CS8501 - Theory of computation

- J.E. Hopcroft, R. Motwani and J.D. Ullman Introduction to Automata Theory, Languages and computations second Edition; pearson education. 2003.

UNIT-I

AUTOMATA FUNDAMENTALS

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

UNIT-II

REGULAR EXPRESSIONS AND LANGUAGES

Regular Expressions - FA and Regular Expressions - proving Languages not to be Regular - closure properties of Regular Languages - Equivalence and Minimization of Automata.

UNIT-III

CONTEXT FREE GRAMMAR AND LANGUAGES

CFG - parse Trees - Ambiguity in Grammars

Languages definition of the Pushdown Automata. Languages of a Pushdown Automata. Equivalence of pushdown Automata and CFG, deterministic pushdown Automata.

UNIT-IV

PROPERTIES OF CONTEXT FREE LANGUAGES

Normal forms for CFG - Pumping Lemma for CFL - closure properties of CFL - Turing Machines - programming Techniques of TM.

UNIT-V

UNDECIDABILITY

Non Recursive Enumerable (RE) Language - Undecidable problem with RE - Undecidable problems about TM - post's correspondence problem, The class P and NP

UNIT-I

Automata Fundamentals

Introduction to formal proof:

The convincing all given that shows the given statement is True.

Formal proof:

The Truth value of the problem statement given can be Solved by sequence of steps and detailed Reasons. This is called as Formal proof.

They are four Types.

- -> deductive proof
- -> Reduction to definitions
- -> If then forms
- -> Theorem that appear not to be if-then.

Deductive proof :

→ It consits of sequence of statement hows truth the conclusion from the given Initial Statment.

→ These sequence of statment are called as hypothesis.

→ In deductive proof each step must follow some accepted Logical principles from either the given factor some of the previously proved statment.

1) Theorem: If $x \geq 4$ then $2^x \geq x^2$

To prove: $2^x \geq x^2$ whenever $x \geq 4$

L·H·S: 2^x Let $x = 5 \Rightarrow 2^5 = 32$

$2^6 = 64$

$2^7 = 128$

whenever x is Increased by 1, then 2^x gets doubled.

growth rate = 2

R·H·S: x^2

growth Rate $= \left(\frac{x+1}{x}\right)^2$

Let $x = 4$

$= \left(\frac{4+1}{4}\right)^2 = \left(\frac{5}{4}\right)^2$

$= 1.5625$

$\therefore$ L·H·S > R·H·S

$\therefore 2^x \geq x^2$

conclusion:

since always $1.5625 < 2$ then each Time x Increases above four then $2^x \geq x^2$

2) If $x = a^2 + b^2 + c^2 + d^2$ then $2^x \geq x^2$ where a, b, c, d are positive Integer.

Sol:

Since a, b, c, d are positive Integers then minimum value is '1' Therefore the minimum value of $x = 4$, then by Modus ponens

If A is True and

$A \rightarrow B$

then B is True

Since $x \geq 4$ then $2^x \geq x^2$ (From theorem 1)

$\therefore$ If $x = a^2 + b^2 + c^2 + d^2$ then $2^x \geq x^2$

is Proof.

Reduction To definitions: (Divide & conquer)

→ when the hypothesis is difficult to solve then convert all the Terms into their definitions.

→ Some Theorems may not have complete problem statement To prove them as its. In such cases this Reduction to definition can be used.

Theorem;

Let 'S' be a finite subset of some Infinite set 'U', Let 'T' be the complement of 'S' w.r.t 'U', prove that 'T' is Infinite.

|| → Length of the set

Sol:

$$||S|| = n$$

The set S has Finite number of elements.

$$||S|| = n$$

If S and T are both subsets of 'U' then ~~Set Union~~ S ∪ T.

vide &
conque

t all
ns.
not
tement
such
finition

Some
rove
ength of
he set
ber of
subsets
of T.

$S \cup T = U$

$S \cap T = \phi$

From $S \cup T = U$

$\therefore \|S\| + \|T\| = \|U\|$

$n + \|T\| = \infty$

$\|T\| = \infty - n$

$\therefore \|T\| = \infty$

$\therefore$ Set T is Infinite

hence proved.

※ If Then forms:

→ If - then

→ If and only if

~~Proof:~~

If - Then:

* H implies C $x \geq 4$ implies $2^x \geq$

*C only if H $2^x \geq x^2$ only if $x \geq$

* C if H $2^x \geq x^2$ if $x \geq 4$

* whenever H holds, C follow

(or) If H holds, then C follows.

whenever $x \geq 4$ holds,

$2^x \geq x^2$ Follow

≡ Equalent to

If and Only if:

* ⇔ iff A iff B
* ⟷ A ⟷ B
* ≡ A ≡ B

Theorem.

Let 'x' be an real number than $\lfloor x \rfloor = \lceil x \rceil$ if and only if 'x' is an Integer

$\lfloor 3.14 \rfloor = 3$ → Flooring operation

$\lceil 3.14 \rceil = 4$ → ceiling operation

only-if part:

Assume $\lfloor x \rfloor = \lceil x \rceil$ and we have to prove x is an Integer

w.k.T,

$\lfloor x \rfloor \leq x \longrightarrow (1)$

$\lceil x \rceil \geq x \longrightarrow (2)$

Substitude (2) in (1);

then $\lceil x \rceil \leq x$

$\therefore \lceil x \rceil = x$

Since ⇔ $\lceil x \rceil = x$ always an Integer than must also be Integer.

If part:

Assume x is an Integer then prove $\lfloor x \rfloor = \lceil x \rceil$

Since x is an Integer $\lfloor x \rfloor = x$

$\lceil x \rceil = x$

$\therefore \lfloor x \rfloor = \lceil x \rceil$

$\therefore \lfloor x \rfloor = \lceil x \rceil$

Therfore the given ~~then~~ $\lfloor x \rfloor = \lceil x \rceil$ if and only if x is an Integer.

hence prove.

Not to be in If - Then Statements:

Trigonometric Theorems;

Ex: $\sin^2\theta + \cos^2\theta = 1$

Additional Forms of proof:

* proofs about sets
* proofs by contradiction
* proofs by counter example

Proofs about Sets:

Sets: It is a collection of Elements.

If 'E' and 'F' are two expression Representing set Then E = F means that they Two Expression are equivalent.

Therefore we have to prove that if 'x' is in 'E' then x is in 'F'.

Then we have to prove that then 'x' is in 'E'.

prove that $R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$

$E = R \cup (S \cap T)$

$F = (R \cup S) \cap (R \cup T)$

if part:

x is in E

i) x is in $R \cup (S \cap T)$ Given

ii) x is in R (or) x is in $(S \cap T)$. (i) and by dfn of Union

iii) x is in R (or) x is in S and T (ii) and by dfn of intersecti

iii) x is in RUS and (3) and by dfn of Set
x is in RUT

iv) x is in (RUS)∩(RUT)
~~what~~ we have reached E

∴ If part is True.

Only if part :

Assume E is True

1) x is in (RUS)∩(RUT)	Given
2) x is in R or S and x is in R or T	From ① & by dfn of union & Intersection
3) x is in R or x is in S and T.	From ② & by dfn of sets
4) x is in RU(S∩T) ∴ x is in E	From ③ & by dfn of union and Intersection

∴ Only if part is also True.

RU(S∩T) = (RUS) ∩ (RUT)

x is in E

hence proved //.

Proof by contradiction :

For the Statement if a then
we will start with the negation of

statement 'a' is not True by assun False (A) Then we will ~~start~~ Try to ge conclusion 'b' when it becomes impossible To reach we contradic our self and accept 'a' is True.

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

Proof:

~~A~~ S Assume A is False.

$\therefore A^{-1} = A \cap (B \cup C)$

1) Let x is in $A \cap (B \cup C)$ Given

2) x is in A and x is in B or C ① & by dfn of Union & Intersection

3) x is in A and B or x is in A and C ② & by dfn of sets

4) x is in $(A \cap B)$ or x is in $(A \cap C)$

5) x is in $(A \cap B) \cup (A \cap C)$

Since we failed to reach B Union assumption is wrong,

$\therefore$ A is True

Then $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ is True hence the proof.

sun

o ge

ic

rue.

uc)

..

f Union
tion

ffer of
Sets

3 Union

:) us

proof by counter Example:

In order to proof certain statements we need to see all possible condition In which the statements Remains True. In some statements there are Infinite number of cases. In such cases Taking some sample and proving the Theorem is false. In Much that is called by proof by Counter Example.

Ex: All primes are odd.

proof: The integer 2 is prime also its even.

∴ The given Statement is not True.

Inductive proof:

As special kind of proof that gives with Recursively defined object This proof follows the principle of Induction.

Basis:

In Basis Step we have shaw the given Statement is True For its most basis value (lowest possible value).

Induction:

In this Step we assign value of given element to Some othee value.

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

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

Now let as assume the Statement as True. then check the Result for the next element

For all $n \geq 0$ prove that $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ by Induction Method.

proof:

Basis : Let $n = 1$

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

$= 1^2 = 1$

Show

For

possible

value

ree

nck

ent

ion

$R \cdot H \cdot S \Rightarrow \frac{n(n+1)(2n+1)}{6}$

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

$= 6/6 = 1$

$\therefore L \cdot H \cdot S = R \cdot H \cdot S$

They eqn is True for $(n=1)$ minimu

basis value.

Induction:

Let $n \geq 0$

by Induction Method Let 'k' be the value for all $k \geq 0$, $\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}$

Assume the Statement is True know we have to check

For $k+1$,

$L \cdot H \cdot S \Rightarrow \sum_{i=1}^{k+1} i^2 = \left(\sum_{i=1}^{k} i^2\right) + (k+1)^2$

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

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

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

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

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

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

R.H.S: $\Rightarrow \frac{K(K+1)(2K+1)}{6}$ substitude $K = K+1$

$$\frac{(K+1)(K+1+1)(2(K+1)+1)}{6}$$

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

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

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

$$= \frac{2K^3+3K^2+6K^2+9K+4K+6}{6}$$

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

$(1) = (2)$

$\therefore$ L.H.S = R.H.S

$\therefore$ The given Statement is True For $K+1$ that implies the statement is True For K also then

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

hence proved.

Finite Automata:

Automata Theory: It is the study of abstract computing the devices It is proposed by alan during in 1930 him Study an abstracted

machine of 100% automatic that had atmost all the capabilities of Todays computers he descriped the boundary between what computer machine can do and what Cannot do to.

2m

The need for automata Theory:

i) It is essential for the study of limits of computation

ii) Designing and checking the behaviour of electronic circuit

iii) batten Seaching in website

iv) Verifying system of all Types that have Finite No. of States such as networking protocols.

Concepts of Automata Theory:

* Alphabet: An alphabet is finit Set of non-empty set of Symbols.

It is denoted by $\Sigma = \{0, 1\}$ - I

is the set of binary Numbers.

$\Sigma = \{A, B, C, \ldots, z\}$ It is the set of upper case letter.

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

Strings: As string over an alphabet is a finite sequence of symbols from that alphabet

Ex:

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

01010 - valid ∴ It has symbol not in alphabet only.

01234 - Invalid

Empty string:

They string with 0 acqurance of symbol

ε - epsilon

Length of the string:

They length of the string is Number of symbols its that string is represented by |a|

If a = 10101 1

|a| = 6

$|\varepsilon| = 0$

Power of an alphabet:

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

$\Sigma^3 = \{000, 001, 010, 011 \ldots\}$

$\Sigma^2 = \{00, 01, 10, 11\}$

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

$\Sigma^0 = \{\varepsilon\}$

Σ^* (Kleene star)

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

Concatenation of strings:

Let x & y be strings then concatenation of x & y denoted by xy (or) $x \cdot y$ is the string created by the string 'x' followed by String 'y'

$x = a_1 a_2 a_3 \ldots a_m$

$y = b_1 b_2 b_3 \ldots b_n$

$x \cdot y = a_1 a_2 a_3 \ldots a_m b_1 b_2 b_3 \ldots b_n$

Language:

Any set ~~string~~ Symbols over a alphabet that has of common Factor is defind as language.

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

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

L = Set of all Strings Starts with a,

$L = \{a, aa, ab, aaa, aab \ldots\}$

Empty Language: (ϕ).

A Language Set of has no elements.

Grammar:

It is defined by four (Tuples) elements $G = \{V, T, S, P\}$

V → Finite Set of objects called variables.

T → Finite Set of objects called Terminals

S → Start symbol $S \in V$

P → Finite Set of production Rules.

Ex:1

Let $\Sigma = (a, b)$ obtain Σ^* give an example for Finite language in Epsilon.

check The String are accepted by the language.

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

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

i) L = The set of all strings has atleast one 'a'

$L = \{a, aa, ab, ba, aaa, \ldots\}$

Let $L = \{a^n b^n ; n \geq 0\}$

aabb $\Rightarrow n = 2$

$\therefore a^n b^n \Rightarrow a^2 b^2$

$= aabb$

$\therefore$ They given String is accepta by language.

abb $\Rightarrow n = 1$

$a^n b^n = a^1 b^1$

$= ab \neq abb$

$\therefore$ They given String is not accepted by language.

iii) $a^2 b^2 ab \Rightarrow aabbab$

$n = 3$

$a^n b^n = a^3 b^3$

$= aaabbb$

∴ They given String is not in the format.

Ex : 2

Let $u = a^2ba^3b^2$

$v = ba$

Find i) uv ii) uv^2 iii) u^2v^2 iv) $|uv|$

Sol:

i) $uv = a^2ba^3b^2 \cdot ba$

$= a^2ba^3b^3a$

ii) $uv^2 = uv \cdot v$

$= a^2ba^3b^2ba \cdot ba$

$= a^2ba^3b^3aba$

iii) $u^2v^2 = u \cdot u \cdot v \cdot v$

$= a^2ba^3b^2 \cdot a^2ba^3b^2 \cdot ba \cdot ba$

$= a^2ba^3b^2a^2ba^3b^3aba$

iv) $|uv| = 10$

∴ Length of uv is Ten.

$a=6, 2=10$
$b=4$
$u = a^2ba^3b^2$
$v = ba$
$|uv| = 10$

Find grammar For $\Sigma = \{a, b\}$ That generate

a) all Strings with exactly one 'a'

b) all Strings with atleast one 'a'

a) Production Rules:

$S \rightarrow aA$

$A \rightarrow bA$

$A \rightarrow \varepsilon$

$G = \{V, T, S, P\}$

$V = \{S, A\}$

$T = \{\varepsilon, a, b\}$

$S = \{S\}$

production Rules $S \rightarrow aA$

$A \rightarrow bA / \varepsilon$

b) all strings with atleast one 'a'

$S \rightarrow aB$

$B \rightarrow aB$

$B \rightarrow bB$

$B \rightarrow \varepsilon$

$S \rightarrow aB$

$B \rightarrow aB / bB / \varepsilon$

$G = \{V, T, S, P\}$

$V = \{S, B\}$

$T = \{\varepsilon, A, B\}$

$S = \{S\}$

production Rules: $S \rightarrow aB$

$B \rightarrow aB$

$B \rightarrow bB$

$B \rightarrow \varepsilon$

$S \rightarrow aB$

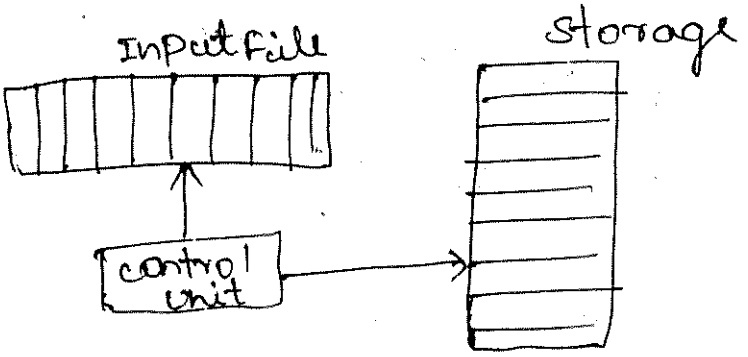
$a \rightarrow aB / bB / \varepsilon$

Finite Automata:

Automaton:

An automaton is an abstract model of the digital computer. It is a device that can take the required i/p's on its own without human intervention.

It is a 100% automated device



An finite automaton has an machanism to read i/p which is as string over the given alphabet.

It is written on i/p file the i/p file is divided into cell

Each can hold one symbols.

Types:

→ DFA

→ NFA

DFA: Deterministic Finite Automata

They term 'deterministic' refers to the ~~back~~ that on each i/p there is only one o/p state ~~to~~ to which the automaton can have Transition from its current state.

DFA is quintuple (5-Tuple)

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

$Q \rightarrow$ Finite set of states

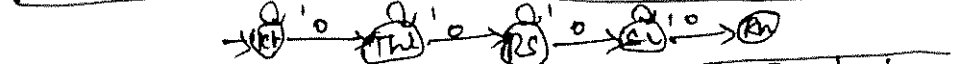
$\Sigma \rightarrow$ Finite set of i/p symbols

$\delta \rightarrow$ Transition function $Q \times \Sigma \rightarrow Q'$

$q_0 \rightarrow$ Initial State $(q_0 \in Q)$

$F \rightarrow$ Set of final states. $(q_F \subseteq Q)$

current state, Read symbol → Resultant State



(K|k, 0) → Thi

(K|k, 1) → K|k

(Thi, 0) → Rsma

(Thi, 1) → Thi

	0	1
K|k	Thi	K|k
Thi	RSm	Thi
Rs		
Sa		

NFA : Non-deterministic Finite Automata

They Term Non-deterministic Transfer the fact that any one of the Transsion Function may have more than one Resultant States.

They NFA is also QuinTuples (5-Tuple) $N = (Q, \Sigma, \delta, q_0, F)$

$Q \rightarrow$ Finite set of states

$\Sigma \rightarrow$ Finite set of i/p symbols

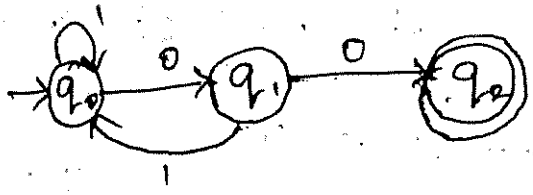
$\delta \rightarrow$ Transition Function $Q \times \Sigma \rightarrow Q'$

$q_0 \rightarrow$ Initial State $(q_0 \in Q)$

$F \rightarrow$ Set of Final State $(q_F \subseteq Q)$

Design the DFA that accept the following languages over the alphabet $\{0, 1\}$.

1) The set of all strings ending with 00



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

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

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

$q_0 = \{q_0\}$

$F = \{q_2\}$

and δ is given by

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_0$

DFA → one current state one i/p symbol one result state

NFA → two result state

Let q_0 be the Initial State the condition given in the languag is every string ended with two consecutive 0 must be accepted by the DFA Therfore The final state of the DFA must be reachable only with to consecutive 0 at the end of strings Therfore the Transition diagram can be drawn as follows.

Therefore the requiring DFA

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

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

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

$$q_0 = \{q_0\}$$

$$F = \{q_2\}$$

& δ is given by

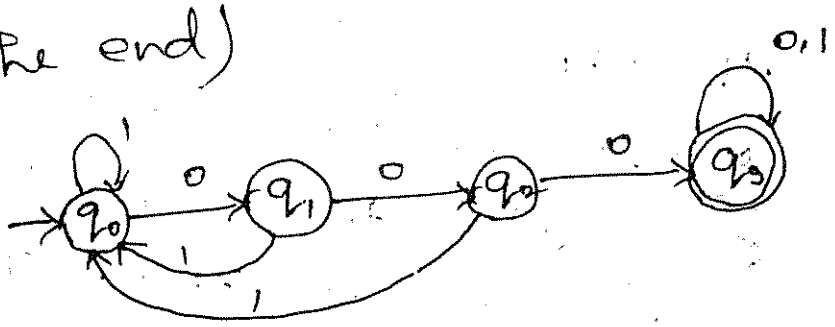
$$\delta(q_0, 0) = q_1$$

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

$$\delta(q_1, 0) = q_2$$

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

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



~~M = {~~

∴ The requiring DFA

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

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

DFA

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

& δ is given by

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_3$

$\delta(q_2, 1) = q_0$

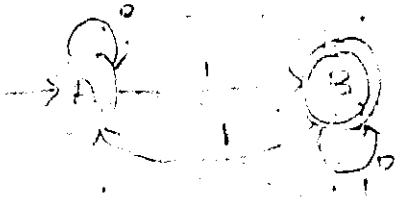
$\delta(q_3, 0) = q_3$

$\delta(q_3, 1) = q_3$

Consider the DFA

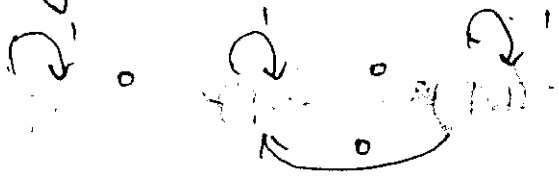
	0	1
A	B	A
*B	B	B

Describe the language accepted by DFA.



Since the Initial state can be Transition to Final state with atleast one '1' therefore language can be described as the set of all strings with atleast one '1'.

constructed at DFA that accept all strings in $(0,1)^*$ having even no: of zeros



3

usually

0,1

1

∴ They Requiding DFA, the given language is

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

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

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

$$q_0 = \{q_0\}$$

$$F = \{q_2\}$$

and δ is given by

$$\delta(q_0, 0) = q_1$$

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

$$\delta(q_1, 0) = q_2$$

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

$$\delta(q_2, 0) = q_1$$

$$\delta(q_2, 1) = q_2$$

Equivalence of NFA & DFA

Theorem:

Let 'L' be a set of language accepted by NFA then Thei exists at DFA That accepts L.

Proof:

Subset construction Method

the

Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

Then the set of states in NFA & DFA has the Relationship of

$Q_D = 2^{Q_N}$

To create The Transtion Functi of DFA

$$\delta'_D(Q_N, \Sigma) = \delta'_N(Q_N, \Sigma)$$

Forexi

$\delta_N(q_0, i) = \{P_1, P_2, P_3\}$

Then

$\delta_D(q_0, i) = [P_1, P_2, P_3]$

They Final State of DFA is the set of all states containing the Final state of NFA in bit.

∴ The Theorem hence proved

then

NFA → DFA:

Given a NFA $m = \{Q, \Sigma, \delta, q_0, F\}$

where δ is given by

Method

1) convert to DFA ~~A = {x}~~ the Following NFA.

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

where δ is given by

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

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

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

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

{ } - NFA
[ ] - DFA

Sol:

Step 1 :- Draw the Transition Table for given NFA.

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$*q_1$	ϕ	$\{q_0, q_1\}$

Step 2 :- Initial State of DFA equal to Initial State of NFA.

∴ Initial state of DFA = $[q_0]$

Step 3 :- Find the Transition of Initial State

$\delta([q_0], 0) = [q_0, q_1]$ ---- New State

$\delta([q_0], 1) = [q_1]$ ---- New State

Step 4:- Find the Transitions of newly Transident State.

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

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

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

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

Step 5:- Since There are no other ne
States they resultant DFA can b
written as $M = (Q_D, \Sigma, \delta, [q_0], f_D$

Where Q_D = Finite Set of States

$$Q_D = \{[q_0], [q_0,q_1], [q_1]\}$$

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

Initial state = $[q_0]$

$$f_D = \{[q_0,q_1], [q_1]\}$$

& δ is given by

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

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

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

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

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

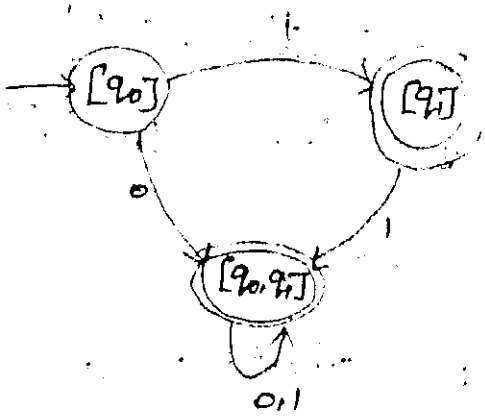
$\delta([q_1], 1) = [q_0, q_1]$

Construct the DFA for the following

Step 6: Draw the Transition Table for the resultant DFA.

	0	1
→ $[q_0]$	$[q_0, q_1]$	$[q_1]$
* $[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
* $[q_1]$	ϕ	$[q_0, q_1]$

Step 7:



Construct the DFA for the following

	0	1
→P	{P,q}	{P}
q	{r}	{r}
r	{s}	∅
*s	{s}	{s}

Step 1: Draw the Transition Table for given NFA.

	0	1
→P	{P,q}	{P}
q	{s}	{r}
r	{s}	∅
*s	{s}	{s}

Step2: Initial State of DFA equal to Initial state of NFA.

Initial State of DFA = [P]

Step 3: Findout the Transitions of Initial State

$\delta([P], 0) = [P, q]$ ----> newsta

$\delta([P], 1) = [P]$

Step 4: Find the Transition newly Transition State

$\delta([P,q],0) = \delta'(P,0) \cup \delta'(q,0)$

$= [P,q] \cup [r]$

$= [P,q,r]$ ----> new state

$\delta([P,q],1) = \delta'(P,1) \cup \delta'(q,1)$

$= [P] \cup [r]$

$= [P,r]$ ----> new state

$\delta([P,q,r],0) = \delta'(P,0) \cup \delta'(q,0) \cup \delta'(r,0)$

$= [P,q] \cup [r] \cup [s]$

$= [P,q,r,s]$ ----> new state

$\delta([P,q,r],1) = \delta'(P,1) \cup \delta'(q,1) \cup \delta'(r,1)$

$= [P] \cup [r] \cup \emptyset$

$= [P,r]$

$\delta([P,r],0) = \delta'(P,0) \cup \delta'(r,0)$

$= [P,q] \cup [s]$

$= [P,q,s]$ ----> new state

$\delta([P,r],1) = \delta'(P,1) \cup \delta'(r,1)$

$= [P] \cup \emptyset$

$= [P]$

$\delta([P,q,r,s],0) = \delta'(P,0) \cup \delta'(q,0) \cup \delta'(r,0) \cup \delta'(s,0)$

$= [P,q] \cup [r] \cup [s] \cup [s]$

$= [P, q, r, S]$

$\delta([P,q,r,S], 1) = \delta'(P,1) \cup \delta'(q,1) \cup \delta'(r$

$\cup \delta'(S,1)$

$= [P] \cup [r] \cup \phi \cup [S]$

$= [P, r, S]$ ----> new stat

$\delta([P,q,S], 0) = \delta'[P,0) \cup \delta'(q,0) \cup \delta'(S,$

$= [P,q] \cup [r] \cup [S]$

$= [P,q,r,S]$

$\delta([P,q,S], 1] = \delta'(P,1) \cup \delta'(q,1) \cup \delta'(S,$

$= [P] \cup [r] \cup [S]$

$= [P, r, S]$

$\delta([P,r,S], 0) = \delta'(P,0) \cup \delta'(r,0) \cup \delta'(S$

$= [P,q] \cup [S] \cup [S]$

$= [P,q,S]$

$\delta([P,r,S], 1) = \delta'(P,1) \cup \delta'(r,1) \cup \delta'(S$

$= [P] \cup \phi \cup [S]$

$= [P,S]$ ----> new state

$\delta([P,S], 0) = \delta'(P,0) \cup \delta'(S,0)$

$= [P,q] \cup [S]$

$= [P,q,S]$

$\delta([P,S],1) = \delta'(P,1) \cup \delta'(S,1)$

$= [P] \cup [S]$

$= [P,S]$

[Final state]

∴ The resultant Initial ~~state~~ no more new state.

$M = (Q_D, \Sigma, \delta, [q_0], F_D)$

$Q_D = 8$

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

$q_0 = P$

$F_D = \{[P,q,r,S], [P,q,S], [P,r,S], [P,S]\}$

and δ is Transition function,

Step 6: Draw the ~~resultant~~ Transition table for that resultant

Final stat)

Step

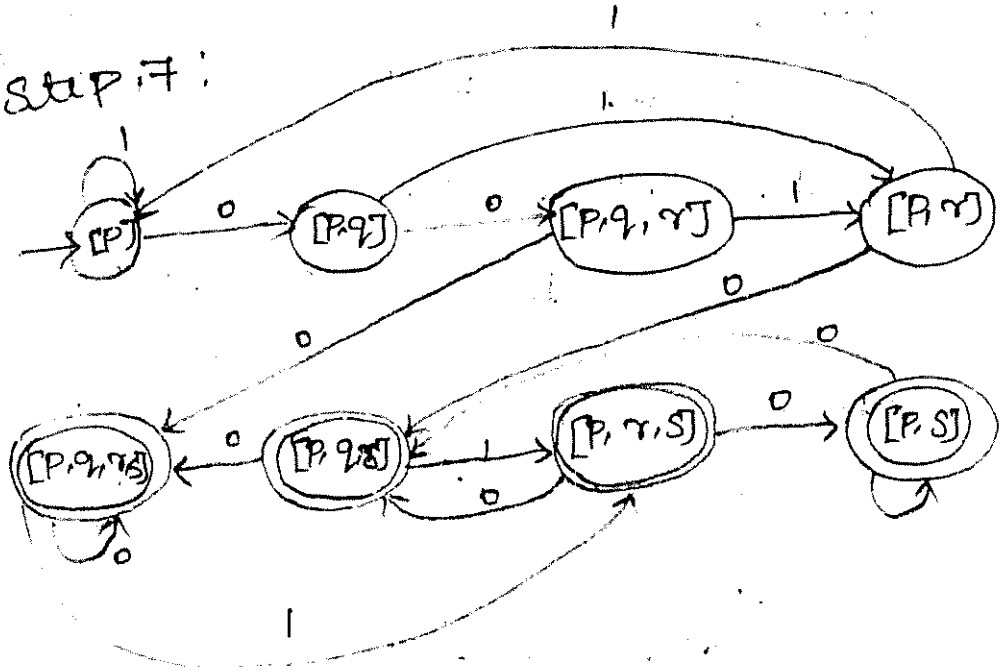
], [P, r, S]
S]}

tion,

xabl

	0	1
→ [P]	[P, Q]	[P]
[P, Q]	[P, Q, r]	[P, r]
[P, Q, r]	[P, Q, r, S]	[P, r]
[P, r]	[P, Q, S]	[P]
* [P, Q, r, S]	[P, Q, r, S]	[P, r, S]
* [P, Q, S]	[P, Q, r, S]	[P, r, S]
* [P, r, S]	[P, Q, S]	[P, S]
* [P, S]	[P, Q, S]	[P, S]

Step 7:



H.W

1)

	a	b
→ P	{P}	{P, Q}
Q	{r}	{r}
* r	∅	∅

2)

	0	1
→ a	{b,d}	{b}
*b	{c}	{b,c}
c	{d}	{a}
*d	ϕ	{a}

1) Step 1: Draw the Transition Table for given NFA

	a	b
→ p	{p}	{p,q}
q	{r}	{r}
*r	ϕ	ϕ

Step 2:

Initial State of DFA = Initial State of NFA therefore

∴ Initial state of DFA = [p]

Step 3:

Find the Transition of Initial State

$\delta([p], a) = [p]$

$\delta([p], b) = [p, q]$ ---> new State

Step 4:

Find the Transition of newly Transition Table corresponding State

$\delta([P,q],a) = \delta'[P,a] \cup \delta'[q,a]$

$= [P] \cup [r]$

$= [P,r]$ ---> new state

$\delta([P,q],b] = \delta'[P,b] \cup \delta'[q,b]$

$= [P,q] \cup [r]$

$= [P,q,r]$ ---> new state

$\delta([P,r],a) = \delta'[P,a] \cup \delta'[r,a]$

$= [P] \cup [\phi]$

$= P$

$\delta([P,r],b) = \delta'[P,b] \cup \delta'[r,b]$

$= [P,q] \cup \phi$

$= [P,q]$

$\delta([P,q,r],a) = \delta'[P,a] \cup \delta'[q,a] \cup \delta'[r,a]$

$= [P] \cup [r] \cup \phi$

$= [P,r]$

$\delta([P,q,r],b) = \delta'[P,b] \cup \delta'[q,b] \cup \delta'[r,b]$

$= [P,q] \cup [r] \cup \phi$

$= [P,q,r]$

No. of state = 4.

Step 5:

Since there are no other new state the resultant DFA can be

written as $M = (Q_D, \Sigma, \delta, q_0, F_D)$

where $Q_D = \{[P], [P,Q], [P,R], [P,Q,R]\}$

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

Initial state = $\{[P]\}$

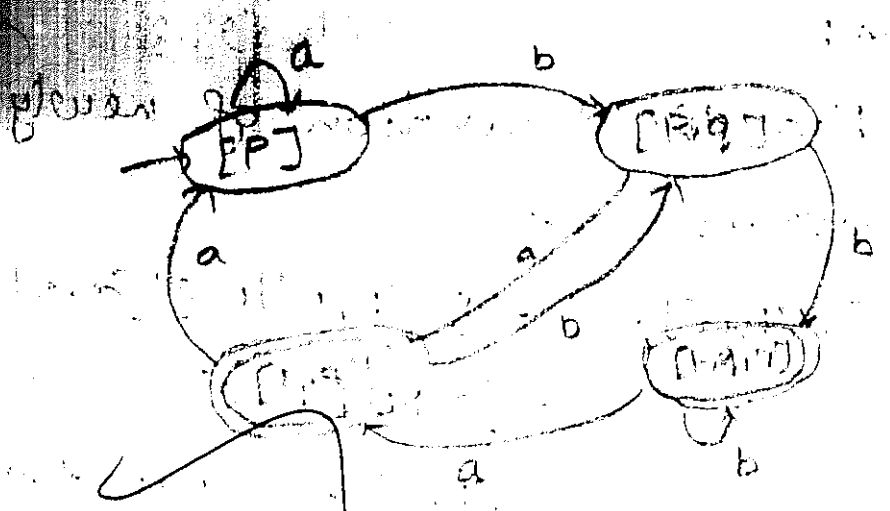
δ = Transition function

$F_D = \{[P,Q,R,S], [P,Q,S], [P,R,S], [P,S]\}$

Step 6:

Draw the Transition Table for the resultant DFA

	a	b
→ [P]	[P,Q]	[P,Q]
[P,Q]	[P,Q,R]	[P,Q,R]
[P,Q,R]	[P,Q,R,S]	[P,Q,R]
[P,R]	[P,Q,S]	[P,Q]
* [P,Q,R,S]	[P,Q,R,S]	[P,R,S]
* [P,Q,S]	[P,Q,R,S]	[P,R,S]
* [P,R,S]	[P,Q,S]	[P,S]
* [P,S]	[P,Q,S]	[P,S]



2) Step 1:

Draw the Transition of given NFA

	0	1
→a	{a,d}	{b}
*b	{c}	{b,c}
c	{d}	{a}
*d	ϕ	{a}

step 2:

The Initial State of DFA = The Initial State of NFA

∴ Initial state of DFA = [a]

step 3:

Findout the Transition of Initial State

δ [a], 0) = [b,d] ---> newstat

δ ([a], 1) = [b] ---> newstat

Step4:

Find the Transition of newly Transition State.

$\delta([b,d],0) = \delta'[b,0] \cup \delta'[d,0]$

$= [c] \cup [b,c]$

$= [b,c]$ ---> newstate

$\delta([b,d],1) = \delta'[b,1] \cup \delta'[d,1]$

$= [c] \cup \emptyset$

$= [c]$ ---> newstate

$\delta([b,1]) = [b,c]$

$\delta([b,0)] = \delta'[b,0)$

$= [c]$

$\delta[(b,c),0] = \delta'[b,0] \cup \delta'[c,0]$

$= [c] \cup [d]$

$= [c,d]$ --> new state

$\delta'(c,1) = [a]$

$\delta[(b,c),1] = \delta'[b,1] \cup \delta'[c,1]$

$= [b,c] \cup [a]$

$= [b,c,a]$ ---> newstate

$\delta[(c,d),0] = \delta'[c,0] \cup \delta'[\emptyset,0]$

$= [d] \cup \emptyset$

$= [d]$ ---> newstate

$\delta[(c,d),1] = \delta'[c,1] \cup \delta'[d,1]$

$= [a] \cup [a]$

$= [a]$ ----> new State

$\delta([b,c,a],0) = \delta'[b,0] \cup \delta'[c,0] \cup \delta'[a,0]$

$= [c] \cup [d] \cup [b,d]$

$= [c,b,d]$ ---> new State

$\delta([b,c,a],1) = \delta'[b,1] \cup \delta'[c,1] \cup \delta'[a,1]$

$= [b,c] \cup [a] \cup [b]$

$= [b,c,a]$

$\delta[d,0] = \phi$

$\delta[d,1] = [a]$

$\delta[a,0] = [b,d]$

$\delta[a,1] = [b]$

$\delta([c,b,d],0) = \delta'[c,0] \cup \delta'[b,0] \cup \delta'[d,0]$

$= [d] \cup [c] \cup \phi$

$= [d,c]$ ---> new State

$\delta([c,b,d],1) = \delta'[c,1] \cup \delta'[b,1] \cup \delta'[d,1]$

$= [a] \cup [b,c] \cup [a]$

$= [a,b,c]$ ---> new state

$$\delta([d,c],0) = \delta'[d,0] \cup \delta'[c,0]$$

$$= \phi \cup [d]$$

$$= [d]$$

$$\delta([d,c],1) = \delta'[d,1] \cup \delta'[c,1]$$

$$= [a] \cup [a]$$

$$= [a]$$

$$\delta([a,b,c],1) = \delta'(a,1] \cup \delta'[b,1] \cup \delta'[c,1]$$

$$= [b] \cup [b,c] \cup [a]$$

$$= [b,c,a]$$

$$\delta([a,b,c],0) = \delta'[a,0] \cup \delta'[b,0] \cup \delta'[c,0]$$

$$= [b,d] \cup [c] \cup [d]$$

$$= [b,d,c]$$

$$\delta([b,d,c],0) = \delta'[b,0] \cup \delta'[d,0] \cup \delta'[c,0]$$

$$= [c] \cup \phi \cup [d]$$

$$= [c,d] \dashrightarrow \text{new state}$$

$\delta([c,d],0) = \delta'[c,0] \cup \delta'[d,0]$

$= [d]$

$\delta([c,d],1) = \delta'[c,1] \cup \delta'[d,1]$

$= [a] \cup [a]$

$= [a]$

$\delta([c,d],1) = [a]$

∴ The result initial number and there is no more new state.

