PAGE NO.
DATE : / / 20

Dt: 11th july '18

INTRODUCTION ...

* The TC deals with capabilities and limitations of the computers.
* Three major Sections / components / areas of TC :-
1) Complexity theory
2) Computability theory.
3) Automata theory.

→ The Automata theory deals with the definations and the properties of the mathematical models of computatn. It includes :
1) Finite Automata.
2) Push down Automata.
3) Turing machine.

→ The Computability theory deals with the classificatn of problems that are solvable & unsolvable.

→ The Complexity theory deals with the classificatn of problems as easy and hard problems. That means the problems are analyzed with respect to the computational difficulty.

Dt: 13th july '18

* Mathematical Concepts :-
1) Sets
2) Relation & funcn.
3) Graph
4) Proofs.

* Set :-

$A = \{3, 5, 2, 8\}$

↳ A set is a collectn of unordered objects where the objects are k/a elements or members

Teacher's Signature

DATE : / /20

Introduction

of the Set :

Membership : $\in$

$3 \in A$ $\quad 9 \notin A$

$B = \{3, 9, 8, 7, 2\}$

$A \subset B$ (proper subset)

$A \subseteq B$ (subset)

* $A \subset B$ if $x \in A$, then $x \in B$.
* $A \subseteq B$ if $x \in A$ then $x \in B$ & $A = B$.

* **Set Representation :-**

i) Statement form $\quad A = \{$set of natural no.s less than 6$\}$

ii) Roster form $\quad A = \{1, 2, 3, 4, 5\}$

iii) Set Builder form (rule) $\quad A = \{x \mid x \in N \text{ and } x < 6\}$

* **Operations :-**

1) Union $\rightarrow A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
2) Intersection $\rightarrow A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
3) Set Difference $\rightarrow A - B = \{x \mid x \in A \text{ and } x \notin B\}$
4) Complement $\rightarrow \bar{A} = \{x \mid x \notin A\}$

Ex :

$B = \{$Set of all even no.s less than 10$\}$

$B = \{2, 4, 6, 8\}$

$B = \{x \mid x = 2K, K \in N \text{ and } K \leq 4\}$

* **Cartesian Product ($\times$) :-**

$\rightarrow A \times B = \{(x, y) \mid x \in A \text{ and } y \in B\}$

$A = \{2, 4, 5\}$ $\quad B = \{3, 6\}$

$A \times B = \{(2,3), (2,6), (4,3), (4,6), (5,3), (5,6)\}$

DATE : / /20

* If $|A| = m$ and $|B| = n$, then $|A \times B| = mn$

* **Power Set :-**

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

$P(A) = \{\phi, \{a\}, \{b\}, \{a, b\}\}$

$\rightarrow$ repetation doesn't affect the set.

$A = \{2, 3, 3, 7, 7\}$, $B = \{2, 3, 7\}$ } Both sets are equal.

pg-no : upto pg 25.

* If $|A| = n$, then $|P(A)| = 2^n$.
* **Empty Set :-** $\phi = \{\}$ $\quad |\phi| = 0$.

* **Relation & function :-**

Let A & B are two sets, then R be a relation from A to B is defined as :

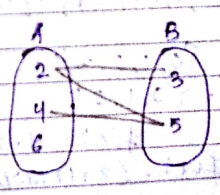
$R \subseteq A \times B$.

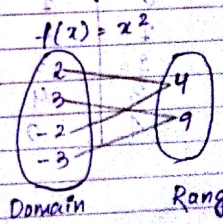
Let $A = \{2, 4, 6\}$, $B = \{3, 5\}$

$R =$ "is less than" (x is less than y)

$A \times B = \{(2,3), (2,5), (4,3), (4,5), (6,3), (6,5)\}$

$R = \{(2,3), (2,5), (4,5)\}$





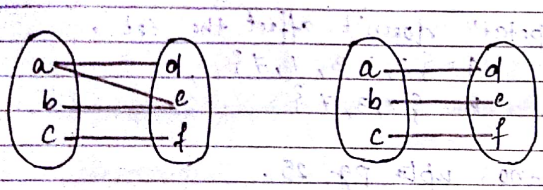
* Let $f: D \to R$ be a func^n

Here f: func^n name

D: set of inputs K/a Domain.

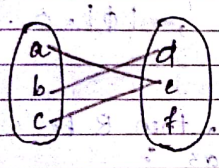
R: set of possible o/ps: K/a Range.

Ex: $f: \{2, 3, -2, -3\} \to \{4, 9\}$



(i)

(ii) onto funct^n.



(iii) into funct^n.

* If $f: D \to R$ and range$(f) = R$ then it is onto funct^n.
* If range$(f) \subset R$ then it is into func^n.

Ex:

x	f(x)
0	1
1	2
2	3
3	4
4	0

$f: \{0,1,2,3,4\} \to \{0,1,2,3,4\}$

$Z_5 = \{0, 1, 2, 3, 4\}$

$Z_3 = \{0, 1, 2\}$

$f(x) = 4 - x$

$f(x) = (x+1) \bmod 5$.

Dt: 16th July '18

Ex: Consider the func^n $g: Z_4 \times Z_4 \to Z_4$ defined as

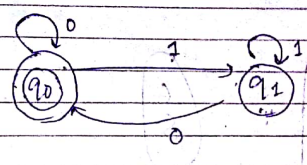
g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$D: \{0,1,2,3\} \times \{0,1,2,3\}$

$R: \{0,1,2,3\}$

$Z_4 = \{0, 1, 2, 3\}$ size: 16.

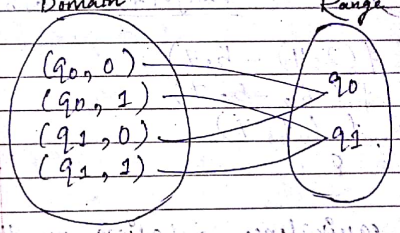
*



δ	0	1
q_0	q_0	q_1
q_1	q_0	q_1

$\delta(q_0, 1) = q_1$

$\delta: \{q_0, q_1\} \times \{0, 1\} \to \{q_0, q_1\}$



Ex:

f	4	5	7
0	6	8	6
1	8	6	8
2	6	6	8

i) find $f(1,7)$

ii) find domain & range of f.

iii) find $f(f(2,4)-5,7)$.

Ans) i) $f(1,7) = ?$

ii) domain $(f) = \{0,1,2\} \times \{4,5,7\}$

$(0,4)$
$(0,5)$
$(0,7)$
$(1,4)$
$(1,5)$
$(1,7)$
$(2,4)$
$(2,5)$
$(2,7)$

6
8

range $(f) = \{6,8\}$

iii) $f(f(2,4)-5,7)$
$= f(6,-5,7)$
$= f(1,7)$
$= 8$.

Q. Define an equivalence relation on the set of natural numbers, i.e. for $i, j \in N$, $i-j$ is a multiple of 7 i.e. $i \equiv_7 j$

→ i) reflexive :
Here $i-i = 0$ that is a multiple of 7
So $i \equiv_7 i$ & it is reflexive.

ii) Symmetric :
Let $i-j$ is a multiple of 7 i.e.
i.e. $i-j = 7K$ $K \in N$.
Now $j-i = -7k$, that is multiple of 7.
So, $j \equiv_7 i$ & it is symmetric.

iii) Transitive :
Let $i-j = 7K_1$, and $j-k = 7K_2$, where $K_1, K_2 \in N$.
Now $i-j+j-K = 7K_1+7K_2 = 7(K_1+K_2)$.
$i-K = 7K_3$ where $K_3 \in N$ and $K_3 = K_1+K_2$.
So, $i-K$ is a multiple of 7, so it is transitive.

Dt: 18th July'18

* Strings & Languages :-

→ Alphabet : It can be defined as the non-empty finite set of symbols.

→ Symbols are the elements of alphabet.
$\Sigma_1 = \{0,1\}$
$\Sigma_2 = \{a,b\}$

→ String : (w) : It is the finite sequence of symbols defined over an alphabet.
$w = 0011$.

→ Strings are written using the symbols one next to another not separated by command.

→ Language (L) :
$L = \{00, 01, 10, 11\}$

→ A language defined over a alphabet is a set of strings.

Teacher's Signature

L = {0, 00, 010, 10, 110, 000, ...}
↳ accepts the strings ending with 0's.

L = {w | w ends in 0}

→ Length of string: |w| - It is the no. of symbols present in the string.
Eg: If w = abcd, then |w| = 4.

L₂ = {w | |w| mod 2 = 0}.
↳ This language consists of those strings which are of even length.

→ Empty String: (ε): It has no symbol and its length is 0.
If w = ε, then |w| = 0.

→ Reverse of the String (w^R):
If w = 011, then w^R = 110.
- The reverse of a string is obtained by writing the string in opposite order, i.e.,
if $w = w_1 w_2 \ldots w_n$
then $w^R = w_n w_{n-1} \ldots w_1$
where w_1, w_2 are symbols.

→ Substring: If w = 01101, then 110 is a substring of 'w'.
- y is a substring of w if y appears consecutively in w.

→ Concatenation of Strings: Let x = 011 & y = 0011, then xy = 0110011.

- It refers to the process of appending one string next to another. (end of other string) i.e.
If $x = x_1 x_2 \ldots x_n$ & $y = y_1 y_2 \ldots y_n$
then $xy = x_1 x_2 \ldots x_n y_1 y_2 \ldots y_n$.

If x = 011, then xx = 011011 = x^2.
x^k = x is concatenated over itself k times.

→ String Order:
- Ex: L = {Set of strings over {0,1} whose length is at most 3}
= {ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111}.
- The string order represents the set of strings in the lexicographic order (dictionary order) and the shorter strings precede the longer strings.

Eg: L = {w | |w| MOD 2 = 0}. write using string order.
→ L = {w | |w| MOD 2 = 0}
= {ε, 00, 01, 10, 11, 0000, 0001, 0010, 1111...}

* Prefix of a string:-
Let y = 0001, x = 00, z = 01,
xz = y. (here x is prefix of y)
→ A string x is a prefix of string y if a string z exists such that xz = y.

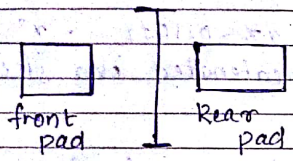
Σ = {0,1} → set of symbols
L = {0,1}

Teacher's Signature

Dt: 20th July '18 **Finite Automata**.

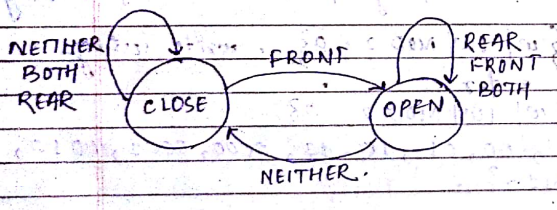
↳ Finite Automata are the simplest computational models with an extremely little amount of memory.

Ex: Automatic door controller.



→ The door has 2 states:
1) OPEN 2) CLOSE.

→ Conditions or Inputs:
1) FRONT 3) BOTH
2) REAR 4) NEITHER.



1) FRONT: Person is standing on front pad.

2) REAR: " " " " Rear pad.

3) BOTH: " " " " both the pads.

4) NEITHER: No one is standing on the pads.

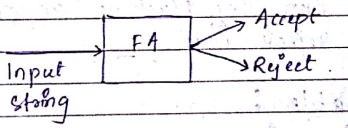
* Transition Table:-

States \ INPUTS	FRONT	REAR	BOTH	NEITHER
CLOSED	OPEN	CLOSED	CLOSED	CLOSED
OPEN	OPEN	OPEN	OPEN	CLOSED.

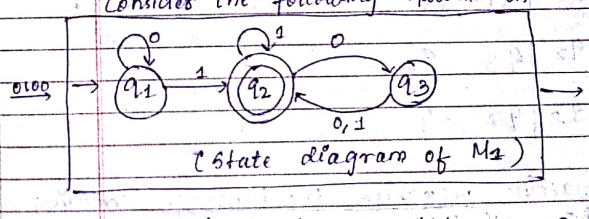
↳ This finite automat^n uses one bit of memory to store the current state only.

↳ The state changes on receiving the input conditions.

* Mathematical formulat^n of FA:-



Consider the following Automat^n M_1.



(State diagram of M_1)

* States: It has 3 states q_1, q_2, q_3.

i) Initial State: The state indicated with an empty arrow is K/a initial state.

ii) Accept State: The state indicated with a double circle is K/a accept/final state.

Teacher's Signature

Here q_1 is the initial state & q_2 is the accept state.

* Transitions :

Let Input String = 0100.

$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} (q_2)$

Input String = 1101

$q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} (q_2)$

→ The Input String is processed from left to right.

→ After the last symbol, if it is in the accept state then the Machine accepts the string o/w rejects it.

– Transition table :

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

→ This machine recognizes the language consisting of strings that have even no. of zeros after the rightmost or last one. (last 1).

→ Smallest length string is 1.

$L = \{1, 100, 10000, 0100, 01100, \ldots\}$

Teacher's Signature

Dt : 23rd July'18

* Formal Definition of FA :

Reasons :

1) Formal Def^n is precise.
2) formal def^n gives notations.

* Components of FA :-

1) Set of states
2) Input alphabet
3) transitions / Rules
4) Initial state
5) Set of Accept states

→ It has 5 elements, so it is a 5-tuple.

→ A finite Automata is a 5-tuple $M(Q, \Sigma, \delta, q_0, F)$ where,

i) Q : Set of states
ii) Σ : Input alphabet
iii) δ : transition func^n defined
iv) q_0 : Initial state
v) F : Set of accept states.

→ In a FA, there is a single initial state but there can be zero or more no. of accept states, i.e. $0 \leq |F| \leq |Q|$

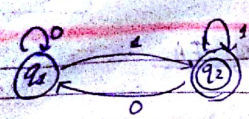
→ In a FA, if the transition func^n specifies exactly one next state for each possible combinat^n of a state & the input symbol then it is called Deterministic finite Automata (DFA).

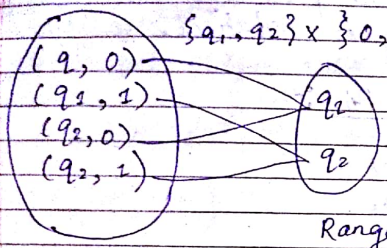
→ In DFA there must be a transition for every i/p symbol from each state.

→ In DFA, we can have more than one accept state.

→ There is exactly one next state for every combinat^n.

Teacher's Signature

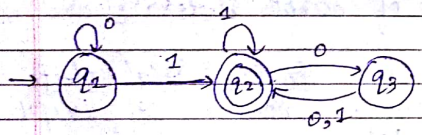




$$\delta : Q \times \Sigma \to Q$$

— In a DFA, the total no. of transitions $= |Q| \times |\Sigma|$.

* Formal Definat^n of FA M_1.



(state diagram of M_1).

M_1 can be defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
i) $Q = \{q_1, q_2, q_3\}$
ii) $\Sigma = \{0, 1\}$
iii) $\delta : \{q_1, q_2, q_3\} \times \{0, 1\} \to \{q_1, q_2, q_3\}$
iv) $q_0 = q_1$ is the initial state.
v) $F = \{q_2\}$
↳ accept state.

p.T.O

So, $M_1(\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_2\})$

where

δ	0	1
→ q_1	q_1	q_2
* q_2	q_3	q_2
q_3	q_2	q_2

→ [for initial state]
* [for accept state]

* Finite State Machine (FSM) :-
→ FA is also k/a finite state Machine (FSM).

* Accepting a string : After processing the string in the machine, if it reaches the accept state, then the string is accepted by the machine or else it is rejected.

→ 110 : string.

$q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} (q_3)$ → not accepted. as q_3 is not in double circle.

— double circle : accept state.
here q_2 : accept state.

→ 1001

$q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2 \xrightarrow{1} (q_2)$
↳ accepted

p.T.O

Teacher's Signature

PAGE NO.
DATE : / /20

If A is the language i.e the set of string accepted by the FSM 'M' then, 'M' recognizes the language A.

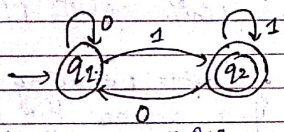
$L(M) = A$ (language of the machine).

where $A = \{w | M \text{ accepts } w\}$.

↳ for FSM M_1,

$L(M_1) = \{w | w \text{ contains at least 1 and even no. of 0s after the last 1}\}$

Q) Consider the following FSM, M_2.



(State diagram of FSM M_2)

→ There is a initial state.

i) $Q = \{q_1, q_2\}$

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

iii) $\delta = \{q_1, q_2\} \times \{0, 1\} \rightarrow \{q_1, q_2\}$

iv) $q_0 = q_1$ is initial state.

v) $F = \{q_2\}$

So, $M_2(\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$

where,

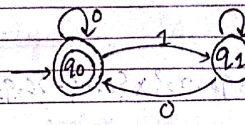
δ	0	1
→ q_1	q_1	q_2
* q_2	q_1	q_2

$L(M_2) = \{w | w \text{ ends in } 1\}$

PAGE NO.
DATE : / /20

Dt: 25th July '18

Q) Consider following DFA, M_3 =



(State diagram of M_3)

The FSM M_3 can be defined as $M_3(Q, \Sigma, \delta, q_0, F)$, where,

i) $Q = \{q_0, q_1\}$

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

iii) $\delta = \{q_0, q_1\} \times \{0, 1\} \rightarrow \{q_0, q_1\}$

$\delta : Q \times \Sigma \rightarrow Q$.

iv) q_0 is the initial state.

v) $F = \{q_0\}$

Set of strings that are accepted =

$\{\epsilon, 0, 00, 10, 000, 010, 100, \dots\}$

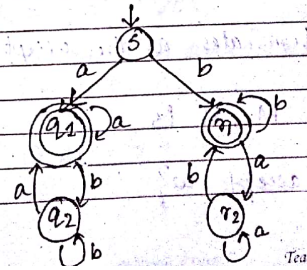
↳ If ϵ is present then initial state will be accept state.

