs in to the initial state or out of the accepting state.

—×—

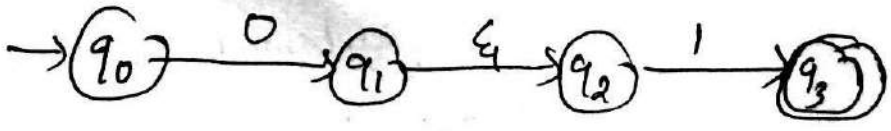
Ex:

Convert the R.E to ε NFA

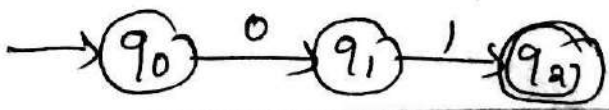
1) 0+1



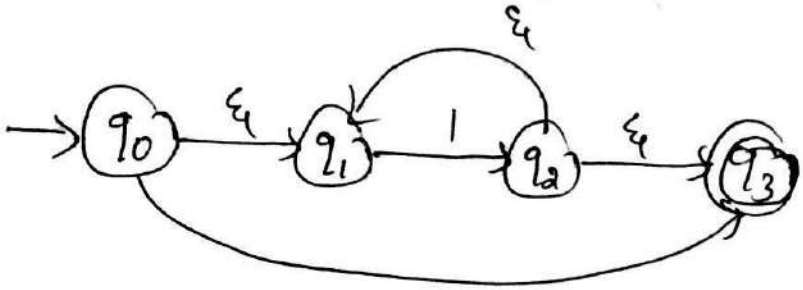
2) 01



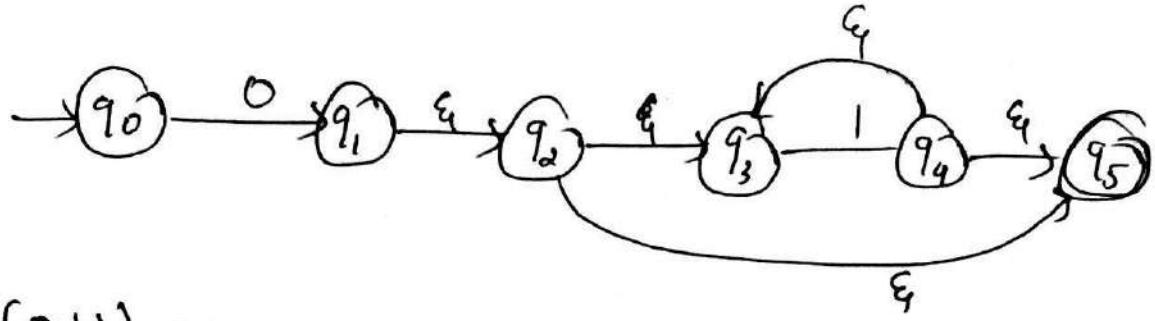
(or)