Step5:

Since there are no more new state the resultant DFA can be written as $M = (Q_D, \Sigma, \delta_D, q_0, F_D)$

Where,

$Q_D = 10$

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

$q_0 = a$

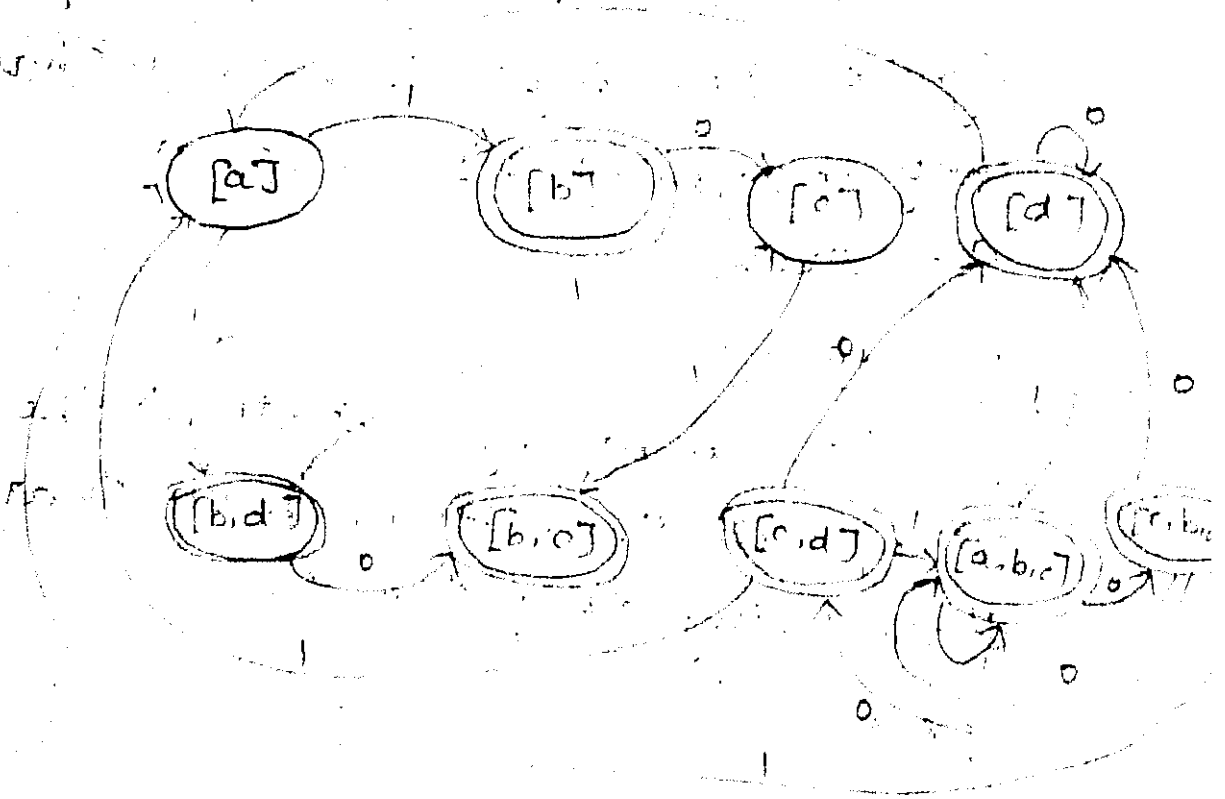
$F_D = 11$

Final State is '4'

δ is a transition functions.

Step 6: Draw the Transition Table for that resultant.

	0	1
→[a]	[b,d]	[b]
*[b,d]	[b,c]	[c]
*[b]	[c]	[b,c]
*[b,c]	[c,d]	[b,c,a]
*[c,d]	[d]	[a]
*[b,c,a]	[c,b,d]	[b,c,a]
*[d]	∅	[a]
*[c,b,d]	[d,c]	[a,b,c]
*[d,c]	[d]	[a]
*[a,b,c]	[b,d,c]	[b,c,a]
*[b,d,c]	[c,d]	[d]
*[c,d]	[d]	[a]


[a]
[b]
[c]
[d]
[b,d]
[b,c]
[c,d]
[a,b,c]

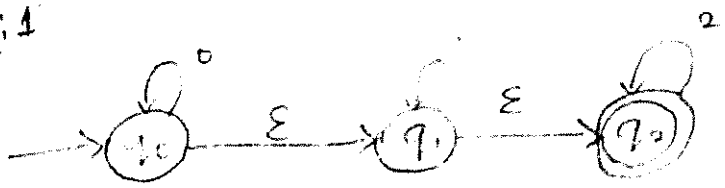
Finite Automata with ε - Transition (ε-NFA)

→ It is possible in NFA that an NFA is allowed to make Transitions without receiving any i/p Symbol spondiesly this move without any i/p symbol is called as ε-move.

ε - closure :

ε-closure of a state is the set of all Transitions from that State using only the ε.

Ex : 1

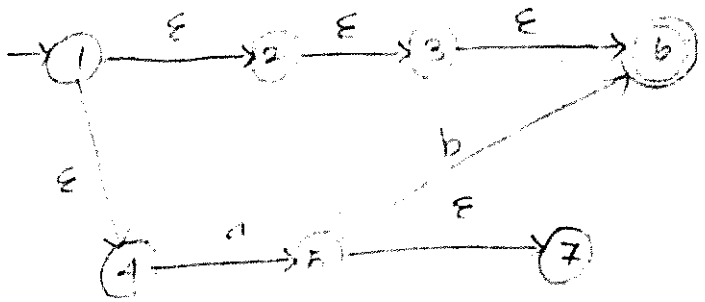


ε - closure $(q_0) = \{q_0, q_1, q_2\}$

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

Ex : 2)



Find ε-closure of 1, 2, 5

ε-closure of (1) = {1, 2, 3, 6, 4}

ε-closure of (2) = {2, 3, 6}

ε-closure of (5) = {5, 7}

Eliminating ε-Transitions:

- It is possible to construct at DFA 'T' that accept same langua as ε-NFA. Then we can construct $D = (Q_D, \Sigma, \delta_D, q_0, F_D$ From $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$

Steps to construct DFA

Step 1: Draw the Transitions Table the given ε-NFA.

Step 2: The Initial State of DFA equal to ε-closure (Initial Sta of ε-NFA.)

Step 3:

Find the Transitions (δ) For newly the arrived States Unt ther are no other new states.

Step 4: Find F_D

Step 5: Draw the resultant DFA

with Transitions Table and diagram.

Equivalence of ε-NFA & DFA:

Theorem:

A language L is accepted by some ε-NFA if and only if L is accepted by some DFA.

proof:

i) If part:

Assume that the language L is accepted by the DFA 'D' this is possible by converting the DFA to ε-NFA by adding $\delta(q, \varepsilon) = \phi$ for all states 'q' in D. provided every Transition $\delta_D(q, a) = P$ of DFA as $\delta_E(q, a) = \{P\}$

Thus the Transitions of E & D all Sa where E & D states that thayare no Transition outof any states on ε.

ii) only If :

Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ε-NFA, we have to prove that

$L(E) = L(D)$ (w→s

where $D = (Q_D, \Sigma, \delta_D, q_0, f_D)$

It is proved by the statement

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

By using Inductive hypothesis

Basis:

If $|w| = 0$ then $w = \varepsilon$

$$\hat{\delta}_E(q_0, \varepsilon) = \varepsilon\text{-closure}(q_0)$$

$$\hat{\delta}_D(q_0, \varepsilon) = \varepsilon\text{-closure}(q_0)$$

∴ It is True For $|w| = 0$

Induction:

Let $|w| = n+1$

$w = xa$ where 'a' is the final symbol of w and $|x| = n$

Also Assume That

$$\hat{\delta}_E(q_E, x) = \hat{\delta}_D(q_D, x) = \{P_1, P_2 \cdots F$$

By the definition of δ

$$\delta_E(q_0, w) = \bigcup_{i=1}^{M} \varepsilon\text{-closure}(r_i) \text{ if}$$

$$\bigcup_{i=1}^{M} \delta_E(p_i, a) = \{r_1, r_2 \ldots r_M\}$$

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

$$= \delta(q_0, q_1, q_2)$$

We can that $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_0, w)$

The statement is True for n+1

By the Inductive hypothesis the

Statement is True for all the string of L therfore the language the language accepted In DFA

$\delta([q_0,$

Design a DFA that eliminates ε-Transit

From the given ε-NFA.

Step 1: Draw the Transition Table for given ε-NFA.

	ε	0	1	2
$\rightarrow q_0$	q_1	q_0	ϕ	ϕ
q_1	q_2	ϕ	q_1	ϕ
$*q_2$	ϕ	ϕ	ϕ	q_2

Step 2: We find Initial state of DFA =

ε-closure (Initial state (ε-NFA))

Initial state (ε-NFA) = q_0

∴ Initial state (DFA) = ε-closure(q_0

= $[q_0, q_1, q_2]$ → Initial st

$\delta([q_0, q_1, q_2], 0)$ = ε-closure ($\delta'(q_0, q_1, q_2), 0)$)

= ε-closure ($\delta'(q_0, 0) \cup \delta'(q_1, 0) \cup \delta'(q_2, 0)$)

= ε-closure (q_0)

= $[q_0, q_1, q_2]$

$\delta([q_0, q_1, q_2], 1)$ = ε-closure ($\delta'((q_0, q_1, q_2), 1)$

= ε-closure ($\delta'(q_0, 1) \cup \delta'(q_1, 1) \cup \delta'(q_2, 1)$)

= ε-closure (q_1)

= $[q_1, q_2]$ ---> new state

$\delta([q_0, q_1, q_2], 2)$ = ε-closure ($\delta'((q_0, q_1, q_2), 2)$)

= ε-closure ($\delta'(q_0, 2) \cup \delta'(q_1, 2) \cup \delta'(q_2, 2)$)

= ε-closure (q_2)

= $[q_2]$ ---> new state

Step 3:-

$\delta([q_1, q_2], 0) = \varepsilon\text{-closure}(\delta'(q_1, 0) \cup \delta'(q_2, 0))$

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

$= \phi$

$\delta([q_1, q_2], 1) = \varepsilon\text{-closure}(\delta'(q_1, q_2), 1)$

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

$= [q_1, q_2]$

$\delta([q_1, q_2], 2) = \varepsilon\text{-closure}(\delta'(q_1, q_2), 2)$

$= \varepsilon\text{-closure}(q_2)$

$= [q_2]$

$\delta([q_2], 0) = \varepsilon\text{-closure}(\delta'(q_2, 0))$

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

$= \phi$

$\delta([q_2], 1) = \varepsilon\text{-closure}(\delta'(q_2, 1))$

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

$= \phi$

$\delta([q_2], 2] = \varepsilon\text{-closure}(\delta'(q_2, 2))$

$= \varepsilon\text{-closure}(q_2)$

$= [q_2]$

Step 4: Find the final state of DFA

Since q_2 is the final state of NFA

They Final State of DFA are,

$[q_0, q_1, q_2]$, $[q_1, q_2]$, $[q_2]$

Step 5:

The Resultant DFA is

$$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$

$Q_D = \{[q_0, q_1, q_2], [q_1, q_2], [q_2]\}$

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

δ_D = Transition State of DFA

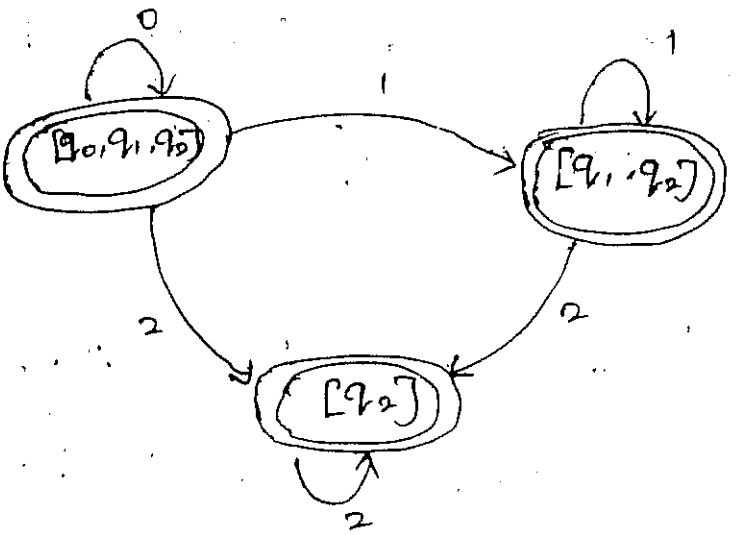
q_0 Initial State = $[q_0, q_1, q_2]$

$F_D = \{[q_0, q_1, q_2], [q_1, q_2]$

Step 6:

	0	1	2
→ * $[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$	$[q_2]$
* $[q_1, q_2]$	ϕ	$[q_1, q_2]$	$[q_2]$
* $[q_2]$	ϕ	ϕ	$[q_2]$

Step 7:



∴ The resultant DFA has be construction by eliminates the ε- Transition from the given NFA.

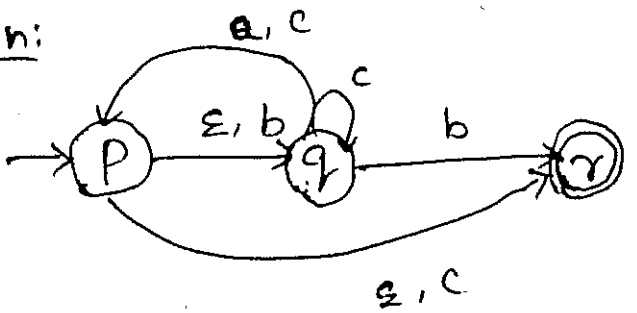
Consider the following ε - NFA ~~to~~

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

i) Compute ε - closure of P, q, r

ii) Convert it into DFA.

Soln:



ε-closure(P) = {P, q, r} --> Initial State

ε-closure(q) = {q}

ε-closure(r) = {r}

Step 1: Draw the Transition Table For ε-NFA.

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

Step 2: We Find the Initial State of DFA = ε-closure (Initial State (ε-NFA))

Initial State (ε-NFA) = P

∴ Initial State (DFA) = ε-closure (P

= [P, q, r] --> Initial State

δ ([P, q, r], a) = ε-closure (δ'(P, q, a)

= ε-closure (δ'(P, a) ∪ δ'(q, a) ∪ δ'(r, a)

= ε-closure (P)

= {P, q}

$\delta([P,q,r],b) = \varepsilon\text{-closure}(\delta'(P,q,r),$

$= \varepsilon\text{-closure}(\delta'(P,b) \cup$

$\delta'(q,b) \cup \delta'(r,b))$

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

$= [q]$

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

$= \varepsilon\text{-closure}\{q,r\} \rightarrow$ new state

$\delta([P,q,r],c) = \varepsilon\text{-closure}$

$(\delta'(P,q,r),c)]$

$= \varepsilon\text{-closure}(\delta'(P,c) \cup$

$\delta'(q,c) \cup \delta'(r,c))$

$= \varepsilon\text{-closure}(\{r\} \cup \{P,q\} \cup$

$\phi)$

$= \varepsilon\text{-closure}\{P,q,r\}$

$= \{P,q,r\}$

Step 3:

$\delta([q],a) = \{q,a\} \cup \{r,a\}$

$= \varepsilon\text{-closure}(\delta'(q,a) \cup$

$\delta'(r,a))$

$= \varepsilon\text{-closure}\{p\} \cup \{\phi\}$

$= \{p\}$

$\delta([q,r],b) = \varepsilon\text{-closure}(\delta'(q,b) \cup \delta(r,b))$

$= \varepsilon\text{-closure}\{r\} \cup \{\phi\}$

$= \{r\} \longrightarrow$ new state

$\delta([q,r],c) = \varepsilon\text{-closure}(\delta'(q,c) \cup \delta'(r,c))$

$= \varepsilon\text{-closure}\{p,q\} \cup \{\phi\}$

$= \{p,q\} \longrightarrow$ new state

Step 4:

$(\{r,a\}) = \phi \Rightarrow \varepsilon\text{-closure}(\delta'(r,a))$

$(\{r,b\}) = \phi \Rightarrow \varepsilon\text{-closure}(\delta'(r,b))$

$(\{r,c\}) = \phi \Rightarrow \varepsilon\text{-closure}(\delta'(r,c))$

Step 5:

$([p,q],a) = \varepsilon\text{-closure}(\delta'(p,a) \cup \delta'(q,a))$

$= \varepsilon\text{-closure}\{\phi\} \cup \{p\}$

$= \{p\}$

$([p,q],b) = \varepsilon\text{-closure}(\delta'(p,b) \cup \delta'(q,b))$

$= \varepsilon\text{-closure}\{q\} \cup \{r\}$

$= \{q,r\}$

$([p,q],c) = \varepsilon\text{-closure}(\delta'(p,c) \cup \delta'(q,c))$

$= \varepsilon\text{-closure}\{r\} \cup \{p,q\}$

$= \{p,q,r\}$

Step6 : To Find the Final State of DFA
Since 'r' is the Final state

The Final State of DFA is
$[P, q, r]$, $[q, r]$, $[r]$

Step 7 :
To Find the Result DFA is

$D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$Q_D = \{[P, q, r], [q, r], [r]\}$

$\Sigma = [a, b, c]$

$q_0 = [P, q, r]$

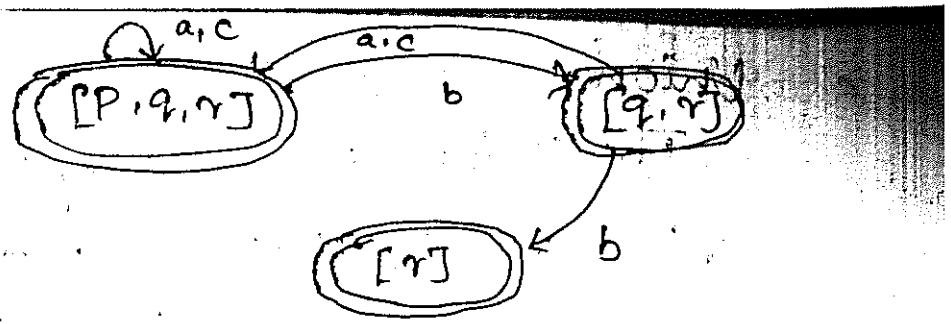
$F_D = \{[P, q, r], [q, r], [r]\}$

Step8 :
Draw the Transition Table for resultant DFA.

	a	b	c
→ * $[P, q, r]$	$[P, q, r]$	$[q, r]$	$[P, q, r]$
* $[q, r]$	$[P, q, r]$	$[r]$	$[P, q, r]$
* $[r]$	ϕ	ϕ	ϕ

Draw the Transition diagram.

DFA



UNIT-II

Regular Expressions & languages

Regular Expressions:

Definition:

The languages accepted by an finite automata can be easily described by simple expressions are Terminals call Regular expressions.

operations of Regular expression

Three operations on language

i) Union.

ii) concatenation

iii) closure.

Union:

The Union of 2 languages L_1 & L_2 is given by $L_1 \cup L_2$ and its the set of all strings either in L_1 or L_2

Ex:

$L_1 = \{0, 111, 110\}$

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

$L_1 \cup L_2 = \{0, 111, 110, \varepsilon, 01\}$

Concatenation:

The Concatenation of Two languages L_1 & L_2 is given by $L_1 \cdot L_2$ (or) $L_1 L_2$ and its formed by choosing each String from L_1 & concatenate it with all the Strings of L_2.

$\therefore L_1 \cdot L_2 = \{xy \mid x$ is from L_1 & y is from $L_2\}$

$L_1 = \{10, 1\}$

$L_2 = \{011, 11\}$

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

$= \{10011, 1011, 111\}$

$L_2 \cdot L_1 = \{0111, 0111, 1110, 111\}$

closure:

We have Two Types.

i) Kleene closure (L^*)

ii) positive closure.

Kleene closure:

The Kleene closure of language of 'L' denoted by L^* defined as the set of all strings that can be formed by taking any no. of string from 'L'.

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

positive closure:

The positive closure of 'L' denoted by L^+ is defined as set of all strings that can be formed by taking any no. of strings from 'L' expect ε.

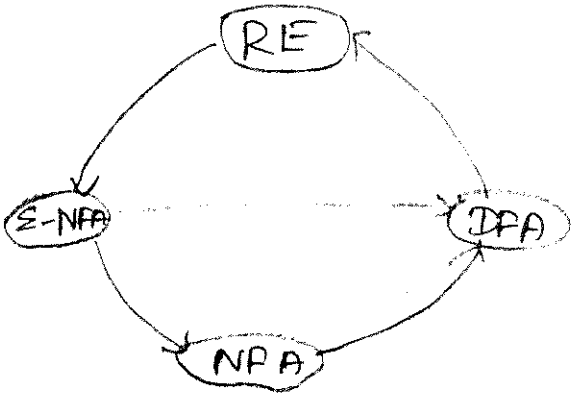
$L = \{0, 1\}$

$L^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$

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

Constructions of Regular Expressions:

We have four Types.



Thompson's construction Method :-

Union : $A \cup B \rightarrow A + B$

$+ \rightarrow$ or

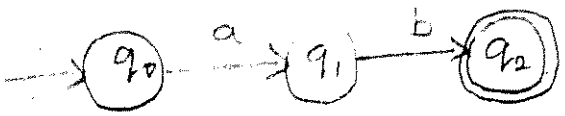
Concatenation : $A \cdot B$ $a \cdot b$

$\cdot \rightarrow$ and

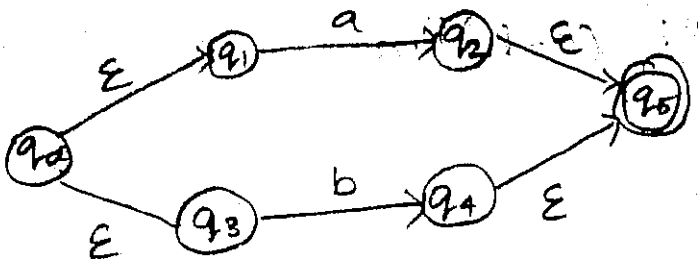
closure : $*$

a^*

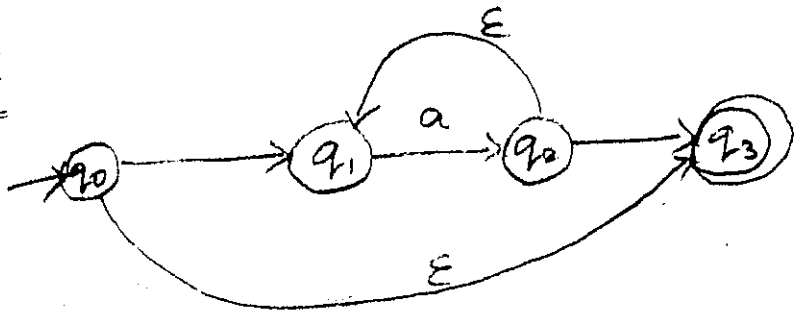
$a \cdot b$



a+b



a*



1) Construct the ε-NFA for the given Regular expression

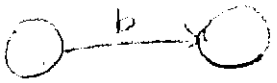
(a+b)*ab

Sol:

Step 1:

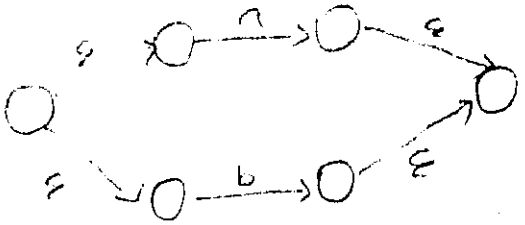
a =>

b =>



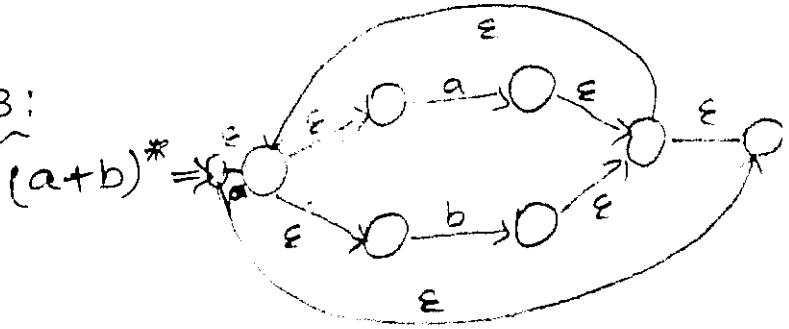
Step 2:

a+b =>

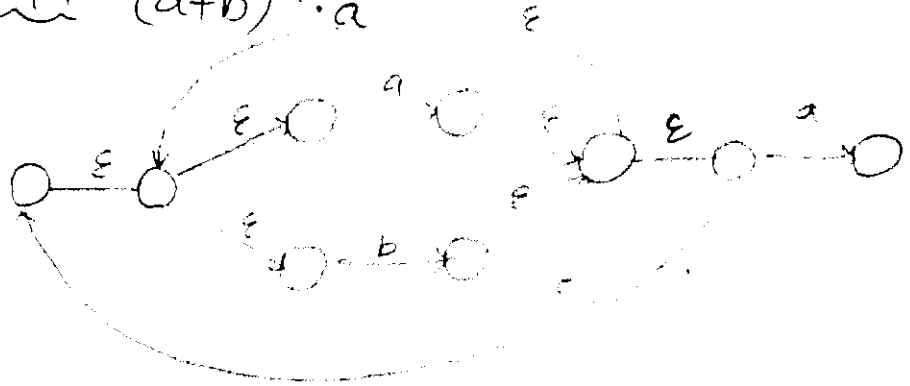


Step 3:

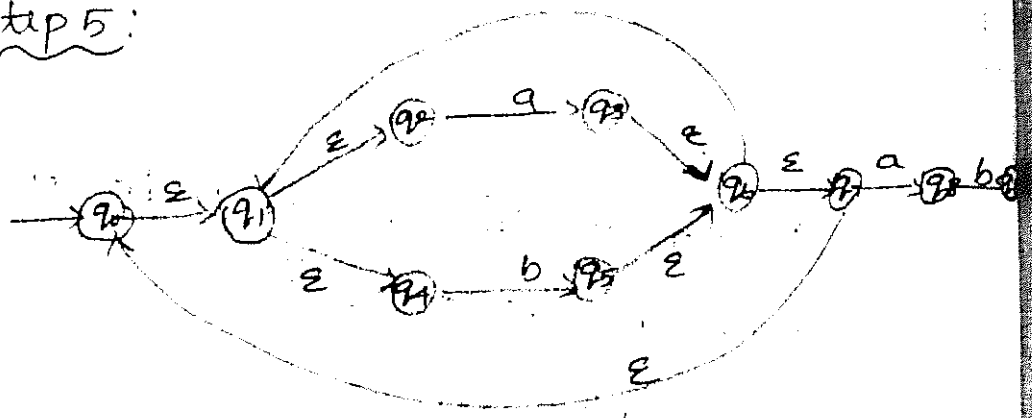
(a+b)* =>



Step 4: $(a+b)^*.a$



Step 5:



H.w

1) $abc^*(a+b)^*$

ii) $(a+b)+c^*$

iii) $(a*b+c)^*+d$

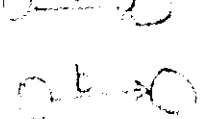
i) $abc^*(a+b)^*$

Step 1:

c^*

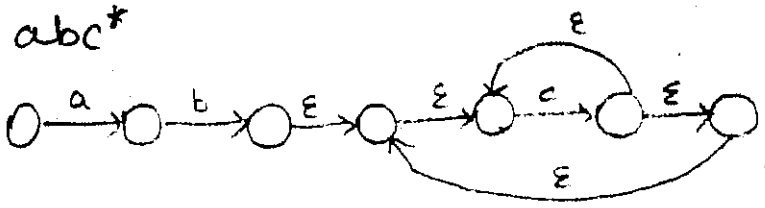
Step 2:

ab



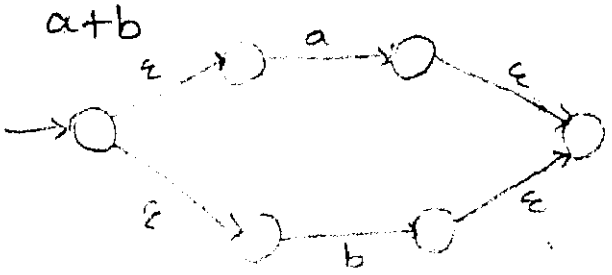
Step 3:

abc^*



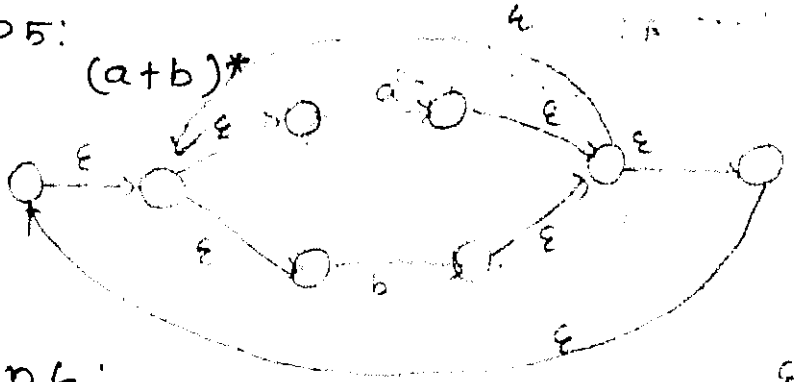
Step 4:

$a+b$



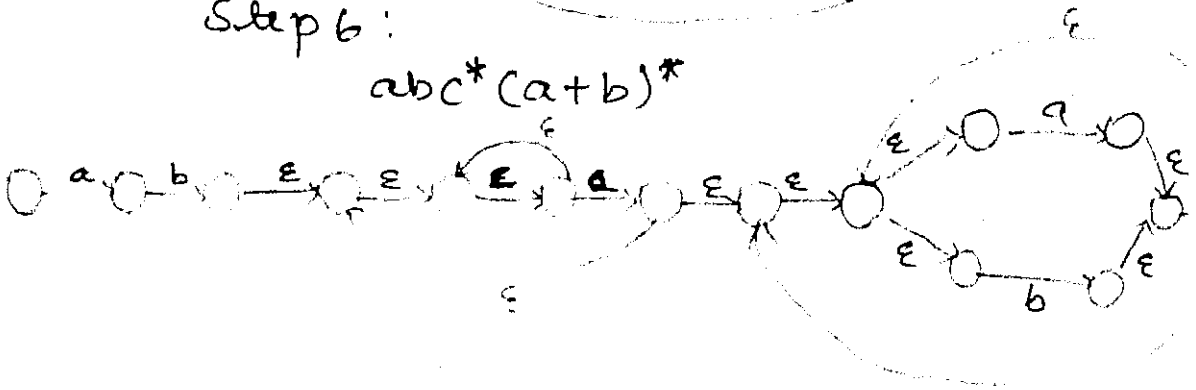
Step 5:

$(a+b)^*$

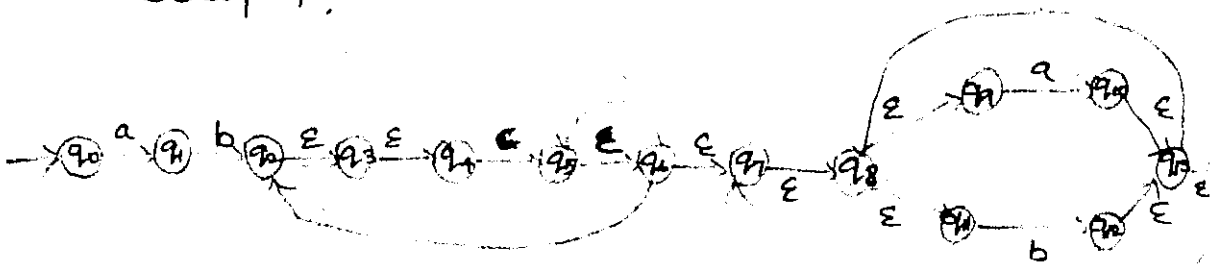


Step 6:

$abc^*(a+b)^*$



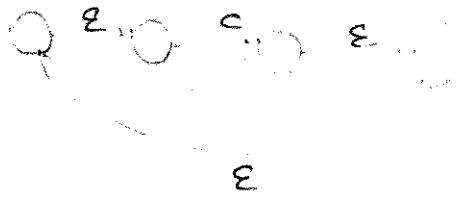
Step 7:



2) $(a+b)+c^*$

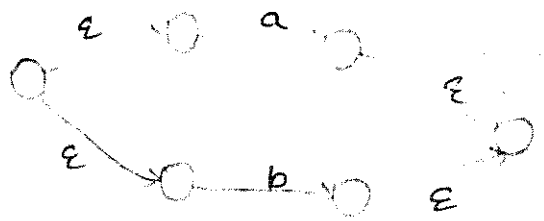
Step 1:

c^*



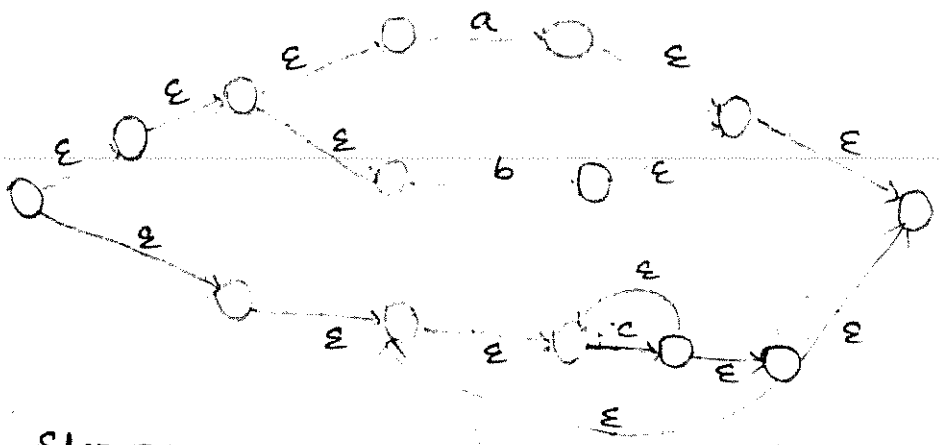
Step 3:

$a+b$

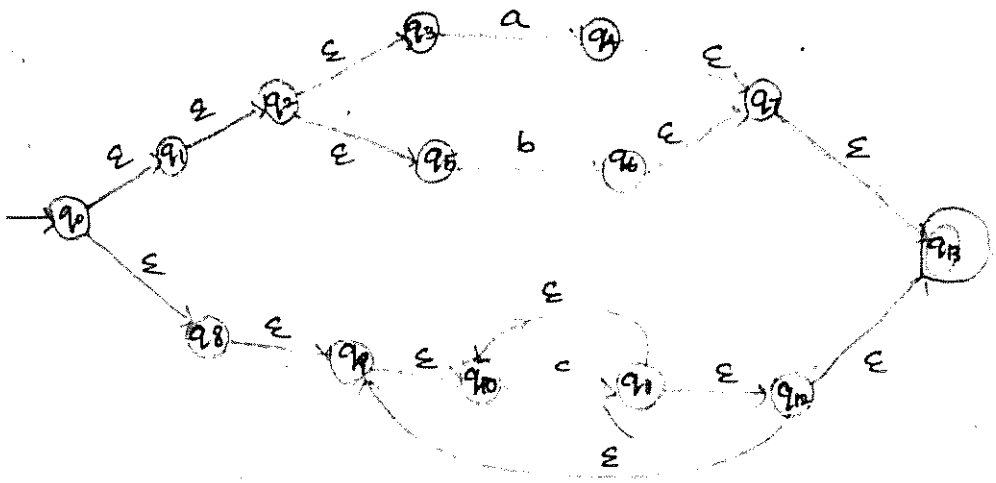


Step 4:

$(a+b)+c^*$



Step 5:

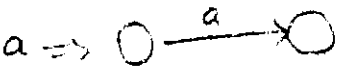


3) $(ab+c)^* + d$

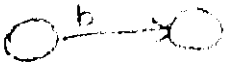
Step 1:

ab

a =>

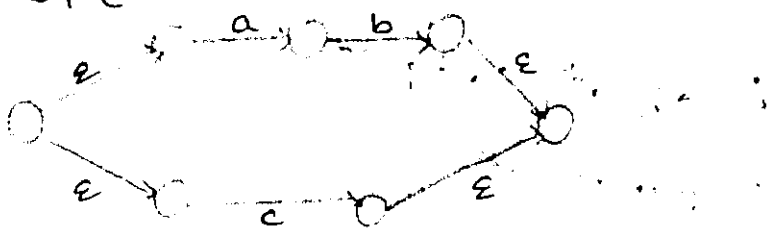


b =>



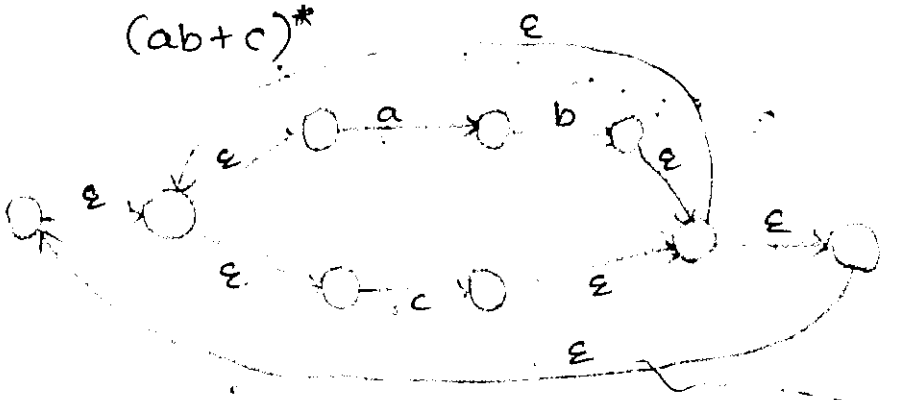
Step 2:

ab + c

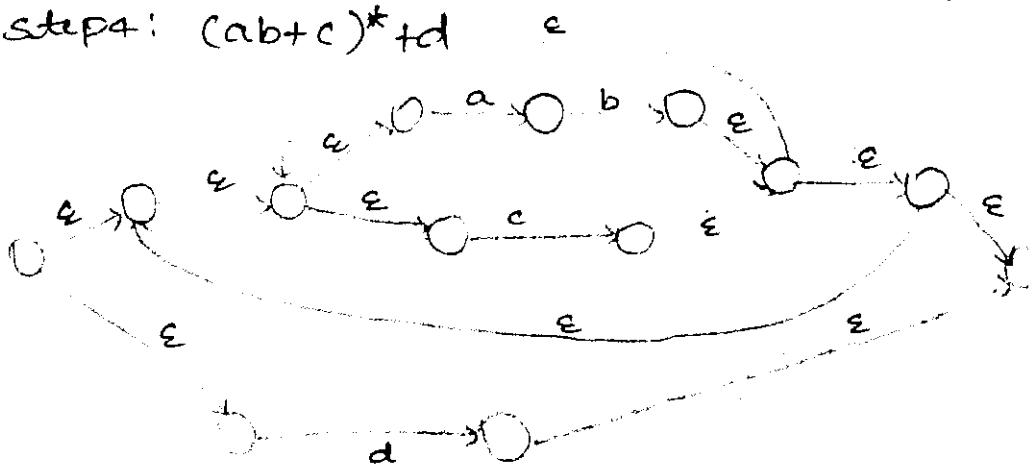


Step 3:

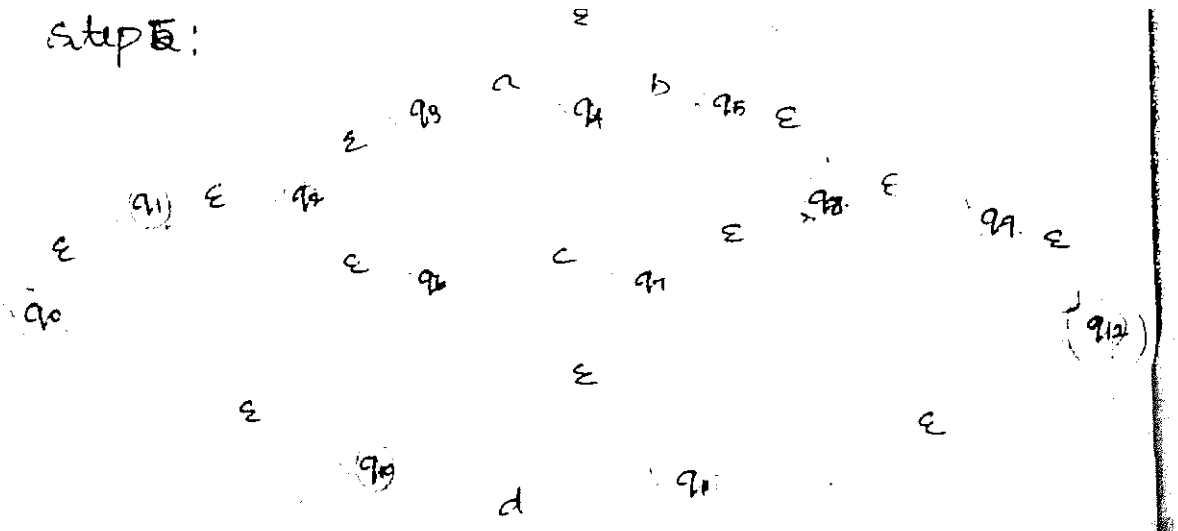
$(ab+c)^*$



Step 4: $(ab+c)^* + d$

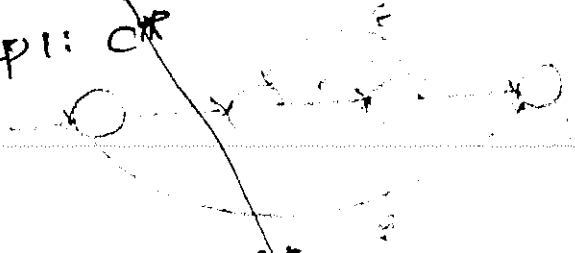


Step 5:

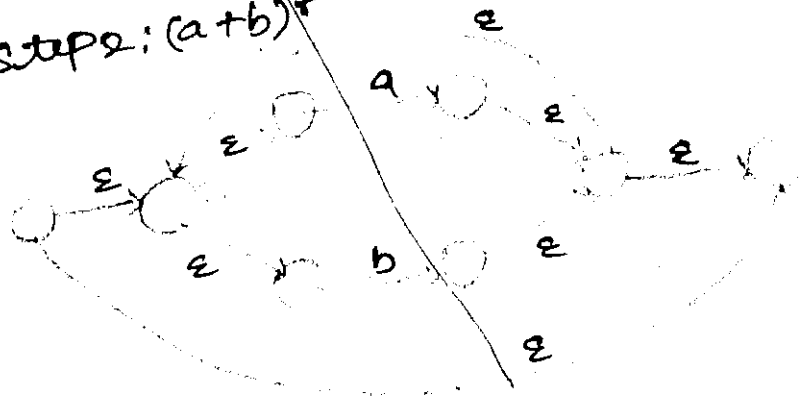


1) $abc^*(a+b)^*$

step 1: c^*



step 2: $(a+b)^*$

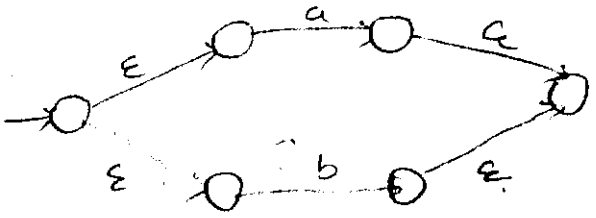


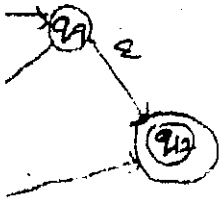
Smart Regular expression to DFA:

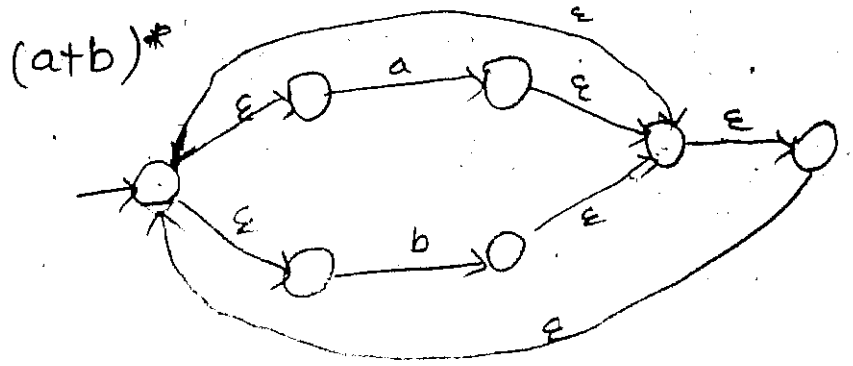
$(a+b)^*ab$

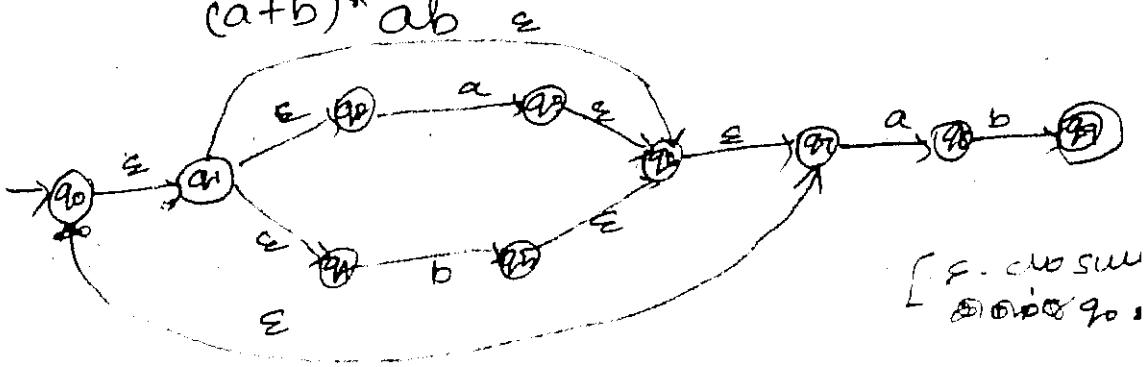
step 1: convert the given Regular expression into ε-NFA.

$a+b$









Step 2: convert the ε-NFA into DFA.

Initial state of DFA = ε-closure (Initial state (ε-NFA))

= ε-closure(q_0)

= $[q_0, q_1, q_2, q_4, q_7]$ → A

∴ Initial State of DFA = A

MOVE (A, a) = ε-closure ($\delta'(q_0, a) \cup \delta'(q_1, a) \cup \delta'(q_2, a) \cup \delta'(q_4, a) \cup \delta'(q_7, a)$)

= ε-closure ($\phi \cup \phi \cup q_3 \cup \phi \cup q_8$)

$= \varepsilon\text{-Closure}(q_3, q_8)$

$= [q_3, q_6, q_7, q_1, q_2, q_4, q_8]$

$= [q_1, q_2, q_3, q_4, q_6, q_7, q_8] \rightarrow B$

MOVE $(A, b) = \varepsilon$-closure $[(\delta'(q_0, b) \cup \delta'(q_1, b) \cup \delta'(q_2, b) \cup \delta'(q_4, b) \cup \delta'(q_7, b)]$

$= \varepsilon$-closure $(\phi \cup \phi \cup \phi \cup q_5 \cup \phi)$

$= \varepsilon$-closure(q_5)

$= [q_5, q_6, q_7, q_1, q_2, q_4]$

$= [q_1, q_2, q_4, q_5, q_6, q_7] \rightarrow C$

MOVE $(B, a) = \varepsilon$-closure $[(\delta'(q_0, a) \cup \delta'(q_1, a) \cup \delta'(q_2, a) \cup \delta'(q_4, a) \cup \delta'(q_7, a)]$

$= \varepsilon$-closure $(\phi \cup \phi \cup q_3 \cup \phi \cup q_8)$

$= \varepsilon$-closure(q_3, q_8)

$= [q_3, q_6, q_7, q_1, q_2, q_4, q_8]$

$= [q_1, q_2, q_3, q_4, q_6, q_7, q_8]$

MOVE (B, b) = ε - closure [(δ'(q0, b) ∪
δ(q1, b) ∪ δ'(q2, b) ∪ δ'(q4,
∪δ'(q6, b) ∪ δ'(q7, b) ∪ δ(
= ε - closure (ϕ ∪ ϕ ∪ ϕ ∪ ϕ ∪ q
ϕ
= ε - closure (q5, q9)
= [q5, q6, q7, q1, q2, q4, q9]
= [q1, q2, q4, q5, q6, q7, q9]
↳ D

MOVE (C, a) = ε - closure [(δ'(q0, a
∪ δ'(q1, a) ∪ δ'(q2, a) ∪ δ'(
∪ δ'(q7, a)]
= ε - closure [ϕ ∪ ϕ ∪ q3 ∪ ϕ ∪ q
= ε - closure [q3, q8]
= [q3, q6, q7, q1, q2, q4, q
= [q1, q2, q3, q4, q6, q7, q8]

MOVE (C, b) = ε-closure [(δ'(q1, b)
∪ δ'(q2, b) ∪ δ'(q3, b) ∪ δ'
(q4, b) ∪ δ'(q6, b) ∪ δ'(q7, b)
∪ δ'(q8, b)]
= ε-closure [φ ∪ φ ∪ φ ∪ q5 ∪ φ
∪ φ ∪ q9]
= ε-closure [q5, q9]
= [q1, q2, q4, q5, q6, q7, q9]

MOVE (D, a) = ε-closure (δ'(q1, a)
∪ δ'(q2, a) ∪ δ'(q4, a) ∪ δ'
(q5, a) ∪ δ'(q6, a) ∪ δ'
(q7, a) ∪ δ'(q9, a)
= ε-closure (φ ∪ q3 ∪ φ ∪ φ
∪ q8 ∪ φ)
= ε-closure (q3, q8)
= [q1, q2, q3, q5, q6, q7,
q8]

$\delta'(q_1, b)$
$, b) \cup \delta'$
$\cup \delta'(q_7, b)$
$\delta, b)]$
$\cup q_5 \cup \phi$
$\phi \cup q_9]$

$, q_7, q_9]$

$\delta'(q_1, a)$
$a) \cup \delta'$
$a) \cup \delta'$
$[q_9, a)$
$\cup \phi \cup \phi$

)

$, q_7,$
$, 8]$

$MOVE(D, b) = \varepsilon - closure(\delta'(q_1, b)$
$\delta'(q_2, b) \cup \delta'(q_4, b) \cup \delta'$
$(q_5, b) \cup \delta'(q_6, b) \cup$
$\delta'(q_7, b)$

$= \varepsilon - closure(\phi \cup \phi \cup q_5 \cup$
$\cup \phi \cup q_9)$

$= \varepsilon - closure(q_5, q_9)$

$= [q_5, q_6, q_7, q_1, q_2, q_4, q$

$= [q_1, q_2, q_4, q_5, q_6, q_7,$
$q_9]$

The required DFA is

$$D = \{Q, \Sigma, \delta, q_0, F\}$$

$$Q = \{A, B, C, D\}$$

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

$$q_0 = \{A\}$$

$$F_D = \{D\}$$

& δ given by

$$\delta(A, a) = B$$

$$\delta(A, b) = C$$

$$\delta(B, a) = B$$

$$\delta(B, b) = D$$

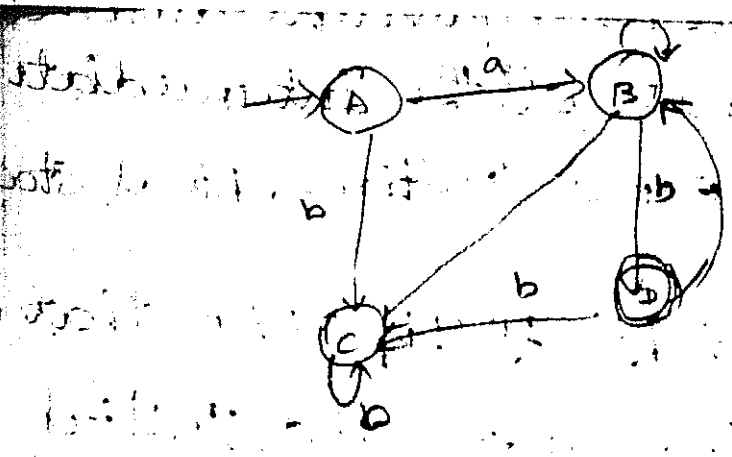
$$\delta(C, a) = B$$

$$\delta(C, b) = C$$

$$\delta(D, a) = B$$

$$\delta(D, b) = C$$

		a	b
→	A	B	C
	B	B	D
	C	B	C
*	D	B	C



conversion of Finite Automata to Regular expression

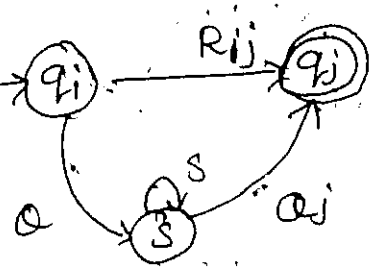
→ State - Elimination Technique

→ Arden's Theorem

→ Regular expression equation Method.

State- Elimination Technique :

The state 's' can be elimina From the Finite automata q_i & q_j



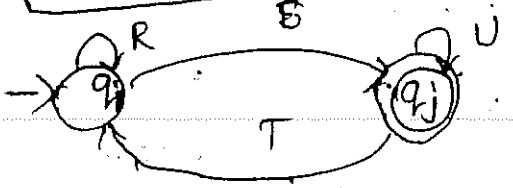
using the Formula. $R_{ij} + Q_i S^* Q_j$

Step 1: For each accepting State apply the Reduants process

& eliminate all the Intermediate changes expect Initial & final state.

Step 2 :- If $q_i \neq q_j$ ~~$q_i = q_j$~~, $q_i \neq 0$ that is accepting state and Initial state are disting then its called Two State automaton.

$$RE = (R + SU^* T)^* SU^*$$



Step 3 :- $q_i = q_j$

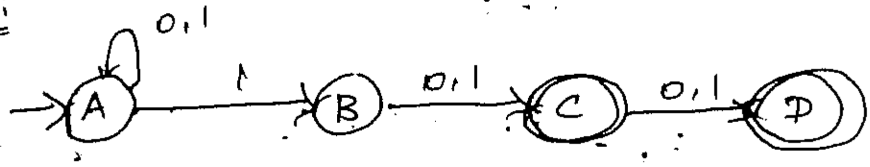
$$RE = (R + SU^* T)^*$$

Step 4 :- They required Regular expression is the Union of all the Regular expression for each accepting state.

tomata
diate
l state
that is
tial
s called
u
n of all
For

1) Convert the following FA to RE using State elimination Technique.

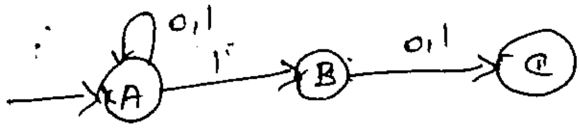
Sol:



Step 1: Initial State = A

Final State = C, D

Step 2: Eliminate State 'B'

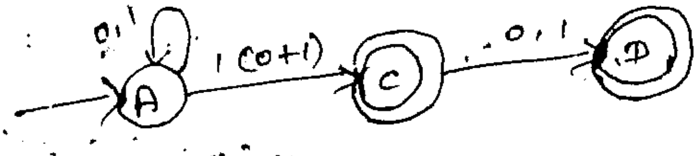


[R_{ij} = A, C)

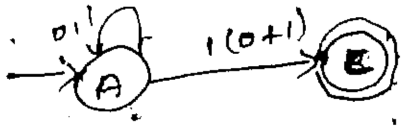
$$RE = R_{ij} + Q_i S^* Q_j$$

$$= \phi + (1)(0+1)$$

$$= 1(0+1)$$



Step 3: Since they are 2 Final State First consider 'C' as the Final State



Since its a 2 State automaton they required RE

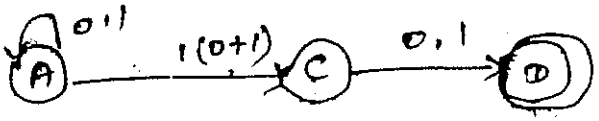
will be

$RE = (R + SU^*T)^*SU^*$

$= ((0+1) + 1(0+1)\phi^*\phi)^* 1(0+1)\phi^*$

$RE_1 = (0+1)^* 1 (0+1)$ $\quad \phi^* = 1$

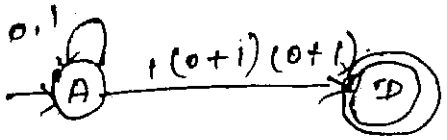
Step4: consider 'D' as the final state



$RE = R_{ij} + Q_1 S^* Q_j$

$= \phi + 1(0+1)(0+1)$

$= 1(0+1)(0+1)$



$RE_2 = (R + SU^*T)^* SU^*$

$= ((0+1) + 1(0+1)(0+1)\phi^*\phi)^*$

$1(0+1)(0+1)\phi^*$

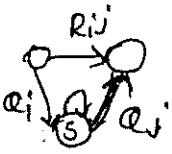
$RE_2 = (0+1)^* 1 (0+1)(0+1)$

Step5: $\therefore$ RE is $RE_1 + RE_2$

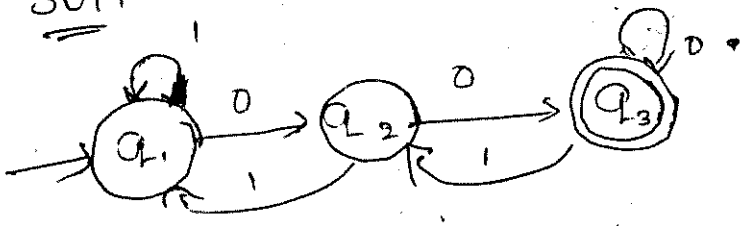
$RE = ((0+1)^* 1 (0+1)) + ((0+1)^* 1 (0+1)(0+1))$

4.10

		0	1
X	$\to q_1$	q_2	q_1
	$*q_2$	q_3	q_1
	$*q_3$	q_3	q_2



Sol:

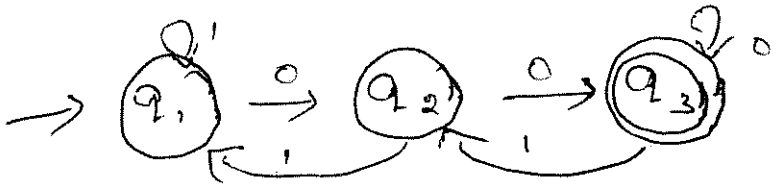


STEP1 :

Initial state = q_1

Final state = q_3

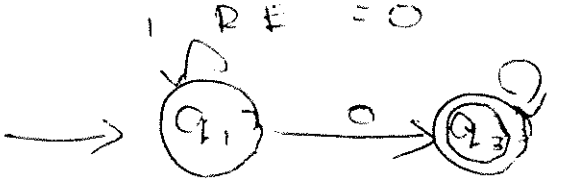
STEP2 : eliminate STEP q_2



$RE = R_{ij} + Q_{is}^{*} Q_j$

$= \phi + 0\,0$

$RE = 0$



$RE = (0 + 1)^{*} 1 0^{*}$

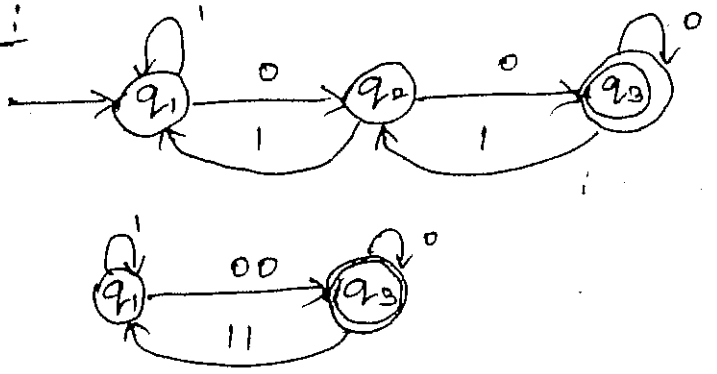
$= (1 + 0 + 0)^{*} 00^{*}$

$RE = (1+0)^{*} 00^{*}$

1)

	0	1
$\rightarrow q_1$	q_2	q_1
q_2	q_3	q_1
$*q_3$	q_3	q_2

Sol:



$$RE = (R + SU^*T)^* SU^*$$

$$= (1 + (00)(0)^* 11)(00)(0)^*$$

Arden's Theorem:

Let 'p' and 'q' be two regular expressions over Σ. If 'P' is not Σ then the regular expression $R = q + RP$ has the solution of $R = qp^*$ where 'R' is the state of automaton.

The arden's Theorum follows Repeated Substitution Methods.

They are 4 rules, To construct the equations.

i) Finite Automata should not h

ε

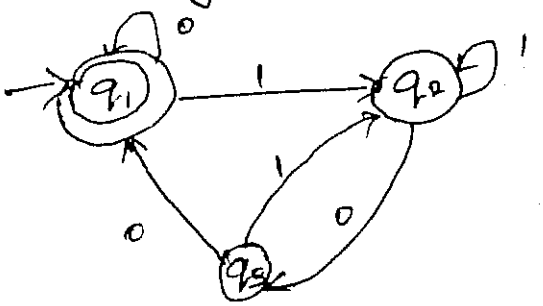
ii) Let the states are $q_1, q_2 \ldots$ c with q_1 as the Initial state.

iii) α_{ij} denoteds the Transition from q_i to q_j.

iv) If they is no Transitions th $\alpha_{ij} = \phi$.

problems:

1) Construct the regular expressi using arden's Theorem for the following automaton.



$q_1 = q_1 0 + q_2 \phi + q_3 0 + \varepsilon$

$q_2 = q_1 1 + q_2 1 + q_3 1$

$q_3 = q_1 \phi + q_2 0 + q_3 \phi$

$q_1 = q_1 0 + q_3 0 + \varepsilon \longrightarrow ①$

$q_2 = q_1 1 + q_2 1 + q_3 1 \longrightarrow ②$

$q_3 = q_2 0 \longrightarrow ③$

Sub ③ in ②

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_2 0 1$

$q_2 = q_1 \cdot 1 + q_2 (1 + 01)$

$q_2 = (q_1 \cdot 1)\ (1+01)^* \longrightarrow ④$ [$\because R = QP^*$]

Sub ⑤ in ①

$q_1 = q_1 0 + q_2 00 + \varepsilon$

$= q_1 0 + q_1 1\ (1+01)^* 00 + \varepsilon$

$= q_1 (0 + 1\ (1+01)^* 00) + \varepsilon$

$q_1 = \varepsilon + q_1\ (0 + 1\ (1+01)^* 00)$

$\therefore R = Q + RP$

$R = QP^*$

$q_1 = \varepsilon \cdot (0 + 1\ (1+01)^* 00)^*$

$q_1 = (0 + 1\ (1+01)^* 00)^*$

$\therefore$ Since q_1 is the final accepting State the required RE for the given Finite automata is

$q_1 = (0 + 1(1+01)^* 00)^*$

Regular Expression Equation Method

$(R_{ij}^{(k)})$

i - Initial state

j - Final state

k - Total no. of states.

Theorem:

For every DFA $A = (Q, \Sigma, \delta, S$ they is Regular expression are such that $L(R) = L(A)$

Proof:

Let 'L' be the set of Langua accepted by DFA $A = (\{q_1, q_2 \ldots q_n\}, \Sigma, \delta, q, F)$ with q_1 as the Initial state.

Let $R_{ij}^{(k)}$ be the Regular expression describing they set of all string 'x' such that $\delta(q_{ij}, x) = q_j$ going theow 'k' no. of Intermediate state.

Basis:

Let $k=0$, $R_{ij}^{(0)}$ denotes the set of strings which are either ε are single ~~Strings~~ symbol.

case (i) :- $i \neq j$

$$\boxed{R_{ij}^{(0)} = \{a \mid \delta(q_i, a) = q_j\}}$$

denotes the set of symbols 'a' such that $\delta(q_i, a) = q_j$

$$R_{ij}^{(0)} = R_{ii}^{(0)} = \{\varepsilon\} \cup \{a \mid \delta(q_i, a) = q_i\}$$

$$R_{ii}^{(0)} = \varepsilon + a \text{ where } \delta(q_i, a) = q_i$$

Induction:

Let $k \geq 1$:

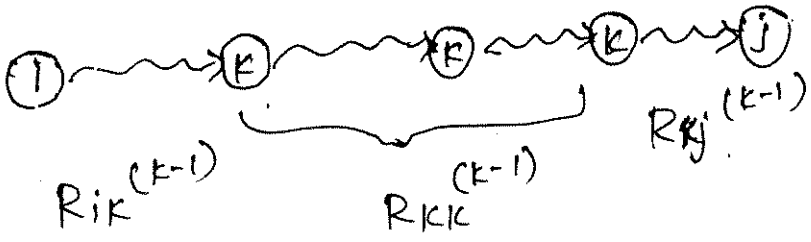
Let there is ~~a~~ path from state 'i' to state 'j' goes to 'k' no. of Intermediate States.

Assume that we have reach $R_{ij}^{(k-1)}$

i) Suppose if this path does not go through 'k' then

$$R_{ij}^{(k)} = R_{ij}^{(k-1)}$$

ii) If the path goes through the State 'k' atleast once then



by the formula.

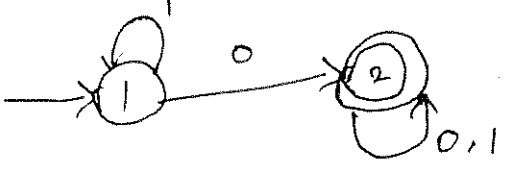
$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^{*} R_{kj}^{(k-1)}$$

Therfore by Induction hypothe we can calculated the required RE using this equation. Therfo For every DFA their is an Regular expression such tha $L(R) = L(A)$ is proved.

hence the proved.

problem:

1) convert the following DFA to Regular expression.



Sol:

Step 1:

$i = 1$

$j = 2$

$k = 2$

We have to find $R_{12}^{(2)}$

Step 2: Let $k = 0$

Find $R_{ij}^{(0)}$ for all the values of i, j

$$R_{ij}^{(0)} = \begin{cases} \varepsilon + a & \text{if } i = j \\ a & \text{if } i \neq j \end{cases}$$

$R_{11}^{(0)} = \varepsilon + 1$

$R_{12}^{(0)} = 0$

$R_{21}^{(0)} = \emptyset$

$R_{22}^{(0)} = \varepsilon + 0 + 1$

Step 3: Let $k = 1$

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$\varepsilon + 1 = 1$

$= (\varepsilon + 1) + (\varepsilon + 1)(\varepsilon + 1)^* (\varepsilon + 1)$

$= 1 + 1 (1)^* (1)$

$= 1 + 1^*$

$= 1^*$

$\left[\because 1 \cdot 1^* = 1^* \right.$
$\left. 1 + 1^* = 1^* \right]$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)}$

$= 0 + (\varepsilon + 1)(\varepsilon + 1)^* 0$

$= 0 + 1 \cdot 1^* \cdot 0$

$= 0 + 1^* \cdot 0$

$= (\varepsilon + 1^*) 0$

$= 1^* 0$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} (R_{21}^{(0)})^* R_{21}^{(0)}$

$= \phi + \varepsilon + 1 \ (\varepsilon + 1)^* \ \phi$

$= \phi$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{22}^{(0)}$

$= \varepsilon + 0 + 1 + (\varepsilon + 1)(\varepsilon + 1)^* \varepsilon + 0 + 1$

$= \varepsilon + 0 + 1 + (1)(1)^* \varepsilon + 0 + 1$

$= \varepsilon + 0 + 1 + (1^*)$

$= 0 + 1$

For $k = 2$:

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$

$= (1^* 0) + (1^* 0)(0 + 1)^* (0 + 1)$

$= (1^* 0) + (1^* 0)(0 + 1)^*$

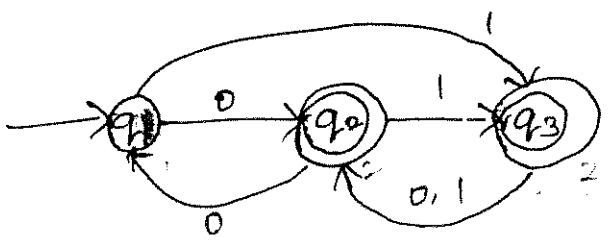
$= (1^* 0)(\varepsilon + (0 + 1)^*)$

$= (1^* 0)(0 + 1)^*$ //

∴ This is a regular expression

For required DFA,

2)



$\varepsilon + 0 = 0$

$\varepsilon \cdot 0 = 0$

$\varepsilon^* = \text{Omit}$

$0 + 0 = 0$

$0 \cdot 0 = 00$

$0 \cdot 0^* = 0^*$

$0 + 0^* = 0^*$

$q_1 = 1$
$q_2 = 2$
$q_3 = 3$

Step 1: $i = 1$

$j = 2, 3$

$K = 3$

$$RE = R_{12}^{(3)} + R_{13}^{(3)}$$

step 2: Let $K = 0$

Find $R_{ij}^{(0)}$ for all the values of i, j

$$R_{ij}^{(0)} = \begin{cases} \varepsilon + a & \text{if } i = j \\ a & \text{if } i \neq j \end{cases}$$

$R_{11}^{(0)} = \varepsilon$ $\quad$ $R_{22}^{(0)} = \varepsilon$

$R_{12}^{(0)} = 0$ $\quad$ $R_{23}^{(0)} = 1$

$R_{13}^{(0)} = 1$ $\quad$ $R_{31}^{(0)} = \phi$

$R_{21}^{(0)} = 0$ $\quad$ $R_{32}^{(0)} = 0 + 1$

$R_{33}^{(0)} = \varepsilon$

ression

+0 = 0
·0 = 0
* = omit
-0 = 0
0 = 00
0* = 0*
·0* = 0*

values

Step 3: Let K = 1

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \varepsilon + \varepsilon (\varepsilon)^* \varepsilon$$

$$= \varepsilon + \varepsilon^*$$

$$= \varepsilon^*$$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= 0 + \varepsilon (\varepsilon)^* 0$$

$$= 0$$

$$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$

$$= 1 + \varepsilon (\varepsilon)^* \cdot 1$$

$$= [illegible]$$

$$= 1$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{21}^{(0)}$$

$$= 0 + 0 (0)^* 0$$

$$= 0$$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= \varepsilon + 0 (\varepsilon)^* 0$$

$$= \varepsilon + 0 \cdot 0 = 0 \cdot 0$$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* (R_{13}^{(0)})^*$

$= 1 + 0 (\varepsilon)^* . 1$

$= 1 + 0 \cdot 1$

$= 1 (\varepsilon + 0)$

$= 1 \cdot 0$

$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{11}^{(0)}$

$= \phi + \phi (\varepsilon)^* \varepsilon$

$= \phi$

$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{21}^{(0)}$

$= 0+1 + \phi (\varepsilon)^* . 0$

$= 0 + 1$

$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)} (R_{11}^{(0)})^* R_{13}^{(0)}$

$= \varepsilon + \phi (\varepsilon)^* \cdot 1$

$= \varepsilon$

For $k = 2$;

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$

$= \varepsilon^* + 0 (00)^* \cdot 0$

$= 0 (0 \cdot 0)^* 0$

$R_{12}^{(0)}$ *

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$$

$$= 0 + 0(00)^* 00$$

$$= 0 + 0(00)^*$$

$$= 0 + (\varepsilon + (00)^*)$$

$$= 0(00)^*$$

$R_{11}^{(0)}$

$$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$$

$$= 1 + 0 \cdot (00)^* 01$$

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} (R_{22})^* R_{21}^{(1)}$$

$R_{21}^{(0)}$

$$= 0 + 00 (00)^* 0$$

$$= 0 + (00)^* 0$$

$$= (\varepsilon + (00)^*) 0$$

$$= (00)^* 0$$

$R_{13}^{(0)}$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$$

$$= 00 + 00(00)^* 00$$

$$= 00 + (00)^*$$

$$= (00)^*$$

$$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$$

$R_{21}^{(1)}$

$$= 01 + 00(00)^* 01$$

$$= 01 + (00)^* 01$$

$$= (\varepsilon + (00)^*) 01$$

$$= (00)^* 1$$

$R_{31}(2) = R_{31}(1) + R_{32}(1) R_{22}(1))^* R_{21}(1)$

$= \phi + (0+1)(00)^* 0$

$= (0+1)(00)^* 0$

$R_{32}(2) = R_{32}(1) + R_{32}(1) (R_{22}(1))^* R_{22}(1)$

$= (0+1) + (0+1)(00)^* (00)$

$= (0+1) + (0+1)(00)^*$

$= (0+1)(\varepsilon + (00)^*)$

$= (0+1)(00)^*$

$R_{33}(2)$

$= R_{33}(1) + R_{32}(1) (R_{32}(1))^* R_{23}(1)$

$= \Sigma + (0+1)(00)^* 01$

$= (0+1)(00)^* 01$

Step final $R_{12}(3) + R_{13}(3)$

$R_{12}(3) = R_{12}(2) + R_{13}(2) (R_{33}(2))$

$= 0(00)^* (1 + 0(00)^* 01$

$((0+1)(00)^* 01)^* ((0+1)(00)^*)$

$R_{13}(3) = R_{13}(2) + R_{13}(2) (R_{33}(2))^* R_{33}(2)$

$= (1+0(00)^* 01) + (1+0(00)^* 01)$

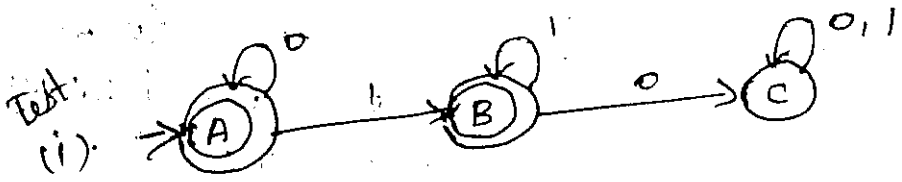
$(0+1)(00)^* 01)^* ((0+1)(00)^* 01)$

$= (1+0(00)^* 01)\cdot(\varepsilon + ((0+1)(00)^* 01)^*)$

$= (1+0(00)^* 01)((0+1)(00)^* 01)^*$

$R_{12}(3) + R_{13}(3) = (0(00)^* + (1+0(00)^*((0+1)(00)^* 01)^*(1\cdot 0+1)\cdot(00)^*) +$

$((1+0(00)^* 01)((0+1)(00)^* 01)^*$



(ii) DFA → RE Theorem

Proving Languages Not to be Regular
↳ pumping Lemma

The regular Languages need Finite amount of memory where as non Regular Languages need Infinite amount of memory which is theoretically not possible.