Transition Table :

δ :	0	1
→ * q_0	q_0	q_1
q_1	q_0	q_1

$L(M_3) = \{w |$ w is the empty string or w ends in 0$\}$

Q) Consider the FSM M_4,



Teacher's Signature

i) $Q = \{S, q_1, r_1, q_2, r_2\}$.

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

iii) $\delta = \{S, q_1, r_1, q_2, r_2\} \times \{a, b\} \to \{S, q_1, r_1, q_2, r_2\}$

iv) S is the initial state.

v) $F = \{q_1, r_1\}$

Set of string that are accepted =

$\{a, b, aa, bb, aaa, aba, bbb, bab, \ldots\}$.

$L(M_4) = \{w \mid w$ starts & ends in a or w starts & ends in $b\}$

* Formal Def° of Computation :-

Let $M(Q, \Sigma, \delta, q_0, F)$ be the finite automata and $w = w_1 w_2 \ldots w_n$ be the input string. M accepts w if it goes through the states $r_0, r_1, \ldots r_n$ with three conditions

i) $r_0 = q_0$: It begins from the initial state.

ii) $\delta(r_i, w_{i+1}) = r_{i+1}$, $0 \le i \le n-1$

iii) $r_n \in F$; It terminates at the accept state.

The language of FA, M is

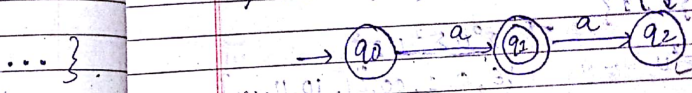
$A = L(M) = \{w \mid M$ accepts $w\}$.

M recognizes the language $L = \{ \}$

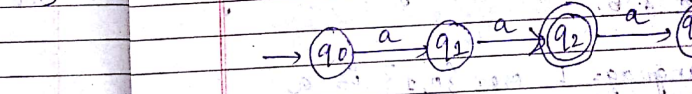
* Designing DFA :-

1) Design a DEA over $\Sigma = \{a\}$ that accepts all strings of
   i) length 1
   ii) length 2
   iii) length 3.

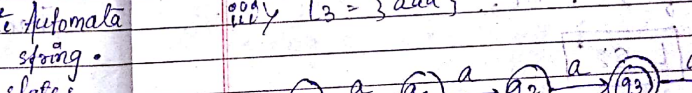
i) $L_1 = \{a\}$



ii) $L_2 = \{aa\}$



iii) $L_3 = \{aaa\}$



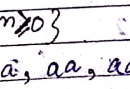
Dt: 27th July 18

2) Design a DFA over $\Sigma = \{a\}$ that recognize

i) $L = \{a^m \mid m \ge 0\}$

$= \{\epsilon, a, aa, aaa, \ldots\}$

$= a^*$

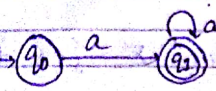


⊙ a^* = zero or more occurances of a.

e.g.

ii) $L = \{a^m \mid m \geq 1\}$ =

$= \{a, aa, aaa, \ldots\} = a^+$



② a^+ = one or more occurances of a.

* Power of Σ :

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

$\rightarrow \Sigma^0 = \{\epsilon\}$ $\rightarrow \Sigma^2 = \{00, 01, 10, 11\}$

$\rightarrow \Sigma^1 = \{0, 1\}$ $\rightarrow \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots\ldots$

$\rightarrow \Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

⊙ Σ^* = set of all possible strings consistency of a & b.

So a language L over $\{0,1\}$ is a subset of Σ^*, i.e,

$$L \subseteq \Sigma^*$$

$\rightarrow \Sigma^* = \{\epsilon\} \cup \Sigma^+$

Q) Design a DFA over $\Sigma = \{0,1\}$ that accepts the string of

i) length exactly 2

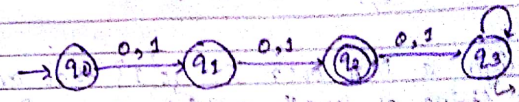
ii) length atleast 2 (2 or more)

iii) length atmost 2. (2 and less)

p.T.O

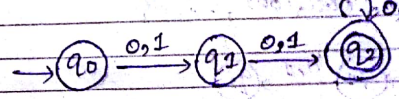
Soln: i) length exactly 2 :

$L = \{00, 01, 10, 11\}$ of



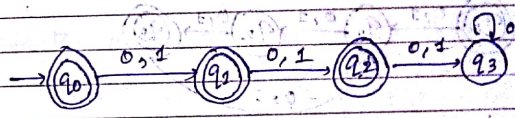
ii) length atleast 2 :

$L = \{00, 01, 10, 11, 000, 001, \ldots\}$



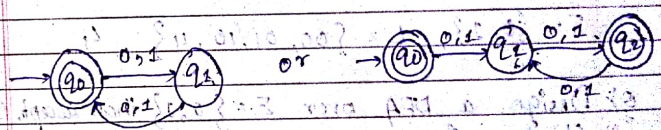
iii) length atmost 2 :

$L = \{\epsilon, 0, 1, 00, 01, 10, 11\}$.



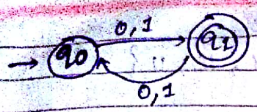
Q) Design a DFA over $\Sigma = \{0,1\}$ that accepts the string of even length i.e $L = \{w \mid |w| \bmod 2 = 0\}$ (even length string)

Soln: $L = \{\epsilon, 00, 01, 10, 11, 0000, \ldots\}$



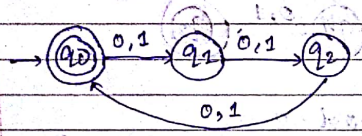
Q) Design a DFA over $\Sigma = \{0,1\}$ that accepts the string of odd length i.e $L = \{w \mid |w| \bmod 2 = 1\}$

Soln: $L = \{0, 1, 000, \ldots\}$

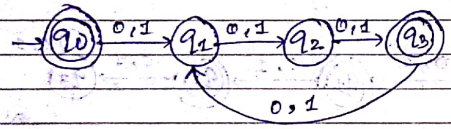


Q) Design a DFA over $\Sigma = \{0,1\}$ that accepts the string of length multiple of 3, i.e., $L = \{w \mid |w| \bmod 3 = 0\}$

Soln: $L = \{\epsilon, 000, 001, \ldots\}$



or



Dt: 28th july '18

* for a string of exact length n, the no. of states in DFA = $n+2$.

$\Sigma = \{a\}$, $L = \{a\}$ ... 3

$\Sigma = \{0,1\}$, $L = \{00, 01, 10, 11\}$ ... 4

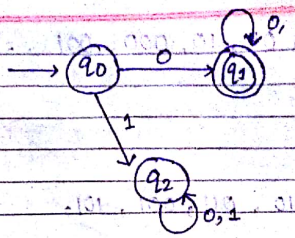
Q) Design a DFA over $\Sigma = \{0,1\}$ that accepts strings

i) starts with 0.

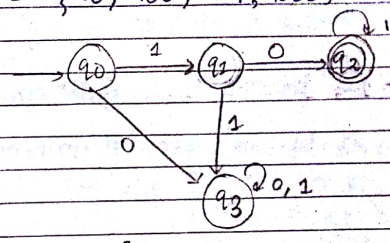
ii) starts with 10

i) $L = \{0, 00, 01, 000, 001, 010, 011, \ldots\}$

P.T.O

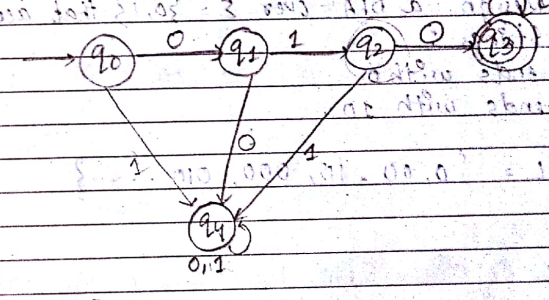


ii) $L = \{10, 100, 101, 1000, 1001, \ldots\}$



iii) starts with 010

$L = \{010, 0100, 0101, \ldots\}$



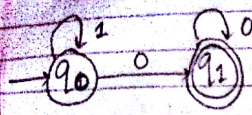
iv) contains 0

v) contains 01

vi) contains 001 as substring.

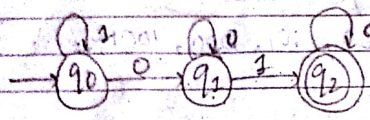
Teacher's Signature

iv) L = {0, 00, 01, 10, 000, 001, ...}

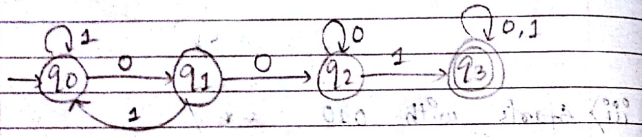


(minimized DFA)

v) L = {01, 010, 011, 001, 101, ...}

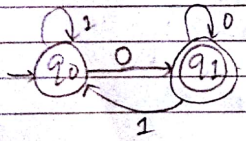


vi) L = {001, ...}

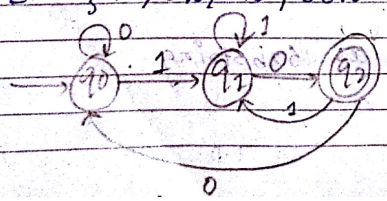


Q) Design a DFA over Σ = {0,1} that accepts strings:
i) ends with 0
ii) ends with 10.

i) L = {0, 00, 10, 000, 010, ...}

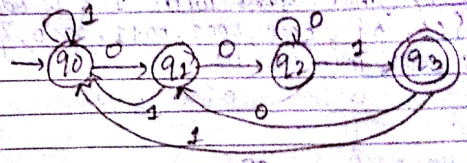


ii) L = {10, 010, 110, 0010 ....}



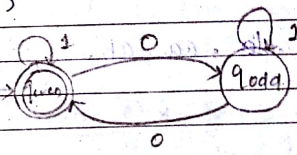
iii) ends with 001

L = {001, 0001, 1001, ...}

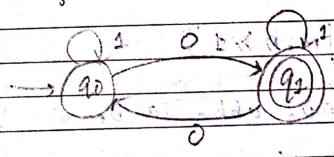


Q) Design a DFA over Σ = {0,1} that accepts strings:
i) Containing even no. of 0s
ii) containing odd no. of 0s

i) L = {ε, 1, 00, 100, 001, ...}

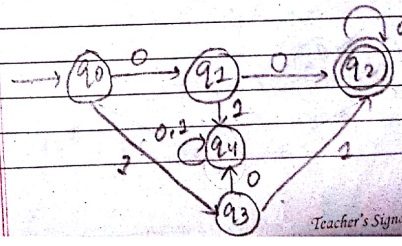


ii) L = {0, 01, 10, ...}



Q) Design a DFA that starts with 00 or starts with 11.

L = {00, 11, 000, 001, 110, 111, ...}

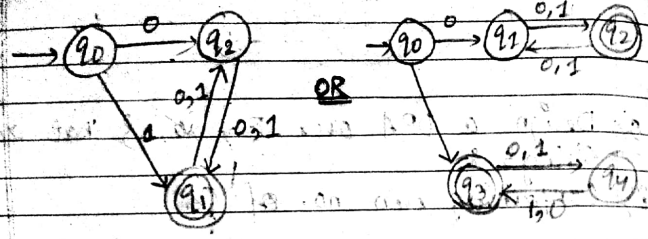


Teacher's Signature

DATE : / /20

DFA over $\Sigma = \{0, 1\}$ that accepts ... starting with 0 and of even length or that starts with 1 & of odd length.

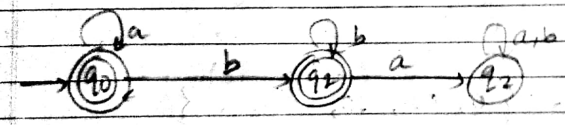
$L = \{00, 01, 0000, 0001, 0011, 0010, \} \cup \{1, 100, 101, 110, 111 \ldots\ldots\}$



ii) Design a DFA over $\Sigma = \{a, b\}$ that accepts

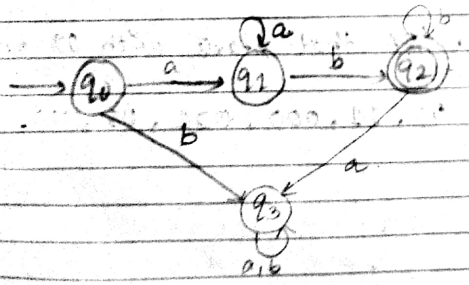
i) $L = \{a^m b^n \mid m, n \geq 0\}$

$L = \{\epsilon, a, b, aa, ab, \ldots\}$



ii) $L = \{a^m b^n \mid m, n \geq 1\}$

$L = \{ab, aab, abb, aaab, \ldots\}$



PAGE NO.
DATE : / /20

* Regular Operations :-

↳ Any language recognized by a DFA is K/a Regular language.

	Objects	operations
Arithmatic	Numbers	+, −, %, ×
Theory of computation	Languages	Union, star, Concatenation

↳ The regular operations are used to manipulate the regular languages.

* Types of Regular Operations :-

↳ Let A and B be two regular languages.

i) Union (∪) : $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

ii) Concatenation (∘) : $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

iii) Star (*) : $A^* = \{x^k \mid x \in A \text{ and } k \geq 0\}$

↳ Star is unary operator.

→ Union & Concatenat^n are binary operators.

Eg: Let $\Sigma = \{0, 1\}$, $L_1 = \{00, 10\}$, $L_2 = \{01, 11\}$.

Find : i) $L_1 \cup L_2$

ii) $L_1 L_2$

iii) L_1^2

Soln: i) $L_1 \cup L_2 = \{00, 01, 10, 11\}$

ii) $L_1 L_2 = \{0001, 0011, 1001, 1011\}$ (L_1 L_2)

iii) $L_1^2 = \{00, 10\}\{00, 10\} = \{0000, 0010, 1000, 1010\}$

Teacher's Signature

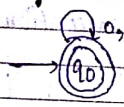
$L_1^0 = \{\epsilon\}$

$L_1^1 = \{00, 10\}$

Q) Is $L_1^* = \Sigma^*$?

→ In L_1^*, we will get all combinat^n of strings. In Σ^*, everything will get accepted.

Q) Design a DFA for Σ^*, if $\Sigma = \{0, 1\}$.



↓ will accept all strings.

Dt: 1st Aug '18

Q) $L_1 = \{good, bad\}$ $\Sigma = \{a, b, \ldots, z\}$

$L_2 = \{boy, girl\}$.

→ $L_1 \cup L_2 = \{good, bad, boy, girl\}$

→ $L_1 L_2 = \{goodboy, goodgirl, badboy, badgirl\}$

→ $L_1^* = \{\epsilon, good, bad, goodbad, badgood, \ldots\}$

* Closure Property :-

$2 \in N$, $5 \in N$, $2+5 \in N$, $2-5 \notin N$.

→ A collect^n of objects is closed under some operation, if applying that operat^n to the members of the collect^n returns an object still belong to that collection.

Eg: Set of natural no.s is closed under addition & multiplicat^n. (Substract^n & divicion are not included).

→ The set of regular languages is closed under Union, Concatenation & Kleen operation.

Teacher's Signature

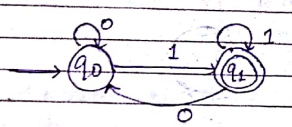
* Closed Under Union :-

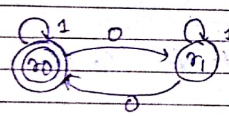
↳ If A is a regular language & B is a regular language then $A \cup B$ is also a regular language.

Ex: Let $\Sigma = \{0, 1\}$

$A = \{w \mid w \in \Sigma^*$ and w ends in $1\}$

$B = \{w \mid w \in \Sigma^*$ and w has even no. of 0s$\}$





$A \cup B = \{w \mid w \in \Sigma^*$ and w ends in 1 or w has even no. of 0s$\}$

→ Let M be the DFA for $A \cup B$ & it is a 5-tuple $(Q, \Sigma, \delta', q_0, F)$

i) $Q = \{q_0, q_1\} \times \{r_0, r_1\} = \{q_0r_0, q_0r_1, q_1r_0, q_1r_1\}$

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

iii) $\delta'(q_ir_i, a) = (\delta'(q_i, a), \delta'(r_i, a))$

$\delta'(q_0r_0, 0) = (\delta'(q_0, 0), \delta'(r_0, 0)) = q_0r_1$

$\delta'(q_0r_0, 1) = (\delta'(q_0, 1), \delta'(r_0, 1)) = q_1r_0$

$\delta'(q_0r_1, 0) = q_0r_0$

$\delta'(q_0r_1, 1) = q_1r_1$

$\delta'(q_1r_0, 0) = q_0r_1$

$\delta'(q_1r_0, 1) = q_1r_0$

$\delta'(q_1r_1, 0) = q_0r_0$

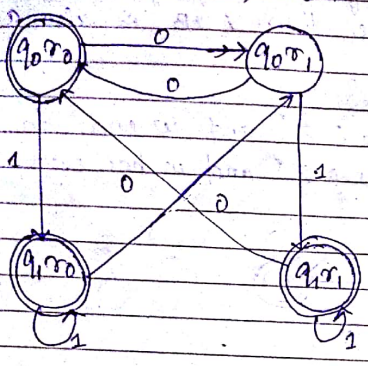
$\delta'(q_1r_1, 1) = q_1r_1$

Teacher's Signature

PAGE NO.
DATE : / /20

$\delta'(q_0, 0) = q_0$.

$\delta'(r_0, 0) = r_1$.



iv) The initial state is $(q_0 r_0)$.

v)

Theorem: The class of regular languages is closed under union operator.

proof: Let A_1 and A_2 be two regular languages, we want to show that $A_1 \cup A_2$ is also a regular language.