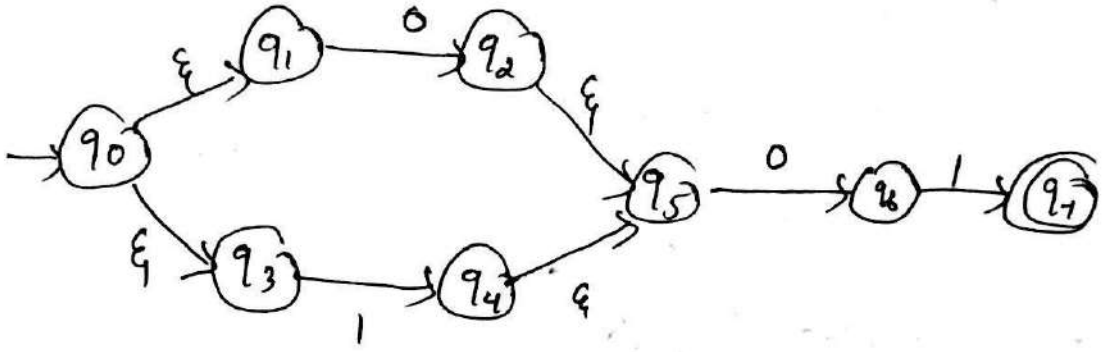
3) 1*



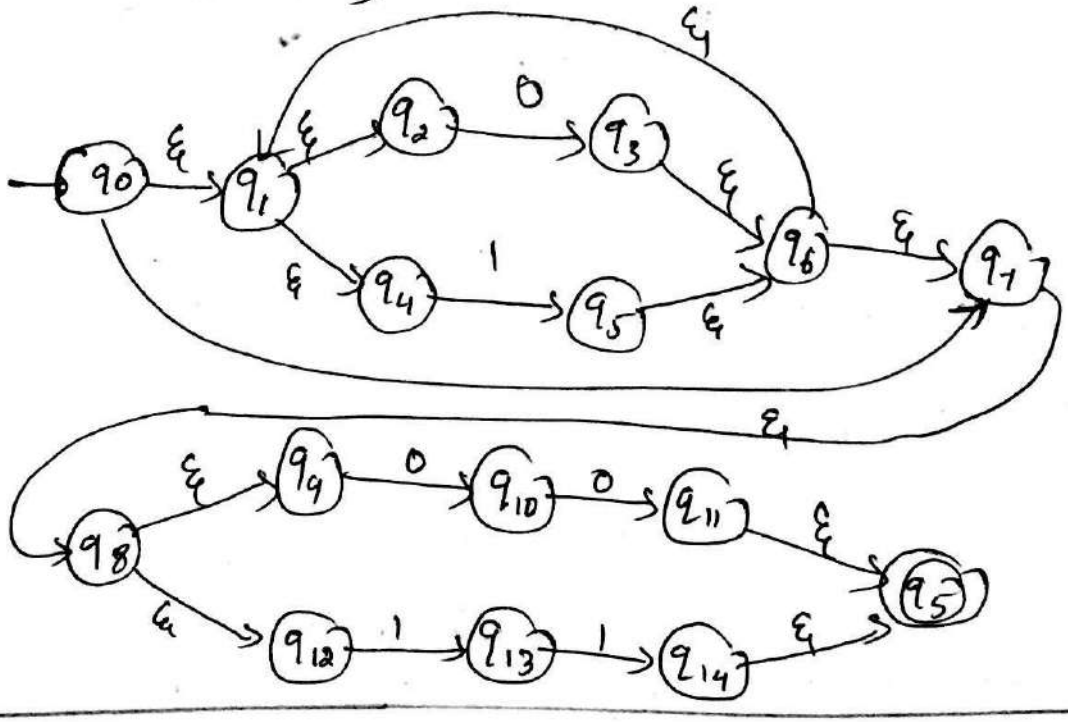
4) 01*



5) (0+1) 01



6) (0+1)* (00+11)



Conversion of ε-NFA into DFA:

Method:

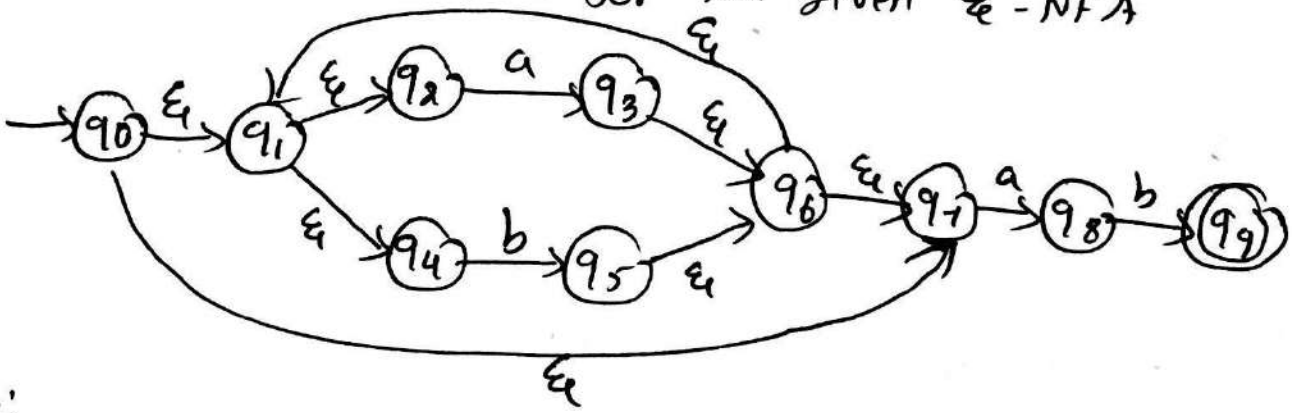
1) Find the ε-closure of the state q_0 from the constructed ε-NFA from state q_0, ε transition to other states are identified as well as ε-transitions from other states are also identified and combined as one set.