Therefore the most powerful Technique To prove as certain languages to be not regular is called pumping Lemma.

Pumping Lemma:

Let 'L' be the regular Language then there exist an Integer $p \geq 1$ such That every string 'w' in L of Length atleast p can written as $w = xyz$ (That is w can be divided into three substrings) satisfying the following conditions.

i) $|y| \geq 1$ ~~Length~~ Loop y to be pumped must be of Length atleast one.

ii) $|xy| \leq p$ Loop must occur within First p characters

iii) For all $i \geq 0$ $xy^iz \in L$

problem;

1) Prove that $L = \{0^i 1^i / i \geq 1\}$ is not Regular.

Step 1: Let 'L' is regular because by the method of contridiction we have to assume the Inverse of given statement to be True.

Step 2: When the language is Regular there exists an Integer $p \geq 1$

Step 3: Since the language is Regular we can use pumping Lemma Method

By the pumping Lemma, $w = xyz$ satisfying the following condition

(i) $|y| \geq 1$

(ii) $|xy| \leq p$

(iii) For all $i \geq 0$ $xy^iz \in L$

For $i = 1$,

$w = 01$

$\underset{x}{0}\underset{y}{0}\underset{z}{11}$

For $i = 2$

$w = \underset{x}{0}\underset{y}{0}\underset{z}{1}\underset{z}{1}$

Let $w = xy^iz \in L$

case i) $y = 0^k$

case ii) $y = 1^l$

case iii) $y = 0^k 1^l$

Let $i = 2$, $\Rightarrow xy^2z$

case (i) $= 0001 \notin L$

case (ii) $= 0111 \notin L$

case (iii) $= 001011 \notin L$

∴ As the structure of the string doesnot match the given language then its not accepted by the language is as the assumption of the given language doesnot accept by finite automata the given language is not Regular.

hence the proved. //

2) Show that $L = \{ww^R : w \in (a,b)^*\}$ is not regular.

sol:

Step 1; Let L is Regular because by the method of contadiction we have to assume the Inverse of the given Statement to be True.

Step 2: When the language is Regular then there exists an Integer $p \geq 1$

Step 3; Since the language is Regular we can use pumping lemma method by the following condition

i) $|y| \geq 1$

ii) $|xy| \leq p$

iii) for all $i \geq 0$ $xy^i z \in L$

For $i = 1$

$w = 0^1 1$

For $i = 2$

$w = 00\,1\,1$

$x\ y\ z$

i) $y = 0^k$

ii) $y = 1^l$

iii) $y = 0^k 1^l$

Let $i = 2 = xy^2z$

case i) $= aaa1 \notin L$

case ii) $= abbb \notin L$

case iii) $= aababbb \notin L$

$w = ab$

$w = abba$ $|y| \geq 1 \Rightarrow$ tru

iii) $xy^iz \in L$ for all $i \geq 0$

Let $w = xy^iz \in L$

$y = a^k$, $y = b^l$

$y = a^k b^l$

for $i = 2$

$w = xy^2z$

$x = a; z$

case i) $w = ac$

case ii) $w = a$

as the structure of the string does not match the given languag then its not accepted by the languag

The assumption of the given language to be not accepted by th Finite automata the given langua is not regular.

CLOSURE PROPERTIES OF REGULAR EXPRESSION:

→ The Union of 2 Regular languages is Regular.

→ The Intersection of 2 Regular languages is Regular.

→ The complement of a regular languages is Regular.

→ The difference of 2 Regular languages is Regular.

→ The Reversal of a regular languages is Regular.

→ The closure of a Regular languages is Regular.

→ The concatenation of Regular languages is Regular.

→ homomorphism of a regular languages is Regular.

→ The Inverse homomorphism of a Regular language is regular.

Union of 2 Regular languages is Regular:

⇒ Closure under Union:-

Let L & M be the regular languages over the alphabet Σ then L ∪ M is regular

$L = L(R)$ } regular expression
$M = L(S)$ }

$L \cup M = L(R+S)$

ii) Closure Under complement :-

Step 1 : convert the regular expression to ε-NFA using Thomson construction method.

Step 2 : convert the ε-NFA into DFA by using Subset ~~construction~~ construction Method.

Step 3 : Complement the accepting States of DFA

Step 4 : convert the complement DFA to RE using State elimination Technique.

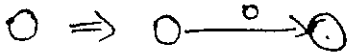
Step 5 : The resultant regular expression is the complement of the given RE.

Problem:

1) Find the complement of $(0+1)^* 01$

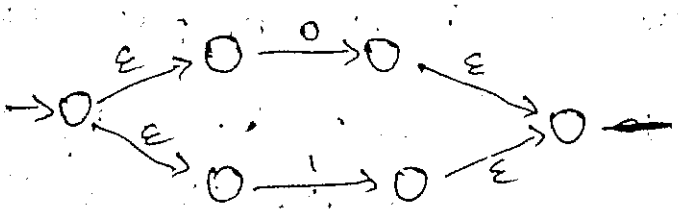
step 1 : convert the Regular expression

to ε- NFA Using thomson construct Methods.

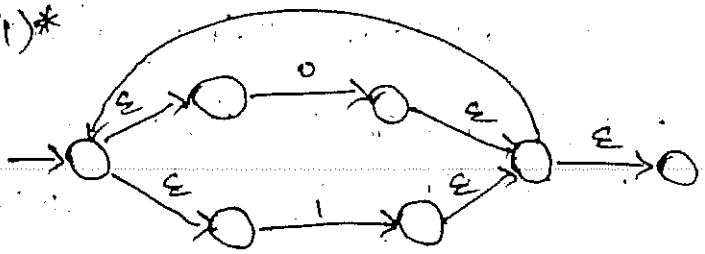
0 ⇒ 

1 ⇒ 

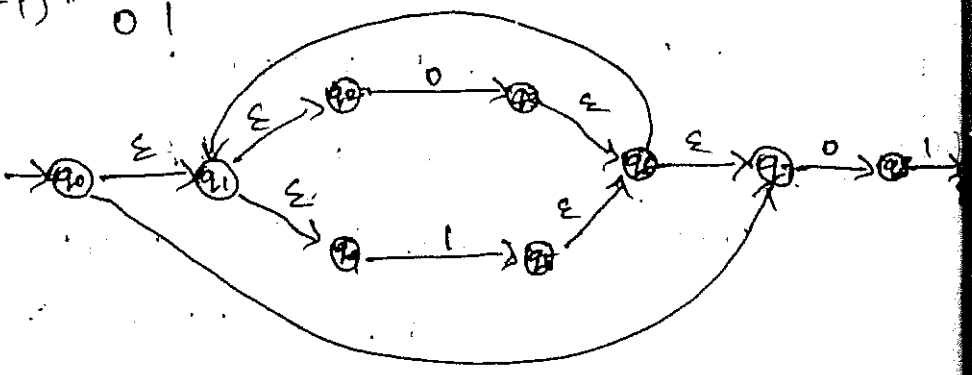
(0+1)



(0+1)*



(0+1)* 0 1



Step 2: Convert the ε - NFA into DFA

Initial state of DFA = ε-closure (Initial state (ε-NFA))

= ε - closure (q_0)

= [q_0, q_1, q_2, q_4, q_7]

↳ A

onstruc

∴ Initial state of DFA = A

MOVE (A, 0) = ε - closure (δ'(q0, 0) ∪
δ'(q1, 0) ∪ δ'(q2, 0) ∪ δ'
(q4, 0) ∪ δ'(q7, 0)

= ε - closure (φ ∪ φ ∪ q3 ∪ φ ∪ q8)

= ε - closure (q3, q8)

= [q3, q6, q7, q1, q2, q4, q8

= [q1, q2, q3, q4, q6, q7, q8] → B.

MOVE (A, 1) = ε - closure (δ'(q0, 1) ∪
δ'(q1, 1) ∪ δ'(q2, 1) ∪ δ'(q4, 1)
∪ δ'(q7, 1))

= ε - closure (φ ∪ φ ∪ φ ∪ q5 ∪ φ

= ε - closure (q5)

= [q5, q6, q7, q1, q2, q4]

= [q1, q2, q4, q5, q6, q7] → C

MOVE (B, 0) = ε - closure (δ'(q0, 0) ∪ δ'
(q1, 0) ∪ δ'(q2, 0) ∪ δ'
(q4, 0) ∪ δ'(q7, 0))

= ε - closure (q3, q8)

= [q3, q6, q7, q1, q2, q4, q8]

= [q1, q2, q3, q4, q6, q7, q8]

MOVE $(B, 1) = \varepsilon$ - closure $(\delta'(q_1, 0) \cup$
$\delta'(q_2, 0) \cup \delta'(q_3, 0) \cup \delta'$
$(q_4, 0) \cup \delta'(q_6, 0) \cup \delta'(q_7, 0) \cup \delta'$
$(q_8, 0))$

$= \varepsilon$ - closure $(\phi \cup \phi \cup \phi \cup q_5 \cup q_9$

$= \varepsilon$ - closure (q_5, q_9)

$= [q_5, q_6, q_7, q_1, q_2, q_4, q_9]$

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

MOVE $(C, 0) = \varepsilon$ - closure $(\delta'(q_0, 0) \cup$
$\delta'(q_1, 0) \cup \delta'(q_2, 0) \cup \delta'(q_4, 0) \cup$
$\delta'(q_7, 0))$

$= \varepsilon$ - closure $(\phi \cup \phi \cup q_3 \cup \phi \cup q_8]$

$= \varepsilon$ - closure (q_3, q_8)

$= [q_3, q_8, q_7, q_1, q_2, q_4, q_8]$

$= [q_1, q_2, q_3, q_4, q_6, q_7, q_8]$

MOVE $(C, 1) = \varepsilon$ - closure $(\delta'(q_1, 1) \cup \delta$
$(q_2, 1) \cup \delta'(q_3, 1) \cup \delta'(q_4, 1) \cup$
$\delta'(q_6, 1) \cup \delta'(q_7, 1) \cup \delta'(q_8, 1)$

$= \varepsilon$ - closure $(\phi \cup \phi \cup \phi \cup q_5 \cup$
$\phi \cup q_9]$

$= \varepsilon$ - closure (q_5, q_9)

$= [q_1, q_2, q_4, q_5, q_6, q_7, q_8]$

MOVE $(D, 0) = \varepsilon$ - closure $(\delta'(q_1, 0) \cup \delta'(q_2, 0) \cup \delta'(q_4, 0) \cup \delta'(q_5, 0) \cup \delta'(q_6, 0) \cup \delta'(q_7, 0) \cup \delta'(q_8, 0))$

$= \varepsilon$ - closure $(\phi \cup q_3 \cup \phi \cup \phi \cup q_8 \cup \phi)$

$= \varepsilon$ - closure (q_3, q_8)

$= [q_1, q_2, q_3, q_5, q_6, q_7, q_8]$

MOVE $(D, 1) = \varepsilon$ - closure $(\delta'(q_1, 1) \cup \delta'(q_2, 1) \cup \delta'(q_4, 1) \cup \delta'(q_5, 1) \cup \delta'(q_6, 1) \cup \delta'(q_7, 1))$

$= \varepsilon$ - closure $(\phi \cup \phi \cup q_5 \cup \phi \cup \phi \cup q_9]$

$= \varepsilon$ - closure (q_5, q_9)

$= [q_1, q_2, q_4, q_5, q_6, q_7, q_9]$

The required DFA is

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{A, B, C, D\}$

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

$q_0 = \{q_0\}$

$F = \{q_7\}$

and δ is given by

$\delta(A,0) = B$

$\delta(A,1) = C$

$\delta(B,0) = B$

$\delta(B,1) = D$

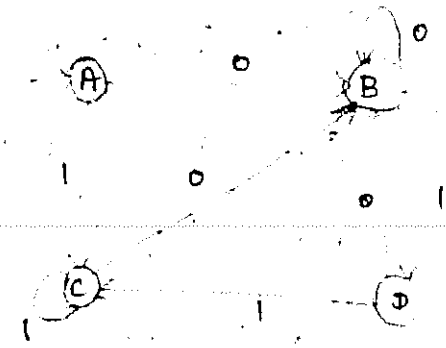
$\delta(C,0) = B$

$\delta(C,1) = C$

$\delta(D,0) = B$

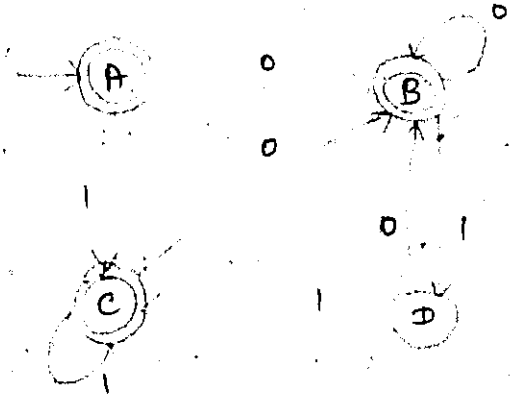
$\delta(D,1) = C$

	0	1
→A	B	C
B	B	D
C	B	C
*D	B	C



Step-3:

. Complement the accepting states of DFA.



Step 4: convert the complement DFA to RE using state elimination method

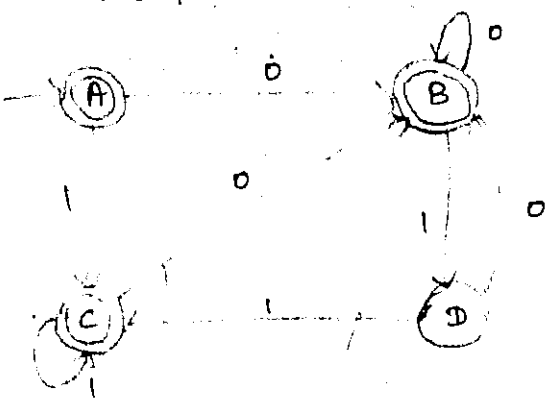
Initial State = A

Final state = B, C, D

eliminate state D — 3 ato b

$RE = R_{ij} + Q_i S^* Q_j$

$= 0 + 0$

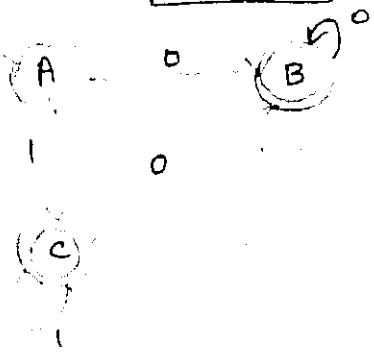


eliminate state D

$$RE = R_{ij} + Q_i S^* Q_j$$

$= 0 + \phi \phi^* 0$

$$RE = 0$$

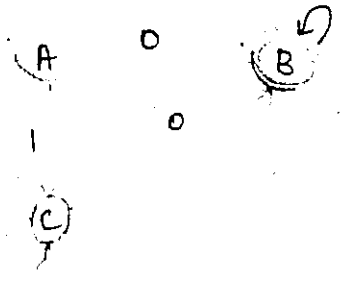


A as a final state

$RE_1 = (R + S0^* T)^*$

$RE_1 = \phi^*$

B as a final state



$RE = R_{ij} + Q_i S^* Q_j$

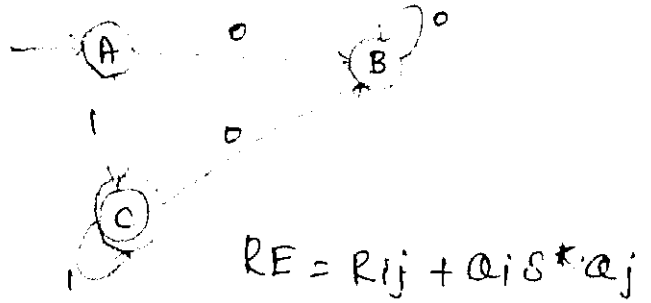
$= 0 + 11^* 0$

$RE_2 = (R + SU^* T)^* SU^*$

$= (\phi + 00^* \phi)^* 00^*$

$\boxed{RE_2 = 00^*}$

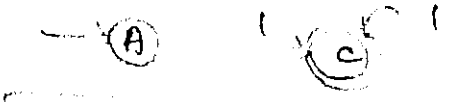
C as a final state



$RE = R_{ij} + Q_i S^* Q_j$

$= 1 + 00^* \phi$

$= 1$



$RE_3 = (R + SU^* T)^* SU^*$

$= (\phi + 11^* \phi)^* 11^*$

$= \phi^* 11^*$

$RE_3 = 11^*$

$RE = RE_1 + RE_2 + RE_3$

$= \phi^* + (0 + 11^*) + (11^*)$

$= (0 + 11^* 0) + (11^*)$

$= (0 + 1^* 0) + 1^*$

$= (\varepsilon + 1^*) 0) + 1^*$

$= 1^* 0 + 1^*$

$= 1^* (0 + \varepsilon)$

$\boxed{RE = 1^* 0}$

Closure Under Intersection:

Theorem:

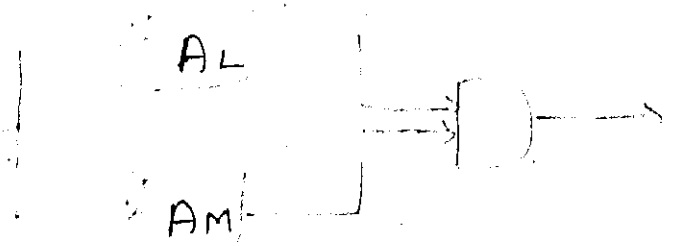
If L & M are Regular languag
then $L \cap M$ is also regular.

proof:

Let L & M be the languages
automata.

$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$

$A_M (Q_M, \Sigma, \delta_M, q_M, F_M)$



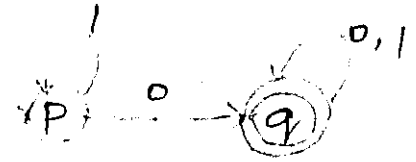
$A = A_L \cap A_M$

$A = (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)$

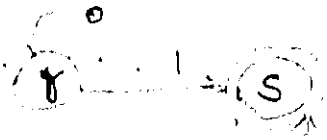
$\delta((P, q), a) = (\delta_L(P, a) \cup \delta_M(q, a))$

problem:

A ⇒



B ⇒



0, 1

Construct $A \cap B = \{Q_A \times Q_B, \Sigma, \delta, (q_A \times q_B), F_A \times F_B\}$

$Q_A \times Q_B = \{(P, r), (P, S), (q, r), (q, S)\}$

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

Initial state $= (P, r)$

Final state $= (q, S)$

and δ is Given by

$$\delta'((P, r), 0) = \delta_A'(P, 0) \cup \delta_B'(r, 0)$$
$$= (q \cup r)$$
$$= (q, r)$$

$$\delta((P, r), 1) = \delta_A'(P, 1) \cup \delta_B'(r, 1)$$
$$= (P, S)$$

$$\delta((P, S), 0) = \delta_A'(P, 0) \cup \delta_B'(S, 0)$$
$$= (q, S)$$

$$\delta((P, S), 1) = \delta_A'(P, 1) \cup \delta_B'(S, 1)$$
$$= (P, S)$$

$$\delta((q, r), 0) = \delta_A'(q, 0) \cup \delta_B'(r, 0)$$
$$= (q, r)$$

$$\delta((q, r), 1) = \delta_A'(q, 1) \cup \delta_B'(r, 1)$$
$$= (q, S)$$

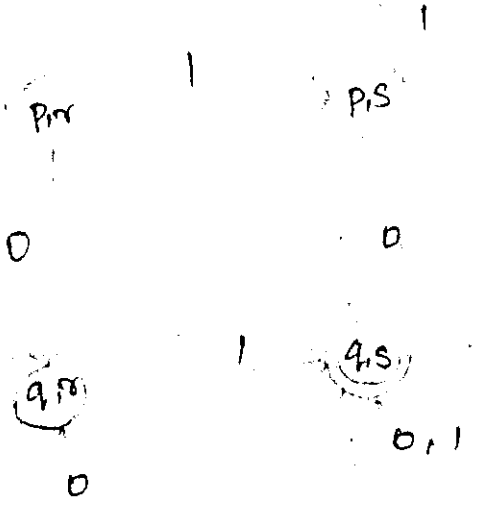
$\delta((q,s),0) = \delta_A'(q,0) \cup \delta_B'(s,0)$

$= (q,s)$

$\delta((q,s),1) = \delta_A'(q,1) \cup \delta_B'(s,1)$

$= (q,s)$

	0	1
→(p,r)	(q,r)	(p,s)
(p,s)	(q,s)	(p,s)
(q,r)	(q,r)	(q,s)
*(q,s)	(q,s)	(q,s)



closure and Their different:

Theorem:

If L & M are Regular languages then L − M is also regular.

$L - M = L \cap \bar{M}$

by the Theorem of complement if M is regular $\bar{M}$ is also Regular. Therefore $L \cap M$ can be written as $L \cap \bar{M}$ by the Theorem of Intersection $L \cap \bar{M}$ is also regular. $L - M$ is regular.

~~Hence the proof~~

Closure Under Reversal:

The reversal of a string 'w' denoted as w^R is the string written in backwords.

$w = 0010$

$w^R = 0100$

The Reversal of a language 'L' is the language consisting of the Reversal of all its strings. It is denoted by L^R.

$L = \{100, 110, 011\}$

$L^R = \{001, 011, 110\}$

Theorem :-

If L is the regular languag
the L^R is also regular.

proof :

Assume that L is defined b
a Regular expression 'E'

The proof is the structural
Induction of the Strings with size
of 'E'.

Basis :

If 'E' has only one Charact
it may be either Σ, ϕ, a.

E^R of the string is the Sam
of E. Therefore E^R is also regular
when E is Regular.

Induction :

They are 3 cases defining on
the form of E.

case i) $E = E_1 + E_2$

then $E^R = E_1^R + E_2^R$

case ii): $E = E_1 \cdot E_2$

$$E^R = E_1^R \cdot E_2^R$$

case iii): $E = E_1^*$

$$E^R = (E_1^R)^*$$

Since in all the 3 cases we are able to constructed the Regular expression E^R then L^R is also regular.

∴ hence the proved.

Homomorphism:-

The homomorphism is a Function on strings of the language that substitute every symbol of the string with some other symbols.

$w = 0011$

$h(0) = a$, $h(1) = b$

$h(w) = aabb$

Theorem:

If 'L' is the regular language over Σ and h is its homomorphism then $h(L)$ is also regular.

Proof:

Let $L = L(E)$ for some regular Expression E. If E is the regular expression with symbols from Σ and let $h(E)$ be the expression obtained by replacing each symbo a in E by $h(a)$.

Basis:

case i): $E = \varepsilon$ (or) ϕ

then $h(E) = E$

then $\boxed{L(h(E)) = h(L(E))}$

case ii): $E = a$

$L(E) = \{a\}$

$h(L(E)) = \{h(a)\}$

$L(h(E)) = \{h(a)\}$

$\therefore \boxed{h(L(E)) = L(h(E))}$

Induction: $|E| > 1$

Case (1) $E = F + G_1$

$h(E) = h(F + G_1)$

$= h(F) + h(G_1)$

$L(h(E)) = L(h(F)) + L(h(G))$

$= L(h(F) \cup h(G_1)) \longrightarrow (1$

$E = F + G$

$L(E) = L(F) \cup L(G)$

$h(L(E)) = h(L(F) \cup L(G))$

$= h(L(F)) \cup h(L(G))$

$= L(h(F)) \cup L(h(G))$

$= L(h(F) \cup h(G)) \longrightarrow (2)$

$(1) = (2)$

$\therefore L(h(E)) = h(L(E))$

case(ii): $E = F \cdot G$

$h(E) = h(F \cdot G)$

$= h(F) \cdot h(G)$

$L(h(E)) = L(h(F) \cdot h(G))$

$= L(h(F) \cap h(G)) \longrightarrow (1)$

R.H.S By the definition of Intersection

$E = F \cdot G$

$L(E) = L(F) \cap L(G)$

$h(L(E)) = h(L(F) \cap L(G))$

$= h(L(F)) \cap h(L(G))$

$= L(h(F)) \cap L(h(G))$

$= L(h(F) \cap h(G)) \longrightarrow (2)$

$(1) = (2)$

$\therefore L(h(E)) = h(L(E))$

case(iii): $E = F^*$

$h(E) = h(F^*)$

$L(h(E)) = L(h(F)^*) \longrightarrow (1)$

$L(E) = L(F^*)$

$h(L(E)) = h(L(F^*))$

$= L(h(F^*)) \longrightarrow (2)$

$(1) = (2)$

$\therefore L(h(E)) = h(L(E))$

Inverse homomorphism:

If $h(L)$ is Regular then it
Inverse $h(L)^{-1}$ that is equivalen
to L is also regular.

$L = 00$

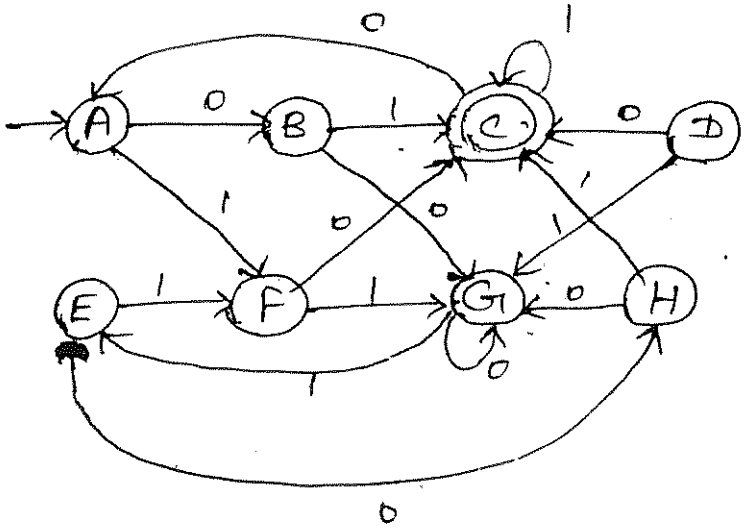
$h(0) = xy$

$h(L) = xy\,xy$

$h^{-1}(xy) = 0$

$h^{-1}(h(L)) = 00$

Minimized DFA:



step 1:

	0	1	
→ A	B	F	(A, E) (A, G)
B	G	C	(B, H)
* C	A	C	(D, A)
D	C	G	(E, G)
E	H	F	
F	C	G	
G	G	E	
H	G	C	

Step 2:

Take (A, E)

(i) Check for acceptance

Here A & E are both non-acceptance states

(ii) Check for input behaviour

For input 0:

$\delta(A, 0) = B$ (non-accepting)

$\delta(E, 0) = H$ (non-accepting)

For input 1:

$\delta(A, 1) = F$ (non-accepting)

$\delta(E, 1) = F$ (non-accepting)

Therefore they are A, E equivalent states.

Take (A, G)

i) check for acceptance

Here A & G are both non-acceptance states

ii) check for i/p behaviour

For input 0:

$\delta(A, 0) = B$ (non-accepting)

$\delta(G, 0) = G$ (non-accepting)

For input 1:

$\delta(A, 1) = F$ (non-accepting)

$\delta(G, 1) = E$ (non-accepting)

Since the behaviour of A & G is are different resultant states they are non equivalent.

Take (B, H)

i) check for acaptance

Here B & H are non-accepting states.

ii) check for i/p behaviour

For input 0:

$\delta(B, 0) = G$ (non-accepting)

$\delta(H, 0) = G$ (non-accepting)

For input 1:

$\delta(B, 1) = C$ (non-accepting)

$\delta(H, 1) = C$ (non-accepting)

They are B & H are equivalent State

Take (D, F)

i) check for acceptance

Here D & F are non-accepting states.

ii) Check for i/p behaviour.

For input 0:

$\delta(D, 0) = C$ (non-accepting)

$\delta(F, 0) = C$ (non-accepting)

For input 1:

$\delta(D, 1) = G$ (non-accepting)

$\delta(F, 1) = G$ (non-accepting)

They are D & F are equivalent states.

Take (E, G)

i) check for acceptance

Here E & G are non-accepting states.

ii) Check for i/p behaviour

For input 0:

$\delta(E, 0) = H$ (non-accepting)

$\delta(G, 0) = G$ (non-accepting)

For input 1:

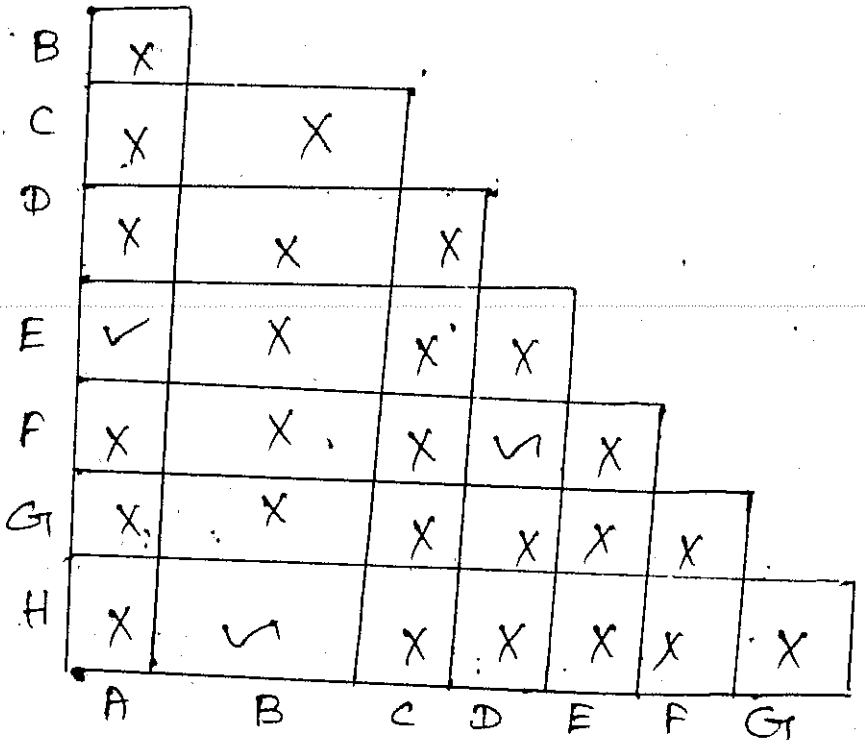
$\delta(E, 1) = F$ (non-accepting)

$\delta(G, 1) = E$ (non-accepting)

Since the behaviour of E & G is are different resultant states They are non equivalent.

Step 3:

Table For State inequalence



Step 4:

To Find minimized DFA

$D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{(A, E), (B, H), (D, F), (C), (G)\}$

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

Initial state = {A, E}

Final State = {C}

and δ is given by,

$\delta((A,E),0) = \{B,H\}$

$\delta((A,E),1) = \{D,F\}$

$\delta((B,H),0) = (G)$

$\delta((B,H),1) = (C)$

$\delta((D,F),0) = (C)$

$\delta((D,F),1) = (G)$

$\delta((C,0) = (A,E)$

$\delta(C,1) = (E)$

$\delta(G,0) = (G)$

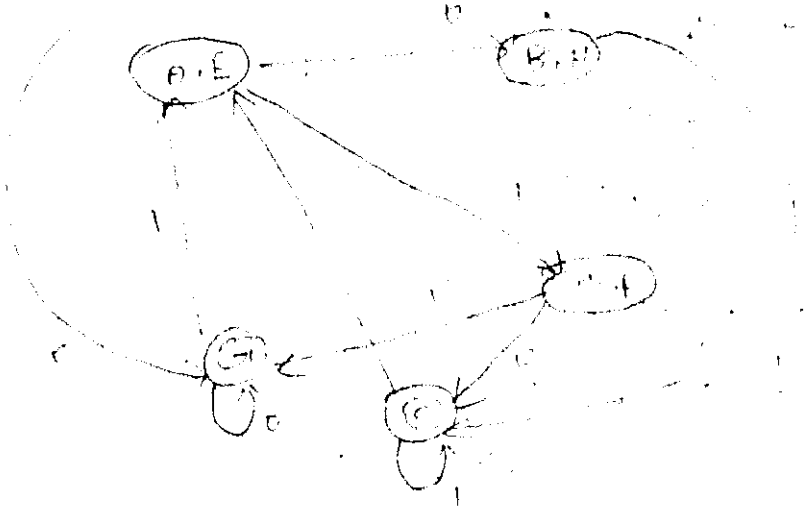
$\delta(G,1) = (A,E)$

Step 5:

Transition Table for the minimized DFA.

	0	1
(A, E)	(B, H)	(D, F)
(B, H)	(G)	(C)
(D, F)	(C)	(G)
(C)	(A, E)	(C)
(G)	(G)	(A, E)

Step 6:



2) Construct minimised DFA

		0	1
→	A	B	A
	B	A	C
	C	D	B
*	D	D	A
	E	D	F
	F	G	E
	G	F	G
	H	G	D

(A, B)
(A, F)
(A, G)
(B, F)
(B, G)
(C, E)
(F, G)

Sol:

Step 1: Take (A, B)

i) Check for acceptance

Here A & B are both non acceptance states.

ii) Check For i/p behaviour

For i/p 0:

$\delta(A, 0) = B$ (non accepting)

$\delta(B, 0) = A$ (non accepting)

For i/p 1:

$\delta(A, 1) = A$ (non-accepting)

$\delta(B, 1) = C$ (non-accepting

∴ They are (A, E) equivalent since the behaviour of A & B are different resultant states that are not equivalent

Take (A, F):

i) check For acceptance

Here A & F are both non acceptance

ii) check For i/p behaviour

For i/p 0:

$\delta(A, 0) = B$ (non accepting)

$\delta(F, 0) = G$ (non accepting

$\delta(A,1) = A$ (non-accepting)

$\delta(F,1) = B$ (non-accepting)

Since they are A & F not equiv

Take (A, G1)

i) Check for acceptance

Here A & G are accept

State

ii) Check For i/p behaviour

For i/p 0:

$\delta(A,0) = B$ (non-accepting)

$\delta(G,0) = F$ (non-accepting)

For i/p 1:

$\delta(A,1) = A$ (non-accepting)

$\delta(G1,1) = G1$ (non-accepting)

Since the behaviour of A & G1 are different resultant States They are non equivalent.