Let, $M_1 (Q_1, \Sigma, \delta_1, q_1, F_1)$ be the FA recognizing A_1, and

$M_2 (Q_2, \Sigma, \delta_2, q_2, F_2)$ be the FA recognizing A_2.

Now,

Construct a FA $M(Q, \Sigma, \delta, q_0, F)$ that recognizes $A_1 \cup A_2$ using the following steps.

i) $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

ii) Σ is same as that of M_1 & M_2.

If Σ_1 and Σ_2 are two different alphabets for M_1 & M_2 respectively, then,

$\Sigma = \Sigma_1 \cup \Sigma_2$.

Teacher's Signature _____

PAGE NO.
DATE : / /20

iii) δ is defined as follows:

for each state $(r_1, r_2) \in Q$ and for each symbol $a \in \Sigma$,

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$

iv) $q_0 = q_1 q_2$

The initial state of resulting DFA is a combination of the initial state of both the DFA's.

v) $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

i.e accept state of resulting DFA is the state containing accept states of either of DFA.

$\therefore F = \{Q_1 \times F_2\} \cup \{F_1 \times Q_2\}$

Dt: 3rd Aug '18

$F = \{Q_1 \times F_2\} \cup \{F_1 \times Q_2\}$

$Q_1 = \{q_0, q_1\}$

$F_1 = \{q_1\}$

$Q_2 = \{r_0, r_1\}$

$F_2 = \{r_0\}$

$F = \{(q_0 r_0), (q_1 r_0), (q_1 r_1)\}$

$Q_1 \times F_2 = \{(q_0 r_0), (q_1 r_0)\}$

$F_1 \times Q_2 = \{(q_1, r_0), (q_1, r_1)\}$

— for intersection operation :-

$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$

— Complement :

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

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

find $\bar{L}$:

Now, $\bar{L} = \{\epsilon, 0, 1, 000, 001, \ldots\}$

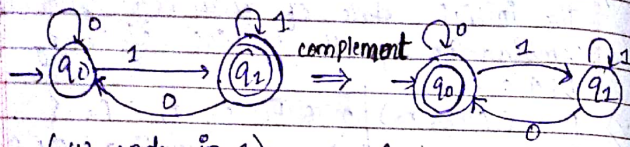
Teacher's Signature _____

$\Sigma = \Sigma^* - 1$

$\Sigma = \{ w \mid w \in \Sigma^* \text{ and } |w| \neq 2 \}$

Q) Design a DFA over $\Sigma = \{0,1\}$ that does not end in 1.

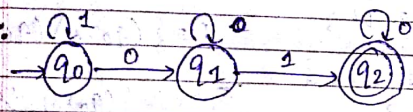
Ans:



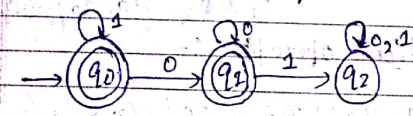
The complement of a DFA can be designed by altering the accept & non-accept states.

Q) Design a DFA that accepts a string that does not contain 01 as substring.

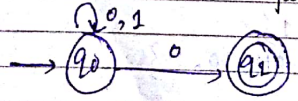
Ans:



⇓ complement.



* Nondeterminison (Nondeterministic Finite Automata (NFA))

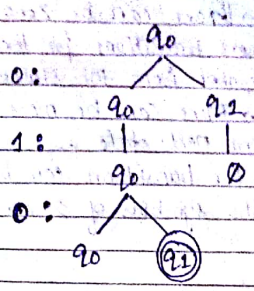


(w ends in 0)

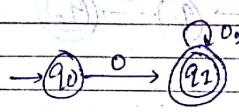
Teacher's Signature

Let w: 010.

→ for every combined we have multiple next states in NFA.



w starts with 0



→ In Non deterministic machine, when it reads the next i/p symbol in a state, several choices exist for the next state i.e. there can be more than one next state.

→ But in DFA there is only one next state.

→ Every DFA is a NFA.

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

DFA: $\delta'(q_0, 0) = q_1$

→ NFA is the generalizat^n of DFA.

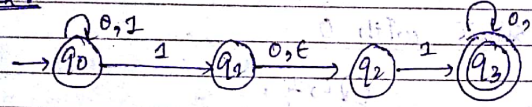
Q) Diff. b/w DFA & NFA

P.T.O.

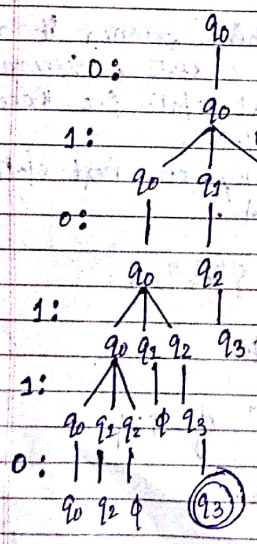
Teacher's Signature

DFA	NFA
→ There is a transition for each i/p symbol from each state.	→ There can be zero or more transitions for the i/p symbol from the states.
→ There is only one next state.	→ There can be more than one next state.
→ The transition is only for the symbols of Σ.	→ The transition can be for the symbols of Σ or ε.

NFA Ex:



w = 010110

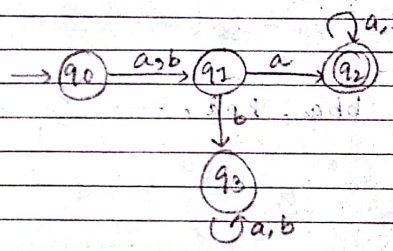


→ During the string processing of NFA, if the accept state appears in any of the exact path, then the string is said to be accepted else rejected.

Dt: 4th Aug '18 PROBLEM SOLVING SESSION

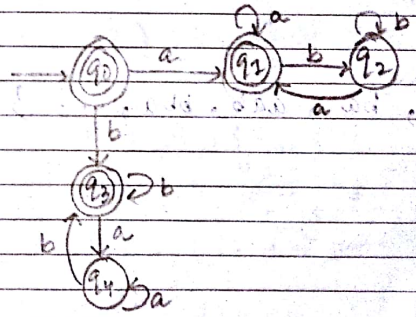
Q1) Design a DFA over Σ = {a, b} where the second symbol from LHS is a.

L = {aa, ba, aaa, baa, aab, ...}



Q2) Design a DFA that starts & ends with same symbol.

L = {ε, a, b, aa, bb, aba, aaa, ...}



P.T.O.

Teacher's Signature

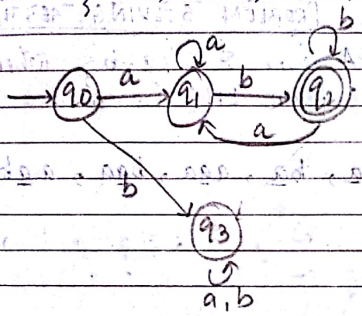
Teacher's Signature

PAGE NO.
DATE : / /20

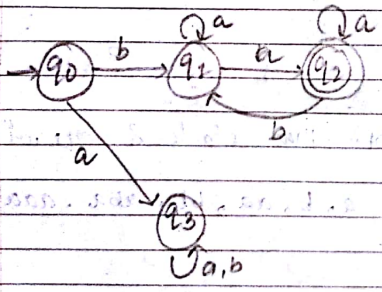
3Q) Design a DFA :

i) Starts with a and ends with b.
ii) starts with b and ends with a.
iii) Starts & ends with diff. symbols.

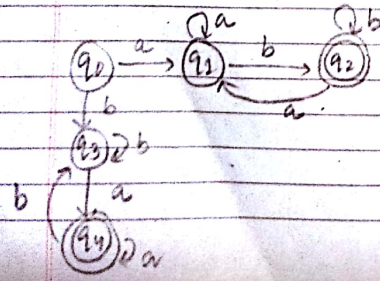
i) L = { ab, aab, abb, ... }



ii) L = { ba, bba, baa, ... }



iii) L = { ab, ba, aab, bba, ... }

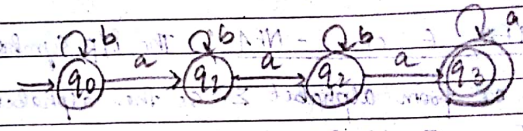


Teacher's Signature

PAGE NO.
DATE : / /20

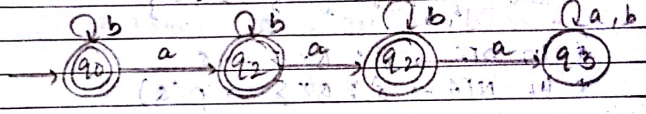
Q4) A: String contains atleast three 'a'.

L = { aaa, baaa, bababa, ... }



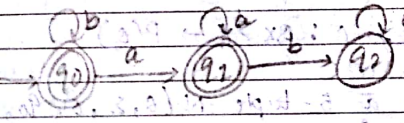
Q5) Atmost two 'a'.

L = { ε, a, b, aa, baa, aab, ... }

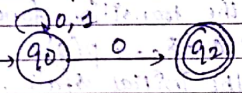


Q6) a never followed by b.

L = { ε, b, bb, ..., a, ba, ... }



* Formal Definition of NFA :-



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

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

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

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

Teacher's Signature

→ In NFA, the transition func^n takes a state & i/p symbol from Σ or ϵ and produces a set of possible next states.

* NFA - ϵ or ϵ - NFA : The i/p symbol is ϵ or from alphabet Σ. So the alphabet is $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

Ex: $\{a, b, \epsilon\}$ $\{0, 1, \epsilon\}$

* In DFA → δ : $Q \times \Sigma \to Q$ (power set of Q)
* In NFA → δ : $Q \times \Sigma \to P(Q)$

* Here $Q = \{q_0, q_1\}$
$P(Q) = \{\phi, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$

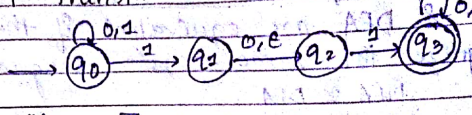
* In ϵ-NFA, δ : $Q \times \Sigma_\epsilon \to P(Q)$.

→ The NFA is a 5-tuple $N(Q, \Sigma, \delta, q_0, F)$ where i) Q is the set of states
ii) Σ is the input alphabet
iii) δ is the transition func^n defined as: $\delta : Q \times \Sigma_\epsilon \to P(Q)$
iv) q_0 is the initial state.
v) F is the set of accept states.

Transition Table :

δ	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
* q_1	ϕ	ϕ

Q) Find transition table :

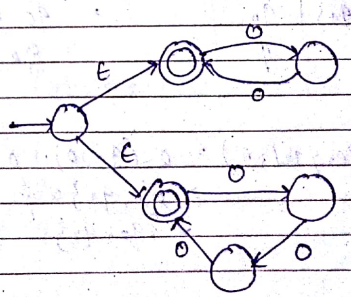


Transition Table :

δ	0	1	ϵ
→ q_0	$\{q_0\}$	$\{q_0, q_1\}$	ϕ
q_1	$\{q_2\}$	ϕ	$\{q_2\}$
q_2	ϕ	$\{q_3\}$	ϕ
* q_3	$\{q_3\}$	$\{q_3\}$	ϕ

* Formal defn of computatn :-
Let $N(Q, \Sigma, \delta, q_0, F)$ be the NFA and $w = y_1 y_2 \ldots y_n$ be the i/p string N accepts w if for each $y_i \in \Sigma_\epsilon$ and it goes through the states $r_0, r_1, r_2 \ldots r_n$ satisfying the three conditions :
i) $r_0 = q_0$.
ii) $r_{i+1} \in \delta(r_i, y_{i+1})$
iii) $r_n \in F$.

Ex:



$\Sigma = \{0\}$

$L = \{\epsilon, 00, 000 \ldots$

$L = \{w \mid w \in \Sigma^* \text{ and } |w| \text{ is multiple of 2 or 3 or } w \text{ is } \epsilon\}$

Teacher's Signature

Dt: 6th Aug '18

* Equivalence of NFA & DFA :-

→ NFA & DFA are equivalent if they recognize the same regular language.

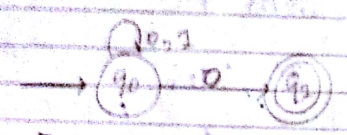
NFA ≅ DFA

→ Every NFA has an equivalent DFA.

Ex: Let $\Sigma = \{0, 1\}$

$L = \{w \mid w \text{ ends in } 01\}$

Construct NFA & find its equivalent DFA.



Transition Table :-

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
* q_1	ϕ	ϕ

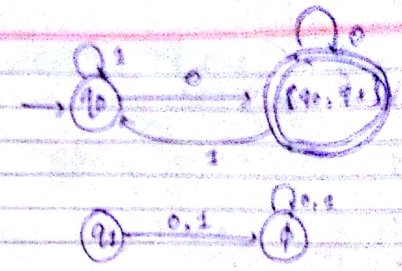
→ If the NFA has 'K' no. of states then its equivalent DFA has 2^K no. of states.

Transition Table for DFA :-

δ_{DFA}	0	1
→ q_0	$[q_0, q_1]$	q_0
* q_1	ϕ	ϕ
* $\{q_0, q_1\}$	$\{q_0, q_1\}$	q_0
ϕ	ϕ	ϕ

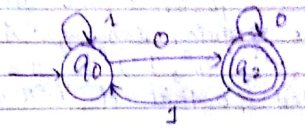
$\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
$= q_0 \cup \phi$
$= q_0$

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

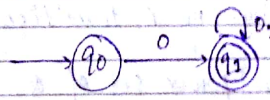


Here q_1 and ϕ are unreachable from q_0.

The equivalent DFA is :



Q) Let $\Sigma = \{0, 1\}$, $L = \{w \mid w \text{ starts with } 0\}$ construct the NFA & find its equivalent DFA



$P(Q) = \{\{q_0\}, \{q_1\}, \{q_0, q_1\}, \phi\}$

Transition table :-

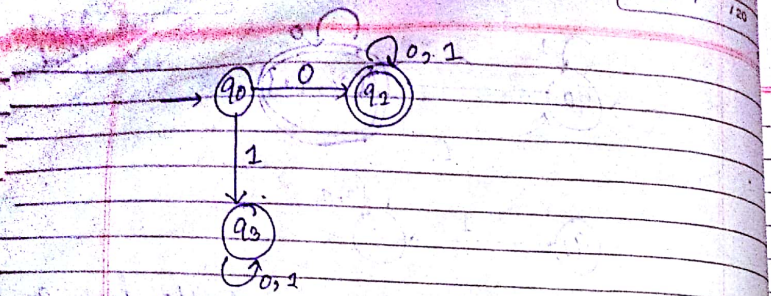
δ	0	1
→ q_0	q_1	ϕ
* q_1	q_1	q_1

Transition Table for DFA

δ_{DFA}	0	1
→ q_0	q_1	ϕ
* q_1	q_1	q_1
* $\{q_0, q_1\}$	q_1	q_1
ϕ	ϕ	ϕ

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

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



Theorem: Every NFA has an equivalent DFA.

proof: Let $N(Q, \Sigma, \delta, q_0, F)$ be the given NFA recognizing the regular language A.
Let $M(Q^1, \Sigma, \delta^1, q_0^1, F^1)$ be the DFA & it is constructed so that it will recognize the language A.
Let k = no. of states in NFA, then its equivalent DFA has 2^k no. of states.
Assume that NFA has no. ε-transition.

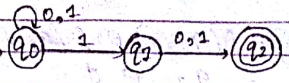
i) $Q' = P(Q)$ i.e the power set of Q.
ii) Σ is same.
iii) $\delta^1(R, a) = \bigcup_{r \in R} (\delta(r, a))$

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

iv) $q_0' = q_0$ i.e start state of NFA will be start state of DFA.
v) F' = States of DFA containing the accept states of NFA.
$= \{R \mid R \in Q' \text{ and } R \text{ contains accept states of } F\}$

Dt: 8th Aug'18

Q) Let $\Sigma = \{0, 1\}$. Consider following NFA. find it's equivalent DFA.



L = {10, 11, 010, 011, 110, 111, ...}

L = {w | w has a 1 in the 2nd pos^n from RHS}

Transition table:

① δ_{NFA}

	0	1
→ q_0	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
* q_2	ϕ	ϕ

② δ_{DFA}

	0	1
→ q_0	q_0	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_2]$	$[q_0, q_1, q_2]$
* $\{q_0, q_2\}$	q_0	$\{q_0, q_1\}$
* $\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

$\delta([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= \{q_0\} \cup \{q_2\}$

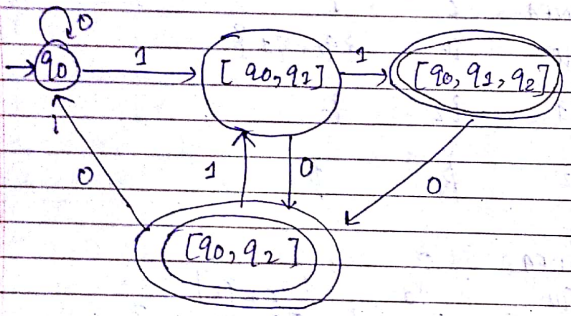
$\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
$= (q_0, q_1) \cup q_2$
$= [q_0, q_1, q_2]$

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

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

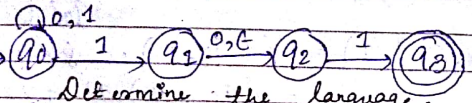
$\delta([q_0,q_2],0) = \delta[q_0,0) \cup \delta(q_2,0)$
$= q_0 \cup \phi$

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



* Conversion of ε-NFA to DFA :
→ If the NFA contains transitions for ε symbol, then find ε-closure of the states.