2) Perform the following steps until there are no more new states as been constructed.

   a) Find the transition of the given RE symbols over Σ from the new state.

   b) Find the ε-closure of Move(new state, symbol).

Ex1: Construct the DFA for the given ε-NFA



Soln:

$$\varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2, q_4, q_7\} \rightarrow A$$

$$\text{Mov}(A, a) = \varepsilon\text{-closure}[\{q_3, q_8\}]$$

$$= \varepsilon\text{-closure}(q_3) \cup \varepsilon\text{-closure}(q_8)$$

$$= \{q_3, q_6, q_7, q_1, q_2, q_4\} \cup \{q_8\}$$

$$= \{q_1, q_2, q_3, q_4, q_6, q_7, q_8\} \rightarrow B$$

ε(MOV(A, b)) = ε-closure(q_5)

= { $q_5, q_6, q_7, q_1, q_2, q_4$ } → C

ε[MOV(B, a] = ε-closure(q_3, q_8)

= ε-closure (q_3) ∪ ε-closure (q_8)

= { $q_3, q_6, q_7, q_1, q_2, q_4$ } ∪ { q_8)

= {$q_1, q_2, q_3, q_4, q_6, q_7, q_8$} → B

ε [MOV(B, b]) = ε-closure(q_5, q_9)

= ε closure (q_5) ∪ ε-closure(q_9)

= { $q_6, q_6, q_7, q_1, q_2, q_4$ } ∪ { q_9 }

= [$q_1, q_2, q_4, q_5, q_6, q_7, q_9$} → D

MOV (c, a) = { q_3, q_8}

ε-closure(MOV(c, a)] = ε-closure([q_3, q_8])

= ε-closure(q_3] ∪ ε-closure(q_8]

= { $q_1, q_2, q_3, q_4, q_6, q_7, q_8$ } → B

MOV(c, b) = { q_5 }

ε-closure [MOV(c, b)] = ε-closure [q_5)

= { $q_1, q_2, q_4, q_5, q_6, q_7$} → C

MOV(D, a) = { q_3, q_8 }

ε-closure (MOV [D, a]) = ε-closure(q_3) ∪ ε-closure (q_8)

= [$q_1, q_2, q_3, q_4, q_6, q_7, q_8$} → B

$Mov(D, b) = \{q_5\}$

$\varepsilon\text{-closure}(Mov(D, b)) = \varepsilon\text{-closure}(q_5)$

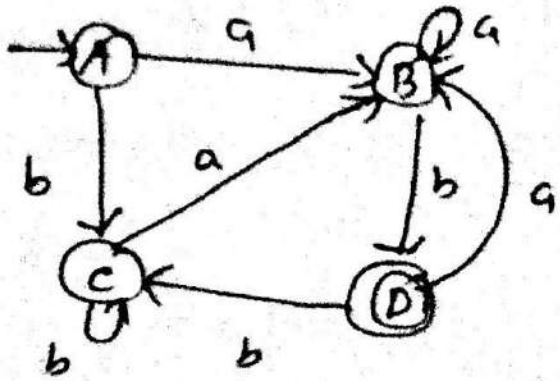
$= \{q_1, q_2, q_4, q_5, q_6, q_7\} \rightarrow C$

$F' = \{D\}$

Transition Table:

δ	Input	
	a	b
→A	B	C
B	B	D
C	B	C
→D	B	C

Transition diagram



—x—

1) Construct a DFA with reduced state equivalent to the regular expression $10 + (0+11)0^*1$

2) Construct a NFA for the given RE $((0+1)(0+1))^*$

3) Construct an NFA equivalent to $(0+1)^*(00+11)$

THEORY OF COMPUTATION

2 Mark Questions and Answers

(M7) (43)

UNIT-I

1) What is finite automata?