∴ Thiy A & G1 are equivalence State

Take (B, F)

i) Check for acceptance

Here B & F are both non-acceptance states.

ii) Check for i/p behaviour

For i/p 0:

$\delta(B, 0) = A$ (non-accepting)

$\delta(F, 0) = G$ (non-accepting)

For i/p 1:

$\delta(B, 1) = C$ (non-accepting)

$\delta(F, 1) = E$ (non-accepting)

Since the behaviour of B & F are different resultant states that are not equivalent.

Take (B, G)

i) Check for acceptance

Here B & G are both non-acceptance states.

ii) Check for i/p behaviour

For i/p 0:

$\delta(B, 0) = A$ (non-accepting)

$\delta(G, 0) = F$ (non-accepting)

For i/p 1:

$\delta(B, 1) = C$ (non-accepting)

$\delta(G, 1) = F$ (non-accepting)

since the behaviour B & G are different resultants states that are non equivalents.

Take (C, E)

i) check for acceptance states.

Here C & E are both non accaptance states.

ii) Check for i/p behaviour

For i/p 0:

$\delta(C, 0) = D$ (non-accepting)

$\delta(E, 0) = D$ (non-accepting)

For i/p 1:

$\delta(C, 1) = B$ (non-accepting)

$\delta(E, 1) = F$ (non-accepting)

They (C, E) are equivalent States.

Take (F, G)

i) check for acceptance

Here (F, G) are both non-accepting states.

ii) check for i/p behaviour

are
that are

ates.
acceptance

ng)
ng)

g)
ng)
valent

h non-

tion

For input 0:

$\delta(F, 0) = G$ (non-accepting)

$\delta(G, 0) = F$ (non-accepting)

For input 1:

$\delta(F, 1) = E$ (non-accepting)

$\delta(G, 1) = G$ (non-accepting)

∴ Since the behaviour of F & G is F & G are different resultant State they are not equivalent

Step 3:

Table for state inequalen

	A	B	C	D	E	F	G
B	X						
C	X	X					
D	X	X	X				
E	X	X	✓	X			
F	X	✓	X	X	X		
G	X	X	X	X	X	X	
H	X	X	X	X	X	X	X

Step 4: To Find minimized DFA

$$D = \{Q, \Sigma, \delta, q_0, F\}$$

$$Q = \{(A, G), (B, F), (C, E), (D)\}$$

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

$$q_0 = \{(A, G)\}$$

$$F = \{(D)\}$$

and δ is given by

$\delta((A, G), 0) = (B, F)$

$\delta((A, G), 1) = (A, G)$

$\delta((B, F), 0) = (A, G)$

$\delta((B, F), 1) = (C, E)$

$\delta((C, E), 0) = (D, A)$

$\delta((C, E), 1) = (B, F)$

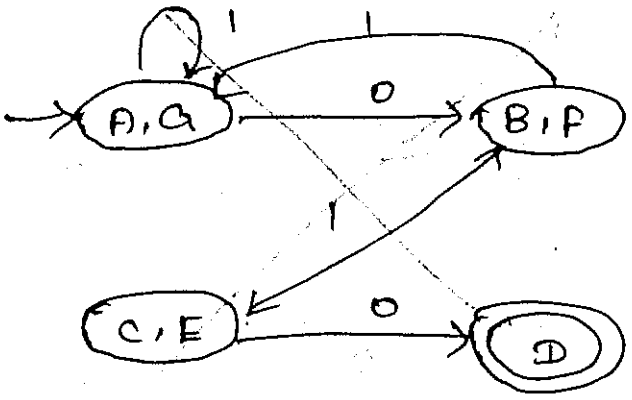
$\delta(D, 0) = D$

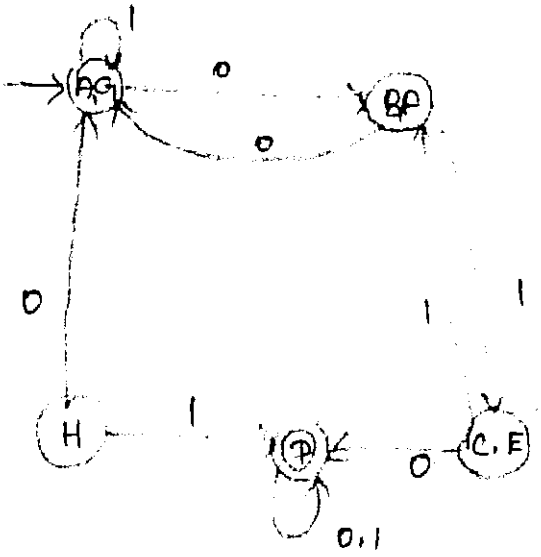
$\delta(D, 1) = A$

	0	1
→ (A, G)	(B, F)	(A, G)
(B, F)	(A, G)	(C, E)
(C, E)	(D A)	(B, F)
* (D)	(D)	(A) (D)
H	(A, G)	

DFA

F), (D)





1) Theorem:

For every DFA $A=(Q, \Sigma, \delta, S, F)$ they is Regular expression are such that $L(R)=L(A)$

proof:

Let 'L' be the set of language accepted by DFA $A=(\{q_1, q_2 \ldots q_n\}, \Sigma, \delta, q_1, F)$ with q_1 as the Initial State.

Let $R_{ij}^{(k)}$ be the regular expression describing the set of all strings 'x' such that $\delta(q_i, x)=q_j$ going throw 'k' no. of Intermediate state.

Basis:

Let $k=0$. $R_{ij}^{(0)}$ denotes the set of string which are either ε are single Symbol.

case(i): $i \neq j$

$$R_{ij}^{(0)} = \{a / \delta(q_i, a) = q_j\}$$

denotes the set of Symbol 'a' such that $\delta(q_i, a) = q_j$

$$R_{ij}^{(0)} = R_{ii}^{(0)} = \{\varepsilon\} \cup \{a / \delta(q_i, a) = q_j\}$$

$$R_{ii}^{(0)} = \varepsilon + a \text{ where } \delta(q_i, a) = q_i$$

Induction:

Let $k \geq 1$. Let there is path from State 'i' to state 'j' goes to 'k' no. of Intermediate State.

Assume that we have reach $R_{ij}^{(k-1)}$

i) Suppose if this path does not go through 'k' then $R_{ij}^{(k)} = R_{ij}^{(k-1)}$

ii) If the path goes through the state 'k' atleast once then

$R_{ik}^{(k-1)}$ $R_{kk}^{(k-1)}$ $R_{kj}^{(k-1)}$

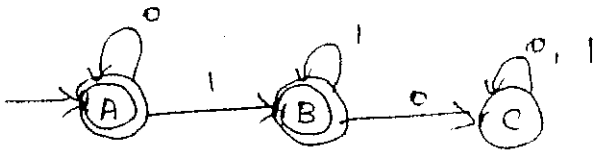
by the Formula,

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

Therefore by the Induction hypothesis we can Calculated they required Regular expression using this equation. Therefore For every DFA their is an Regular expression such that $L(R) = L(A)$ isproved.

hence the proved //

2)



$i = 1$

$j = 1, 2$

$K = 3$

$RE = R_{11}^{(3)} + R_{12}^{(3)}$

$$R_{ij} = \begin{cases} \epsilon + a & , i = j \\ a & , i \neq j \end{cases}$$

If $k=0$,

$R_{11}^{(0)} = \varepsilon + 0$

$R_{12}^{(0)} = 1$

$R_{21}^{(0)} = \phi$

$R_{22}^{(0)} = \varepsilon + 1$

$R_{13}^{(0)} = \phi$

$R_{23}^{(0)} = 0$

$R_{33}^{(0)} = \varepsilon + 0 + 1$

$R_{31}^{(0)} = \phi$

$R_{32}^{(0)} = \phi$

Let $k = 1$,

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= (\varepsilon + 0) + (\varepsilon + 0)(\varepsilon + 0)^* (\varepsilon + 0)$$

$$= 0 + 0 + (\varepsilon + 0)^* \cdot 0 + 00^* 0$$

$$= 0 + 0^*$$

$$= 0^*$$

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= 1 + (\varepsilon + 0)(\varepsilon + 0)^* 1$$

$$= 1 + 0 \cdot 0^* 1$$

$$= 1 + 0^* 1$$

$$= (\varepsilon + 0)^* 1 = 0^* 1$$

$$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \phi + \phi (\varepsilon + 0)^* (\varepsilon + 0)$$

$$= \phi + \phi = \phi$$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= (\varepsilon + 1) + \phi (\varepsilon + 0)^* 1$$

$$= (\varepsilon + 1) = 1$$

If $K = 2$

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$$

$$= \phi^* 0^* + \varepsilon^* 1 1^* \phi$$

$$= \varepsilon^*$$

$$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$

$$= \phi + (\varepsilon + 0)(\varepsilon + 0)^* \phi = \phi$$

$$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$

$$= 0 + \phi (\varepsilon + 0)^* \phi$$

$$= 0 + \phi = 0$$

$$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$$

$$= \phi + \phi (\varepsilon + 0)^* (\varepsilon + 0)$$

$$= \phi$$

$$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

$$= \phi + \phi (\varepsilon + 0)^* 1$$

$$= \phi$$

$$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)} \left(R_{11}^{(0)}\right)^* R_{13}^{(0)}$$

$$= (\varepsilon + 0 + 1) + \phi (\varepsilon + 0)^* \phi$$

$$= \varepsilon + 0 + 1 = 0 + 1$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$

$$= 0^* 1 + 0^* 1 1^* 1$$

$$= 0^* 1 + 0^* 1^*$$

$$= 0^* (1 + 1^*$$

$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$

$= \phi + 1 1^* \phi = \phi$

$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$

$= 1 + 11^* 1$

$= 1 + 1^*$

$R_{13}^{(2)} = R_{13}^{(2)} + R_{12}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$

$= \phi + \phi^* 11^* 0$

$= \phi + 0^* 1^* = 0^* 1^*$

$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$

$= 0 + 11^* 0$

$= 0 + 1^* 0 = (\varepsilon + 1^*) 0$

$= 1^* 0$

$R_{31}^{(2)} = R_{31}^{(2)} + R_{32}^{(1)} (R_{22}^{(1)})^* R_{21}^{(1)}$

$= \phi + \phi 1^* \phi = \phi$

$R_{32}^{(2)} = R_{32}^{(1)} + R_{32}^{(1)} (R_{22}^{(1)})^* R_{22}^{(1)}$

$= \phi + \phi 1^* 1$

$= \phi$

$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)} (R_{22}^{(1)})^* R_{23}^{(1)}$

$= (0 + 1) + \phi 1^* 0$

$= (0 + 1) + \phi$

$= 0 + 1$

If $K = 3$

$$R_{11}^{(3)} = R_{11}^{(2)} + R_{13}^{(2)} \left(R_{33}^{(2)}\right)^* R_{31}^{(2)}$$

$$= 0 + 0^* 1^* (0+1)^* \phi$$

$$= 0 + \phi$$

$$= 0$$

$$R_{12}^{(3)} = R_{12}^{(2)} + R_{13}^{(2)} \left(R_{33}^{(2)}\right)^* R_{32}^{(2)}$$

$$= 0^* (1+1)^* + 0^* 1^* (0+1)^* \phi$$

$$= 0^* (1+1)^*$$

$$RE = R_{11}^{(3)} + R_{12}^{(3)}$$

$$= 0 + 0^* (1+1^*)$$

$$RE = 0 + 0^* (1+1^*)$$

UNIT - III

context free Grammars & Language

An CFG is way of Discribing Language by recursive rules called as production rules (or) Subtraction rules.

Every CFG has Four Tuples

$G = \{V, T, S, P\}$

V - Set of variable also called as non Terminal

T - Set of i/p symbol also called as non - Terminal

S - set of start symbol non Terminal

P - set of production rule.

$A \rightarrow \alpha$

A → variable

α → for string of zero or more Terminals & non Terminals

1) Problem: construct the representing a set of palindromes over (0,1)

Sol:

palindromes is a set of string which give the same meaning when the string is reused:

i) For Length = 1

$S \leftarrow 0 | 1 | \varepsilon$

For Length > 1

$S \leftarrow 0S0$

$S \leftarrow 1S1$

∴ CFG is given by $G = \{V, T, S, P\}$

V = Set of variable $\{S\}$

$T = \{0, 1, \varepsilon\}$

$S = \{S\}$

2) construct CFG for the languages

$L = \{a^n / n \text{ is odd}\}$

Sol:

we have Two type of Length

Length = 1

Length = 2n + 1

Length = 1 $\boxed{A \rightarrow a}$

Length 2n+1

$\boxed{A \rightarrow aaA}$

∴ CFG is given by

$G_1 = \{V, T, S, P\}$

V = A

T = {a}

S = {a}

P = production rule

$A \rightarrow \alpha$

T, S, P}

3) $L = \{wcw^R / w$ is a string in $(0,1)\}$

Sol:

$S \rightarrow c$

$S \rightarrow 0c0$

$S \rightarrow 1c1$

$A \rightarrow 0A0$

$A \rightarrow 1A1$

$A \rightarrow c$

rges

ength

For Length = 1

$A \rightarrow c$ (constant)

For Length > 1

$A \rightarrow 1A1$

$A \rightarrow 0A0$

∴ CFG is given by G = {V, T, S, P}

V = {A}

T = {0, 1}

$S = \{A\}$

production rule

$$A \rightarrow \alpha$$

3) $L = \{ww^R / w \text{ is a string in } (a,b)\}$

Sol:

$A \rightarrow \varepsilon$

$A \rightarrow aAa$

$A \rightarrow bAb$

For Length = 1

$S \rightarrow A$ (constant)

For Length > 1

$A \rightarrow aAa$

$A \rightarrow bAb$

Therfore CFG is given by

$G_1 = \{V, T, S, P\}$

$V = \{S\}$

$T = \{a, b\}$

$S = \{S\}$

Derivations:-

It is the process of apply the production rules from the start symbol of CFG and expecting it till the given strin is reached.

Ex: For the given language ⇒ check whether L = { wcw^R / u (0+1)* } the strings 01C10, 10C10 are acceptable or not.

Sol:- construct the CFG for the given language.

$S \rightarrow C$

$S \rightarrow 0C0$

$S \rightarrow 1C1$

$A \rightarrow 0A0$ ($\because A \rightarrow 0A0$

$A \rightarrow 01A101$ ($\because A \rightarrow 1A1$)

$A \rightarrow 01C10$ ($\because A \rightarrow C$)

$\therefore$ the given strings accept by the language.

10C10

$A \rightarrow 1A1$ ($\because A \rightarrow 1A1$)

$A \rightarrow 10A01$ ($\because A \rightarrow 0A0$)

$A \rightarrow 10C01$ ($\because A \rightarrow C$)

Since the derived strings does not match with the given string, the given string is not accepted by the languages.

Types of Derivations:

- → Left most derivation (LMD)
- → Right most derivation (RMD)

Left most derivations:

At each step In the derivation the production Rule should be applied to the left most ~~dddd~~ variables.

Right most derivation:

At each step in the derivation the production Rule should be applied to the Right most variables.

; does
tring,
epted

(LMD)
n (RMD)

e
Rule
left

. Rule
e

problems:

Let $G = \{V, T, P, S\}$

$V = \{E\}$

$T = \{+, *, id\}$

$S = \{E\}$:

and P ⇒ $E \to E+E$

$E \to E*E$

$E \to id$

construct LMD & RMD for the strin
id+id * id

Left most derivation:

$E \underset{lm}{\Rightarrow} E+E$ ($\because E \to E+E$)

$\underset{lm}{\Rightarrow} id+E$ ($\because E \to id$)

$\underset{lm}{\Rightarrow} id+E*E$ ($\because E \to E*E$)

$\underset{lm}{\Rightarrow} id+id*E$ ($\because E*id$)

$\underset{lm}{\Rightarrow} id+id*id$ ($\because E*Id$)

Right most derivation:

$E \underset{rm}{\Rightarrow} E+E$ ($\because E \to E+E$)

$\underset{rm}{\Rightarrow} E+E*E$ ($\because E \to E*E$)

$\underset{rm}{\Rightarrow} E+E*id$ ($\because E*Id$)

$\underset{rm}{\Rightarrow} E+id+id$ ($\because E*id$)

$\Rightarrow_{rm}$ id + id * id (∵ E → id)

Parse Trees:

The derivations can be represented by Tree like structures called parse Trees.

condition 1 → Each Interial node (non-leaf node) is labeled by a variable

condition 2 → Each leaf node should be variable by a Terminal or ε.

condition 3:

yield of the parse Trees.

It is the string obtained by concatenating all the leaf nodes of parse Trees from the left most end towards right is called the yield of the parse Trees.

Every yield should be derived from the root of the the Tree. It is called the Start Symbol.

1) construct the parse Tree for that yield 1001 from the grammer 'G' defined by the production rules

$S \rightarrow A1B$

$A \rightarrow 0A / \varepsilon$

$B \rightarrow 0B / AB / \varepsilon / 1B$

Sol:

$S \Rightarrow A1B$

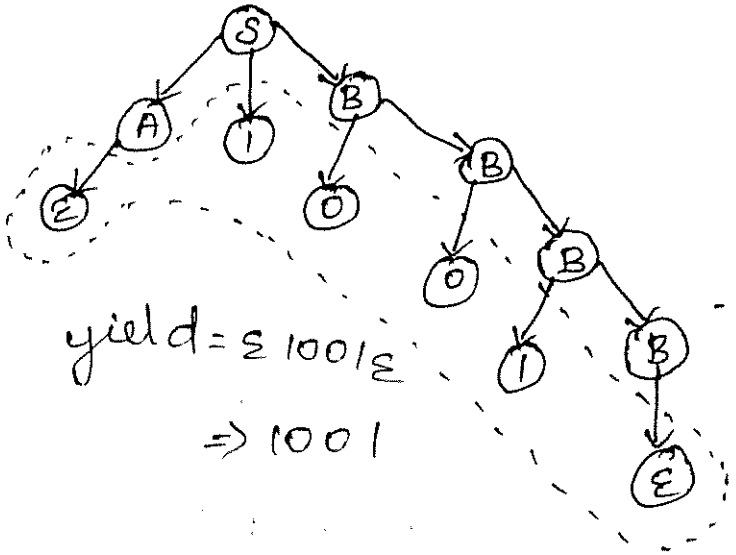
$\Rightarrow A10B \quad (\because B \rightarrow 0B)$

$\Rightarrow A100B \quad (\because B \rightarrow 0B)$

$\Rightarrow A1001B \quad (\because B \rightarrow 1B)$

$\Rightarrow 1001B \quad (\because A \rightarrow \varepsilon)$

$\Rightarrow 1001 \quad (B \rightarrow \varepsilon)$



ii) 00101

Sol:

$S \Rightarrow A1B$

$\Rightarrow A10B \quad (\because B \rightarrow 0B)$

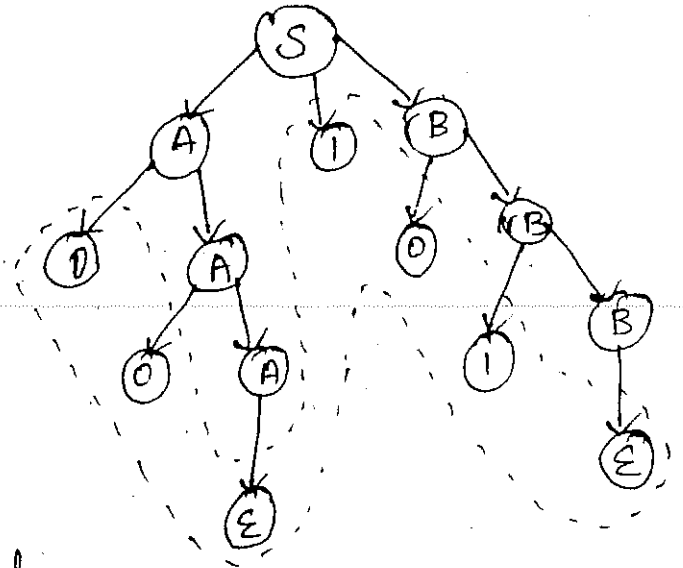
$\Rightarrow$ A 101 B ($\because$ B $\rightarrow$ 1B)

$\Rightarrow$ 0A101B ($\because$ A $\rightarrow$ 0A)

$\Rightarrow$ 00A101B ($\because$ A $\rightarrow$ 0A)

$\Rightarrow$ 00101B ($\because$ A $\rightarrow \varepsilon$)

$\Rightarrow$ 00101ε ($\because$ B $\rightarrow \varepsilon$)



yield = ε00101ε

= 00101

iii) 00011

Sol:

$S \Rightarrow$ A 1 B ($\because$ B $\rightarrow$ 0B)

$\Rightarrow$ A 101 B ($\because$ A $\rightarrow$ 0A)

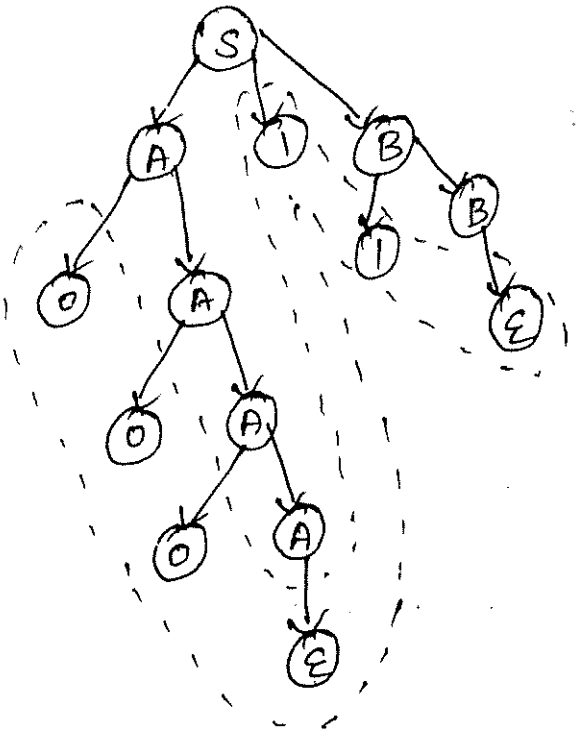
$\Rightarrow$ 0A11 B ($\because$ A $\rightarrow$ 0A)

$\Rightarrow$ 00A11B ($\because$ A $\rightarrow$ 0A)

$\Rightarrow$ 000A11B ($\because$ A $\rightarrow$ 0A)

$\Rightarrow 00011B$ $(\because A \to \varepsilon)$

$\Rightarrow 00011$ $(\because B \to \varepsilon)$



Ambiguity:

A grammer is set to be ambiguos if there is atleast one strings (w) in that grammer is having Two diffe parse Trees. each with the Sar Root and same yield.

1) Show that the grammer define by the production rules

$S \to SS$

$S \to aS/bS/\varepsilon$ is ambiguity.

$S \to SS$
$asbS$
$aaSbbS$
$aabb$

Sol:

Let $W = aabb$

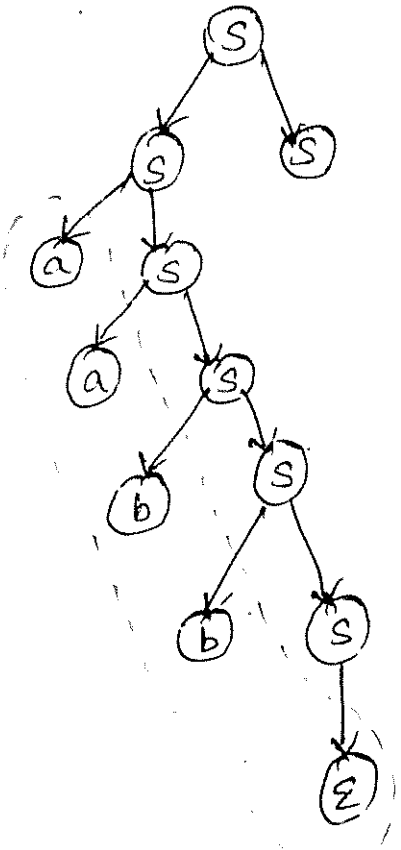
Left most derivation:

$S \Rightarrow SS$ [$\because S \to aS$]
$\Rightarrow aSS$ [$\because S \to aS$]
$\Rightarrow aaSS$ [$\because S \to bS$]
$\Rightarrow aabSS$ [$\because S \to bS$]
$\Rightarrow aabbSS$ [$\because S \to \varepsilon$]
$\Rightarrow aabbS$ [$\because S \to \varepsilon$]
$\Rightarrow aabb$

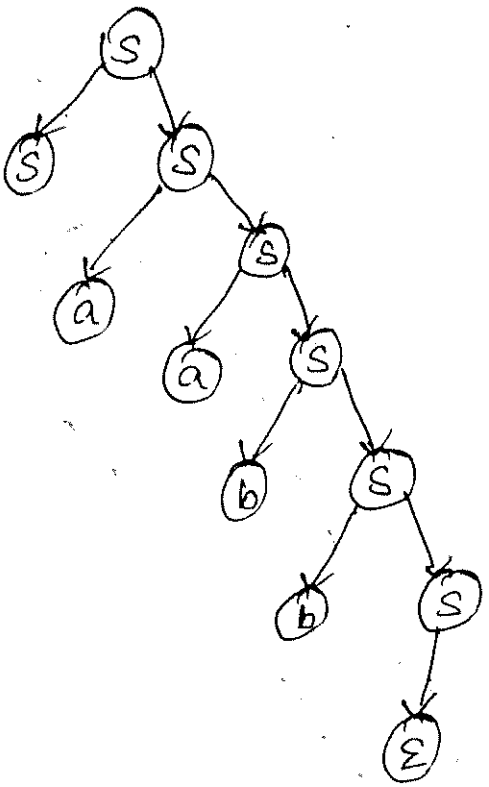
Right most derivations:

$S \Rightarrow SS$ [$\because S \to aS$]
$\Rightarrow Sas$ [$\because S \to aS$]
$\Rightarrow SSaa$ [$\because S \to bS$]
$\Rightarrow SSaab$ [$\because S \to bS$]
$\Rightarrow SSaabb$ [$\because S \to \varepsilon$]
$\Rightarrow Saabb$ [$\because S \to \varepsilon$]
$\Rightarrow aabb$

$S \rightarrow SS$
$aSbS$
$aaSbbS$
$aabb$



yield = aabb



yield = aabb

i) $S \to SbS / a$

Sol:

Let $w = aba$

LMD:

$S \Rightarrow SbS$

$\Rightarrow abS$ $(\because S \to a)$

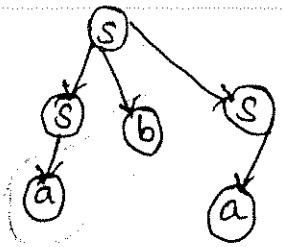
$\Rightarrow aba$ $(\because S \to a)$

RMD:

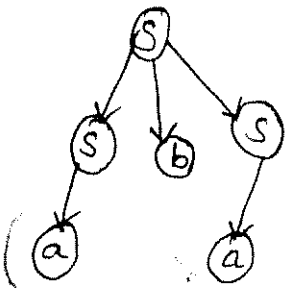
$S \Rightarrow SbS$

$\Rightarrow Sba$ $(\because S \to a)$

$\Rightarrow aba$ $(\because S \to a)$



yield = aba



yield = aba

ii)

iii) $S \to AA$

$A \to AAA$

$A \to a$

$A \to bA$

$A \to Ab$

Sol:

Let = abba

LMD:

$S \Rightarrow AA$

$\Rightarrow aA$ $[A \to a]$

$\Rightarrow abA$ $[A \to bA]$

$\Rightarrow abbA$ [A → bA]

$\Rightarrow abba$ [A → a]

RMD:

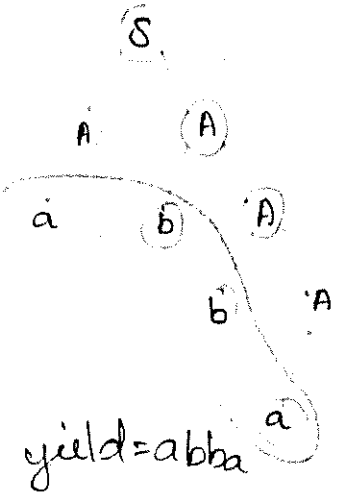
$S \Rightarrow AA$

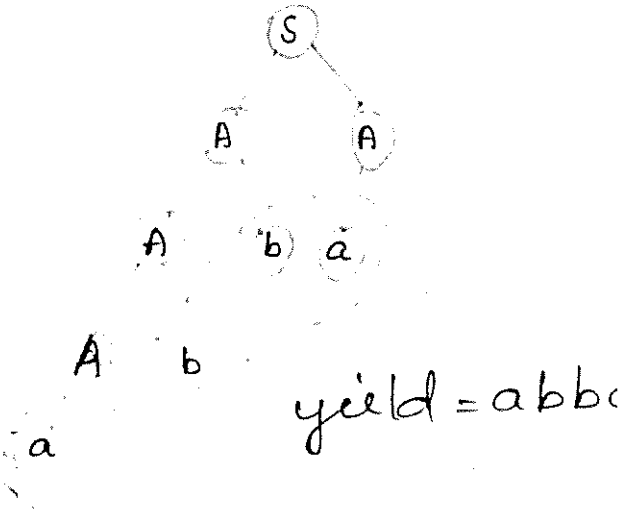
$\Rightarrow Aa$ [A → a]

$\Rightarrow Aba$ [A → Ab]

$\Rightarrow Abba$ [A → Ab]

$\Rightarrow abba$ [A → a]





ii) S → ictS

S → ictses

S → a

S → b

Sol: Let w = ictaeb

LMD:

$S \Rightarrow ictses$

$\Rightarrow ictaes$ (∵ S → a)

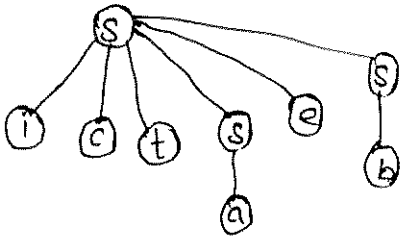
$\Rightarrow ictaeb$ (∵ S → b)

RMD:

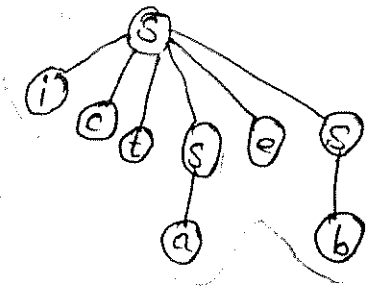
$S \Rightarrow ictses$

$\Rightarrow ictseb$ (∵ S → b)

$\Rightarrow ictaeb$ (∵ S → a)



yield = ictaeb



yield = ictaeb

2m (or) 6m

1) Find LMD & RMD For + * - xy xy

For the grammer

$E \rightarrow + EE$

$E \rightarrow * EE$

$E \rightarrow - EE$

$E \rightarrow x$

$E \rightarrow y$

2) Let G1 be the grammer $S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

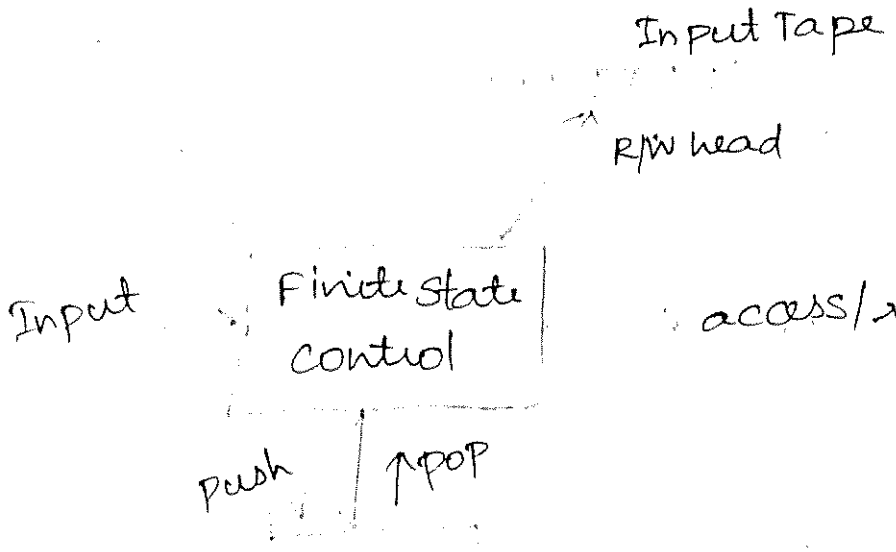
$B \rightarrow 1 \mid 1S \mid 0BB$

Find Lmd & rmd For the string 00110101 with parse tree.

PUSH DOWN AUTOMATA (PDA):

It is an Finite automata with the control of both an Input Tag and a stack on which it can

store a string of stack symbol.



A push down automata consits of 7 Tuples,

$P = (Q, \Sigma, \vdash, \delta, q_0, z_0, F)$

Q → For Finite non empty set of States

Σ → Finite set of Input Symbols

$\vdash$ → Finite set of Stack Symbo (Push down Symbo

δ → production rules

q_0 → Initial state of PDA

z_0 → Initial start symbol of stack

F → Set of accepting states.

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

$$\delta(q_0, 0, 0) \longrightarrow (q_1, 0)$$

Operations of PDA:

- Push
- Pop

PUSH:

It is the process of Inserting a Input symbol into the Stack.

current state → q_0, I/P symbol → 0, stack symbol → z_0, Resultant state → q_0, Resultant stack → $0z_0$

$$\delta(q_0, 0, z_0) = (q_0, 0z_0)$$

$$\delta(q_0, 1, z_0) = (q_0, 1z_0)$$

$$\delta(q_0, 1, 1) = (q_0, 11)$$

POP:

$$\left.\begin{aligned}\delta(q_0, 0, 0) &= (q_1, \varepsilon)\\ \delta(q_0, 1, 1) &= (q_1, \varepsilon)\end{aligned}\right\} \text{accepted}$$

$$\left.\begin{aligned}\delta(q_0, 0, 1) &=\\ \delta(q_0, 1, 0) &=\end{aligned}\right\} \text{Rejected}$$

1) construct PDA to accept the langu

$L = \{ww^R / w \text{ is in } (0+1)^*\}$

Sol:-

General steps:

Step1: The construction of PDA sta
with state q_0 In this state Read
i/p symbols and store them an
to the stack by Pushing a copy
of each i/p symbol.

to the
state

sultant
stack

step2: When its Finished at
Transitions occured to the
next state q_1

step3: Ones the PDA q_1 now
compare the i/p symbols with th
stack symbol if the match the
its poped out of the stack. If

epted

the do not match the entri
string is Rejected.

steps: When all the symbols of
stack are accepted then the
Final state is Reached. It
has Two Types of acaptana

i) Acceptance by empty stack

ii) Acceptance by Final state

1) construct PDA to accept the language $L = \{WW^R / W \text{ is in } (0+1)^*\}$ by empty stack.

Sol:

3-state q_0, q_1, q_F

The required PDA For the given language is

$$P = \{Q, \Sigma, \vdash, \delta, q_0, z_0, F\}$$

Where $Q = \{q_0, q_1, q_F\}$.

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

$\vdash = \{z_0, 0, 1\}$

q_0 Initial state $= \{q_0\}$

z_0 Initial state $= \{z_0\}$

Final state $= \{q_F\}$

and δ is given by,

① Push

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 1, z_0) = (q_0, 1z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_0, 1, 0) = (q_0, 10)$

$\delta(q_0, 0, 1) = \delta(q_0, 01)$

$\delta(q_0, 1, 1) = (q_0, 11)$

(ii) State Transition:

$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$

$\delta(q_0, \varepsilon, 0) = (q_1, 0)$

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

(iii) POP:

$\delta(q_1, 0, 0) = (q_1, \varepsilon)$ } accepted

$\delta(q_1, 1, 1) = (q_1, \varepsilon)$

$\delta(q_1, 0, 1)$ } Rejected

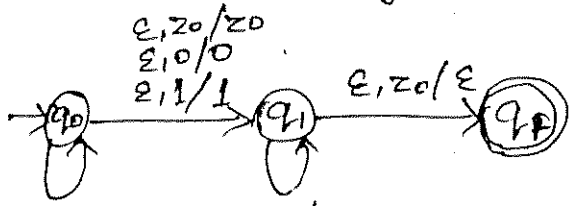
$\delta(q_1, 1, 0)$ }

(iv) Reach the ~~Final~~ empty stack:

$\delta(q_1, \varepsilon, z_0) = (q_F, \varepsilon)$

Since We have reach the Empty stack the PDA can accepted The String.