* ε-closure of a state :



Determine the language.
L = {ω ends with 101 or 11}
ε({q0}) = {q0 or states reachable from q0 with ε-transition}
= {q0}

ε({q1,3}) = {q1, q2} ε({q3}) = {q3}
ε({q2}) = {q2}

1) Start state of DFA = ε(start state of NFA)
ε(q0) = q0.
2) $\delta_{DFA}(q_0,0) = \varepsilon(\delta_{NFA}(q_0,0)) = \varepsilon(q_0) = q_0$.
$\delta_{DFA}(q_0,1) = \varepsilon(\{q_0,q_1\}) = \{q_0,q_1,q_2\}$

δ DFA	0	1
→ q0	q0	{q0, q1, q2}

Steps of conversion from ε-NFA to DFA :-

Let $N(Q, \Sigma, \delta, q_0, F)$ be the ε-NFA and it's equivalent DFA $M(Q', \Sigma, \delta', q_0, F')$ is constructed using following steps :-

i) $Q' = P(Q)$

ii) Σ is same to that of NFA.

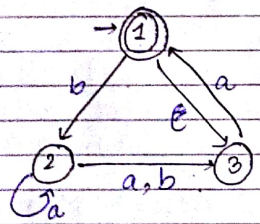
iii) $\delta'(R, a) = \{q \mid q \in \epsilon(\delta(r, a))$ for each $r \in R$

iv) $q_0' = \epsilon(q_0)$

v) F' = states of DFA containing accept states of NFA.

Dt : 10th Aug '18

Q) Find the equivalent DFA of the following NFA.



δ_{NFA}	a	b	ε
1	φ	2	3
2	2,3	3	φ
3	1	φ	φ

Soln : The possible states of DFA = $\{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$

Let $M(Q, \Sigma, \delta, q_0, F)$ be the DFA.

q_0 = Start state of DFA = ϵ(start state of NFA)

$= \epsilon(1) = \{1, 3\}$

Teacher's Signature

$= \epsilon(\{1\}) = \{\{1, 3\}, \{2\}\}$

δ_{DFA}	a	b
→ {1,3}	[1,3]	[2]
[2]	[2,3]	[3]
[2,3]	[1,2,3]	[3]
[3]	[1,3]	φ
* [1,2,3]	[1,2,3]	[2,3]
φ	φ	φ

Transitions of DFA :-

$\delta'(\{1,3\}, a) = \epsilon(\delta(1,a), \delta(3,a)) = \epsilon(1) = \{1,3\}$ ($\delta(3,a) = \phi$)

$\delta'(\{1,3\}, b) = \epsilon(\{2\}) = \{2\}$

$\delta'(2, a) = \epsilon(\{2,3\}) = \{2,3\}$

$\delta'(2, b) = \epsilon(\{3\}) = \{3\}$

$\delta'([2,3], a) = \epsilon(\{1,2,3\}) = [1,2,3]$

$\delta'([2,3], b) = \epsilon(\{3\}) = \{3\}$

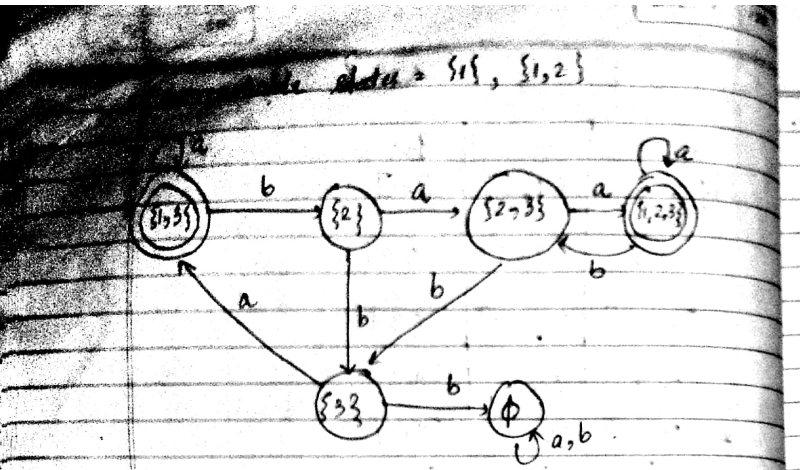
$\delta'(3, a) = \epsilon(1) = \{1,3\}$

$\delta'(3, b) = \epsilon(\phi) = \phi$.

$\delta'([1,2,3], a) = \epsilon(\{1,2,3\}) = \{1,2,3\}$

F = Accept states of DFA = $\{[1,3], [1,2,3]\}$

Teacher's Signature

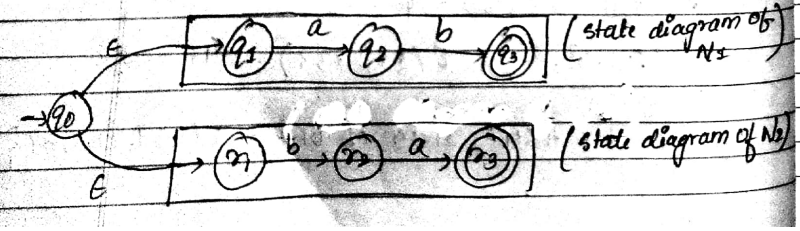


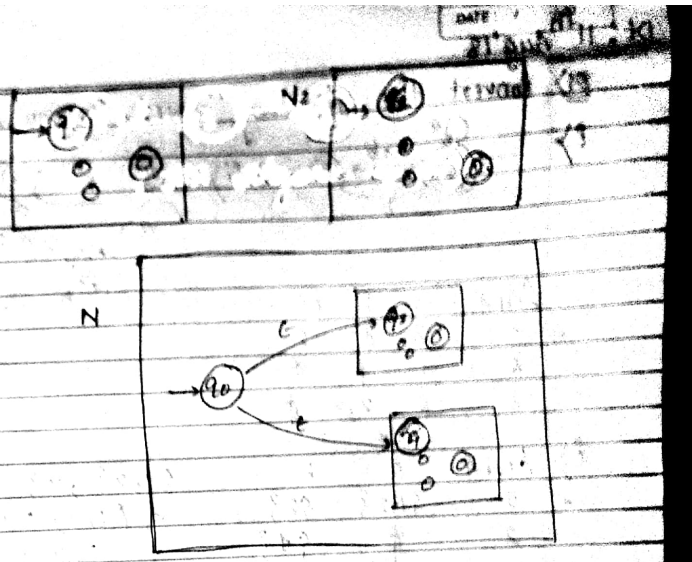
* Closure under Regular Operations :-

Theorem: The class of regular languages is closed under union operation.

Proof: Let A & B be two regular languages, i.e., they are recognized by two NFA's N_1 & N_2 respectively. We want to show that $A \cup B$ is a regular language i.e. Construct a NFA N that recognizes $A \cup B$.

Q) $\Sigma = \{a, b\}$
$A = \{w \mid w \in \Sigma^* \text{ and } w = ab\}$
$B = \{w \mid w = ba\}$.
$A \cup B = \{ab, ba\}$





Let $N(Q, \Sigma, \delta, q_0, F)$ be the constructed NFA & $N_1(Q_1, \Sigma, \delta_1, q_1, F_1)$ & $N_2(Q_2, \Sigma, \delta_2, q_2, F_2)$ be the two NFA.

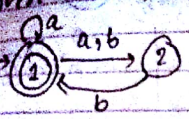
i) $Q = \{q_0\} \cup Q_1 \cup Q_2$,
ii) Σ is same.
iii) q_0 is the start state.
iv) $F = F_1 \cup F_2$
v) The transition function δ is defined as

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \{q_1, r_1\} & \text{if } q = q_0 \text{ and } a = \epsilon \\ \phi & \text{if } q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

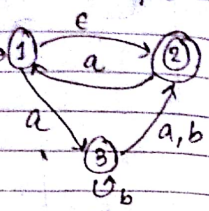
Dt: 11th Aug'18

Q) Convert the following 2 NFAs to equivalent DFA

i)



ii)



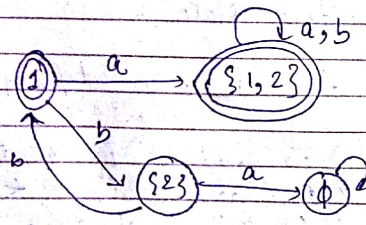
δNFA	a	b	ε
→ 1	3	ϕ	2
2	1	ϕ	ϕ
3	2	2,3	ϕ

i)

δNFA	a	b
* 1	{1,2}	{2}
2	ϕ	{1}
{1,2}	{1,2}	{2,1}
* ϕ	ϕ	ϕ
[1]	{1,2}	{2}
{2}	ϕ	{1}

{1,2}, a} = (1,a) ∪ (2,a) = 1,2 ∪ ϕ

(1,b) ∪ (2,b) = {2,1}



ii) Start state = ε(1) = {1,2} = Start state of DFA

P.T.O

δDFA	a	b
→ [1,2]	[1,2,3]	ϕ
* [1,2,3]	[1,2,3]	[2,3]
ϕ	ϕ	ϕ
* [2,3]	[1,2]	[2,3]

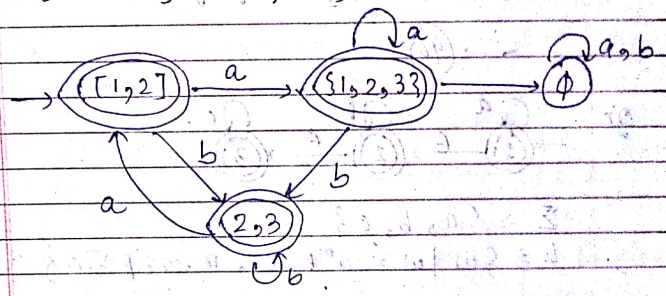
δ([1,2],a) = ε({1,3}) = [1,3,2]

δ([1,2],b) = ε(ϕ) = ϕ

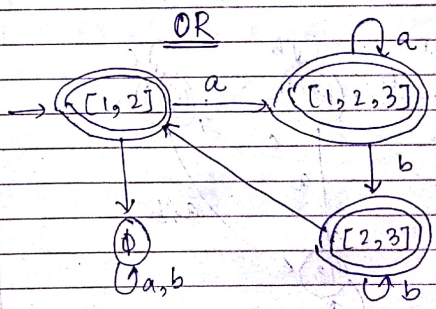
δ(1,2,a)

δ[2,3,a] = ε(1,2} = {1,2}

δ[(2,3),b] = ε{2,3} = {2,3}



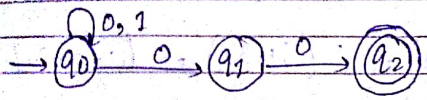
OR



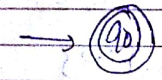
Q) Design a NFA for following language:

i) $\{w \mid w \text{ ends with } 00\}$ with three states.

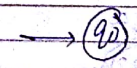
$L = \{00, 000, 100, 1100, 0000, \ldots\}$



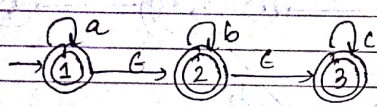
ii) $\{w \mid w = \epsilon\}$



iii) $\{w \mid w = \}$ Any language.



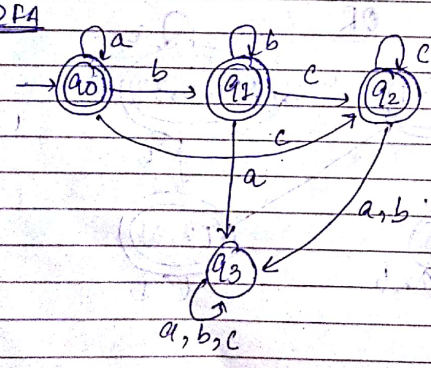
Q)



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

$L = \{w \mid w = a^m b^n c^p, m, n, p \geq 0\}$

DFA



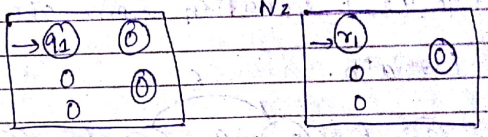
Theorem: The clan of regular languages is closed under concatenation operation.

Let $A = \{w \mid w = ab\}$ → $q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3$

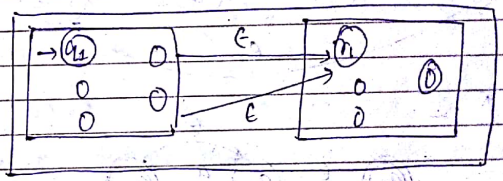
$B = \{w \mid w = bc\}$ → $r_1 \xrightarrow{b} r_2 \xrightarrow{c} r_3$

$A \circ B = AB = \{abbc\}$

Proof: N1 N2



N

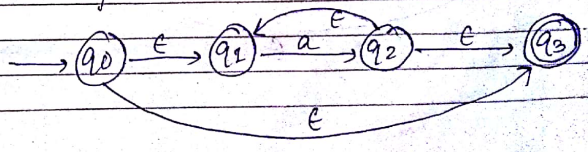


Theorem: The clan of regular languages is closed under star i.e., if A is a Regular language then A^* is also a regular language.

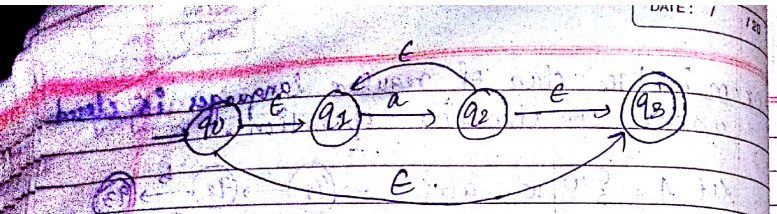
letter (letter/digit)*

$L = \{a\}$

$L^* = \{\epsilon, a, aa, aaa, \ldots\}$



Teacher's Signature

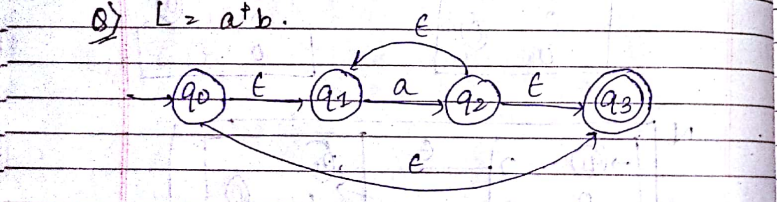


L^+ : one or more occurance.

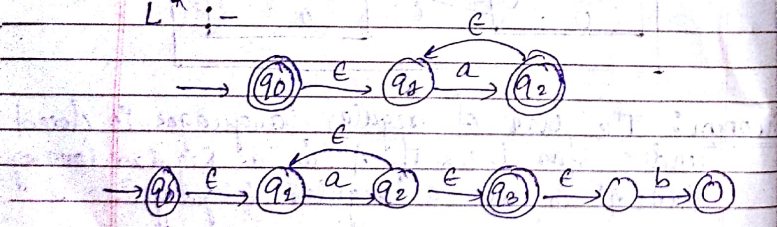
L^+ : {a, aa, ... }

Design the NFA for L^+ :- Remove the (q_0, q_3) ε transition from above diagram.

Q) $L = a^+b$.

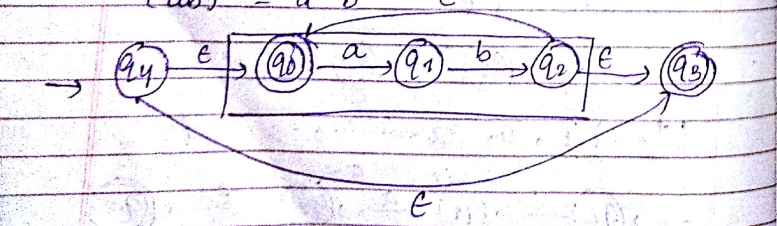


L^* :-



Q) Design NFA for $(ab)^*$

$(ab)^* = a^*b^*$

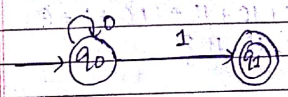


Dt: 8th Aug'18

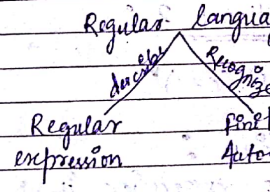
* Regular Expressions : 0^*1 = Regular language

$0^* = \{\epsilon, 0, 00, 000, \ldots\}1$

$= \{1, 01, 001, 0001, \ldots\}$.



DFA/NFA



* RE can be build using regular operations that describes a regular Language. The value of RE is a Regular Language (RL).

Ex: $(0 \cup 1)0^*$ = {0, 1, 00, 10, 000, 100, ... }

= Set of strings that starts with 0 or 1 followed by zero or any no. of 0s.

* Union : ∪, +, | Eg: $(0+1)^*$

$= \{\epsilon, 0, 1, 00, 01, 10, 11, (0+1)\}$

= Set of all possible strings are 0 and 1

$= \Sigma^*$

Q) write RE that describes string of exactly 2. Assume $\Sigma = \{0, 1\}$

$RE = (0+1)^2 = \{00, 01, 10, 11\}$

or $(0+1)(0+1) = \{00, 01, 10, 11\}$

* String length almost 2 :

RE = ε + 0 + 1 + (0+1)(0+1) = (0+1+ε)
= (0+1+ε)
= 00 + 01 + 0 + 10 + 11 + 1 + 0 + 1 + ε

L = {ε, 0, 1, 00, 01, 10, 11}

* String length atleast 2 :

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

* Formal Definition :

→ R is a regular expression if R is :

i) a, for each symbol a ∈ Σ
ii) ε
iii) ϕ
iv) $R_1 \cup R_2$, where R_1 & R_2 are regular expressions.
v) $R_1 R_2$
vi) R_1^*

Regular Expression	Language
1) 0	{0} = Single 0
2) ε	{ε}
3) ϕ	{ }
4) Σ^*	All possible strings over 0 and 1
5) $\Sigma^* 1$	Strings end with 1.
6) $0^* 1 0^*$	Set of strings that contains a single 1.
7) $\Sigma^* 1 \Sigma^*$	{w | w contains at least one 1}
$\begin{pmatrix} \epsilon & 1 & \epsilon \\ 0 & 1 & 0 \end{pmatrix}$	

8) $\Sigma^* 010 \Sigma^*$ {w | w contains 010 as a substring}

9) $(10+1)(0+1)^*$

10) $1^* \phi = \phi$

11) $1^* \epsilon = 1^*$

12) $\phi^* = \epsilon$

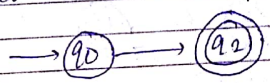
13) $R \cup \phi = R$

14) $R \cup \epsilon \neq R$ $\{00, 01\} \cup \{\epsilon\} = \{\epsilon, 00, 01\}$

* Equivalence of RE with FA

NFA from regular exp.