FA consist of set of states and transitions from occurs on i/p symbols chosen from alphabet Σ. FA is denoted by a 5 tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of states, Σ - finite i/p alphabet, δ - transition, q_0 - initial state, F - Final state. TYPES: NFA, DFA, ϵ NFA

2) Define Deterministic Finite Automata (DFA).

A DFA consist of finite set of states and a finite set of i/p symbols. In DFA, only a single transition occurs from one state to another state. $M = (Q, \Sigma, \delta, q_0, F)$. $(\delta : Q \times \Sigma \to Q)$

3) Define Non Deterministic Finite Automata (NFA or NDFA):

A NFA consist of finite set of states and a set of i/p symbols. In NFA, zero or more transition occurs from one state to another state. $M = (Q, \Sigma, \delta, q_0, F)$. $(\delta : Q \times \Sigma \to 2^Q)$.

4) What is Transition diagram and Table?

Transition diagram: FA can be represented by a directed graph namely "transition diagram". 

Transition Table: FA can be represented by a tabular representation of "δ" transition function

δ	0	1
$\to q_0$	q_0	q_0
$* q_1$	q_1	q_1

6) Define language accepted by NFA and DFA.

DFA: The language of DFA, $M=(Q, \Sigma, \delta, q_0, F)$ is denoted by $L(M)$ and it's define as $L(M)=\{w \mid \delta'(q_0, w) \text{ is in } F\}$

NFA: The language of DFA, $M=(Q, \Sigma, \delta, q_0, F)$ is denoted by $L(M)$ and it is define as $L(M)=\{w \mid \delta'(q_0, w) \cap F \neq \phi\}$.

6) Define ϵ-NFA [ϵ-Transition].

It is defeie as, $M=(Q, \Sigma, \delta, q_0, F)$ where
Q - Set of State, Σ - Set of i/p symbol ($\Sigma \cup \{\epsilon\}$),
δ - Transition function ($Q \times \Sigma \cup \{\epsilon\} \to 2^Q$), q_0 - Initial state
F - Set of finital state.

7) Define Regular Expression (RE).

RE's are the type of language defining Notation, used to describe the regular language.

Ex: $(0+1)^* 0$

8) Define Kleen closure and ~~positive~~ positive closure.

Kleen closure: Regular language L includes zero or more instance of the occurrance. $L^* = \bigcup_{i=0}^{\infty} L^i$

Positive closure: Regular language L includes one or more instance of the occurrence. $L^+ = \bigcup_{i=0}^{\infty} L^i$

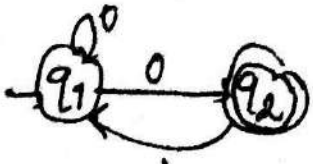
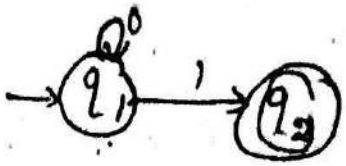
9) State the Pumping lemma for Regular language.

Let L be a Regular language, then there exist a constant n such that w in L, $|w| \geq n$, we can break w into 3 strings: $w = xyz$
such that
i) $y \neq \epsilon$
ii) $|xy| \leq n$
iii) For all $k \geq 0$, the string $xy^k z$ is also L.

10 Mention the closure properties of regular language.

1. Union 2. Difference 3. Concatenation 4. Intersection
5. Complementation 6. Transpose 7. Kleene star
8. Homomorphism 9. Inverse homomorphism.