Transition diagram



0,z0/0z0
1,z0/1z0
0,0/00
1,1/11
0,1/01
1,0/10
1,1/11

0,0/ε
1,1/ε

To check the designed PDA

W = 1011

Input string = WW^R = 10111101

Push:

$\delta(q_0, 10111101, z_0) \vdash (q_0, 0111101, 1z_0)$

$\vdash (q_0, 111101, 01z_0)$

$\vdash (q_0, 11101, 101z_0)$

$\vdash (q_0, \varepsilon 1101, 1101z_0)$

State Transition

$\vdash (q_1, 1101, 1101z_0)$

POP:

$\vdash (q_1, 101, 101z_0)$

$\vdash (q_1, 01, 01z_0)$

$\vdash (q_1, 1, 1z_0)$

$\vdash (q_1, \varepsilon, z_0)$

$\vdash (q_f, \varepsilon)$

∴ Since we have reached the empty stac
the given string is accepted. //.

2) $L = \{ wCw^R / w \text{ is in } (0+1)^* \}$ const
to the PDA accept to the
Final stack.

Sol:

The Required PDA for the
given language is

$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

where $Q = \{q_0, q_1, q_F\}$

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

$\Gamma = \{z_0, 0, 1\}$

q_0 Initial state $= \{q_0\}$

z_0 Initial state $= \{z_0\}$

Final state $= \{q_F\}$

and δ is given by,

(1) push

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 1, z_0) = (q_0, 1z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_0, 1, 0) = (q_0, 10)$

$\delta(q_0, 0, 1) = (q_0, 01)$

$\delta(q_0, 1, 1) = (q_0, 11)$

(ii) State Transition:

$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$

$\delta(q_0, \varepsilon, 0) = (q_1, 0)$

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

(iii) POP:

$\delta(q_1, 0, 0) = (q_1, \varepsilon)$
$\delta(q_1, 1, 1) = (q_1, \varepsilon)$ } accepted

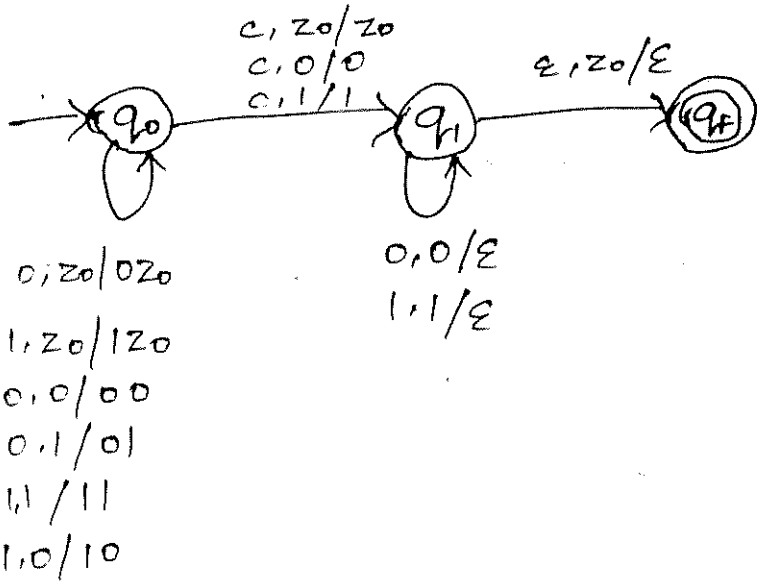
$\delta(q_1, 0, 1)$
$\delta(q_1, 1, 0)$ } Rejected

(iv) Reach the Final Stack:

$\delta(q_1, \varepsilon, z_0) = (q_F, \varepsilon)$

Since we have reach the Final stack the PDA can accepted the string.

Transition diagram:



To check the designed PDA

W = 1011

Input string = WCW^R = 1011C1101

Push:

$\delta(q_0, 1011C1101, z_0) \vdash (q_0, 0111101, 1z_0$

$\vdash (q_0, 111101, 01z_0)$

$\vdash (q_0, 11101, 101z_0)$

$\vdash (q_0, 1101, 1101z_0)$

state Transition:

$\vdash (q_1, 1101, 1101z_0)$

pop:

$\vdash (q_1, 101, 101z_0)$

$\vdash (q_1, 01, 01z_0)$

$\vdash (q_1, 1, 1z_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_F, \epsilon)$

∴ Since we have reached the empty stack the given string is accepted.

(1) construct the PDA accepting $L = \{a^n b^m a^n / m, n \geq 1\}$ by empty stack

2) construct the PDA accepting $L = \{a^n b^n / n \geq 1\}$ by Final stack.

1) Ans:

The required PDA for given language is

$P = \{Q, \Sigma, \vdash, \delta, q_0, z_0, F\}$

where $Q = \{q_0, q_1, q_2, q_F\}$

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

$\vdash = \{z_0, a\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_F\}$

and δ is given by

Push

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aa)$

State Transition:

$\delta(q_0, b, z_0) = (q_1, z_0)$

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

$\delta(q_1, b, z_0) = (q_1, z_0)$

$\delta(q_1, b, a) = (q_1, a)$

$\delta(q_1, a, a) = (q_2, \varepsilon)$

$\delta(q_2, a, a) = (q_2, \varepsilon)$

POP:

$\delta(q_1, a, a) = (q_2, \varepsilon)$

$\delta(q_2, a, a) = (q_2, \varepsilon)$ } accepted

$\delta(q_1, a, z_0)$

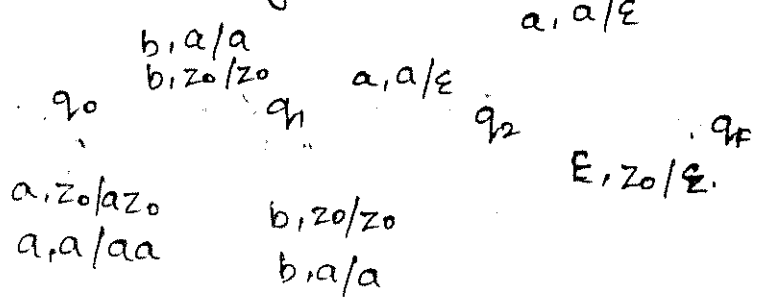
$\delta(q_2, a, z_0)$ } Rejected

Reach the empty stack.

$\delta(q_0, \varepsilon, z_0) = (q_F, \varepsilon)$

Since we have empty stack then the PDA can accept the string.

Transition diagram:



To check designed PDA

Let n = 3 m = 2

Input string - aaabbaaa

$\delta(q_0, aaabbaaa, z_0) \vdash (q_0, aabbaaa, az_0)$ (Push)

$\vdash (q_0, abbaaa, aaz_0)$ (Push)

$\vdash (q_0, bbaaa, aaaz_0)$ (Push)

State Transition:

$\delta(q_0, bbaaa, aaaz_0) \vdash (q_1, baaa, aaaz_0)$ (Transition)

$\vdash (q_1, aaa, aaaz_0)$

State Transition & POP operation

$\delta(q_1, aaa, aaaz_0) \vdash (q_2, aa, aaz_0)$ (Transition)

$\vdash (q_2, a, az_0)$ (POP)

$\vdash (q_2, \varepsilon, z_0)$ (POP)

$\vdash (q_f, \varepsilon)$ (Reached)

∴ Since we have reach the empty stack the given string is accepted.

2) Ans:

The required PDA for given language is

$P = \{Q, \Sigma, \vdash, \delta, q_0, z_0, F\}$

$Q = \{q_0, q_1, q_f\}$

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

$\vdash = \{a, z_0\}$

$q_0 = \{q_0\}$

$Z_0 = \{Z_0\}$

$F = \{q_F\}$

and δ is given by

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$ (Push)

$\delta(q_0, a, a) = (q_0, aa)$ (Push)

$\delta(q_0, \varepsilon, Z_0) = (q_1, Z_0)$ (Transition)

$\delta(q_0, \varepsilon, a) = (q_1, a)$ (Transition)

$\delta(q_0, b, Z_0) \rightarrow$ rejected

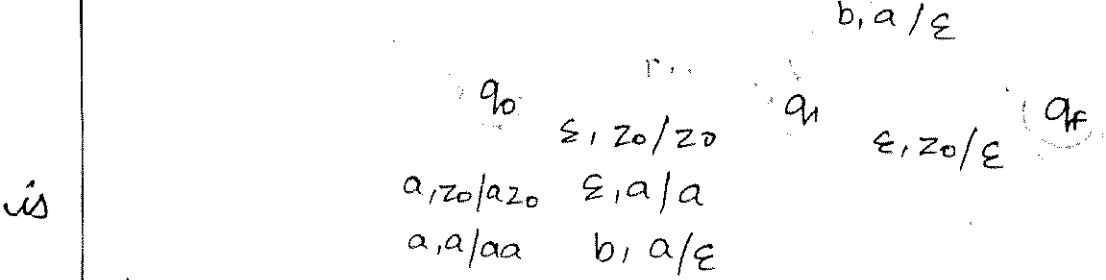
$\delta(q_0, b, a) = (q_1, \varepsilon)$ (Transition + pop)

$\delta(q_1, b, a) = (q_1, \varepsilon)$ (pop)

$\delta(q_1, b, Z_0)$
$\delta(q_1, a, Z_0)$
$\delta(q_1, a, a)$
$\delta(q_1, \varepsilon, a)$
} Rejected

$\delta(q_1, \varepsilon, Z_0) = (q_F, \varepsilon)$ (Reached)

Transition diagram.



To check designed PDA

Let n = 3

Input string : aaabbb

$\delta(q_0, aaabbb, z_0) \vdash (q_0, aabbb, az_0)$ (Push)

$\vdash (q_0, abbb, aaz_0)$ (Push)

$\vdash (q_0, bbb, aaaz_0)$ (Push)

$\vdash (q_1, bb, aaz_0)$ (Transition + POP)

$\vdash (q_1, b, az_0)$ (POP)

$\vdash (q_1, \varepsilon, z_0)$ (POP)

$\vdash (q_F, \varepsilon)$ (Reached)

Since we have reach the Final state the given string is accepted.

Equivalence of PDA & CFG:

CFG → PDA :-

Let $G = \{V, T, P, S\}$ now we are going to construct the PDA that accepts the language of the grammer $L(G)$

$P = \{\{q_0\}, T, VUT, \delta, q, s\}$

where

T → Terminals

V - variables

For each variable 'A'

$\delta(q, \varepsilon, A) = \{q, B\}$ $\therefore A \to B$

For every Terminal 'a'

$$\delta(q, a, a) = \{q, \varepsilon\}$$

1) Convert the CFG to PDA

$E \rightarrow E + E$

$E \rightarrow id$

Sol:

They equivalent PDA for the grammer is

$$P = \{\{q\}, T, V \cup T, \delta, q, S\}$$

$$P = \{\{q\}, \{id, +\}, \{E, Id, +\}, \delta, q, S\}$$

Where δ is given by

$$\delta(q, \varepsilon, E) = \{(q, E+E), (q, id)\}$$

$$\delta(q, id, id) = \{q, \varepsilon\}$$

$$\delta(q, +, +) = \{q, \varepsilon\}$$

$$W = id + Id + id$$

$$\delta(q, w, S) \vdash \delta(q, Id+Id+id, E)$$

$$\vdash \delta(q, Id+id+Id, E+E) \quad (\because E \rightarrow E+)$$

$$\vdash \delta(q, id+Id+id, id+E) \quad (E \rightarrow Id)$$

$$\vdash \delta(q, +id+id, +E) \quad (POP)$$

$$\vdash \delta(q, id+id, E) \quad (POP)$$

$\vdash \delta(q, id+id, E+E)$ $(\because E \to E+E)$

$\vdash \delta(q, id+id, id+E)$ $(\because E \to id)$

$\vdash \delta(q, +id, +E)$ $(\because POP)$

$\vdash \delta(q, id, E)$ $(\because POP)$

$\vdash \delta(q, id, id)$ $(E \to id)$

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

2) convert the CFG to PDA

$S \to aA$

$A \to aAbc \mid bB \mid a$

$B \to b$

$C \to c$

Sol:

The equivalent PDA for the grammer is

$P = \{\{q\}, T, VUT, \delta, q, S\}$

$P = \{\{q\}, a, b, c, SA, B, C, a, b, c, \delta, q, S\}$

where δ is given by,

~~$\delta(q, \varepsilon, S, A, B, C) = \{(q, aA)\}$~~

$\delta(q, \varepsilon, S) = \{q, aA\}$

$\delta(q, \varepsilon, A) = \{(q, aABC), (a, bB)\}$

$\delta(q, a, B) = (a, b)$

$\delta(q, a, c) = (a, c)$

$\delta(q, a, a) = (q, \varepsilon)$

$\delta(q, b, b) = (q, \varepsilon)$

$\delta(q, a, c) = (q, \varepsilon)$

Take a sample

$w = aaabc$

$\delta(q, w, s) \vdash \delta(q, aaabc, s)$

$\vdash \delta(q, aaabc, aA)$

$\vdash \delta(q, aabc, A)$

$\vdash \delta(q, aabc, aaB)$ (Pop)

$\vdash \delta(q, abc, aBc)$ (A → a)

$\vdash \delta(q, bc, Bc)$ (Pop)

$\vdash \delta(q, bc, bc)$ (B → b)

$\vdash \delta(q, c, c)$ (Pop)

$\vdash \delta(q, c, c)$ (c → c)

$\vdash \delta(q, a)$ (Pop)

∴ The given string is accepted the PDA. ✓

PDA to Grammar:

$G_1 = \{V, T, P, S\}$

$V = [\{S\} \cup \{[q, z, q']\} \{(q, q' \in Q), z \in T\}$

For constructing production rules (P)

$R_1 : S \rightarrow$ productions

$S \rightarrow [q_0, z_0, q]$ For every q in Q.

R_2 : POP (erasing moves)

$\delta(q, a, z) = (q', \varepsilon)$

$[q, z, q'] \rightarrow a$

R_3 : PUSH (Non-Erasing moves)

$\delta(q, a, z) = (q', z_1 z_2 z_3 \ldots)$

$[q, z, q'] \rightarrow a\, [q_1, z_1, q_2] [q_1 q_2 q_3 \ldots]$

problem:

1) $M = (\{q_0, q_1\}, \{a, b\}, \{z_0, z_1\}, \delta, q_0, z_0, \phi\}$

Where δ is given by

$\delta(q_0, b, z_0) = (q_0, zz_0)$ push

$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$ pop

$\delta(q_0, b, z) = (q_0, zz)$ push

$\delta(q_0, a, z) = (q_1, z)$ push

$\delta(q_1, a, z_0) = (q_0, z_0)$ push

$\delta(q_1, b, z) = (q_1, \varepsilon)$ pop

construct CFG.

Sol:

Step1: The resultant CFG for the given PDA is $G = \{V, T, P, S\}$

T (Terminals) = $\{a, b\}$

$V_n = \{S, [q_0, z_0, q_0], [q_0, z_0, q_1],$

$[q_0, z_1, q_0], [q_0, z_1, q_1], [q_1, z_0, q_0]$

$[q_1, z_0, q_1], [q_1, z_1, q_0], [q_1, z_1, q_1]\}$

Step 2:

S-productions:

To construct S-productions

$S \to [q_0, z_0, q]$ for every q in Q

- S-productions are,

$S \to [q_0, z_0, q_0]$ ---- (P1)

$S \to [q_0, z_0, q_1]$ ---- (P2)

Step 3:

For Erasing moves:

$\delta(q, a, z) = (q', \varepsilon)$

$\Rightarrow [q, z, q'] \to a$

For $\delta(q_0, z, z_0) = (q_0, z)$

$\Rightarrow \boxed{[q_0, z_0, q_0] \rightarrow \varepsilon}$ ----> (P3)

For $\delta(q_1, b, z) = (q_1, \varepsilon)$

$\Rightarrow \boxed{(q_1, z, q_1) \rightarrow b}$ ----> (P4)

Step 4:

Non-Erasing moves

$\delta(q_1, a, z) = (q_1, z_1 z_2 z_3 \ldots)$

$\Rightarrow (q_1, z, q_1) \rightarrow a\,[q_1 z_1 q_2]\,[q_2, z_2, q_3]$

* $\delta(q_0, b, z_0) = (q_0, z z_0)$

$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_0]$ ----> (P5)

$[q_0, z_0, q_0] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_0, q_0]$ ----> (P6)

$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z_0, q_1]$ ----> (P7)

$[q_0, z_0, q_1] \rightarrow b\,[q_0, z, q_1]\,[q_1, z_0, q_1]$ ----> (P8)

State $q_0 q_1$
$q_0 \rightarrow$ first
$q_0 \rightarrow$ second
$q_1 \rightarrow$ second

* $\delta(q_0, b, z) = (q_0, z z)$

$[q_0, z, q_0] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_0]$ ----> (P9)

$[q_0, z, q_0] \rightarrow b\,[q_0, z, q_1]\,[q_1, z, q_0]$ ----> (P10)

$[q_0, z, q_1] \rightarrow b\,[q_0, z, q_0]\,[q_0, z, q_1]$ ----> (P11)

$[q_0, z, q_1] \rightarrow b\,[q_0, z, q_1]\,[q_1, z, q_1]$ ----> (P12)

* $\delta(q_0, a, z) = (q_0, z)$

$[q_0, z, q_0]$

$[q_0, z, q_0] \rightarrow a[q_1, z, q_0] \rightarrow (P_{13})$

$[q_0, z, q_1] \rightarrow a[q_1, z, q_1] \rightarrow (P_{14})$

* $\delta(q_1, a, z_0) = (q_0, z_0)$

$[q_1, z_0, q_0] \rightarrow a[q_0, z_0, q_0] \rightarrow (P_{15})$

$[q_1, z_0, q_1] \rightarrow a[q_0, z_0, q_1] \rightarrow (P_{16})$

$z_2, q_3]$

$z_0, q_0] \rightarrow (P_5)$

$z_0, q_0] \rightarrow (P_6)$

$z_0, q_1] \rightarrow (P_7)$

$z_0, q_1] \rightarrow (P_8)$

$\therefore$ The Rules from P_1 to P_{16} are they required production Rules of the CFG.

Does the context free gramm has been constructed successfully from the given PDA.

$] \rightarrow (P_9)$

$] \rightarrow (P_{10})$

$] \rightarrow (P_{11})$

$] \rightarrow (P_{12})$

Deterministic PDA (DPDA):

A push down automata is deterministic if it has atmost one move for a given state, input symbol and stack symbol

A PDA = $\{Q, \Sigma, \vdash, \delta, q_0, z_0, F\}$

is set to be deterministic if and only if $\delta(q, a, x)$ has atmost one number for any q in Q, a in Σ, x in Γ.

A DPDA is more powerful then that of NPDA because NPDA produces ambigious grammar by reaching it's final state or by empty in the stack. but DPDA produces only unambigious grammar.

1) construct CFG From

$\delta(q_0, 0, z_0) = (q_0, XZ_0)$ } push
$\delta(q_0, 0, X) = (q_0, XX)$ }

$\delta(q_0, 1, X) = (q_1, \varepsilon)$ }
$\delta(q_1, 1, X) = (q_1, \varepsilon)$ } pop
$\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$ }
$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$ }

Sol:

Step 1: The resultants CFG For the given PDA is $G = \{V, T, P, S\}$

where,

$T = \{0, 1\}$

and
one
in

' then

ι by
ε by
DA
s

For
P, S}

$V = \{S, [q_0, z_0, q_0], [q_0, z_0, q_1], [q_1, z_0, q$

$[q_1, z_0, q_1], [q_0, X, q_0], [q_0, X, q_1],$

$[q_1, X, q_0] [q_1, X, q_1]$

Step 2:

s - production:

Rule 1:

To construct s-productions

$S \to [q_0, z_0, q]$ for every q in Q

s-productions are,

$S \to [q_0, z_0, q_0]$ → (P1)

$S \to [q_0, z_0, q_1]$ → (P2)

Step 3:

Rule 2: For Erasing moves: (pop)

For $\delta(q_0, 1, X) = (q_1, \varepsilon)$

$\Rightarrow [q_0, X, q_1] \to 1$ → (P3)

For $\delta(q_1, 1, X) = (q_1, \varepsilon)$

$\Rightarrow (q_1, X, q_1) \to 1$ → (P4)

For $\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$

$\Rightarrow (q_1, X, q_1) \to \varepsilon$ → (P5)

For $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

$\Rightarrow \boxed{[q_1, z_0, q_1] \to \varepsilon}$ ----> (P6)

Step 4:

Non-Erasing moves: (push)

Rule 3: push

$$\delta(q, a, z) = (q', z, z_2 z_3 \ldots)$$

$\Rightarrow (q, , z, q_1) \Longrightarrow a[q_1 z_1 q_2][q_2, z_2, q_3]$ 1)

* $\delta(q_0, 0, z_0) = (q_0, X z_0)$

$[q_0, z_0, q_0] \to 0\,[q_0, X, q_0]\,[q_0, z_0, q_0]$ ---> P_7

$[q_0, z_0, q_0] \to 0\,[q_0, X, q_1]\,[q_1, z_0, q_0]$ ---> P_8

$[q_0, z_0, q_1] \to 0\,[q_0, X, q_0]\,[q_0, z_0, q_1]$ ---> P_9

$[q_0, z_0, q_1] \to 0\,[q_0, X, q_1]\,[q_1, z_0, q_1]$ ---> P_{10}

* $\delta(q_0, 0, X) = (q_0, XX)$

$[q_0, X, q_0] \to 0\,[q_0, X, q_0]\,[q_0, X, q_0]$ ---> P_{11}

$[q_0, X, q_0] \to 0\,[q_0, X, q_1]\,[q_1, X, q_0]$ ---> P_{12}

$[q_0, X, q_1] \to 0\,[q_0, X, q_0]\,[q_0, X, q_1]$ ---> P_{13} 2.

$[q_0, X, q_1] \to 0\,[q_0, X, q_1]\,[q_1, X, q_1]$ ---> P_{14}

∴ The Rules From P_1 to P_{14} are They required productions Rules of the CFG.

Does the context Free gramma has been constructed successfully From the given PDA.

1) construct CFG for

$P = \{(P, q), \{0, 1\}, \{x, z_0\}, \delta, q_0, z_0$

where δ is given by.

$\delta(q, 1, z_0) = (q, x z_0)$

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

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

$\delta(q, \varepsilon, z_0) = (q, \varepsilon)$

$\delta(P, 1, x) = (P, \varepsilon)$

$\delta(q, 0, z_0) = (q, z_0)$

2) construct PDA For $L = \{0^n 1^n / n \geq 1\}$ and convert into its equivalent CFG.

* Sol:

The required PDA for the given Language is

$$P = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

Where,

$$Q = \{q_0, q_1, q_F\}$$

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

$$\Gamma = \{0, 1, z_0\}$$

$$q_0 = \{q_0\}$$

$$z_0 = \{z_0\}$$

$$F = \{q_F\}$$

and δ is given by,

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$ (push)

$\delta(q_0, 0, 0) = (q_0, 00)$ (push)

~~$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$~~ (Transition)

~~$\delta(q_0, \varepsilon, 0) = (q_1, 0)$~~ (Transition)

$\delta(q_0, 1, z_0) \rightarrow$ rejected

$\delta(q_0, 1, 0) = (q_1, \varepsilon)$ (Transition +pop)

$\delta(q_1, 1, 0) = (q_1, \varepsilon)$ (pop)

State Transition:

$$\delta(q_0, 1, 0) = (q_1, \varepsilon)$$

Pop:

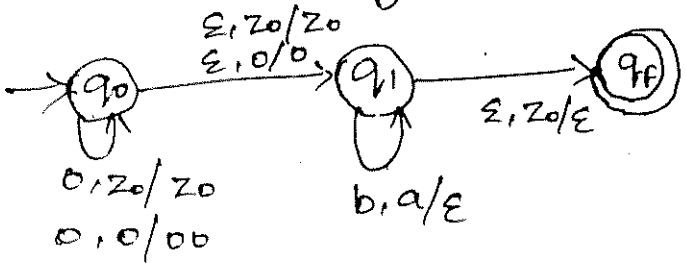
$$\delta(q_1, 1, 0) = (q_1, \varepsilon)$$

$\delta(q_1, 1, z_0)$
$\delta(q_1, 0, z_0)$
$\delta(q_1, 0, 0)$
$\delta(q_1, \varepsilon, 0)$
} Rejected

each Final State

$$\delta(q_1, \varepsilon, z_0) = (q_F, \varepsilon)$$

Transition diagram:



Sample String: Let n = 3

Input String = 000111

Push:

$$\delta(q_0, 000111, z_0) \vdash (q_0, 00111, 0z_0)$$
$$\vdash (q_0, 0111, 00z_0)$$
$$\vdash (q_0, 111, 000z_0)$$
$$\vdash (q_0, \varepsilon 111, 000z_0)$$

Transition:

$\vdash (q_1, 111, 000Z_0)$

POP:

$\vdash (q_1, 11, 00Z_0)$

$\vdash (q_1, 1, 0Z_0)$

$\vdash (q_1, \varepsilon, Z_0)$.

Reach Final state:

$\vdash (q_F, \varepsilon)$

∴ Since we have reached the Final State.

∴ The given string is accepting

Conversion of CFG:

$P = \{(q_0, q_1, q_F), (0,1), \{0, Z_0\}, \delta, q_0, Z_0\}$

where δ is given by

$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_0, 1, 0) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, Z_0) = (q_F, \varepsilon)$

$\delta(q_1, 1, 0) = (q_1, \varepsilon)$

The required CFG for given PDA is

Step-1:

$G = \{V, T, P, S\}$

$T = \{0, 1\}$

$V_n = \{ S, [q_0, z_0, q_0], [q_0, z_0, q_1], [q_0, z_0, q_F],$

$[q_0, 0, q_0], [q_0, 0, q_1], [q_0, 0, q_F],$

$[q_1, z_0, q_0], [q_1, z_0 q_1], [q_1, z_0, q_F]$

$[q_1, 0, q_0], [q_1, 0, q_1], [q_1, 0, q_F],$

$[q_F, z_0, q_0], [q_F, z_0, q_1] [q_F, z_0, q_1$

$[q_F, 0, q_0], [q_F, 0, q_1], [q_F, 0, q_F] \}$

Step 2:

$S \rightarrow$ productions

To construct $S \rightarrow$ productions

$S \rightarrow [q_0, z_0, q]$ for q in Q

$S \rightarrow$ productions

$S \rightarrow [q_0, z_0, q_0] \dashrightarrow (P_1)$

$S \rightarrow [q_0, z_0, q_1] \dashrightarrow (P_2)$

$S \rightarrow [q_0, z_0, q_F] \dashrightarrow (P_3)$

Step 3:

For Erasing moves

$\delta(q, a, z) = (q', \varepsilon)$

$[q, z, q'] \rightarrow a$

$\delta(q_0, 1, 0) = (q_1, \varepsilon)$

$\Rightarrow [q_0, 0, q_1] \longrightarrow 1 \dashrightarrow (P_4)$

$\delta(q_1, \varepsilon, z_0] = (q_F, \varepsilon)$

$\Rightarrow [q_1, z_0, q_F] \longrightarrow \varepsilon \dashrightarrow (P_5)$

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

$\Rightarrow \delta[q_1, 0, q_1] \longrightarrow 1 \text{ ----->} (P_6)$

Step 4:

For non-Erasing moves

$\delta(q_1, a, z) = (q_1, z_1 z_2 z_3) \cdots$

$\Rightarrow [q_1, z, q] \rightarrow a\,[q_1, z_1, q_2][q_2, z_2, q_3]$

$\delta(q_0, 0, z_0) = (q_0, 0 z_0)$

$[q_0, z_0, q_0] \rightarrow 0[q_0, 0, q_0][q_0, z_0, q_0] \text{ ---->} (P_7)$

$[q_0, z_0, q_0] \rightarrow 0\,[q_0, 0, q_1]\,[q_1, z_0, q_0] \text{ ---->} (P_8)$

$[q_0, z_0, q_0] \rightarrow 0\,[q_0, 0, q_F]\,[q_F, z_0, q_0] \text{ ---->} (P_9)$

$[q_0, z_0, q_1] \rightarrow 0\,[q_0, 0, q_0]\,[q_0, z_0, q_1] \text{ ---->} (P_{10})$

$[q_0, z_0, q_1] \rightarrow 0[q_0, 0, q_1]\,[q_1, z_0, q_1] \text{ ---->} (P_{11})$

$[q_0, z_0, q_1] \rightarrow 0[q_0, 0, q_F]\,[q_F, z_0, q_1] \text{ ---->} (P_{12})$

$[q_0, z_0, q_F] \rightarrow 0\,[q_0, 0, q_0]\,[q_0, z_0, q_F] \text{ ---->} (P_{13})$

$[q_0, z_0, q_F] \rightarrow 0\,[q_0, 0, q_1]\,[q_1, z_0, q_F] \text{ ---->} (P_{14})$

$[q_0, z_0, q_F] \rightarrow 0\,[q_0, 0, q_F]\,[q_F, z_0, q_F] \text{ ---->} (P_{15})$

$\delta(q_0, 0, 0) = (q_0, 00)$

$[q_0, 0, q_0] \rightarrow 0\,[q_0, 0, q_0]\,[q_0, 0, q_0] \text{ ---->} (P_{16})$

$[q_0, 0, q_0] \rightarrow 0\,[q_0, 0, q_1]\,[q_1, 0, q_0] \text{ ---->} (P_{17})$

$[q_0, 0, q_0] \rightarrow 0\,[q_0, 0, q_F]\,[q_F, 0, q_0] \text{ ---->} (P_{18})$

$[q_0, 0, q_1] \to 0\ [q_0, 0, q_0]$ $[q_0, 0, q_1]$ ... (P11)

$[q_0, 0, q_1] \to 0\ [q_0, 0, q_1]$ $[q_1, 0, q_1]$ ... (P12)

$[q_0, 0, q_1] \to 0\ [q_0, 0, q_F]$ $[q_F, 0, q_1]$ ... (P13)

$[q_0, 0, q_F] \to 0\ [q_0, 0, q_0]$ $[q_0, 0, q_F]$ ... (P14)

$[q_0, 0, q_F] \to 0\ [q_0, 0, q_1]$ $[q_1, 0, q_F]$ ... (P15)

$[q_0, 0, q_F] \to 0\ [q_0, 0, q_F]$ $[q_F, 0, q_F]$ ... (P16)

The rule from P_1 to P_{14} are the requi production rule of CFG.

Thus a content free Grammar has been constructed successfully from the given PDA.

1) Sol:

Step 1:

The required CFG for given PDA is $G = \{V, T, P, S\}$

where,

$T = \{0, 1\}$

$V_n = \{S, [q_0, x, q_0], [q_0, x, p], [q_0, z_0, q_0], [q_0, z_0, p], [p, x, q_0], [p, x, p], [p, z_0, q_0], [p, z_0, p]\}$

Step 2:

$S \rightarrow$ production

To constructs $S \rightarrow$ productions

$S \rightarrow [q_0, z_0, q]$ For every q in Q

$S \rightarrow [q, z_0, q] \dashrightarrow (P_1)$

$S \rightarrow [q, z_0, p] \dashrightarrow (P_2)$

Step 3:

For Erasing moves.

For $\delta(q_1, a, z) = (q', \varepsilon)$

$\Rightarrow [q, a, q'] \rightarrow a$

For $\delta(q, \varepsilon, z_0) = (q_1, \varepsilon)$

$\Rightarrow [q, z_0, q_1] \longrightarrow \varepsilon \dashrightarrow (P_3)$

For $\delta(p, 1, x) = (p, \varepsilon)$

$\Rightarrow [p, x, p] \longrightarrow 1 \dashrightarrow (P_4)$

Step 4:

non - Erasing moves.

* $\delta(q, 1, z_0) = (q, x z_0)$

$[q, z_0, q] \rightarrow 1\,[q, x, q]\,[q, z_0, q] \dashrightarrow (P_5)$

$[q, z_0, q] \rightarrow 1\,[q, x, p]\,[p, z_0, q] \dashrightarrow (P_6)$

$[q, z_0, p] \rightarrow 1\,[q, x, q]\,[q, z_0, p] \dashrightarrow (P_7)$

$[q, z_0, p] \rightarrow 1\,[q, x, p]\,[p, z_0, p] \dashrightarrow (P_8)$

* $\delta(q, 1, x) = (q, xx)$

$[q, x, q] \rightarrow 1\ [q, x, q]\ [q, x, q]$ ---> (P9)

$[q, x, q] \rightarrow 1\ [q, x, p]\ [p, x, q]$ ---> (P10)

$[q, x, p] \rightarrow 1\ [q, x, q]\ [q, x, p]$ ---> (P11)

$[q, x, p] \rightarrow 1\ [q, x, p]\ [p, x, p]$ ---> (P12)

* $\delta(q, 0, x) = (p, x)$

$[q, x, q] \rightarrow 0\ [p, x, q]$ ---> (P13)

$[q, x, p] \rightarrow 0\ [p, x, p]$ ---> (P14)

* $\delta(q, 0, z_0) = (q, z_0)$

$[q, z_0, q] \rightarrow 0\ [q, z, q]$ ---> (P15)

$[q, z_0, p] \rightarrow 0\ [q, z_0, p]$ ---> (P16)

The Rules from P_1 to P_{16} are they required production rules of CFG.

Does the context free gramm has been constructed successfull from the given PDA.

(P5)

(P6)

(P7)

(P8)

UNIT-IV

Properties of context free Languages

* If a Language is CFL then it should have a grammar in sum specific form. To find this as CFG can be simplified & this mechanism is called as normal forms.

  * CNF (Chomsky Normal Form)
  * GNF (Greibach Normal Form)

CNF:

A CFL is said to be in CNF if the language genuated by CFG has the production of the form $A \to BC$

$A \to a$

where A, B, C → Variables

a → Terminal

steps to convert as CFG into CNF.

Step 1: Eliminate useless Symbols or Terminals which to not appear in any derivation of at Termi String from a Start Symbol.

Step 2: Eliminate ε - productions. these production will be of th Form $A \to \varepsilon$

Step 3: Eliminate unit production which of the form $A \to B$

Eliminate useless productions:

1. $S \to AB / a$

   $A \to b$

   $S \to AB$

   $S \to a$

   $A \to b$

Since no string can be derived From 'B' then we can eliminate 'B' by removing its production rule

$\therefore S \to AB$ can be eliminated

Then we have $S \to a$

$A \to b$

Here there is no variable A in S therefore production A also useless & it can be eliminated.

∴ They Resultant grammar will be $\boxed{S \to a}$ after the elimi-nations of useless Symbol.

Eliminate ε - productions:

A production which of the form $A \to \varepsilon$ are called as ε - production. They elimination of ε - production should not effect the CFG.

Ex: Consider $S \to AB$

$A \to aAA / \varepsilon$

$B \to bBB / \varepsilon$

Step 1: Find null production

$A \to \varepsilon$

$B \to \varepsilon$

Step 2: Apply this null productions to the variables and find all possible production of Grammar.

A in
)
ted.
mai
ini

f the

ination
effect

ductions
d find
tion of

For $S \to AB$:

$S \to AB$ (without applying)

$S \to B$ (apply $A \to \varepsilon$)

$S \to A$ (apply $B \to \varepsilon$)

For $A \to aAA$:

$A \to aAA$ (without applying)

$A \to aA$ (apply $A \to \varepsilon$ to First A)

$A \to aA$ (apply $A \to \varepsilon$ to Second A)

$A \to a$ (apply $a \to \varepsilon$ to all A)

For $B \to bBB$:

$B \to bBB$ (without applying)

$B \to bB$ (apply $B \to \varepsilon$ to First B)

$B \to bB$ (apply $B \to \varepsilon$ to Second B)

$B \to b$ (apply $b \to \varepsilon$ to all B)

∴ The Required grammar will b

$S \to AB / A / B$

$A \to aAA / aA / a$

$B \to bBB / bB / b$

∴ It is the resultant gramma
after Eliminating ε-production.

Eliminate unit production:

A unit production is of the form $\boxed{A \to B}$ where both A & B are variables.

Ex: consider the Grammar

$E \to T / E + T$

$T \to F / T * F$

$F \to id$ Eliminate unit production.

step 1: Find the unit production

$E \to T$

$T \to F$

step 2: Replace them by applying their production rules untill the Terminals is Rich.

$E \to T$

$E \Rightarrow T \Rightarrow F \Rightarrow Id$

$\boxed{E \to Id}$

$T \to F$

$T \Rightarrow F \Rightarrow id$

$T \rightarrow id$

$E \rightarrow id \mid E + T$

$T \rightarrow id \mid T * F$

$F \rightarrow id$

1) consider the grammar $G = \{(S, A, B), (a, b), P, S\}$ the production

$S \rightarrow bA / aB$

$A \rightarrow bAA / aS / a$

$B \rightarrow aBB / bS / b$

sol:

Step 1: Check the given gramma For useless symbol there is no useless symbol in this Gramma

Step 2: Check for ε-production They is no ε-production In the grammar.

Step 3: Check unit production There is no unit production in this grammar.

Step 4: Find the production which are in already in CNF.

$A \rightarrow a$

$B \rightarrow b$

This production are in already in CNF.

Step 5: Replace the Terminals on RHS.