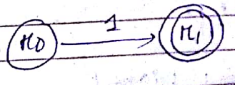
i) Let R = 0



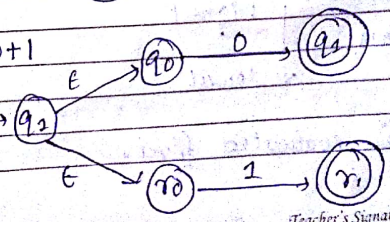
ii) R = ε

iii) R = ϕ

iv) R = 1

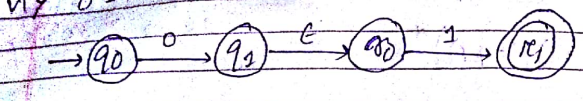


v) 0+1

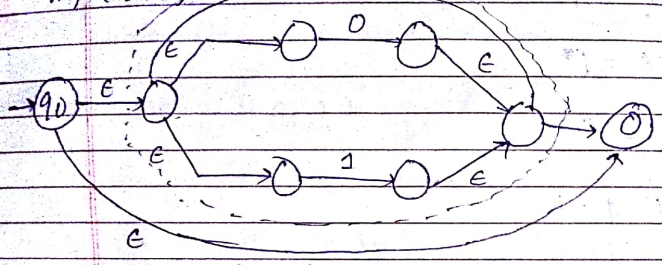


Teacher's Signature

vi) 01



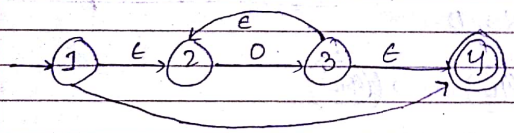
vii) (0+1)*



$(0+1)^+ \cup \{\epsilon\} = (0+1)^*$

0^+ = one or more occurance of 0s.

0^* = one or more occurance zero or more.



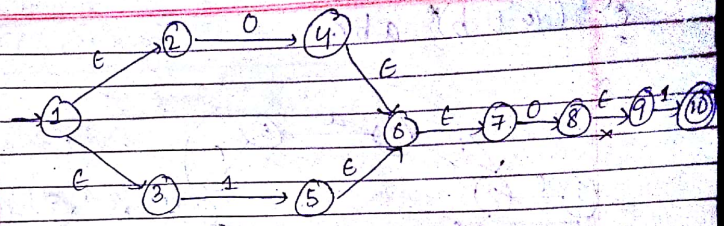
1) Add one extra start state & 1 accept state with ϵ - transition.
2) $1 \xrightarrow{\epsilon} 4$ is added for ϵ.
3) $3 \xrightarrow{\epsilon} 2$ is added for repeatation.
4) (0+1)01.

* Evaluation Order :

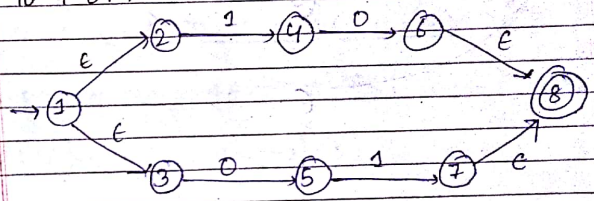
i) Star
ii) Concatenation
iii) Union

highest ↓ lowest.

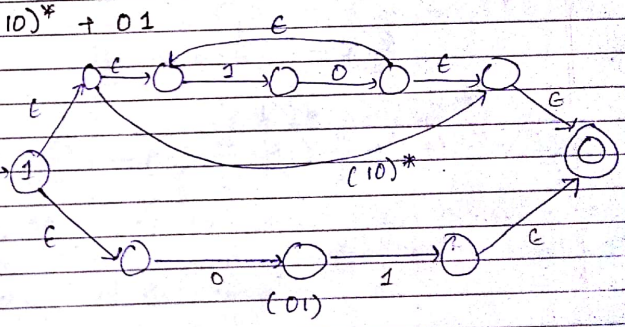
Parenthesis is evaluated first.



* 10 + 01.



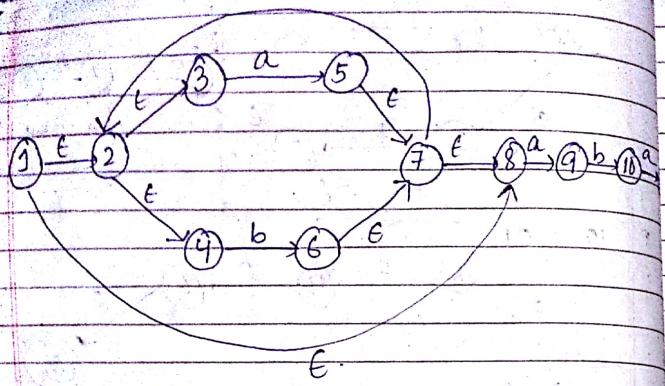
* (10)* → 01



P.70

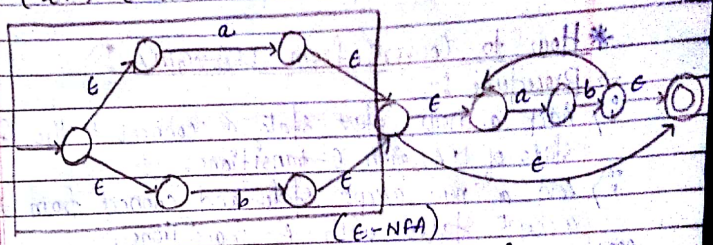
Teacher's Signature

Q) $(a \cup b)^* aba$



Dt: 20th Aug '18

Q) $(a+b)(ab)^*$

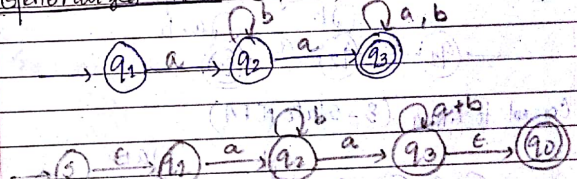


* Convert DFA to regular Expression :-

i) RE → ε-NFA → DFA

ii) DFA → GNFA → RE

* Generalized NFA (GNFA) :



→ GNFA is a NFA where the transition arrows may have regular expression as its labels, not only just the symbols.

* Conditions / Properties :-

i) There is no incoming transition to the start states from other states.

ii) There is a single start state & single accept state.

iii) There is no outgoing transition from accept state.

iv) Except the start & accept states, all other states are connected with each other with

Teacher's Signature

transitions labelled as ϕ.

* How to Convert from DFA → GNFA :

→ Procedure :-

i) Add a new start state & connect to the start state of DFA with ϵ-transitions.

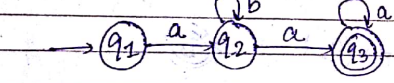
ii) Add a new accept state and connect from the accept states with ϵ-transitions.

iii) If there are multiple labels b/w the arrows of 2 states then replace them with the union operation.

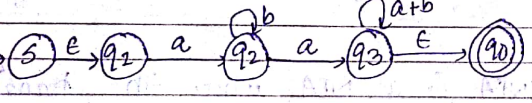
iv) If 2 states are not connected in the DFA, then connect them with transitions as ϕ.

* Convert GNFA to Regular Expression :

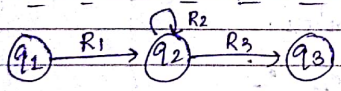
Q1)



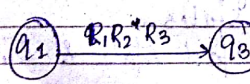
Convert to GNFA (3-state NFA)



(5-State GNFA)



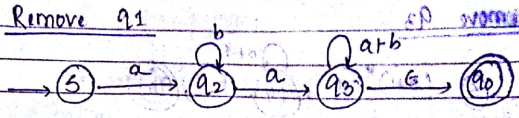
remove q_2.



$1^* = \{\epsilon, 1, 11, \ldots\}$

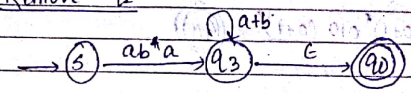
continued

Q1) Remove q_1



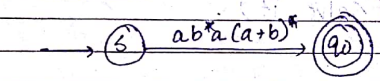
(4-State GNFA)

Remove q_2



(3-State GNFA)

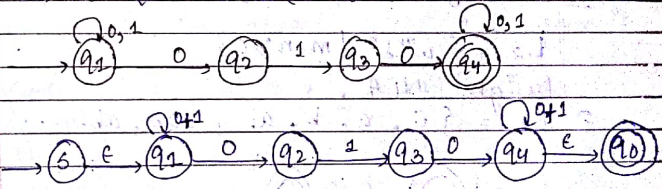
Remove q_3



(2-State GNFA)

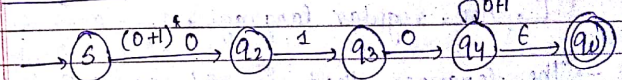
So, the RE = $ab^*a(a+b)^*$

Q2)

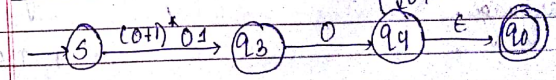


(6-State GNFA)

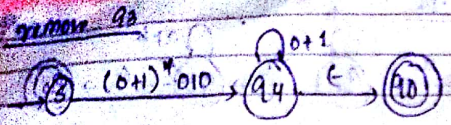
remove q_1.



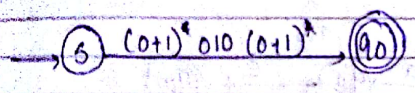
remove q_2



remove q_3



remove q_4



RE = $(0+1)^* 010 (0+1)^* = \Sigma^* 010 \Sigma^*$

Pg-no: 76, Ex 1.68 (Solve it).

Dt: 24th Aug '18

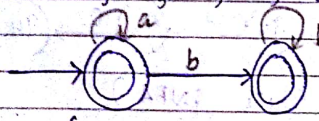
* Non-Regular language :-

Q) $L_1 = \{ a^n b^n \mid n \geq 0 \}$

$L_2 = \{ a^m b^n \mid m, n \geq 0 \}$

Design NFA.

① $L_2 = \{ \epsilon, a, b, ab, aab, abb \ldots \}$



② $L_1 = \{ \epsilon, ab, aabb, aaabbb \ldots \}$

↳ non-regular language.

→ The language that is not recognized by any finite automata is K/a Non-Regular language.

P.T.O.

PAGE NO:
DATE: / /20

Languages

(regular) finite | Infinite

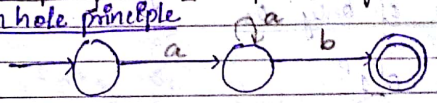
Ex: $L = \{ w \mid |w| = 2 \}$
$= \{ 00, 01, 10, 11 \}$

$L = \{ w \mid w$ starts with $0 \}$
- Regular.
- Non-Regular.

- $L_3 = \{ w \mid w$ contains equal no. of a & b$s \}$.

imp * Pumping lemma for Regular languages :-

pigeon hole principle



→ The pumping lemma states that all regular languages have a special property that means all the strings of the language can be pumped if its length is atleast equal to the pumping length.

→ Such strings contain a section that can be repeated any no. of times.

→ The pumping lemma is used to prove that the language is not regular, but it cannot be used to prove if is regular.

→ Let A be a regular language then there is a number p (pumping length) where if s is a string in A of length at least p, then s can be divided into three parts x, y & z satisfying the following conditions:

i) for each $i \geq 0$, $xy^i z \in A$.

ii) $|y| > 0$

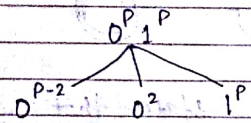
iii) $|xy| \leq p$.

Imp

Ex: Prove that $A = \{0^n 1^n \mid n \geq 0\}$ is a non-regular language.

Proof: Let's assume that A is a regular language.
So acc. to pumping lemma of regular languages, A has a pumping length.
Let 'p' be the pumping length.
Take a string s of A where length is at least p i.e $|s| \geq p$.
Let $\boxed{s = 0^p 1^p}$, $\boxed{|s| = 2p > p}$.
Divide s into 3 parts x, y and z.

i) y consists of only 0s.

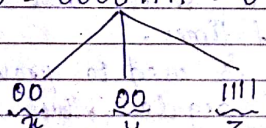


$y^2 = 0^2 0^2 = 0^4$

So $xy^2z = 0^{p-2} 0^4 1^p = 0^{p+2} 1^p \notin A$, for $i = 2$.

Ex: Let $p = 4$

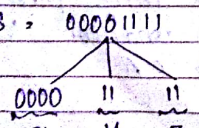
$s = 00001111 = 0^4 1^4$



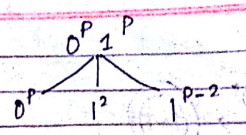
$xy^2z = xyyz = 000000111 = 0^6 1^4 \notin A$

ii) y consists of only 1s.

Ex: $s = 00001111$



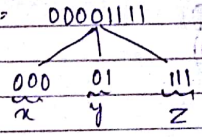
$xyyz = 000011111$
$= 0^4 1^6 \notin A$.

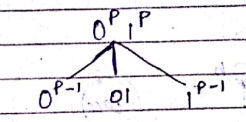


So $xy^2z = 0^p 1^{p-2} \notin A$.

iii) y consists of both 0s & 1s

Ex: $s = 00001111$ $xy^2z = 00001011111 \notin A$.



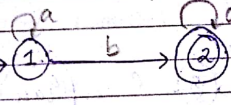


Here, $xy^2z = 0^{p-1}01011^{p-1} \notin A$.

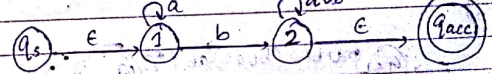
→ The condn 1 of pumping lemma is not satisfied for all possible divisions. It violates our assumption that means language A is non-regular.

Dt: 25th Aug '18

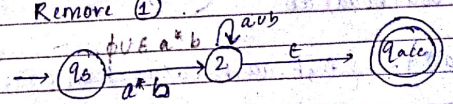
Q: Convert into equivalent R.G



Step-1:
GNFA



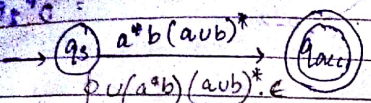
Step-2: Remove (1)



Teacher's Signature

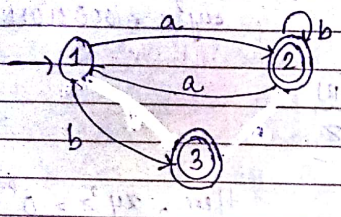
DATE : / /20

Remove (2)

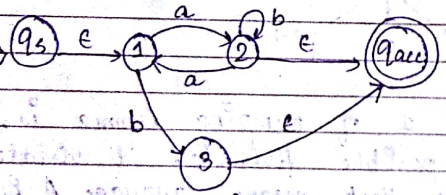


$\phi \cup (a^*b)(a \cup b)^* \cdot \epsilon$

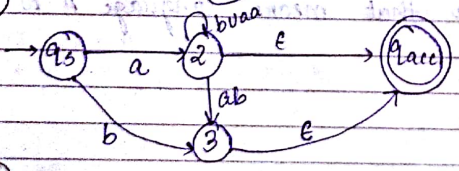
Q) Convert into equivalent R.E.



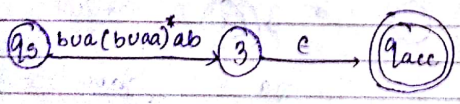
Step-1 :
GNFA



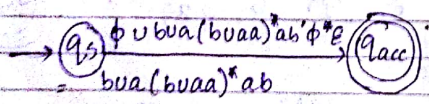
Remove (1)



Remove (2)



Remove (3)



$L((bvaa)^*) = \{\epsilon, b, aa, bb, aaaa, baa, aab, \ldots\}$

PAGE NO.
DATE : / /20

CH-2

Dt : 31st Aug '18

* Context Free Grammar (CFG) :

Regular language
- generates → CFG
- Described by / recognized by → Regular expⁿ / finite Automata

* Productions :

CFG $\begin{array}{|l} A \to 0A1 \\ A \to \epsilon \end{array}$

$A \Rightarrow 0A1$
$\Rightarrow 00A11$
$\Rightarrow 000A111$
$\Rightarrow 000111$

$A \Rightarrow 0A1$
$\Rightarrow 01$

$= \{\epsilon, 01, 0011, 000111, \ldots\}$

$= \{0^n 1^n \mid n \geq 0\}$

→ A Grammar is a set of substitutⁿ rules or productions.

→ In each productⁿ the LHS & RHS is separated by an arrow.

* Ex : $A \to 0A1$

A : Variables / non-terminals
0A1 : strings consist of variables & terminals.

- variable = {A}
- Terminals = {0, 1}

→ The variables or non-terminals are written using capital letters.

→ Terminals are written using small letters.

→ Every grammar must have a start variables.
Start Variable = {A}

DATE :

The languages generated by CFG are known as Context Free Languages (CFL).

* Derivation :

→ It is the sequence of substitutions to obtain a string from the start variable.

Ex: $S \to 0A1$
$A \to \#$

Is $S \overset{*}{\Rightarrow} 00\#11$?

↑ Substitutn with one or more steps.

$S \Rightarrow 0A1$
$\Rightarrow 0\#1$ $(A \to \#)$

Ex: $A \to 0A1$
$A \to B$
$B \to \#$

$S \overset{*}{\Rightarrow} 0\#1$
$00\#11$ ....