11) Enumerate the difference b/w NFA and DFA.

NFA	DFA
→ δ is a transition function that takes a state and i/p symbol as argument but returns a zero or more state	→ δ is a transition function that takes a state and i/p symbol as arguments but returns exactly one state
→ $M = (Q, \Sigma, \delta, q_0, F_0)$ Q - set of states Σ - set of i/p symbol δ - transition $Q \times (\Sigma \cup \{\epsilon\}) \to_2 Q$ q_0 - Initial State F - Final state.	→ $(M = Q, \Sigma, \delta, q_0, F)$ Q - set of states Σ - set of i/p symbol δ - transition $Q \times \Sigma \to Q$ q_0 - Initial state F - Final state
 $\delta(q_1, 0) = \{q_1, q_2\}$	 $\delta(q_1, 0) = \{q_1\}$

— x —

• 8) What are the applications of finite automata?

1) Compiler construction 2) Switch circuit, 3) Pattern searching
4) To verify the correctness of a program
5) Design and analysis of complex s/w and H/w systems

12) Define Proof.

A Proof is single line/multi line derivation that provide a convincing argument to make a statement true.

Two Forms of proofs: 1. Deductive Proofs 2. Inductive Proofs.

Mathematical Induction, Structural Induction

12) What is Deductive proof?

These are sequence of statements which are derived from any assumption (hypothesis) or a given initial statement to a conclusion statement.

Ex: If $x \geq 4$, $2^x \geq x^2$

Hypothesis statement $\Rightarrow x \geq 4$

conclusion statement $\Rightarrow 2^x \geq x^2$

13) What is Inductive proof?

It has a sequence of recursive/parametrical statement that handle the statements with lower value of its parameters.

Types: 1. Mathematical Induction 2. Structual Induction

14) What is Mathematical Induction?

This follows induction principle that follows two steps:

* Basis of Induction: we start with lowest possible value

Ex: To prove $f(n)$. we take $n=0$ or 1 initially

* Inductive step: Here we prove if $f(n)$ is true, $f(n+1)$ is also true.

16) What is structural Induction?

Structural Induction follows the mathematical induction concept but applies for trees and expressions.

16 Define Epsilon closure (ε-closure).

If state p is ε-closure(q), and there is a transition from state p to state r labled ε, then r is in ε-closure (q). $\bigcup_{q \text{ in } p} \varepsilon\text{-closure}(q)$

UNIT - II

1) Define Grammar. (or) Context Free Grammar (CFG)

A grammar is a set of production rules for generating string in a language. $G = (V, T, P, S)$ where V - variable, T - Terminal, P - Production, S - Starting symbol.

2) **Types:** of Grammar.

1. Type 0 - Recursive Grammar, 2. Type 1 - Context Sensitive
3. Type 2 - Context Free Grammar 4. Type 4 - Regular Grammar

3) Define derivation.

set of terminal strings derived from the start symbol.

Ex: $S \to 0S1 / \varepsilon$

$S \Rightarrow 0S1$ [S→ε

$\Rightarrow 01$

Types:

1. Left Most Derivation (LMD)
2. Right Most Derivation (RMD)

4) Define LMD and RMD.

LMD: we replace the left most non terminal by one of its production in the grammar it's represented by $\Rightarrow_{lm}$

Ex: $E \to E+E / E*E / (E) / a$

$E \Rightarrow_{lm} (E)$

$\Rightarrow_{lm} (E+E)$

$\Rightarrow_{lm} (a+E) \Rightarrow_{lm} (a+a)$

www.BrainKart.com

Anna University

for Affilated Engineering College - 2021 Regulation

CSE Department

1st Semester | 2nd Semester | 3rd Semester | 4th Semester

5th Semester | 6th Semester | 7th Semester | 8th Semester

Click on Subject/Paper under Semester to enter.