Take $S \rightarrow bA$

Introduce new variable

$S \rightarrow C_1 A$ (C_1 - new variable)

$\boxed{C_1 \rightarrow b}$

for $S \rightarrow aB$

$S \rightarrow C_2 B$ (C_2 - new variable)

$C_2 \rightarrow a$

for $A \rightarrow bAA$

$A \rightarrow C_1 AA$ ($\because C_1 \rightarrow b$)

$C_3 \rightarrow AA$ (C_3 - new variable)

$A \rightarrow C_1 C_3$

for $A \rightarrow aS$

$\boxed{A \rightarrow C_2 S}$

For $B \rightarrow aBB$

$C_4 \rightarrow BB$ (new variable)

For $B \rightarrow C_2 C_4$

$B \rightarrow bS$

$B \rightarrow C_1 S$

The resultant production in CNF.

$S \rightarrow C_1 A \mid C_2 B$

$A \rightarrow C_1 C_3 \mid C_2 S \mid a$

$B \rightarrow C_2 C_4 \mid C_1 S \mid b$

$C_1 \rightarrow b$

$C_2 \rightarrow a$

$C_3 \rightarrow AA$

$C_4 \rightarrow BB$

∴ The resultant CNF.

2) Convert the following grammar into CNF.

$S \rightarrow AACD$

$A \rightarrow aAb / \varepsilon$

$C \rightarrow aC / a$

$D \rightarrow aDa / bDb / \varepsilon$

step1: Check the given grammar for useless symbol there is no useless symbol in this Grammar

step2: check for ε-production

$A \rightarrow \varepsilon$

$D \rightarrow \varepsilon$

$S \to AACD$

$S \to ACD$ $(A \to \varepsilon)$

$S \to ACD$ $(A \to \varepsilon)$

$S \to AAC$ $(D \to \varepsilon)$

$S \to AC$ $(A \to \varepsilon \ \& \ D \to \varepsilon)$

$S \to AC$ $(A \to \varepsilon \ \& \ D \to \varepsilon)$

$S \to C$ $(A \to \varepsilon, A \to \varepsilon, D \to \varepsilon)$

$S \to CD$ $(A \to \varepsilon, A \to \varepsilon)$

$A \to aAb$

$A \to ab$ $(A \to \varepsilon)$

$D \to aDa$

$D \to aa$ $(D \to \varepsilon)$

$D \to bDb$

$D \to bb$ $(D \to \varepsilon)$

Step 3: Eliminate Unit production

$S \to C$

$S \Rightarrow C \Rightarrow a$

$\boxed{S \to a}$

The resultant production in CNF

$S \to AACD / ACD / AAC / AC / CD / a$

$A \to aAb / ab$

$C \to aC / a$

$D \to aDa / aa / bDa / bb$

Step 4: The production which are CNF

$S \rightarrow a$

$C \rightarrow a$

Step 5: Replace the Terminal on R.H.S

For $S \rightarrow AACD$

$C_1 \rightarrow AA$

$C_2 \rightarrow CD$

$S \rightarrow C_1 C_2$

for $S \rightarrow ACD$

$S \rightarrow AC_2$

$C_2 \rightarrow CD$

For $S \rightarrow AAC$

$S \rightarrow C_1 C$

$C_1 \rightarrow AA$

For $A \rightarrow aAb$

$C_3 \rightarrow a$

$C_4 \rightarrow b$

For $A \rightarrow C_3 A C_4$

$C_5 \rightarrow C_3 A$

$A \rightarrow C_5 C_4$

for $A \rightarrow ab$

$C_3 \rightarrow a$

$C_4 \rightarrow b$

$A \rightarrow C_3 C_4$

$C \rightarrow aC$

$C_3 \rightarrow a$

$C \rightarrow C_3 C$

For $D \rightarrow aDa$

$C_3 \rightarrow a$

$D \rightarrow C_3 D C_3$

$C_6 \rightarrow C_3 D$

$D \rightarrow C_6 C_3$

$D \rightarrow aa$

$D \rightarrow C_3 C_3$

$D \rightarrow bDb$

$D \rightarrow C_4 D C_4$

$C_7 \rightarrow C_4 D$

$D \rightarrow C_7 C_4$

$S \rightarrow C_1 C_2 \mid AC_2 \mid C_i C \mid A'C \mid cD \mid a$

$A \rightarrow C_5 C_4 \mid C_3 C_4$

$C \rightarrow C_3 C \mid a$

$D \rightarrow C_6 C_3 \mid C_3 C_3 \mid C_7 C_4 \mid C_4 C_4$

$C_1 \rightarrow AA$

$C_2 \rightarrow CD$

$C_3 \rightarrow a$

$C_4 \rightarrow b$

$C_5 \rightarrow C_3 A$

$C_6 \rightarrow C_3 D$ ∴ The resultant CNF

$C_7 \rightarrow C_4 D$

Greibach Normal form: (GNF)

Every context free language without ε can be generated by a grammar for which every production rules is of the form $\boxed{A \rightarrow a\alpha}$ where 'A' is variable 'a' is Terminal 'α' string of variables.

Step 1:
convert the variables into $A_1, A_2, A_3 \ldots$ based on their occurance into the production

step 2: If $A_i \rightarrow A_j$ & where $i > j$ then substitude the productions of A_j such that either $i = j$ (or) $i < j$

Step3: If $i < j$ then go to Terminal Replacement (step 6)

Step4: If $i = j$ the Introduce an new variable Bi such that $Bi \rightarrow \gamma$, $Bi \rightarrow \gamma Bi$ and remove the production $Ai \rightarrow Aj\gamma$

Step5: For Each $Ai \rightarrow B$ where B does not Stat with Aj then add the production rules $Ai \rightarrow \beta Bi$

Step6: Terminal Replacement after converting all the production rules to the formad of $Ai \rightarrow Aj$ (with $i < j$) we will have only the productions of the form

i) $Ai \rightarrow Aj\gamma$ (with $i < j$)

ii) $Ai \rightarrow aAj$

iii) $Bi \rightarrow \gamma$

Step7: Now replace the first element of RHS untill its the Terminal.

step8: They Final set of production are the required GNF.

1) $S \rightarrow AB$

$A \rightarrow BS / b$

$B \rightarrow SA / a$

convert it into GNF.

Sol:-

Step 1: Since the given productio are not in the standard formar we have to convert them by renaming it.

∴ They production Rule will become.

$A_1 \rightarrow A_2 A_3$ ----> (1)

$A_2 \rightarrow A_3 A_1 / b$ ----> (2)

$A_3 \rightarrow A_1 A_2 / a$ ----> (3)

Step 2: The First two production Rules have $i < j$ the remaining one is $A_3 \rightarrow A_1 A_2 / a$

Sub (1) in (2)

$A_3 \rightarrow A_2 A_3 A_2 / a$ ——> (4)

Sub (2) in (4)

$A_3 \rightarrow A_3 A_1 A_3 A_2 / b A_3 A_2 / a$ ——> (5)

(A_3 = A_i, A_3 = A_j, $A_1 A_3 A_2$ = γ)

Step 3: Here i = j then we can introduce new variable $B_i B_3$

$$B_i \rightarrow \gamma$$

$$B_i \rightarrow \gamma B_i$$

$$B_3 \rightarrow A_1 A_3 A_2$$

$$B_3 \rightarrow A_1 A_3 A_2 B_3$$

Step 4: For the production rules that does not have A_j

$$A_i \rightarrow \beta$$

$$A_i \rightarrow \beta B_1$$

$$A_3 \rightarrow b A_3 A_2$$

$$A_3 \rightarrow b A_3 A_2 B_3$$

$$A_3 \rightarrow a$$

$$A_3 \rightarrow a B_3$$

$A_1 \rightarrow A_2 / A_3$

$A_2 \rightarrow A_3 A_1 / b$

$A_3 \rightarrow$

$$A_1 \rightarrow A_2 A_3$$

$$A_2 \rightarrow A_3 A_1 / b$$

$$A_3 \rightarrow b A_3 A_2 / a / b A_3 A_2 B_3 / a B_3$$

$$B_3 \rightarrow A_1 A_3 A_2 / A_1 A_3 A_2 B_3$$

already in GNF

$$A_2 \rightarrow b$$

$$A_3 \rightarrow b A_3 A_2 / a / b A_3 A_2 B_3 / a B_3$$

Take $A_1 \rightarrow A_2 A_3$

$\Rightarrow A_3 A_1 A_3 / bA_3$

$A_1 \rightarrow A_3 A_1 A_3$

$A_1 \rightarrow bA_3A_2A_1A_3 / aA_1A_3 / bA_3A_2B_3A_1A_3 / aB_2A_3A_2$

$A_2 \rightarrow A_3 A_1$

$A_2 \rightarrow bA_3A_3A_1 / aA_1 / bA_3A_2B_3A_1 / aB_3A_1$

$B_3 \rightarrow A_1 A_3 A_2$

$B_3 \rightarrow bA_3A_3A_2 / bA_3A_2A_1A_3A_3A_2 / aA_1A_3A_3A_2 / bA_3A_2B_3A_1A_3A_3A_2 / aB_3A_1A_3A_3A_2$

$B_3 \rightarrow A_1A_3A_2B_3$

~~$B_3 \rightarrow bA_3A_2A_2A_3B_3 / bA_3A_2A_1A_3A_2$~~

$B_3 \rightarrow bA_3A_2A_2A_3B_3 / bA_3A_2A_1A_3A_2 / B_3 / aA_1A_3A_2A_2B_3 / bA_3A_2B_3A_1A_3A_2A_2B_3 / aB_3A_1A_3A_2A_2B_3$

∴ The resultant GNF are

A_1

$A_1 \rightarrow bA_3$

$A_1 \rightarrow bA_3A_2A_1A_3$

$A_1 \rightarrow aA_1A_3$

$A_1 \rightarrow bA_3A_2B_3A_1A_3$

$A_1 \rightarrow aB_3A_1A_3$

S

$S \rightarrow bB$

$S \rightarrow aSB$

$S \rightarrow bBAB_3SB$

$S \rightarrow aB_3SB$

A_2

$A_2 \to b$

$A_2 \to bA_3A_2A_1$

$A_2 \to aA_1$

$A_2 \to bA_3A_2B_3A_1$

A_2 / aB_2A_1

A

$A \to b$

$A \to bBA_3S$

$A \to aS$

$A \to bB \cdot AB_3S$

$A \to aB_2S$

A_3

$A_3 \to bA_3A_2$

$A_3 \to a$

$A_3 \to bA_3A_2B_3$

$A_3 \to aB_3$

B

$B \to bB \cdot A$

$B \to a$

$B \to bBAB_3$

$B \to aB_3$

B_3

$B_3 \to bA_3A_3A_2$

$B_3 \to bA_3A_2A_1A_3A_3A_2$

$B_3 \to aA_1A_3A_3A_2$

$B_3 \to bA_3A_2B_3A_1A_3A_3A_2$

$B_3 \to aB_3A_1A_3A_3A_2$

$B_3 \to bA_3A_2A_2A_3B_3$

$B_3 \to bA_3A_2A_1A_3A_2A_2B_3$

$B_3 \to aA_1A_3A_2A_2B_3$

$B_3 \to bA_3A_2B_3A_1A_3A_2A_3B_3$

$B_3 \to aB_3A_1A_3A_2A_2B_3$

$B_3 \to bBBA$

$B_3 \to bBASBBA$

$B_3 \to aSBBA$

$B_3 \to bBAB_3SBBA$

$B_3 \to aB_3SBBA$

$B_3 \to bBAABB_3$

$B_3 \to bBASBAAB_3$

$B_3 \to aSBAAB_3$

$B_3 \to bBAB_3SBA$
BB_3

$B_3 \to aB_3SBAA$
B_3

H.W

1) $S \rightarrow AA / a$

$A \rightarrow SS / b$

convert into GNF

2) $S \rightarrow A$

$A \rightarrow aaA / B$

$B \rightarrow bAb$

1) Ans:

step 1: Since the given product are not in the standard format we have to convert them by rename it.

∴ They production Rule will become.

$A_1 \rightarrow A_2 A_2 / a \longrightarrow ①$

$A_2 \rightarrow A_1 A_1 / b \longrightarrow ②$

step 2: The first 1 production rule is $i<j$ the remaining one is

$A_2 \rightarrow A_1 A_1 / b$

Sub ① in ②

$A_2 \rightarrow A_2 A_2 A_1 / b / a \longrightarrow ③$

(A_2 = A_i, A_2 = A_j, A_1 = 8)

Step 3: Here i=j then we introduce a new variable B_2.

$B_i \rightarrow \gamma$

$B_i \rightarrow \gamma B_i$

$B_2 \rightarrow A_2 A_4$

$B_2 \rightarrow A_2 A_1 B_2$

Step 4: For the production rules that does not have A_j

$A_i \rightarrow \beta$

$A_i \rightarrow \beta B_i$

$A_4 \rightarrow b$

$A_4 \rightarrow bB_2$

$A_2 \rightarrow a$

$A_2 \rightarrow aB_2$

$A_1 \rightarrow A_2 A_2 / a$

$A_2 \rightarrow b / bB_2 / aA_1 / aB_2$

$B_2 \rightarrow A_2 A_1 / A_2 A_1 B_2$

Already in GNF

$A_1 \rightarrow a$

$A_2 \rightarrow b / bB_2 / bA_1 / aB_2$

Take

$A_1 \rightarrow A_2 A_2 / a$

$A_1 \rightarrow bA_2 / bB_2 A_2 / aA_1 A_2 / aB_2 A_1 A_2$

$B_2 \rightarrow A_2 A_1$

$B_2 \rightarrow bA_1 / bB_2 A_1 / aA_1 A_1 / aB_2 A_1$

$B_2 \rightarrow bA_2 A_1 B_2$

$B_2 \rightarrow bA_1 B_2 / bB_2 A_1 B_2 / aA_1 A_1 A_2 / aB_2 A_1 A_2$

new

∴ The resultant GNF are

A_1

$A_1 \rightarrow bA_2$	$S \rightarrow bA$
$A_1 \rightarrow bB_2A_2$	$S \rightarrow bB_2A$
$A_1 \rightarrow aA_1A_2$	$S \rightarrow aSA$
$A_1 \rightarrow aB_2A_2$	$S \rightarrow aB_2A$
$A_1 \rightarrow a$	$S \rightarrow a$

: does

A_2

$A_2 \rightarrow b$	$A \rightarrow b$
$A_2 \rightarrow bB_2$	$A \rightarrow bB_2$
$A_2 \rightarrow a$	$A \rightarrow a$
$A_2 \rightarrow aA_1B_2$	$A \rightarrow aSB_2$

B_2

$B_2 \rightarrow bA_1$	$B_2 \rightarrow bS$
$B_2 \rightarrow bB_2A_1$	$B_2 \rightarrow bB_2S$
$B_2 \rightarrow aA_1A_1$	$B_2 \rightarrow aS$
$B_2 \rightarrow aA_1B_2A_1$	$B_2 \rightarrow aB_2S$
$B_2 \rightarrow bA_1B_2$	$B_2 \rightarrow bSB_2$
$B_2 \rightarrow bB_2A_1B_2$	$B_2 \rightarrow bB_2SB_2$
$B_2 \rightarrow aA_1A_2$	$B_2 \rightarrow aSSA$
$B_2 \rightarrow aA_1B_2A_1B_2$	$B_2 \rightarrow aSB_2SB_2$

A_1 A_2

2) $S \rightarrow A$

$A \rightarrow aaA / B$

$B \rightarrow bAb$

Sol:

Step 1:

Since the given productions also not in the standard format we have convert them by Remaining it. ~~Hence~~ Therefore the productions Rules will become.

$A_1 \rightarrow A_2$ ----> ①

$A_2 \rightarrow aaA_2 / A$ ----> ②

$A_3 \rightarrow bA_2b$ ----> ③

Step 2:

Already in GNF

$A_2 \rightarrow aaA_2$

$A_3 \rightarrow bA_2b$

$A_1 \rightarrow A_2$

$A_1 \rightarrow aaA_2$

$A_1 \rightarrow A_3$

$A_1 \rightarrow bA_2b$

$A_2 \rightarrow A_3$

$A_2 \rightarrow bA_2b$

The Resultant GNF are

A_1:

$A_1 \rightarrow aaA_2$

$A_1 \rightarrow bA_2b$

S:

$S \rightarrow aaA$

$S \rightarrow bAb$

A_2:

$A_2 \rightarrow aaA_2$

$A_2 \rightarrow bA_2b$

A:

$A \rightarrow aaA$

$A \rightarrow bAb$

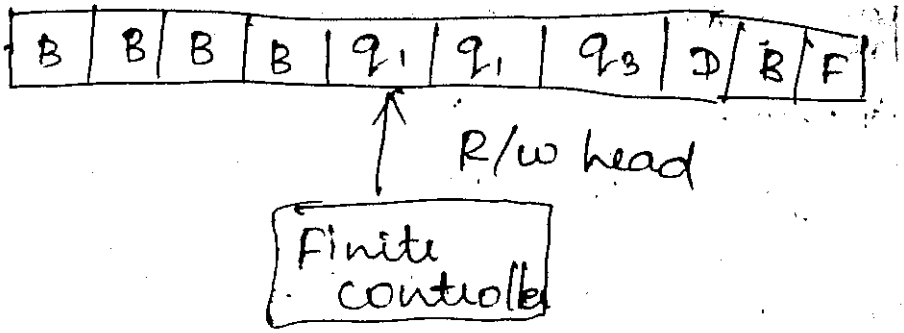
A_3:

$A_3 \rightarrow bA_2b$

B:

$B \rightarrow bAb$

Turing machines :-



* They Turing machines is an Finite automata with an unlimit and understand memory.

* It is made up three components,

(i) Read write head

ii) Input Tape

iii) Finite controller.

* They Input Tape is divided into cells each cell can hold one of an Finite no. of Tape Symbols.

* All other cells all extended infinitly to the Left & right and hold as special called blank (B).

* Initially the Tape head is finding the left most cells that holds their input.

* The Turing machines Transitions is depending on three elements,

  * State of the finite control
  * Tape symbol
  * ~~distance of~~ Direction of head.

* The Turing machines is a 7 Tuple automaton defined by M,

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

$Q \rightarrow$ Finite set of states

$\Sigma \rightarrow$ Finite set of i/p symbols

$\vdash \rightarrow$ Finite Set Tape symbols

$\delta \rightarrow$ Transition function (mapping the states of FA and states Symbol to resultant states, Tape symbol and moment of the head).

$q_0 \rightarrow$ Initial State

$B \rightarrow$ blanks to $\in$

finding
s their

nsitions
uts,

ead.
7
M,

:
ls
mapping
states
states,
ument

F → set of final states

problem:

1) Design a Turing machines to accept a language $L = \{0^n 1^n / n \geq 1\}$

Sol:

The Turing machines for the Language has 'n' no. of 0 zeros Followed by 'n' no. of ones.

step1:

'M' replaces the left most zero by 'x' and move towards right to the Left most 1 & replacing it by 'Y'.

step2:

Then M moves towards left to find the right most x or moves 'M' one cell right.

step2.1:

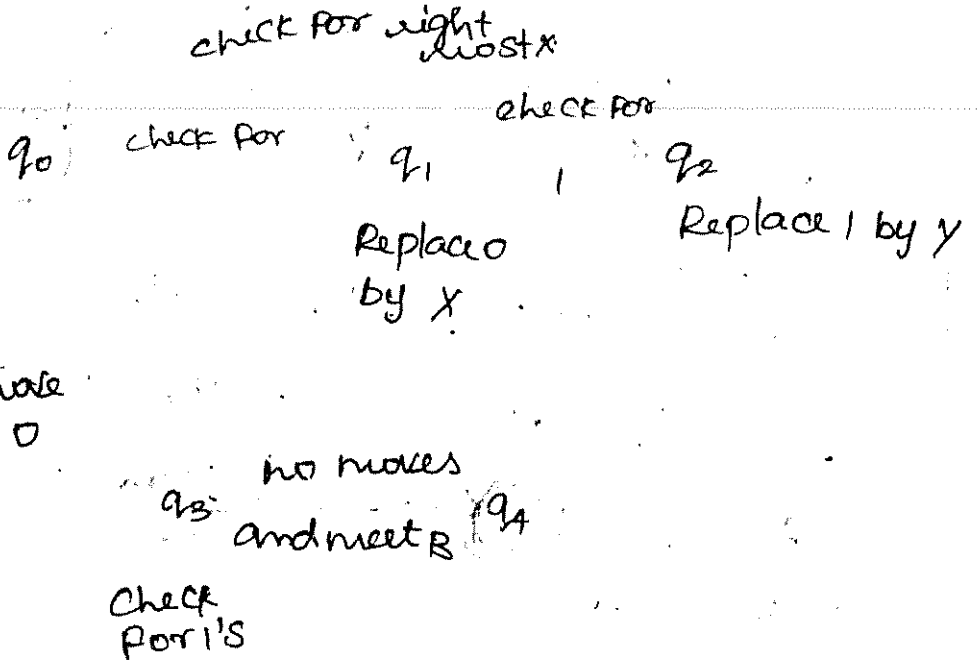
If its a '0' then step(1). repeated.

Step 2.2:

If its 'y' then go for step(3).

Step3:

It search for '1' in the remaining cells towards right. If '1' is found then string must be Rejected else M can reach the Final state upon reading 'B' and accept the string.



The required Turing machines for the given languages is

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

$Q = (q_0, q_1, q_2, q_3, q_4)$

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

tep(s).

ght.
must be
the
and

1 by y

es

$\Gamma = \{0, 1, X, Y, B\}$

$\delta q_0 = \{q_0\}$

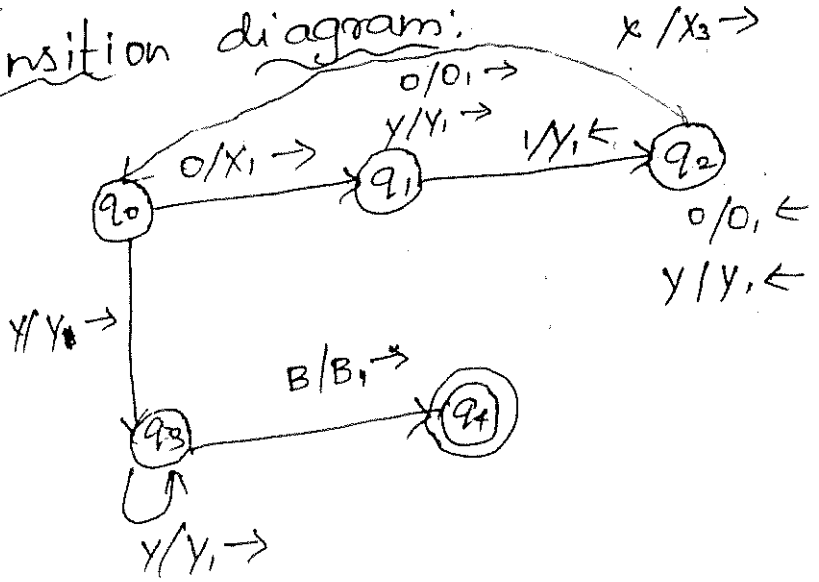
$B = \{B\}$

$F = \{q_4\}$

and δ is given by,

	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	$(q_4, B$
q_4	–	–	–	–	–

Transition diagram:



Let w = 0011

$\vdash \boxed{q_0}\ 0011\ B$

$\vdash X \boxed{q_1}\ 011\ B$

$\vdash X0 \boxed{q_1}\ 11\ B$

$\vdash X0 \boxed{q_2}\ Y1\ B$

$\vdash X \boxed{q_2}\ 0Y1\ B$

$\vdash \boxed{q_0}\ X0Y1\ B$

$\vdash X \boxed{q_1}\ 0Y1\ B$

$\vdash XX \boxed{q_1}\ Y1\ B$

$\vdash XXY \boxed{q_1}\ 1\ B$

$\vdash XXY \boxed{q_2}\ Y\ B$

$\vdash XX \boxed{q_2}\ YY\ B$

$\vdash X \boxed{q_0}\ XYY\ B$

$\vdash XX \boxed{q_3}\ YY\ B$

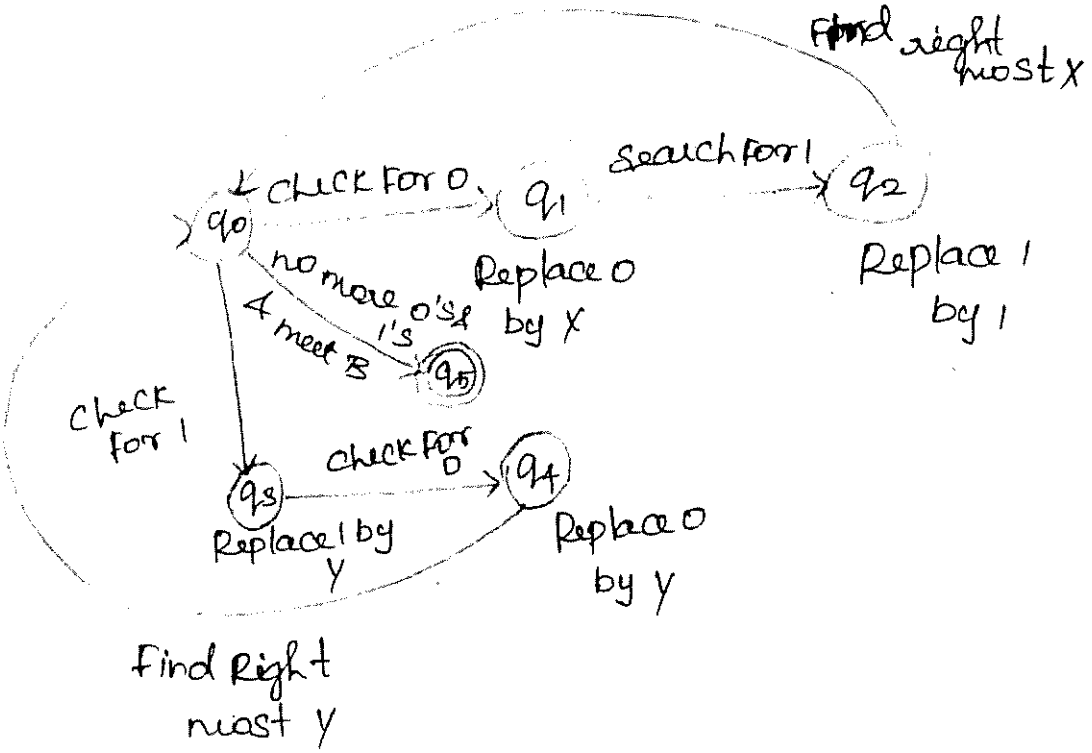
$\vdash XXY \boxed{q_3}\ Y\ B$

$\vdash XXYY \boxed{q_3}\ B$

$\vdash XXYY\ B\ \boxed{q_4}$

∴ Since the Final state is reach the given String is accepted.

2) Design the Turing machines for a languages L = { The set of all stri with equal no. of 0's & 1's }



The required Turing machine For the given languages is

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

$$Q = (q_0, q_1, q_2, q_3, q_4, q_5)$$

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

$$\vdash = \{0, 1, X, Y, B\}$$

$q_0 = \{q_0\}$

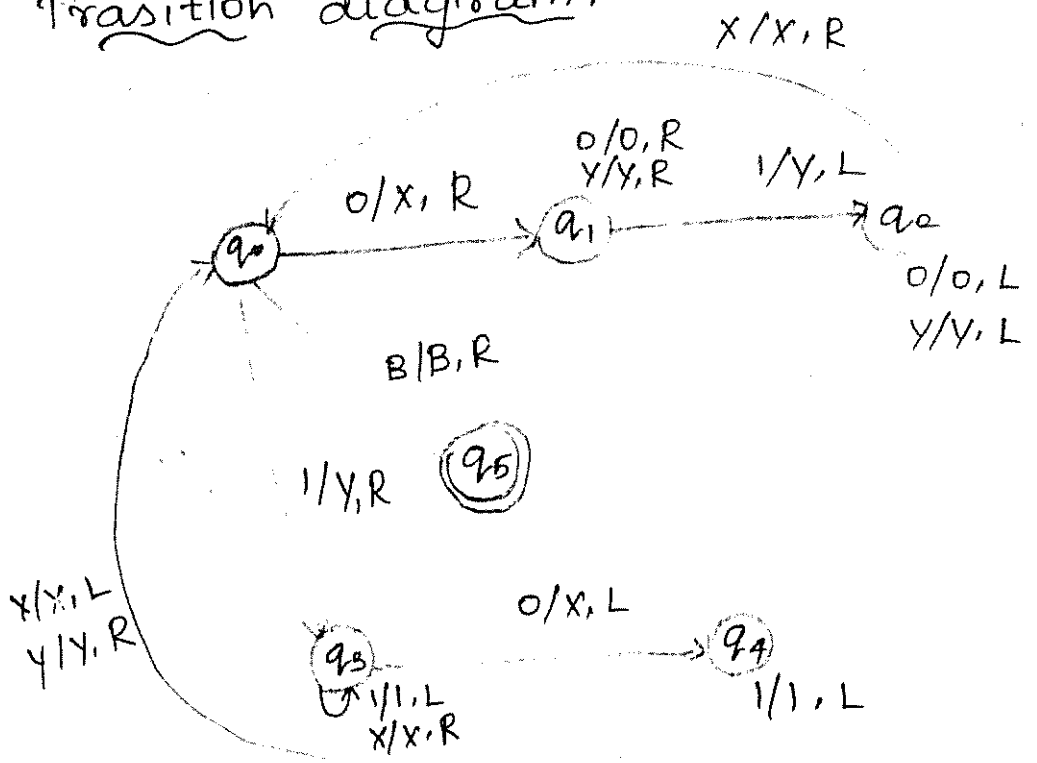
$B = \{B\}$

$F = \{q_5\}$

and δ is given by

	0	1	X	Y	B
q_0	(q_1, X, R)	(q_3, Y, R)	–	–	(q_5, B, R)
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	(q_4, X, L)	$(q_3, 1, R)$	(q_3, X, R)	–	–
q_4	–	$(q_4, 1, L)$	(q_0, X, L)	(q_0, Y, R)	–
q_5	–	–	–	–	–

Trasition diagram:



$w = 0101$

$\vdash \underset{\rightarrow}{[q_0]}\, 0101B$

$\vdash X \underset{\rightarrow}{[q_1]}\, 101B$

$\vdash XY \underset{\rightarrow}{[q_0]}\, 01B$

$\vdash X \underset{\leftarrow}{[q_0]}\, Y01B$

$\vdash X \underset{\rightarrow}{[q_0]}\, Y01B$

$\vdash XY \underset{\rightarrow}{[q_0]}\, 01B$

$\vdash XYX \underset{\rightarrow}{[q_1]}\, 1B$

$\vdash XYXY \underset{\leftarrow}{[q_2]}\, B$

$\vdash XYX \underset{\rightarrow}{[q_0]}\, YB$

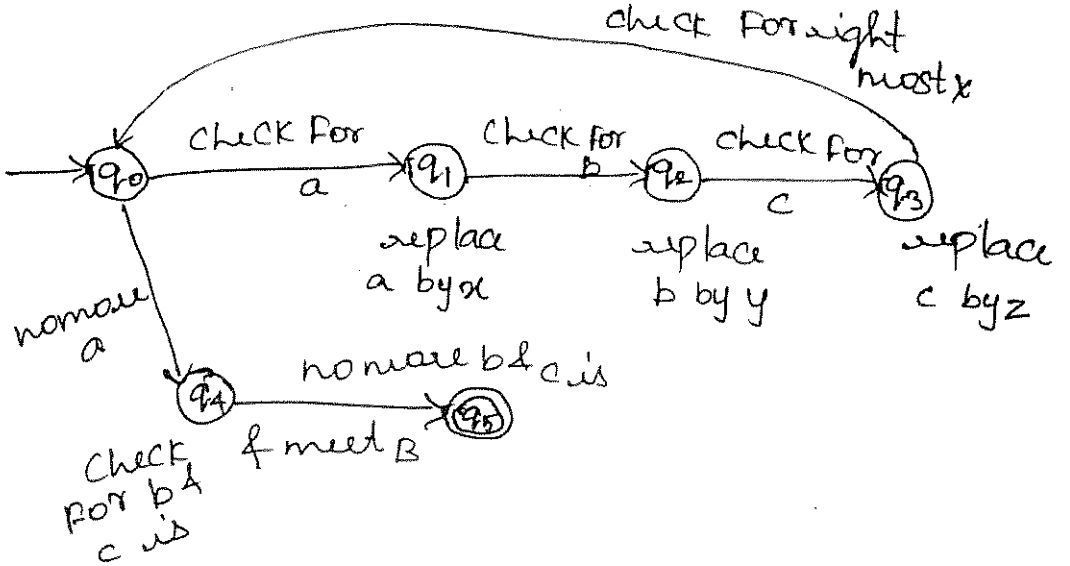
$\vdash XYXY \underset{\rightarrow}{[q_5]}\, B$

$\vdash XYXYB \underset{\rightarrow}{[q_5]}$

∴ Since we have reach the Final state.

∴ The given string is accepted.

2) $L = \{a^n b^n c^n / n \geq 1\}$



The required Turing machine for the given language is

$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

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

$\Gamma = \{a, b, c, X, Y, Z, B\}$

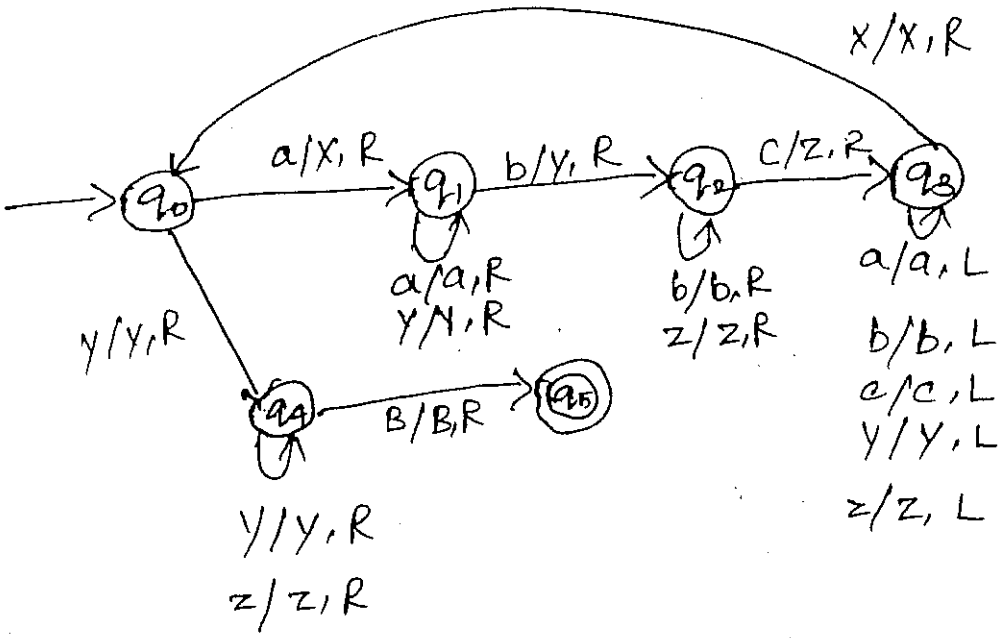
$q_0 = \{q_0\}$

$B = \{B\}$

$F = \{q_5\}$

and δ is given by

	a	b	c	X	Y	Z	
q_0	(q_1, X, R)	-	-	-	(q_4, Y, R)	-	-
q_1	(q_1, a, R)	(q_2, Y, R)	-	-	(q_1, Y, R)	-	-
q_2	-	(q_2, b, R)	(q_3, z, R)	-	-	(q_2, z, R)	-
q_3	(q_3, a, L)	(q_3, b, L)	(q_3, c, L)	(q_0, X, R)	(q_3, X, L)	$(q_3,$	
q_4	-	-	-	-	(q_4, Y, R)	(q_4, z, R)	$(q_5, B$
q_5	-	-	-	-	-	-	-



$w = aabbcc$

$\vdash \underset{\rightarrow}{[q_0]}\ aabbcc\,B$

$\vdash x\underset{\rightarrow}{[q_1]}\ abbcc\,B$

$\vdash xa\underset{\rightarrow}{[q_1]}\ bbcc\,B$

$\vdash xay\underset{\rightarrow}{[q_2]}\ bcc\,B$

$\vdash xayb\underset{\rightarrow}{[q_2]}\ cc\,B$

$\vdash xayb\underset{\leftarrow}{[q_3]}\ z\,cB$

$\vdash xayb\underset{\leftarrow}{[q_3]}\ z\,cB$

$\vdash xay\underset{\leftarrow}{[q_3]}\ bz\,cB$

$\vdash xa\underset{\leftarrow}{[q_3]}\ yb\,z\,cB$

$\vdash x\underset{\rightarrow}{[q_0]}\ ayb\,z\,cB$

$\vdash xx\underset{\rightarrow}{[q_1]}\ yb\,z\,cB$

$\vdash xxy\underset{\rightarrow}{[q_1]}\ bz\,cB$

$\vdash xxyy\underset{\rightarrow}{[q_2]}\ z\,cB$

$\vdash xxyyz\underset{\rightarrow}{[q_2]}\ cB$

$\vdash xxyyzz\underset{\leftarrow}{[q_3]}\ B$

$\vdash X\,x\,y\,y\,z\,\boxed{q_3}\,z\,B$ ←

$\vdash X\,x\,y\,y\,\boxed{q_3}\,z\,z\,B$ ←

$\vdash X\,x\,y\,\boxed{q_3}\,y\,z\,z\,B$ ←

$\vdash X\,x\,\boxed{q_0}\,y\,y\,z\,z\,B$ →

$\vdash X\,x\,\boxed{q_4}\,y\,y\,z\,z\,B$ →

$\vdash X\,x\,y\,\boxed{q_4}\,y\,z\,z\,B$ →

$\vdash X\,x\,y\,y\,\boxed{q_4}\,z\,z\,B$ →

$\vdash X\,x\,y\,y\,z\,\boxed{q_4}\,z\,B$ →

$\vdash X\,x\,y\,y\,z\,z\,\boxed{q_4}\,B$ →

$\vdash X\,x\,y\,y\,z\,z\,B\,\boxed{q_5}$ →

Since we have reach the Final State.