$L = \{0^n \# 1^n \mid n \geq 0\}$ ① $A \Rightarrow B$
$\Rightarrow \#$

② $A \Rightarrow 0A1$
$\Rightarrow 0B1$ $(A \to B)$
$\Rightarrow 0\#1$ $(B \to \#)$

① $A \Rightarrow 0A1$
$\Rightarrow 00A11$
$\Rightarrow 00B11$
$\Rightarrow 00\#11$

② So Generated strings $= \{\#, 0\#1, 00\#11, 000\#111 \ldots\}$

③ $L(G) = \{0^n \#1^n \mid n \geq 0\}$
i.e $A \overset{*}{\Rightarrow} 00\#11$.

PAGE NO.
DATE : / /20

Ex: consider the following grammar.
$S \to aSB \mid aB$
$B \to b$

$A \to a$
$A \to b$ $\}$ $A \to a/b$

Soln:
$S \Rightarrow aB$.
$\Rightarrow ab$ $(\because S \to B \to b)$

$S \Rightarrow aSB$
$\Rightarrow aSb$ $(\because B \to b)$
$\Rightarrow aaSBb$ $(\because S \to aSB)$
$\Rightarrow aaaBBb$ $(\because S \to aB)$
$\Rightarrow aaabBb$ $(B \to b)$
$\Rightarrow aaabbb$. $(B \to b)$

$S \Rightarrow aSB$
$\Rightarrow aaBB$
$\Rightarrow aabB$
$\Rightarrow aabb$.

① So, $L(G) = \{a^n b^n \mid n \geq 1\}$.

Ex: $S \to aS \mid \epsilon$

i) $S \Rightarrow \epsilon$. $S \to \epsilon$

ii) $S \Rightarrow aS$
$\Rightarrow a\epsilon$
$\Rightarrow a$. $S \to aS$

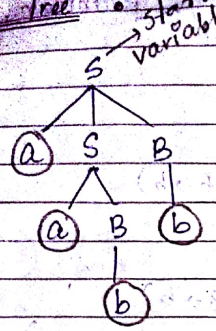
① $L(G) = \{a^n \mid n \geq 0\}$.

iii) $S \Rightarrow aS$
$\Rightarrow aaS$ $(S \to aS)$
$\Rightarrow aa\epsilon$ $(S \to \epsilon)$
$\Rightarrow aa$.

→ Left hand side components : Variables.

Teacher's Signature

* Parse Tree :-

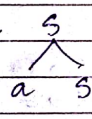


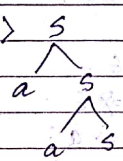
It is the pictorial representation of the derivation of a string from a grammar.

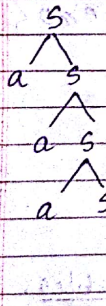
→ In parse tree, the root is always the start variable & the leaf nodes are the terminals. The non-leaf nodes are the variables / non-terminals.

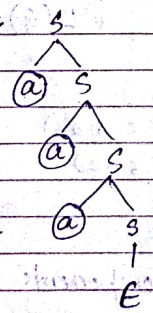
Ex: Construct the parse tree for
$S \overset{*}{\Rightarrow} aaa$ from Grammar-3

i) S

ii)


iii)


iv)


v)


Dt: 1st Sept '18

* Formal Definition Of CFG :-

Ex: $S \rightarrow aS \mid \epsilon$

Components [4 components]

i) Variables $V = \{S\}$
ii) Terminals $T = \{a, \epsilon\} = \Sigma$
iii) 2 rules.
iv) S is the start variable.

→ The CFG is a 4-tuple (V, T, P, S) where
i) V is the set of variables or Non-Terminals.
ii) T is the set of terminals, also denoted as Σ.
iii) P is the set of productions or substitution rules.
iv) S is the start variable.

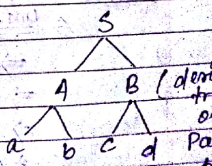
* Ex: $S \rightarrow AB \mid \epsilon$
$A \rightarrow aAb \mid ab$
$B \rightarrow CBd \mid cd$.

Soln: i) $V = \{S, A, B\}$
ii) $T = \{\epsilon, a, b, c, d\}$
iii) 6 productions.
iv) S is the start Variable.

* Language of a grammar :-

→ Let $G(V, T, P, S)$ be the CFG, then the language of Grammar G will be defined as
$L(G) = \{w \mid S \overset{*}{\Rightarrow} w\}$

Ex: i) $S \Rightarrow \epsilon$
ii) $S \Rightarrow AB \Rightarrow aBB \Rightarrow abcd$
So $S \overset{*}{\Rightarrow} abcd$.



(derivation tree or Parse tree)

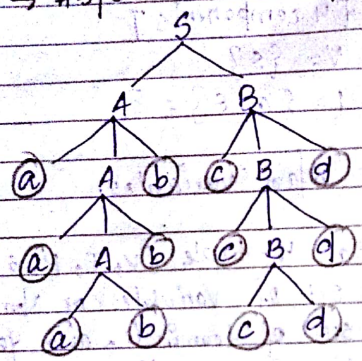
Grammar yields abcd.

DATE :

* 10) Draw the derivation tree of
$S \Rightarrow a^3b^3c^3d^3$.

$S \to AB|\epsilon \quad A \to aAb|ab \quad B \to cBd|cd$

Soln:

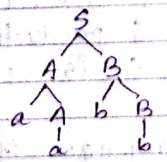


$S \to AB$
$S \to \epsilon$
$A \to aAb$
$A \to ab$
$B \to cBd$
$B \to cd$.

* Types Of Derivations :-
→ It is of 2 types :
i) Leftmost Derivation (lmd)
ii) Rightmost Derivation (rmd).

* Leftmost Derivatn :-

Ex: $S \to AB$
$A \to aA|a$
$B \to bB|b$
let $w = aabb$.



⊙ In the lmd, the leftmost variable of a string is expanded first.

lmd :

$S \xRightarrow{lmd} AB \quad (A \to aA)$
$\Rightarrow aAB \quad (A \to a)$
$\Rightarrow aaB \quad (B \to bB)$
$\Rightarrow aabB \quad (B \to b)$
$\Rightarrow aabb$.

DATE :

⊙ In the rmd, the substitutn is applied to the right most variable first in each state.

rmd :

$S \xRightarrow{rmd} AB \quad (B \to bB)$
$\Rightarrow AbB \quad (B \to b)$
$\Rightarrow Abb \quad (A \to aA)$
$\Rightarrow aAbb \quad (A \to a)$
$\Rightarrow aabb$.

* Ambiguous Grammar :-

Ex: Consider the following Grammar :
$E \to E+E \mid E*E \mid a$
Is $a+a*a$ derived from this grammar.

Soln: Leftmost derivation 1 :

$E \Rightarrow E+E$
$\Rightarrow a+E \quad (E \to E*E)$
$\Rightarrow a+E*E$
$\Rightarrow a+a*E \Rightarrow a+a*a$.

lmd 2 :

$E \Rightarrow E*E \quad (E \to E+E)$
$\Rightarrow E+E*E$
$\Rightarrow a+E*E$
$\Rightarrow a+a*E \Rightarrow a+a*a$.

→ If the grammar generates the same string by using 2 or more lmds then the string is derived ambiguously / ambiguous manner & the grammar is said to be ambiguous in nature.

Teacher's Signature

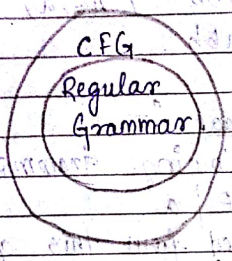
* Designing of CFG :

Imp

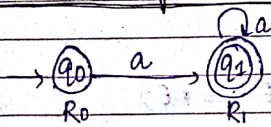
$S \to aSb \mid \epsilon \mid a \mid b$

$\{a^n b^n \mid n \geq 0\} = \{\epsilon, a, b, ab, \ldots\}$

$S \to aS \mid \epsilon \leftarrow L(G) = \{a^n \mid n \geq 0\}$



i) DFA to Regular Grammar



$\odot R_0 \to aR_1 \quad \odot R_1 \to aR_1 \mid bR_1 \mid \epsilon$

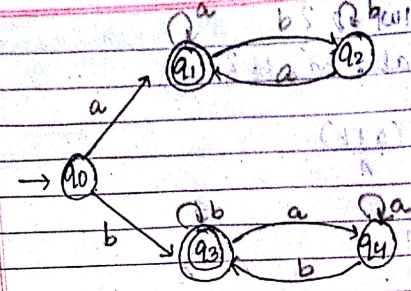
→ Steps of designing grammar from DFA:

Step 1: for every state, give a variable name.
Step 2: for the transition $\delta(q_i, a) = q_j$,
⊙ add the rule $R_i \to aR_j$.
Step 3: for each accept state q_k,
⊙ add $R_k \to \epsilon$.
Step 4: The start state of DFA will be the start variable of grammar.

P.T.O

DATE:

OR



Convert to Grammar

i) $V = \{R_0, R_1, R_2, R_3, R_4\}$
ii) $T = \{a, b, \epsilon\}$
iii) Rules
⊙ $R_0 \to aR_1$
⊙ $R_0 \to bR_3$
• $q_1 \xrightarrow{a} q_1$ ⊙ $R_1 \to aR_1$
• $q_1 \xrightarrow{b} q_2$ ⊙ $R_1 \to bR_2$
• q_1 → accept state ⊙ $R_1 \to \epsilon$
• $q_2 \xrightarrow{b} q_2$ ⊙ $R_2 \to aR_1$
• $q_2 \xrightarrow{a} q_1$ ⊙ $R_2 \to bR_2$
• q_3 → accept state ⊙ $R_3 \to bR_3 \mid aR_4 \mid \epsilon$
• $q_4 \xrightarrow{a} q_4$ ⊙ $R_4 \to aR_4 \mid bR_3$
• $q_4 \xrightarrow{b} q_3$

→ Language: $L(G) = \{$Starts & ends with same symbol$\}$.

* Language to CFG :

Imp
$\Sigma = \{a, b\}$
Ex-1: $L = \{w \mid$ length of w is at most $1\}$
$= \{\epsilon, a, b\}$

✓ Grammar
$S \to a \mid b \mid \epsilon$

Teacher's Signature

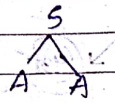
imp

Ex-2: $L = \{w \mid |w| = 2\}$
$= \{aa, ab, ba, bb\}$

$RE = \underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}$.

Grammar:

$S \to AA$
$A \to a|b$.



Ex-3: $L = \{w \mid |w| \leq 2\}$
$\underbrace{(a+b+\epsilon)}_{A}\underbrace{(a+b+\epsilon)}_{A}$

Grammar:

$S \to AA$
$A \to a|b|\epsilon$.

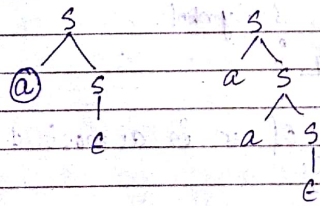
Dt: 3rd Sept' 18

imp

Ex: Design the CFG for the language:
$L = \{a^n \mid n \geq 0\}$
$= \{\epsilon, a, aa, \ldots\}$
$= a^*$

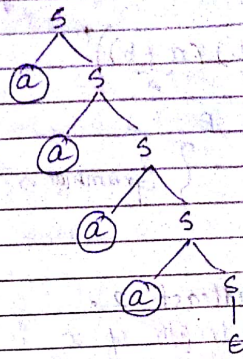
Grammar:

$S \to as|\epsilon$



Teacher's Signature

Q Draw parse tree for $S \Rightarrow aaaa$



Q $L = \{a^n b^m \mid n, m \geq 0\}$
$= \{\epsilon, a, b, aa, bb, ab, aaa \ldots\}$

a^n - A
b^m - B

Grammar:

$S \to AB$
$A \to aA|\epsilon$ — a^n
$B \to bB|\epsilon$ — b^m

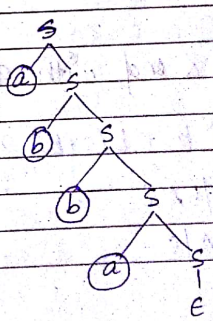
imp

Q $RE = (a+b)^*$
$L = \{w \mid w \in (a,b)^*\}$

Grammar:

$S \to as|bs|\epsilon$.

let $w = abba$



Teacher's Signature

Q. Design the CFG for the language consisting of all even length strings.

$$\underbrace{(\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A})^*}_{B}$$

$S \to BS \mid \epsilon$
$B \to AA$
$A \to a \mid b$.
} Grammar.

Q. length is atleast 2.

$L = \{w \mid \text{length of } w \text{ is } |w| \geq 2\}$

$\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}\underbrace{(a+b)^*}_{B}$

Grammar {
$S \to AAB$
$A \to a \mid b$
$B \to aB \mid bB \mid \epsilon$ — $(a+b)^*$

Q. Strings starting with a & ending with b.

$a(a+b)^*b$

Grammar {
$S \to aAb$.
$A \to aA \mid bA \mid \epsilon$.

Q. Strings start & end with diff. symbols.

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

Grammar {
$S \to aAb \mid bAa$.
$A \to aA \mid bA \mid \epsilon$

Teacher's Signature

Q. Start & end with same symbol.

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

$S \to aAa \mid bAb \mid a \mid b \mid \epsilon$
$A \to aA \mid bA \mid \epsilon$.

imp Q. $L = \{a^n b^n \mid n \geq 0\}$
$= \{\epsilon, ab, aabb, \dots\}$

$S \to aSb \mid \epsilon$

Q. $L = \{\underbrace{a^n b^n}_{A}\underbrace{c^m}_{B} \mid n, m \geq 1\}$

$S \to AB$
$A \to aAb \mid ab$ — $a^n b^n$
$B \to cB \mid c$ — c^m

Q. $L = \{\underbrace{a^n}_{A}\underbrace{b^m c^m}_{B} \mid n, m \geq 1\}$

$S \to AB$.
$A \to aA \mid a$.
$B \to bBc \mid bc$.

Q. $L = \{a^n b^n c^n \mid n \geq 1\}$

↳ It is not a CFG.

Q. $L = \{a^n b^m c^n \mid n, m \geq 1\}$

$S \to aSc \mid aAc$.
$A \to bA \mid b$

Teacher's Signature

Q) $L = \{ \underbrace{a^n b^n}_{A} \underbrace{c^m d^m}_{B} \mid n, m \geq 1 \}$

$S \rightarrow AB$
$A \rightarrow aAb \mid ab$.
$B \rightarrow cBd \mid cd$.

Q) $L = \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$.

$S \rightarrow aSd \mid aAd$.
$A \rightarrow bAc \mid bc$. $\quad A \rightarrow b^m c^m$

Q) $L = \{ a^n b^{2n} \mid n \geq 1 \}$

$S \rightarrow aSbb \mid abb$.

Q) $L = \{ a^{m+n} b^m c^n \mid n, m \geq 1 \}$.

$= a^n a^m b^m c^n$.

$S \rightarrow aSc \mid aAc$.
$A \rightarrow aAb \mid ab$.

Dt: 5th Sept '18

* Chomsky Normal Form (CNF)

$S \rightarrow AB$ $\quad S \rightarrow AB$ } not in CNF.
$A \rightarrow a$ $\quad A \rightarrow aA \mid a$ }
$B \rightarrow b$.

→ A context free Grammar (CFG) is in CNF if every rule is of the form
- ⊙ $A \rightarrow BC$
- ⊙ $A \rightarrow a$

where A, B, C → variables and a → terminal.
$S \rightarrow \epsilon$ is also permitted in CNF.
S is the start variable.

→ In CNF there is a restrict^n in the R.H.S of a rule i.e the RHS must contain 2 variables or a single terminal symbol.

* Conversion from CFG to CNF :-

1) Add the rule $S_0 \rightarrow S$ to the grammar if S (start Variable) is present in the RHS.

2) Eliminate ϵ-rule i.e, the grammar should not contain the rule of the form $A \rightarrow \epsilon$, where A is not the start variable.

3) Eliminate Unit rule i.e It shouldn't have any rule of the form $A \rightarrow B$.

4) Eliminate Useless Variables & Convert the rules to appropriate form.

P.T.O

Teacher's Signature

Imp.

* Elimination of ε-rules :

Ex: S → ABC
A → aA | ε
B → bB | ε
C → c.

i) Eliminate B → ε.

S → ABC | AC.
A → aA | ε
B → bB | b.
C → c

ii) Eliminate A → ε

S → ABC | AC | BC | C
A → aA | a
B → bB | b
C → c

By S → ABAC
A → aA | ε.
B → bB | ε
C → c.

i) Eliminate B → ε

S → ABAC | AAC.
A → aA | ε
B → bB | b
C → c.

ii) Eliminate A → ε

S → ABAC | AAC | BAC | ABC | BC | C.
A → aA | a
B → bB | b
C → c.

* Nullable Variable (NV) :

NV = {A, B}

→ The NV set consists of the variables generating ε-values.

Shortcut

S → ABAC | BAC | ABC | AAC | AC | BC | C.

By S → AB
A → aAA | ε
B → bBB | ε.

Eliminate ε-rule.

S → AB | B | A | ε (∵ S → ε is permitted).
A → aAA | aA | a
B → bBB | bB | b.

Nullable Variables = {A, B, S}.

* Procedure (To eliminate A → ε).

1. If R → aAb is a rule, then add the rule.
   R → ab.
2. If R → aABAd is a rule, then add the rule
   R → aBAd | aABd | aBd

Dt: 7th Sept '18

* Elimination of Unit rules (of the form A → B)

→ here RHS will contain only 1 variable.

Ex: Consider the following CFG.

S → XY
X → a
Y → Z | b
Z → M
M → N
N → a.