1st Semester

- Professional English - I - HS3152
- Matrices and Calculus - MA3151
- Engineering Physics - PH3151
- Engineering Chemistry - CY3151
- Problem Solving and Python Programming - GE3151

2nd Semester

- Professional English - II - HS3252
- Statistics and Numerical Methods - MA3251
- Engineering Graphics - GE3251
- Physics for Information Science - PH3256
- Basic Electrical and Electronics Engineering - BE3251
- Programming in C - CS3251

3rd Semester

- Discrete Mathematics - MA3354
- Digital Principles and Computer Organization - CS3351
- Foundation of Data Science - CS3352
- Data Structure - CS3301
- Object Oriented Programming - CS3391

4th Semester

- Environmental Sciences and Sustainability - GE3451
- Theory of Computation - CS3452
- Artificial Intelligence and Machine Learning - CS3491
- Database Management System - CS3492
- Algorithms - CS3401
- Introduction to Operating Systems - CS3451

5th Semester

- Computer Networks - CS3591
- Compiler Design - CS3501
- Cryptography and Cyber Security - CB3491
- Distributed Computing - CS3551
- Elective 1
- Elective 2

6th Semester

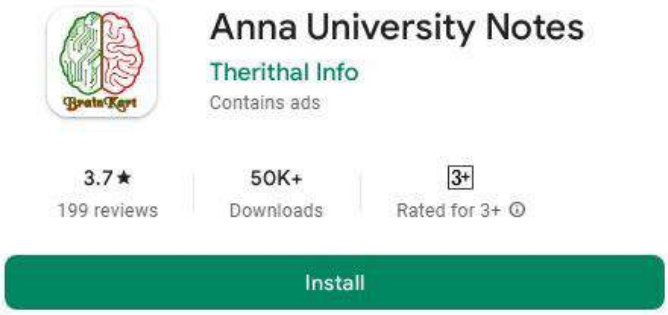
- Object Oriented Software Engineering - CCS356
- Embedded Systems and IoT - CS3691
- Open Elective-1
- Elective-3
- Elective-4
- Elective-5
- Elective-6

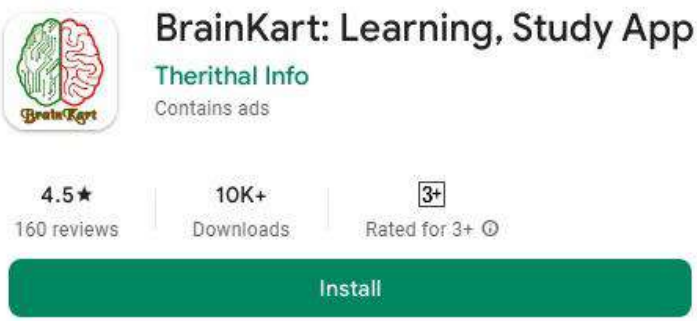
7th Semester

- Human Values and Ethics - GE3791
- Open Elective 2
- Open Elective 3
- Open Elective 4
- Management Elective

8th Semester

- Project Work / Intership





All Computer Engg Subjects - [B.E., M.E.,]		(Click on Subjects to enter)
Programming in C	Computer Networks	Operating Systems
Programming and Data Structures I	Programming and Data Structure II	Problem Solving and Python Programming
Database Management Systems	Computer Architecture	Analog and Digital Communication
Design and Analysis of Algorithms	Microprocessors and Microcontrollers	Object Oriented Analysis and Design
Software Engineering	Discrete Mathematics	Internet Programming
Theory of Computation	Computer Graphics	Distributed Systems
Mobile Computing	Compiler Design	Digital Signal Processing
Artificial Intelligence	Software Testing	Grid and Cloud Computing
Data Ware Housing and Data Mining	Cryptography and Network Security	Resource Management Techniques
Service Oriented Architecture	Embedded and Real Time Systems	Multi - Core Architectures and Programming
Probability and Queueing Theory	Physics for Information Science	Transforms and Partial Differential Equations
Technical English	Engineering Physics	Engineering Chemistry
Engineering Graphics	Total Quality Management	Professional Ethics in Engineering
Basic Electrical and Electronics and Measurement Engineering	Problem Solving and Python Programming	Environmental Science and Engineering