∴ The given String is accepted.

Programming Techniques for TM construction:

*A Turing machines is powerful as a conventional computer for constructing a Turing machines are following Techniques are used.

(i) storage in the Finite control (or) state.

(ii) Multiple Tracks

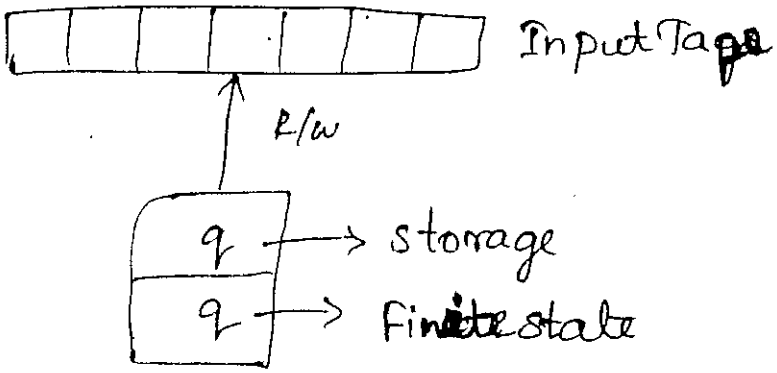
(iii) Subroutine

storage in the finite control (or) state:-

*The Finite control also be used hold the finite amount of Information along the Tracks are Representing the position of TM in the program.

*The Finite control will have a separated memory inside it two hold the Information for limited period of Time.

* The state of TM is written as a pair of elements one for state control and another one for storing a symbol.



Multiple Tracks:

* It is also possible that Turing machine Input Tags can be divided into multiple level of Tracks.

* Each Tracks will have an Infinite amount of cells.

...	B	B	0	0	1	1	B	...
...	B	B	1	0	0	1	B	...
...	B	B	1	0	0	0	B	...

Subroutine:

* When as same Task is to be Repeated for many no. of Times the Turing machine can be constructed with Subroutines. It is a set of

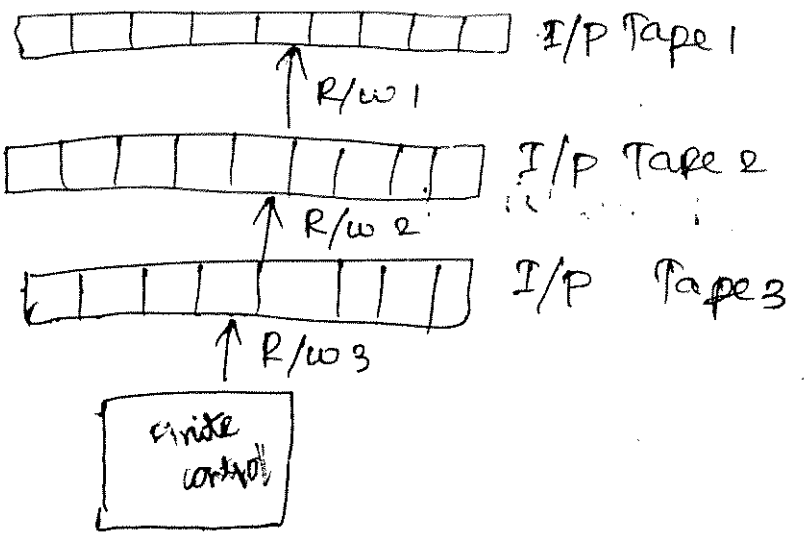
Programming codes that can repeated for different Set of Inputs.

* The Subroutines in a TM is the Set of States that can perform Some usefull process.

* The idea here its to write Some part of Turing machine program that will have its own Initial State & Final accepting State Returning the accepts to the Calling routines.

* TM uses Top down program design for the constructions of Subroutines.

MultiTape Turing machines:



*A Multitape Turing machines will have Finite control with some Finite no. of i/p Tapes.

Each Tape is Infinite in both the direction it has its own Initial State & accepting states.

*The Multi Tape Turing machine Can process the Following one move

⇒ It can Change the state

⇒ It can print new symbol on each of the cells ~~by~~ can by its R/W head.

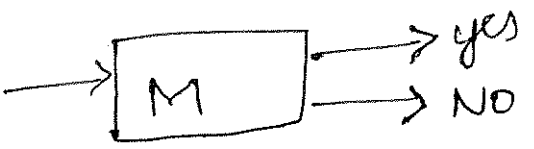
⇒ The Finite control can move each its Tape heads Independently or every Tape Independently.

UNIT- V

Undecidability

Recursive language:

A language is Recursive If there exist a Turing machine. That accepts every String of the Language And rejects that are not a language.



So, the problem whose language is . A problem is set be undecitable If there is no algorithm that Takes as i/p an Instance of the problem & determine whether the result to the Instance either yes (or) no.

Codes for Turing Machine:-

Step 1:

To represent the Turing Machine $M = \{Q, \{0,1\}, \vdash, \delta, q, B, F\}$ as binary string we have to assign Integer to the state & tape symbol & direction.

Step 2:

Assume the states as $Q_1, Q_2 \ldots Q_K$ for some 'K' using Integers in the suffix of each state the string can be represented.

Step3:
Assume the Tape symbol (0, 1, B) as can be represented as X_1, X_2, X_3

Step4:
Assume the direction as D_1 & D_2 For Left & right direction.

Step5:
After representing each stat Tape symbol, direction using Inte the production rule can be represented as follows.

$\delta(q_i, X_j) = (q_K, X_L, D_M)$ the program code

$$C = 0^i 1\, 0^j 1\, 0^k 1 0^l 1\, 0^m$$

$$\delta(q_1, 1) = (q_3, 0, R)$$

$$\delta(q_1, X_2) = (q_3, X_1, D_2)$$

$$C = 0^1 1\, 0^2 1\, 0^3 1\, 0^1 1 0^2$$

$$= 0100100010100$$

Step6:
They code For entire Turing Machine 'M' consits of all the

Strings for all the Transitions can be written as $M = \langle \underline{C_1} 11 C_2 11 C_3 11 \ldots \rangle$

↳ code for each Transition

Problem:

1) Obtain the code for $\langle M, 1011 \rangle$

$M = (\{q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_1, B, \{q_2\})$ and δ is given by

$\delta(q_1, 1) = (q_3, 0, R)$

$\delta(q_3, 0) = (q_1, 1, R)$

$\delta(q_3, 1) = (q_2, 0, R)$

$\delta(q_3, B) = (q_3, 1, L)$

Sol:-

Step 1: The states can be represented as ~~$M = (\{Q_1, Q_2, Q_3\})$~~

$q_1 = Q_1$

$q_2 = Q_2$

$q_3 = Q_3$

Step 2: The Tape Symbol can be represented as $0 = X_1$, $1 = X_2$, $B = X_3$

Step 3:

The directions are

$L = D_1$

$R = D_2$

Step 4:

$$\delta(Q_1, X_2) = (Q_3, X_1, D_2)$$

$$= 0^1 1 0^2 1 0^3 1 0^1 1 0^2$$

$$C_1 = 0\,1\,00\,1\,000\,10\,100$$

$$\delta(Q_3, X_1) = (Q_1, X_2, D_2)$$

$$= 0^3 1 0^1 1 0^1 1 0^2 1 0^2$$

$$C_2 = 000\,10\,10\,100\,100$$

$$\delta(Q_3, X_2) = (Q_2, X_1, D_2)$$

$$= 0^3 1 0^2 1 0^2 1 0^1 1 0^2$$

$$C_3 = 000\,100\,100\,10\,100$$

$$\delta(Q_3, X_3) = (Q_3, X_2, D_1)$$

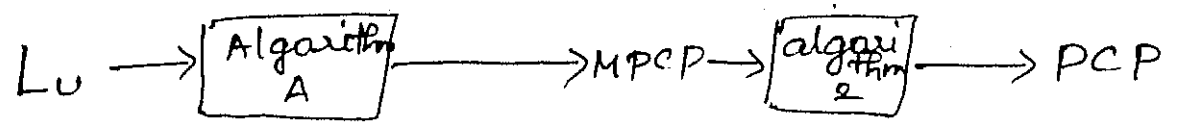
$$= 0^3 1 0^3 1 0^3 1 0^2 1 0^1$$

$$C_4 = 000\,1000\,1000\,10010$$

Step 5:

$M = <0100\,1000\,10100 \,||\, 00010101$
$00100 \,||\, 000\,1001001010 0 \,||\, 0001$
$000\,1000\,10010>$

post's correspondence problem (PCP):



List A = $w_1, w_2, w_3 \ldots w_k$

List B = $x_1, x_2 \ldots x_k$ For some Integer k

$w_1, w_2, w_3 \ldots w_k = x_1, x_2 x_3 \ldots x_k$

* The decidable problem above the Turing machine are reduce into undecidable problem about real Thing.

* Therefore the problem about the String can be proved undecidable.

PCP:

It is an algorithm that Reduces the given Turing machine Into Two list of strings over Σ.

A = $w_1, w_2, w_3 \ldots w_k$

B = $x_1, x_2 \ldots x_k$ For Some Integer 'k'

The Instance of PCP has a solution

PCP)

→ PCP

Integer

'k'

the

to

real

ut the

dable.

ine

Σ.

'Integer

'K'

olution

If there is some sequence of Integers $I_1, I_2 \ldots I_M$ such that

$W_{i_1}, W_{i_2}, W_{i_3} \ldots = X_{i_1}, X_{i_2}, X_{i_3} \ldots$

Is the solution for this Instance of PCP

Example: 1

Let $\Sigma = \{0, 1\}$, A & B be strings Find the inverse of PCP

	List A	List B
i	W_i	X_i
1	1	111
2	10111	10
3	10	0

Sol:

Instance of PCP

$i = 2, 1, 1, 3$

List A = 10111 1 1 10

List B = 10 1 1 1 1 0

Ex:2 Let $\Sigma = \{0, 1\}$. A & B be strings Find the Inverse of PCP

i	List A: wi	List B: xi
1	10	101
2	011	11
3	1011	011

Sol:

$i = 1, 3, 1$

A = 10101101101

B = 10101101101

∴ The given problem is no solution

∴ undecidable.

Modify PCP:

steps for constructing modified PCP From the given Turing machine.

Step 1:

Let $M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$ be a Turing machine and 'w' blanks to Σ^* be and I/P ~~Stream~~ string the Instance MPCP can be constructed as

Inqs A... follows.

Rule 1:

They first pair of MPCP is

ListA	ListB
#	# q0w#

q0 - Initial State
w - given I/P string

Rule 2:

The Tape symbols and the separate # has can be upended to both the list

Solution

ListA	ListB
X	X
#	#

has each X in Σ

id PCP

Rule 3:

For each q in Q for the Transitions Functions

, F }

blanks

ListA	ListB	Transition function
qX	YP	if (q, x) = (P, y, R)
zqX	PzY	if (q, x) = (P, y, L) z - I/P symbol
q#	YP#	if (q, B) = (P, y, R)
zq#	Pzy#	if (q, B) = (P, y, L)

tted as

Rule 4: For each q in F

List A	List B
x q y	q
x q	q
q y	q

Rule 5: For each q in F

List A	List B
q # #	#

To complete the solution we have to use all this rules to make List A & B equals.

problem:

1) convert the Turing machine M Given by

q_i	$\delta(q_i, 0)$	$\delta(q_i, 1)$	$\delta(q_i, B)$
$\rightarrow q_1$	$(q_2, 1, B)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_2	$(q_3, 0, L)$	$(q_1, 0, R)$	$(q_2, 0, R)$
$*q_3$	—	—	—

and the i/p string is Given by w = 01 Find the Solution.

Sol:-

Start with the First pair and Follow the rules one after other Therefore Instance of MPCP can be constructed as Follows.

Rule	List A	List B	Source
1	#	# $q_1$01#	AS q_1 - Initi state & w =
2	0 1 #	0 1 #	Have Σ → {0,1
3	$q_1$0 1q_2 0$q_2$0 1$q_2$0	$q_3$00 $q_3$10	δ(q_1, 0) = (q_2, 1, 1 δ(q_2, 0) = (q_3, 0,
4	$q_1$1 1$q_1$1 $q_2$1 $q_1$1#	$q_2$00 $q_2$10 0$q_1$1 $q_2$01#	δ(q_1, 1) = (q_2, 0 δ(q_2, 1) = (q_1, 0, 1 δ(q_1, B) = (q_2, 1, L
	1q_1#	$q_2$11#	

Rule	List A	List B	Some
	$q_2\#$	$0q_2\#$	$\delta(q_1, B) =$
4)	$0q_30$	q_3	Xq_4Y/q
	$0q_31$	q_3	Xq/q
	$1q_30$	q_3	qY/q
	$1q_31$	q_3	
	$0q_3$	q_3	
	$1q_3$	q_3	
	q_30	q_3	
	q_31	q_3	
5)	$q_3\#\#$	$\#$	

Next we are going to the Instance of MPCP by comparing List A & B equals.

The First pair

A: # (From Rule 1)

B: $\#q_101\#$

Step 2: (From Rule 3)

A: $\# q_10$

B: $\# q_101\#q_2$

Step 3: (From Rule 2)

A : #q101

B : #q101 #1q21

Step 4: (From Rule 2)

A : #q101#

B : #q101#1q21#

Step 5: (From Rule 2)

A : #q101#1

B : #q101#1q21#1

Step 6: (From Rule 3)

A : #q101#1q21

B : #q101#1q21#10q1

Step 7: (From rule 2)

A : #q101#1q21#

B : #q101#1q21#10q1

Step 8: (From Rule 2)

A : #q101#1q21#1

B : #q101#1q21#10q1#1

Step 9: (From Rule 3)

A : #q101 #1921 #1[illegible]091

B : #q101 #1921 #1091 #1921

Step 10: (From Rule 3)

A : #q101 #1921 #[illegible]1091

B : #q101 #1921 #1091 #19201#q3 10

Step 11: (From Rule 2)

A : #q101#1921 # 1091 #19201

B : #q101 #1921 #1091# 19201 #q3101

Step 12: (From Rule 2)

A : #q101 #1921 # 1091# 19201 #

B : #q101 #1921 # 1091 # 19201 #q3101 #.

Step 13: (From Rule 4)

A : #q101 # 1921 # 1091 #19201 # q31

B : #q101 # 19210 1 # 1091 # 19201 #q3101 #q3

Step 14: (From Rule 2)

A : #q101 # 1921 # 1091 # 19201 #19310

B : #q101 # 1921 # 1091 # 19201 # q3101 #

q30

Step 15: (From Rule 2)

A: #q101#1q21#10q1#1q201#q3101

B: #q101#1q21#10q1#1q201#q8101#q301

Step 16: (From Rule 2)

A: #q101#1q21#10q1#1q201#q3101#

B: #q101#1q21#10q1#1q201#q3101#q301#

Step 17: (From Rule 4)

A: #q101#1q21#10q1#1q201#q3101#q30

B: #q101#1q21#10q1#1q201#q3101#q301#q3

Step 18: (From Rule 2)

A: #q101#1q21#10q1#1q201#q3101q301

B: #q101#1q21#10q1#1q201#q3101#q301#q31

Step 19: (From Rule 2)

A: #q101#1q21#10q1#1q201#q3101#q301#

B: #q101#1q21#10q1#1q201#q3101#q3#q31

Step20: (From Rule 4)

A: #q101#1q21#10q1#1q201#q3101#q30
#q31

B: #q101#1q21#10q1#1q201#q3101#q30
#q31#q3

Step21: (From Rule 2)

A: #q101#1q21#10q1#1q201#q3101#
q301#q31#

B: #q101#1q21#10q1#1q201#q3101#q301
#q31#q3#

Step22: (From Rule 5)

A: #q101#1q21#10q1#1q201#q3101#
q301#q31#q3##

B: #q101#1q21#10q1#1q201#q3101#q301
#q31#q3##

class P & NP:

class P - problem solvable in polynomial Time

class NP - problem solvable in non deterministic polynomial Time.

classP:-

A Language is in class P iff they exist some polynomial T(n) such

that $L = L(M)$ for some deterministic Turing Machine M of Time complexity $T(n)$ the classes of problem with efficient come under class P.

Ex: ~~Knap~~ Kruskal Algorithm:

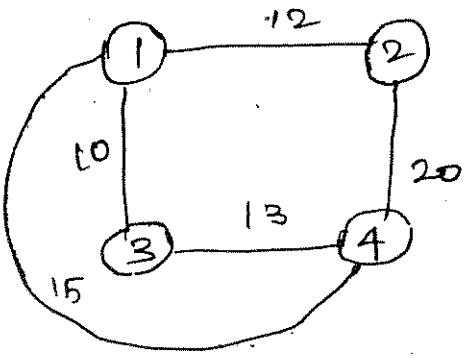
Step 1:
Maintain the connected component (node) for each state Initially the connected component itself.

Step 2:
Sort the edges in ascending order of their ~~use~~ weight 's'.

Step 3:
Select the lowest weight fr the Spanning Tree select the edge for spanning & merge the Two connected component Involved.

Step 4:
Repeat Step (3) until all the connected components are visited.

1) Consider given graph.



Finding shortest path construct its Turing machine code.

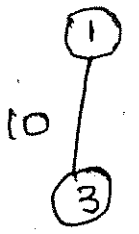
Sol: step1: assending order of weight

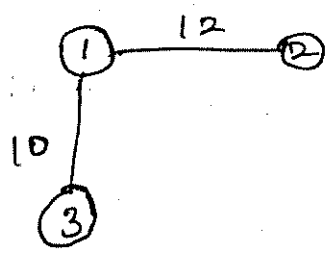
$W = \{10, 12, 13, 15, 20\}$

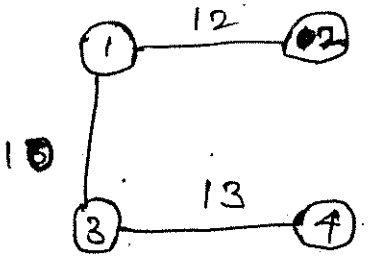
Step2: Total no. of states

$M = 4$

step3: we have to select minimum spanning Tree







minimum weight = 10 + 12 + 13 = 35

Step 4:

Now we have to design the Turing machine that accept the Given graph.

Step 4.1:

Assign values for all the nodes

node 1 = 1

node 2 = 10

node 3 = 11

node 4 = 100

Step 4.2:

The code with M it binary f the minimum weight of the spannin Tree & each edges of MWST all Separate by comman.

Turing Machine code = { M, W, E1, E2 ... }

Step 4.3:

If they is an edge e E between the nodes (i, j) with the weight of K can be written it

Binary format = {i, j, k}

Therefore for the Given problem

M = 4 = 100

W = 35 = 100011

$E_1 = \{1, 2, 12\}$

$= \{1, 10, 1100\}$

$E_2 = \{$

$E_1 = \{1, 3, 10\}$

$= \{1, 11, 1010\}$

$E_2 = \{1, 2, 12\}$

$= \{1, 10, 1100\}$

$E_3 = \{3, 4, 13\}$

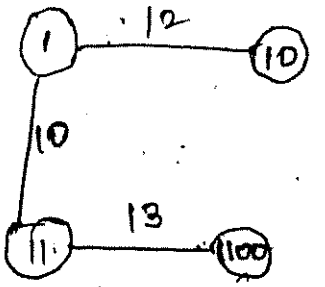
$= \{11, 100, 1101\}$

∴ the Turing machine code for the given problem is

Turing machine code = {M, W, E_1, E_2, E_3 ...}

= {100, 100011, (1, 11, 1010), (1, 10, 1100

(11, 100, 1101)}



Class NP:- Non deturministic polynomial Time

*A Language L' is in class NP If their exist non-deturministic Turing machine with the polynomial Time complexity of T(n) such that L = L(M).

* class P is the propare subset of class NP. class NP has many problem when comparing with class N class P.

NP complete problems:

* A Language L is NP complete If it's satisfies they following condition.

i) L is in NP

ii) For Every Language L' In NP their is a polynomial Time reduction of L' to L.

Ex: → Travelling Salesman problem
→ Hamiltonian problem
→ Knapsack problem
→ Graph colouring problem.

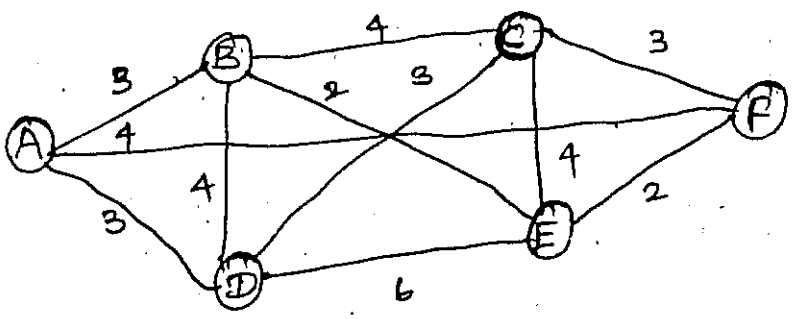
Travelling Salesman problem:

* It can be model as undirected weighted graph such that cities are graph-ots and paths or graph edges.

* They distance between two cities can be labeled as the weight of the edges.

* It is a minimization problem starting and finishing of specialli specified vertex after having visited each other vertices exactly one.

d t(1), t(2) + d t(2), t(3) ... d(t(n), t(1)≤



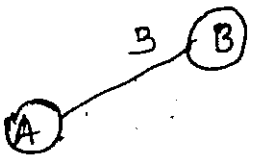
directed
or
edges.

o
eight

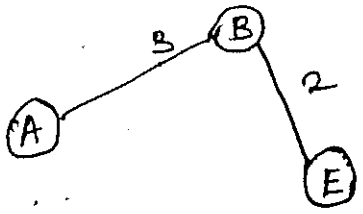
g of
aving
actly

t(1)≤b

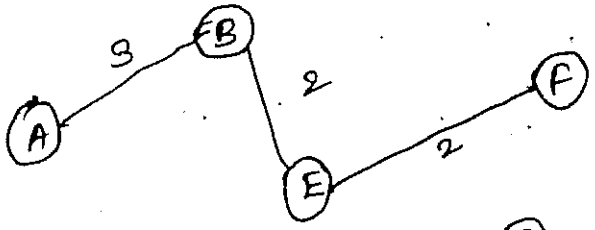
Step 1:-



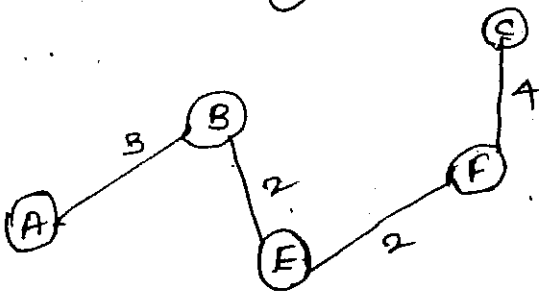
Step 2:



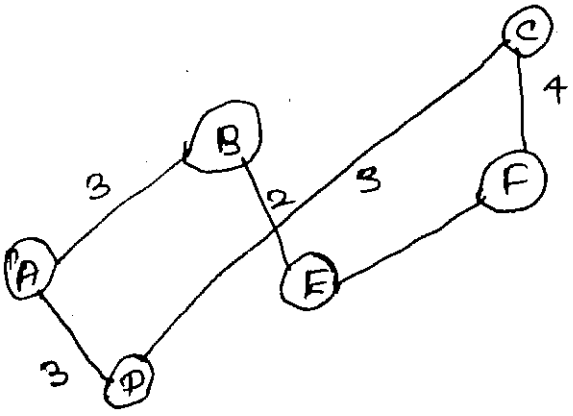
Step 3:



Step 4:



Step 5:



total weight = 38

(b)

$d_{A,B} + d_{B,E} + d_{E,F} + d_{F,C} + d_{C,D} + d_{D,A}$

$= 3 + 2 + 2 + 3 + 3 + 3$

$= 16$

computational complexity of Travelling Salesman problem:

* They problem has been shown to be under the catalogy NP hold & division the problem remains NP had even for the case when cities all in the plan with * distance and as well as in the no. of other restriction cases. Removing the condition of visiting each cities only one doesno Remove the NP hardness.

* Since its sine that in the plane case then is an optimal to that visit each cities only once.

Graph colouring problem:

* It is the way of colouring the vertices of a graph such that no to adjacent vertices share the same colour.

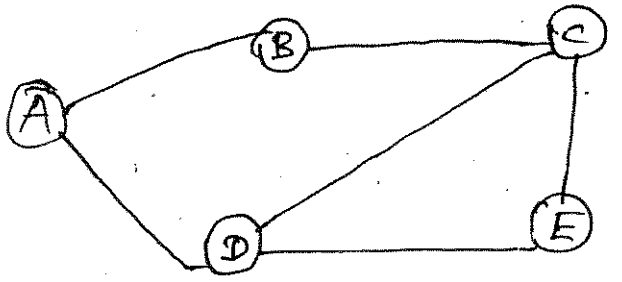
* This is also called as vertex colouring problem.

* Determining If a graph can be coloured with Two colours is equivalent determining whether or not the graph is ~~Diap~~ Bipartite & This is computable only in linear Time using BFS (Breath First Search but it cannot be computable in polynomial Time.

∴ They graph colouring problem comes under the catagaur -zed of NP- complete problem. For Solving the graph colouring problem in ~~deter~~ non deterministic polynomial Time. Brust force Search for a ~~Col~~ k- colouring can be used.

* Considering every vertex k^n assignment of k- colours to n vertices.

∴ It is possible To determine The Problem has a solution is legal without exact answer.



computation complexity For Graph colouring:

Graph colouring is computable only at polynomial Time.

∴ It is NP complete problem decide the If a given graph is atmost K - number of colouring For K Vertices In this K It is NP hard Two compute the gromatic Number The Dec complete problem NP complete catagorized graph of degree 4 where K & n are distinct Number.

The best node approximations algorithm computing the colour of k size vertices at most with in the Factor of $O(h(\log n)^{-3}(\log(\log$ of the Gromatic no. of $h>0$.

Theorem:

Lu is recursively enumerable

proof:

In Order prove this theorem, it is necessary to construct a TM U that accepts Lu the TM U consist of a Three Track Input tape where the First Track holds the Input Tape ($<M, w>$) the Secon Track containts the Tape of M where Tape symbols are written in unary for and the Third Track represents the state q_i which is also in unary form.

Input BB111 code 111 code 211 ... 111W BBB Track 1

Tape of M 0100100001 Track 2

state of M 000 .... 0BB Track 3

The operation U are as follows.

i) First make sure that the code for M is Legitimate code for some TM M otherwise it holds without accepting

ii) Initilize the second Tape with the input w, in its encoded form keep 0 the start state of M on the Tape and move the head of U's second Tape to the first simulator cell.

iii) If 0^i is current state with 0^j the current Input Symbol approach on Track Three and Two respectivity then U Finds the corresponding Transition of the form $0^i10^j10^k10^l10^m$ on the Track 1 and replace 0^i by 0^k and 0^j by 0^l.

iv) Move that next on tape two to the position corresponding to the value of M.

v) The Universal TM U accepts $0^i10^j10^k10^m$ if M has the Transition of the form

$\delta(0^i, 0^j) = (0^k 0^l 0^m)$, otherwise M halts without accepting.

Thus U simulator M and accept w. Thus L_u is Recursively enumerable.

The Diagonalization Languages:

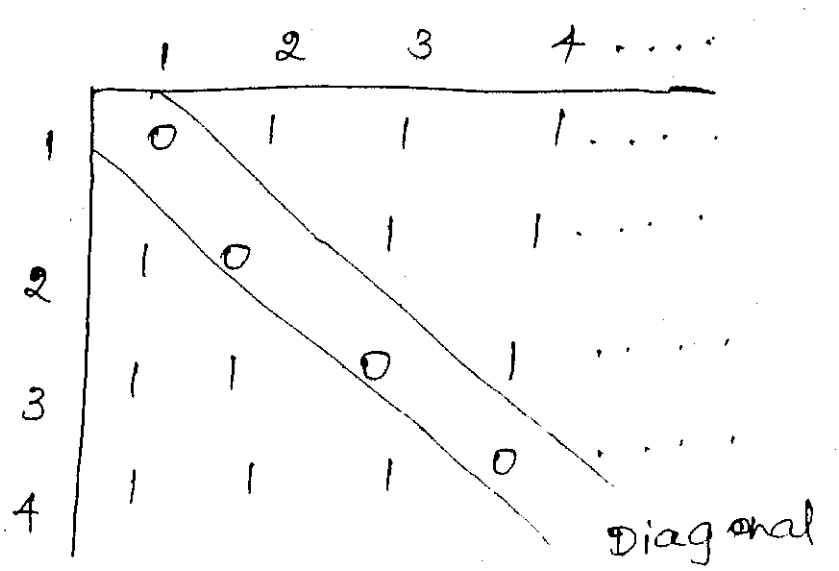
Some Integers do not belong to any TM. If w_i is not a valide TM code, Then take the TM M_i to be the TM with One State and no Transitions so for these values of i, M_i is a TM that halls on any Input.

$\therefore L(M_i) = \emptyset$ If $w_i \neq$ valid TM code

Definition:

The diagonalization languages L_d is the set of strings w_i where w_i not in $L(M_i)$.

L_d consist of all strings w such that the TM M whose code in w does not accepts the Input w.



Form the Table, it is clear that whether the TM Mi accepts the Input string Wj or not.

Diagonalization:-

The process of complementing the diagonal to construct the characteristic vector of a languages can't be the language that appears in any row is called diagonalization.

Then the complement of diagonal can't be the characteristics vector of Any TM.

Ld is not recursively Enumerable:

Theorem:

Ld is not recursively Enumerable

language i.e. There is no TM that accept Ld

Proof:-

Suppose, Ld is accepted by some TM defined by L(M) Since Ld is a languages over alphabet {0,1}, M would be in the List of TM constructed, where it is Include all TM with Input alphabet {0,1} So there may be atleast one code for M, say i. that is M = Mi

Then whether Wi is in Ld.

By definition,

$Ld = \{ Wi / Mi \text{ does not accepts } Wi \}$

Here we have two possibilities,

→ $Wi \in Ld$

This means that (i, i) entry is '0' and So Mi does not accepts Wi But our assumption here is that there exist a TM Mi, Which accepts Wi there is contradiction.

→ $Wi \notin Ld$

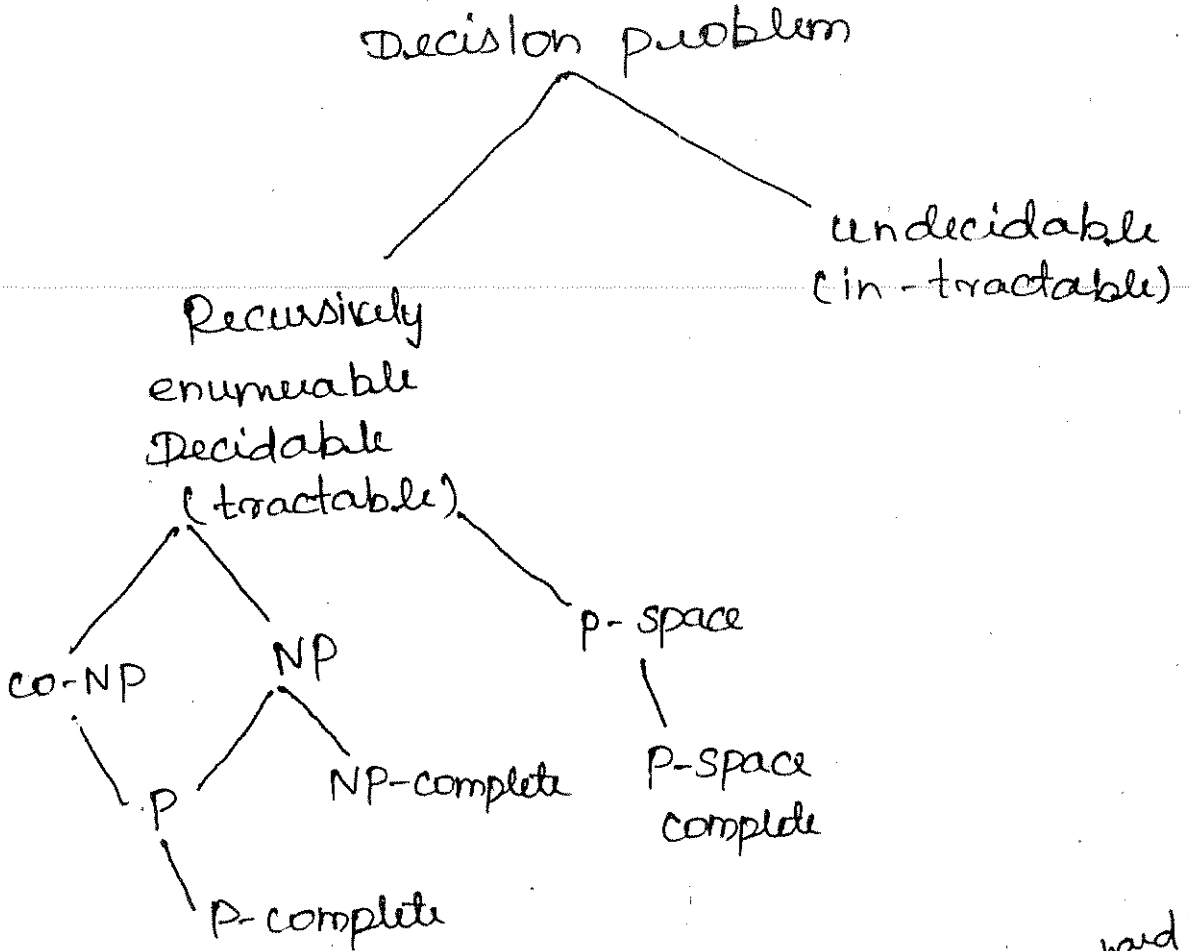
This means that (i, i) entry is '1' and So Mi accepts the Wi But by

definition of Ld, Mi does not accepts wi - so there is a contradiction.

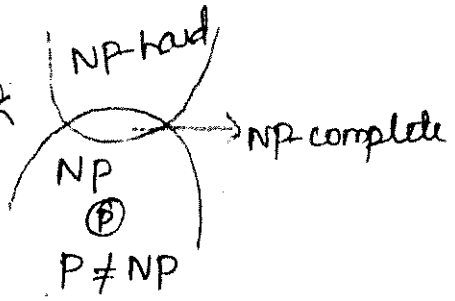
Thus it is clear that Ld is not Recursively enumerable and Ld is not recursive too.

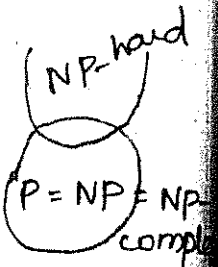
---

class P & class NP:



Hamiltonian path





Hamiltonian path problems:

-> Approximation
-> probabilistic
-> Heuristic

NP-complete problems } - sudoko puzzles

Knapsack problem:

It is the Type of the NP problem and time complexity = $O(W^n)$

eg: consider the given Items that has the corresponding weights.

Item	weight	cost
1	3	10
2	4	12
3	6	13
4	7	20
5	9	22

capacity W = 12

Item	weight	totalvalues	
1	3	10	
2	4	12	
3	6	13	
4	7	20	
5	9	22	
1, 2	7	22	- Feasible
1, 3	9	23	- Feasible
1, 4	10	30	- Feasible
1, 5	12	32	- Feasible
3, 4	13 > 12	33	- Not feasible

From this the Feasible Solution of Items (1,2) (1,3) (1,4), (1,5) . . .

Non Feasible solution are (3,4) . . .

Subset Sum problems:

The Subset Sum problems can be solved only by the BackTracking Method Strategy.

BackTracking eg: Chess board