Teacher's Signature

a) Eliminate $M \to N$

$M \to N \to a$

So Add $M \to a$

$S \to XY$

$X \to a$

$Y \to Z|b$

$Z \to M$

$M \to a$

$N \to a$

b) Eliminate $Z \to M$

$Z \to M \to a$

So, Add $Z \to a$

$S \to XY$

$X \to a$

$Y \to Z/b$

$Z \to a$

$M \to a$

$N \to a$

c) Eliminate $Y \to Z$

$Y \to Z \to a$

So, Add $Y \to a$

$S \to XY$

$X \to a$

$Y \to b$

$Y \to a$

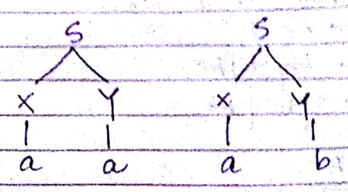
$Z \to a$

$M \to a$

$N \to a$

Teacher's Signature

* Eliminatⁿ of Useless Symbols:



$S \to XY$

$X \to a$

$Y \to a|b$

→ The variable that are not reachable from the start Variable (S) are the Useless Variable & should be eliminated.

Q) $S \to Aa|B$ Remove the unit variable

$B \to A|bb$

$A \to a|bc|B$.

i) Eliminate $A \to B$

$A \to B$ → bb, → A

Add

$A \to bb$.

So the grammar becomes,

$S \to Aa|B$

$B \to A|bb$

$A \to a|bc|bb$.

ii) Eliminate $B \to A$

$B \to A$ → a, → bc

Teacher's Signature

$B \to a$ and $B \to bc$

The grammar is now,

$S \to Aa \mid B$

$B \to bb \mid a \mid bc$

$A \to a \mid bc \mid bb$.

iii) Eliminate $S \to B$

$S \to B \to bb$
$\to a$
$\to bc$

So add $S \to a \mid bb \mid bc$.

* Convert the rules into appropriate form

$S \to Aa$

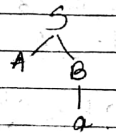
$A \to b$

In order to convert it into appropriate form :-

$S \to AB$

$B \to a$

$A \to b$.



ii) $S \to \underbrace{a}_{A}\underbrace{bbc}_{B}$

So, $S \to AB$

$A \to a$

$B \to bbc$.

Again $B \to XY$

$X \to b$

$Y \to bc$.

Again $Y \to MN$

$M \to b$

$N \to c$.

The Grammar becomes,

$S \to AB$

$A \to a$

$B \to XY$

$X \to b$

$Y \to MN$

$M \to b$

$N \to c$.

b) Convert the following CFG into CNF.

$S \to ASA \mid aB$

$A \to B \mid S$

$B \to b \mid \epsilon$.

Conversion:

Step 1 : Add $S_0 \to S$ as S is present in RHS of the rules.

$S_0 \to S$

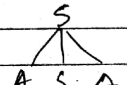
$S \to ASA \mid aB$.

$A \to B \mid S$

$B \to b \mid \epsilon$.

Step 2 : Eliminate ϵ - rules

Nullable variables = $\{B, A\}$



So the grammar becomes,

$S_0 \to S$

$S \to ASA \mid aB \mid SA \mid AS \mid S \mid a$

$A \to B \mid S$

$B \to b$.

6 Sept '18

Q) $S \to ASA \mid AS \mid SA \mid S \mid aB \mid a$

$A \to B \mid S$

$B \to b$

ii) Elimination of unit rules

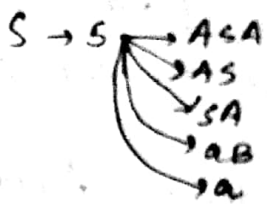
a) Eliminate $S \to S$

$S_0 \to S$

$S \to ASA \mid AS \mid SA \mid aB \mid a$

$A \to B \mid S$

$B \to b$.



b) Eliminate $S_0 \to S$

$S_0 \to ASA \mid AS \mid SA \mid aB \mid a$

$S \to ASA \mid AS \mid SA \mid aB \mid a$

$A \to B \mid S$

$B \to b$.

c) Eliminate $A \to B$

$S_0 \to ASA \mid SA \mid AS \mid aB \mid a$

$S \to ASA \mid AS \mid SA \mid aB \mid a$

$A \to S \mid b$

$B \to b$

d) Eliminate $A \to S$

$S_0 \to ASA \mid SA \mid AS \mid aB \mid a$

$S \to ASA \mid SA \mid AS \mid aB \mid a$

$A \to b \mid ASA \mid SA \mid AS \mid aB \mid a$

$B \to b$.

iii) Convert to appropriate form

$S_0 \rightarrow AX \mid SA \mid AS \mid YB \mid a$

$S \rightarrow AX \mid SA \mid AS \mid YB \mid a$

$A \rightarrow b \mid AX \mid SA \mid AS \mid YB \mid a$.

$B \rightarrow b$

$X \rightarrow SA$

$Y \rightarrow a$.

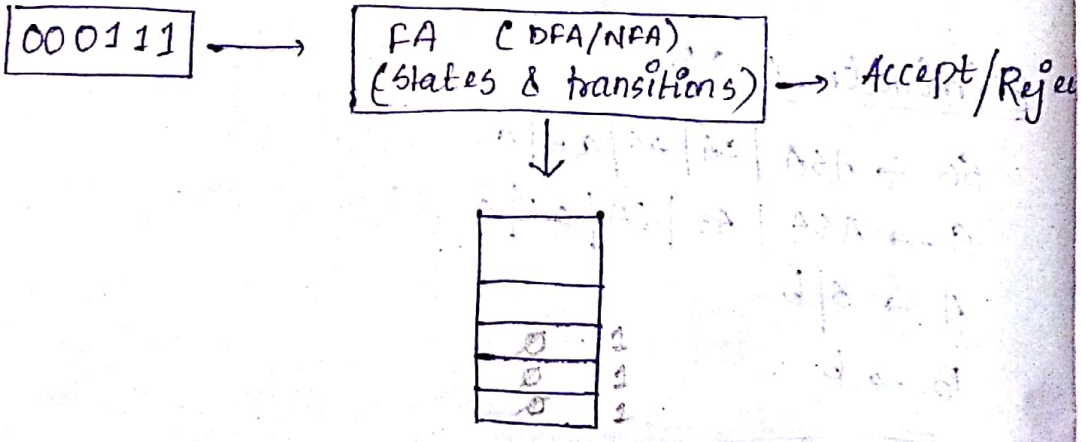
This is the CNF of the given grammar.

* Push-down Automata (PDA):

- FA recognize → Regular Language.
- PDA recognize → Context Free Language

- PDA = FA + stack.

Ex: $L = \{0^n 1^n \mid n \geq 1\}$

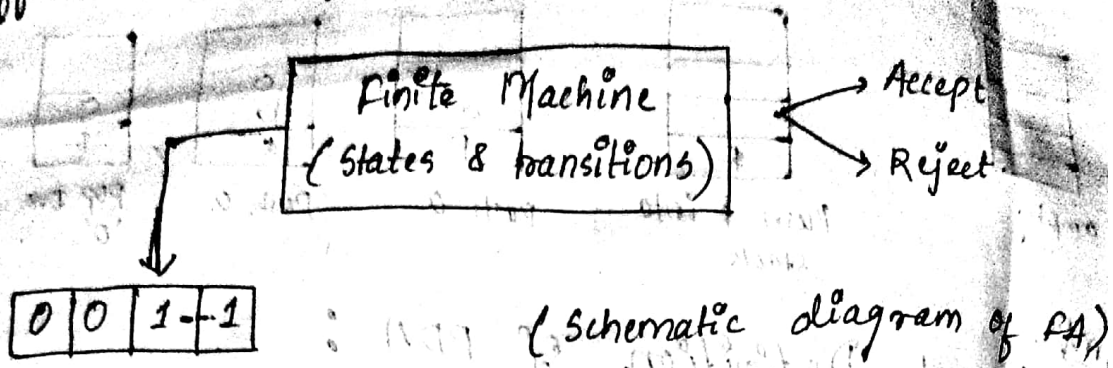


→ PDA is a computational model used to recognize Context Free Languages in the simila way FA is used to recognize regular languag

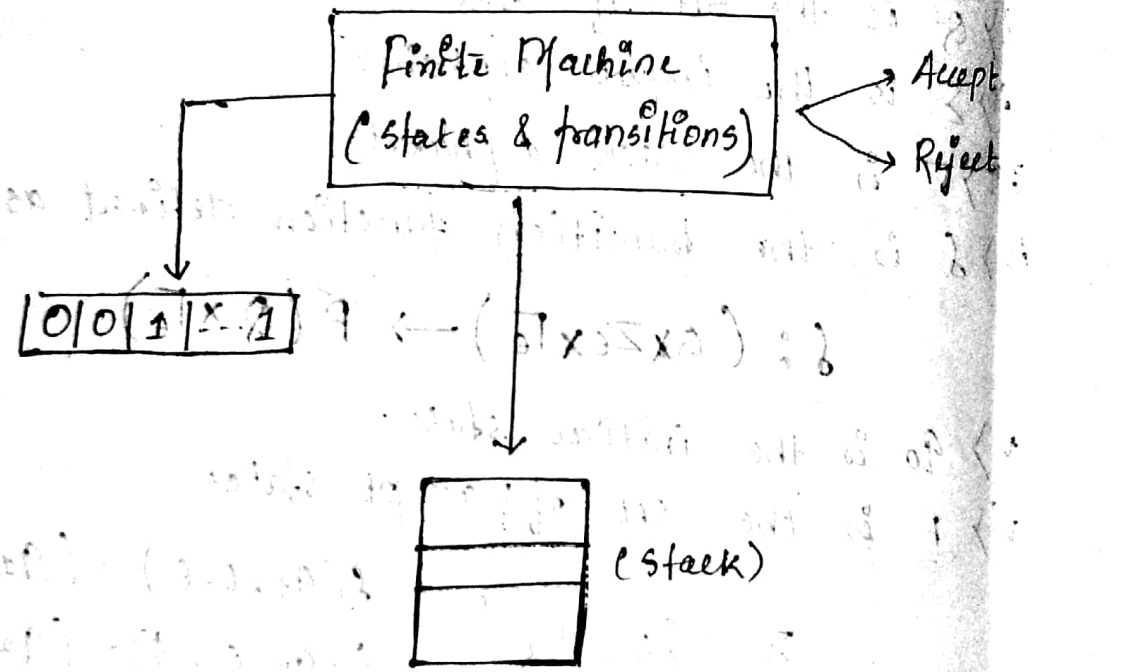
→ PDA is a NFA with an extra component call Stack.

→ Stack provides the memory to process the input string.

→ PDA is more powerful than FA as it has sufficient amt of memory.

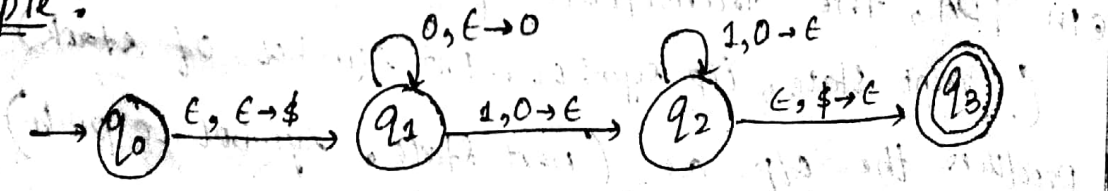


(Schematic diagram of FA)



(Schematic diagram of PDA).

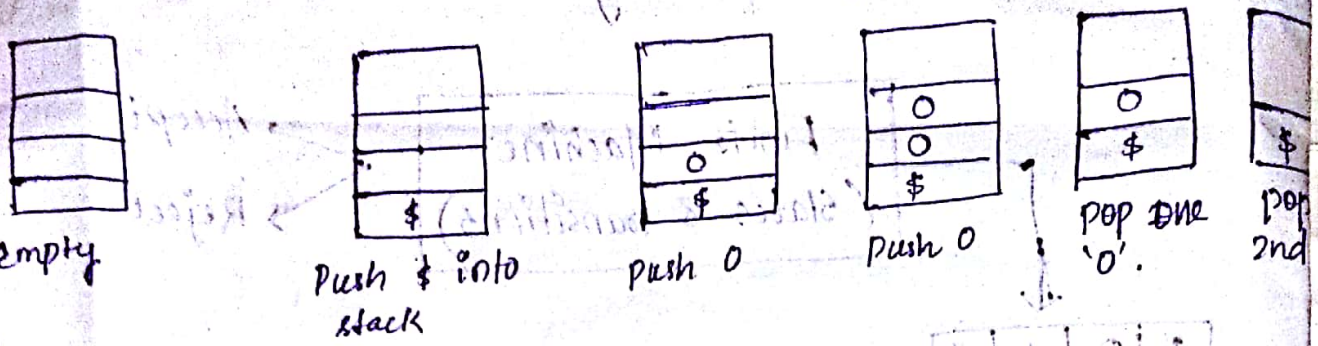
Example:



(PDA for $L = \{0^n 1^n \mid n \geq 1\}$)

a, b → c : On reading input symbol a, it pops b from the stack and push c into the stack.

$q_0 \xrightarrow{\epsilon, \epsilon \to \$} q_1 \xrightarrow{0, \epsilon \to 0} q_1 \xrightarrow{0, \epsilon \to 0} q_1 \xrightarrow{1, 0 \to \epsilon} q_2 \xrightarrow{1, 0 \to \epsilon} q$



imp.

* Formal Definition of PDA :

→ A pushdown Automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

i) Q is the set of states.
ii) Σ is the input alphabet.
iii) Γ is the stack alphabet
iv) δ is the transition function defined as

$$\delta : (Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \to P(Q \times \Gamma_\epsilon)$$

v) q_0 is the initial state.
vi) F is the set of accept states.

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

$\delta(q_1, 0, \epsilon) = (q_1, 0)$
$\delta(q_2, \epsilon, \$) = (q_3, \epsilon)$

* Transition function :

⊙ In PDA, the transition func^n takes the i/p as (current state, i/p symbol, top symbol of stack) & produces the o/p as (next state, symbol to push).

$$\delta(q, a, b) = (q_5, c)$$

Transition	i/p symbol	stack elem to pop	stack e to pu
$a, b \to c$	a	b	c
$a, \epsilon \to c$	a	ϵ	c
$a, b \to \epsilon$	a	b	ϵ
$\epsilon, b \to c$	ϵ	b	c
$\epsilon, b \to \epsilon$	ϵ	b	ϵ
$\epsilon, \epsilon \to c$	ϵ	ϵ	c
$\epsilon, \epsilon \to \epsilon$	ϵ	ϵ	ϵ

* Components of PDA M_1 :

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

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

iii) $\Gamma = \{0, \$\}$

iv) $\delta : (Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \to P(Q \times \Gamma_\epsilon)$

v) q_0 is the initial state

vi) $F = \{q_0, q_3\}$ is the set of accept states

: The transitions of PDA M_1 are

i) $\delta(q_0, \epsilon, \epsilon) = \{(q_1, \$)\}$

ii) $\delta(q_1, 0, \epsilon) = \{(q_1, 0)\}$

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

iv) $\delta(q_2, 1, 0) = \{(q_2, \epsilon)\}$

v) $\delta(q_2, \epsilon, \$) = \{(q_3, \epsilon)\}$

* Transition Table :

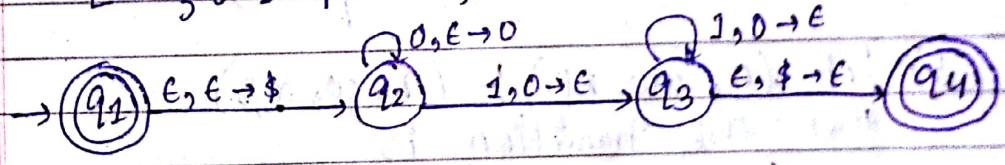
Input Symbol	0			1			ϵ		
Stack Symbol	0	$	ϵ	0	$	ϵ	0	$	ϵ
→* q_0									$\{(q_1,\$)\}$
q_1			$\{(q_1,0)\}$	$\{(q_2,\epsilon)\}$					
q_2				$\{(q_2,\epsilon)\}$			$\{(q_3,\epsilon)\}$		
q_3									

The blank cells have ϕ values.

DL: 12th Sept' 18

* Processing String :

$L = \{0^n 1^n \mid n \geq 0\}$



- let i/p string $w = 0011$.

$(q_1, 0011, \epsilon)$ – configuration.

⊢ : turnstyle notation.

$(q_2, 0011, \$)$

⊢

$(q_2, 011, 0\$)$

⊢

$(q_2, 11, 00\$)$

Teacher's Signature

$\vdash$
$(q_3, 1, 0\$)$

$\vdash$
$(q_3, \epsilon, \$)$

$\vdash$
$(q_4, \epsilon, \epsilon)$

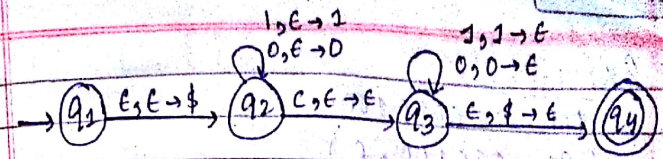
→ During string processing in PDA, it goes from configuration to configuration in response to the i/p symbol.
→ The PDA configuration consists of 3 parts:
i) State (q)
ii) remaining part of the string (w)
iii) stack contents (Y)
→ So, the configuration is a triplet (q, w, Y).
→ The triplet is also k/a ID of the PDA.
→ ID (Instantaneous Description).

Q) Let $(q, aw, X\beta) \vdash (r, w, \alpha\beta)$ then the transition is
$\delta(q, a, X) = (r, \alpha)$
X is popped & α is pushed.

* Designing of PDA:

Ex-1: let $L = \{ wcw^R \mid w \in \{0,1\}^* \}$
$= \{0c0, 01c10, 00c00, \ldots\}$
↳ palindrome.

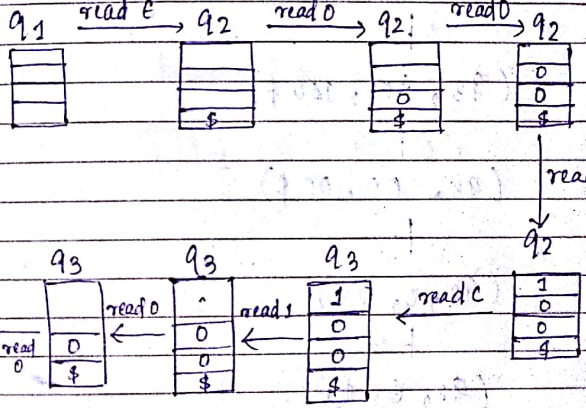
P.70



* Design steps:

i) Initially, push $ to the stack.
ii) If it reads 0 or 1, then push it to the stack.
iii) If it reads symbol c, then change the state & do nothing.
iv) for each read operation of 0 & 1; pop 0 & 1 respectively.
v) pop the $ symbol & accept the string.

Let $w = 001c100$



P.70

Teacher's Signature

* Execution

Let $w = 001c100$

$(q_1, 001c100, \epsilon)$

$\vdash$

$(q_2, 001c100, \$)$

$\vdash$

$(q_2, 001c100, 0\$)$

$\vdash$

$(q_2, 1c100, 00\$)$

$\vdash$

$(q_2, c100, 100\$)$

$\vdash$

$(q_3, 100, 100\$)$

$\vdash$

$(q_3, 00, 00\$)$

$\vdash$

$(q_3, 0, 0\$)$

$\vdash$

$(q_3, \epsilon, \$)$

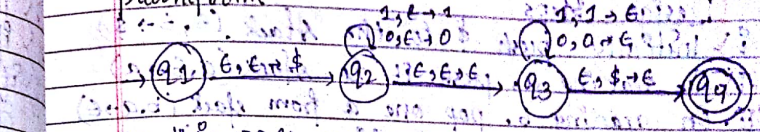
$\vdash$

$(q_4, \epsilon, \epsilon)$.

P.T.O.

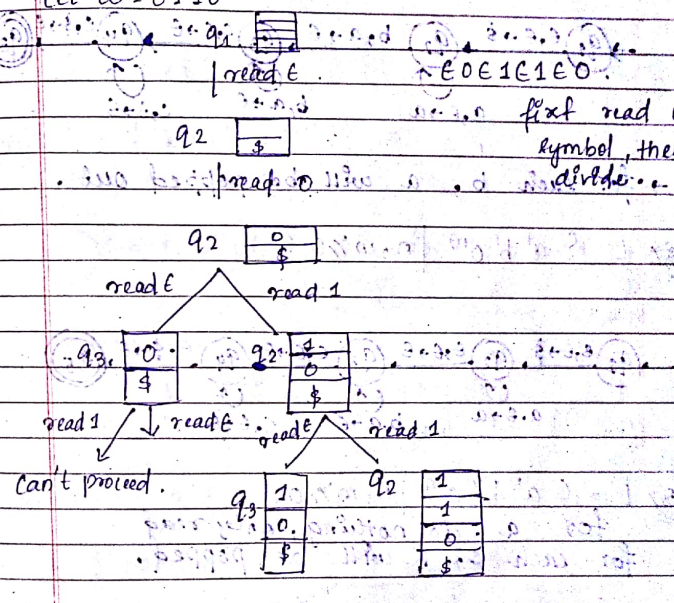
Q) Let $L = \{ ww^R \mid w \in \{0,1\}^* \}$

even length palindrome $= \{0110, 1001, 1111, \ldots\}$



→ In this PDA, the middle pos. of the string is non-deterministically determined (using $\epsilon \to$ move).

Let $w = 0110$.



Dt: 14th Sept'18

* Design of PDA :-

Q) $L = \{ a^n b^n c^m \mid n, m \geq 1 \}$

= Design steps

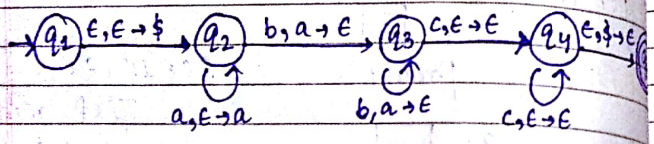
i) Initially push $ to the stack. $(\epsilon, \epsilon \to \$)$

ii) On reading a, push it to stack $(a, \epsilon \to a)$

iii) On reading b, pop one a from stack $(b, a \to \epsilon)$

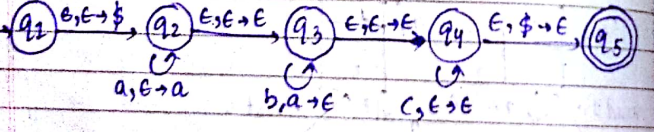
iv) On reading c, do nothing $(c, \epsilon \to \epsilon)$

v) pop $ from stack & accept $(\epsilon, \$ \to \epsilon)$



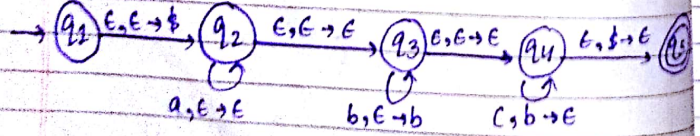
→ for each b, a will be popped out.

Q) $L = \{ a^n b^n c^m \mid n, m \geq 0 \}$



Q) $L = \{ a^n b^m c^m \mid n, m \geq 0 \}$

for a, do nothing, only read
for each c, b will be popped.



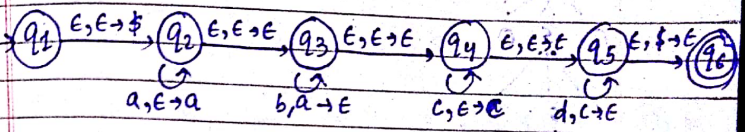
Teacher's Signature

PAGE NO.
DATE: / /20

Q) $L = \{ a^n b^n c^n \mid n \geq 0 \}$

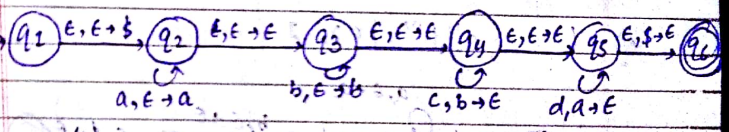
It is not a context free language, so no PDA.

Q) $L = \{ a^m b^m c^n d^n \mid n, m \geq 0 \}$



for each b, pop a.
for each d, pop c.

Q) $L = \{ a^m b^n c^n d^m \mid n, m \geq 0 \}$

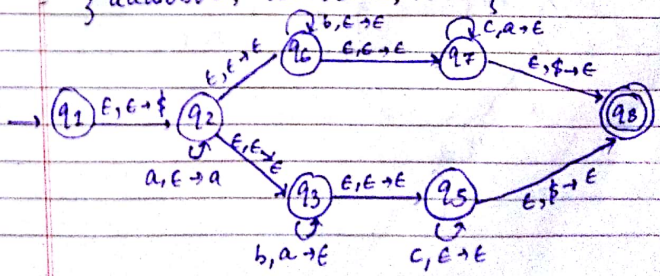


Q) $L = \{ a^n b^m c^n d^m \mid n, m \geq 0 \}$

It is not a CFL. so no PDA.

Q) $L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k \}$

$= \{ aaabbbc, aaabbccc, \ldots \}$



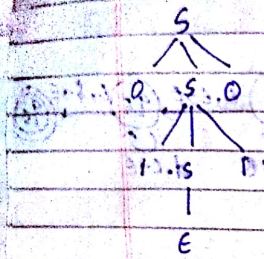
Teacher's Signature

* CFG for palindrome :-

Even length palindrome

$S \to 0S0 \mid 1S1 \mid \epsilon$

$L = \{\epsilon, 00, 11, 0110, \ldots\}$

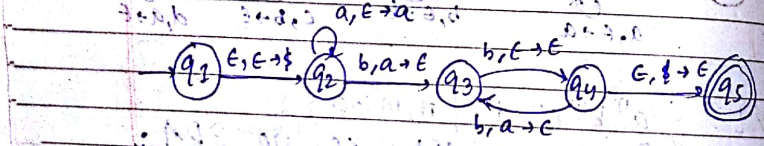


Odd length palindrome :

$S \to 0S0 \mid 1S1 \mid 0 \mid 1$

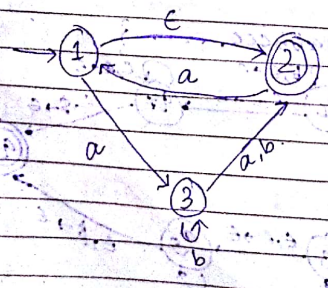
$L = \{0, 1, 000, \ldots\}$

Q) $L = \{a^n b^{2n} \mid n \geq 1\}$



Dt: 15th Sept '18

Q) Convert the NFA to DFA.



Teacher's Sig...

Sol^n :- Transition Table

δ	a	b	ε
→ 1	3	ϕ	2
* 2	1	ϕ	ϕ
3	2	{2,3}	ϕ

Transition Table (DFA).

Start state of DFA = $E(\{1\}) = \{1, 2\}$

δDFA	a	b
*→ [1,2]	{1,2,3}	ϕ
* [1,2,3]	{1,2,3}	{2,3}
ϕ	ϕ	ϕ
* [2,3]	{1,2}	{2,3}

① $\delta([1,2], a) = E(3,1) = \{1,2,3\}$

② $\delta([1,2], b) = E(\phi) = \phi$

③ $\delta([2,3], a) = E(1,2) = \{1,2\}$

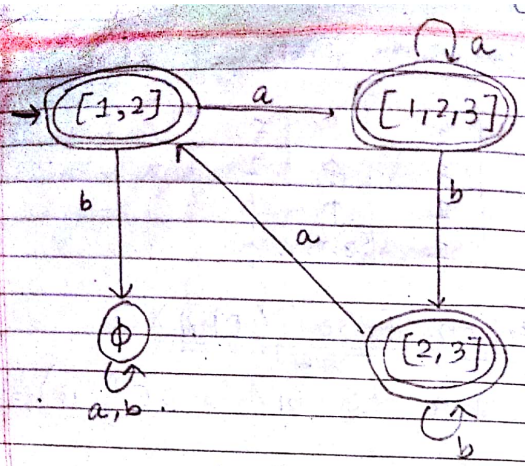
④ $\delta([2,3], b) = E(2,3) = \{2,3\}$

⑤ $\delta([1,2,3], a) = E(3,1,2) = \{1,2,3\}$

⑥ $\delta([1,2,3], b) = E(3,2) = \{2,3\}$

P T

Teacher's Signature



* Equivalence of PDA with CFG :

Theorem: A language is Context Free iff some pushdown automata recognizes it.

So, it has 2 parts :-

i) If some language is context free, then there is some PDA that recognizes it.

[CFG → PDA]

ii) If some PDA recognizes some language, then it is context free.

[PDA → CFG]

i) Convert CFG to PDA :

Consider the following CFG.

$S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \epsilon$.

p.T.O

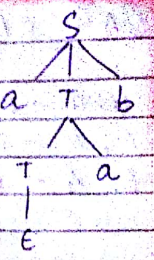
Teacher's Signature

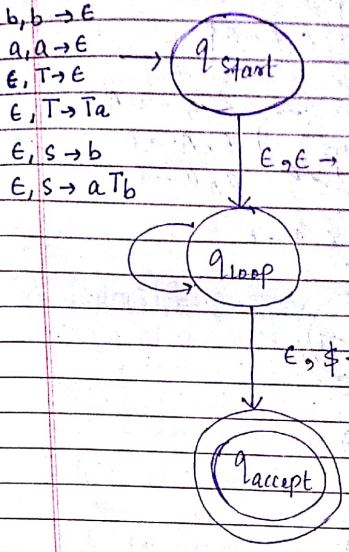
PAGE NO.
DATE : / /

Ex: $S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \epsilon$.

$L = \{b, ab, aab, \ldots\}$





$\overset{A}{S} \rightarrow \overset{\omega}{aTb}$

$\epsilon, \underset{A}{S} \rightarrow \underset{\omega}{aTb}$

i) for each production $A \rightarrow \omega$, add the transition $\epsilon, A \rightarrow \omega$.

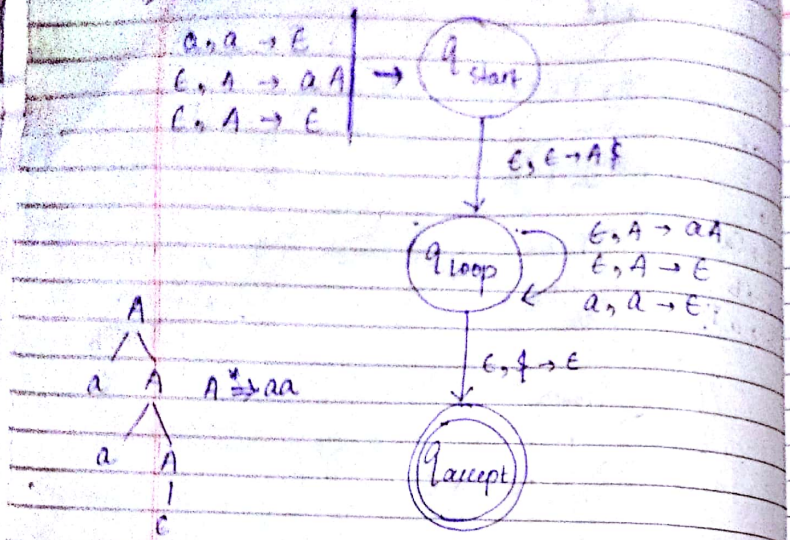
ii) for each terminal symbol a, add the transition $a, a \rightarrow \epsilon$.

Ex: $A \rightarrow aA \mid \epsilon$.

p.T.O

Teacher's Signature

Q) $A \to aA \mid \epsilon$ — A → Start state



$A \Rightarrow aa$

$(q_{start}, aa, \epsilon)$
↓
$(q_{loop}, aa, A\$)$
↓
$(q_{loop}, aa, aA\$)$
↓
$(q_{loop}, a, A\$)$
↓
$(q_{loop}, a, aA\$)$
↓
$(q_{loop}, \epsilon, A\$)$
↓
$(q_{loop}, \epsilon, \$)$
↓
$(q_{accept}, \epsilon, \epsilon)$

* Construct of PDA :-

→ Let $P(Q, \Sigma, \Gamma, \delta, q_0, F)$ be the PDA

i) $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup E$

where E is the addition states reqd. for implementing Short-hand representation.

ii) $q_0 = q_{start}$

iii) $F = \{q_{accept}\}$

iv) Transitions

a) Initially add $\epsilon, \epsilon \to S\$$ to q_{start} i.e.,

$$\delta(q_{start}, \epsilon, \epsilon) = (q_{loop}, S\$)$$

b) for each production $A \to w$, where w is string, then add $\epsilon, A \to w$ in q_{loop}, i.e.,

$$\delta(q_{loop}, \epsilon, A) = (q_{loop}, w)$$

c) for each terminal symbol a, add $a, a \to \epsilon$ in q_{loop}.

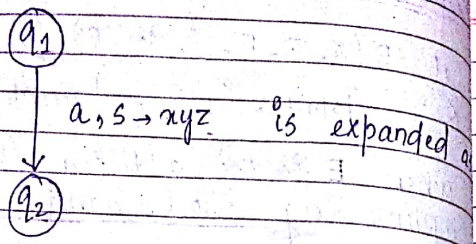
i.e $\delta(q_{loop}, a, a) = (q_{loop}, \epsilon)$

d) finally, add $\epsilon, \$ \to \epsilon$ in q_{loop} i.e

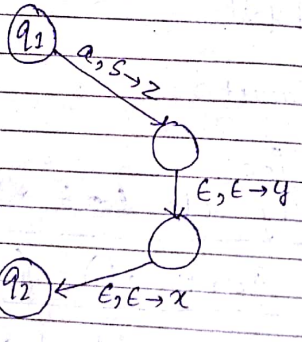
$$\delta(q_{loop}, \epsilon, \$) = (q_{accept}, \epsilon)$$

Teacher's Signature

* Shorthand representation :

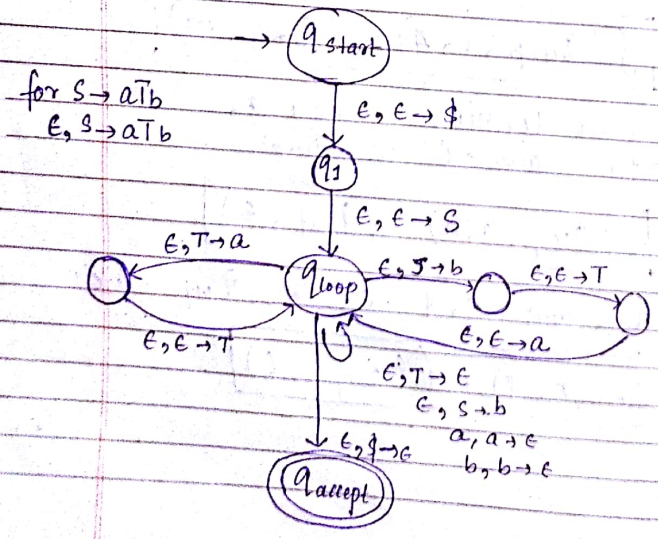


x
y
z



(expanded form)

* Without Shorthand form :

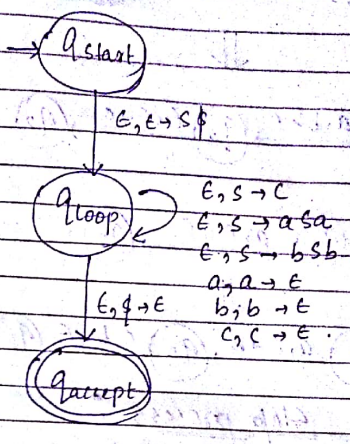


for $S \to aTb$
$\epsilon, S \to aTb$

Teacher's Signature

PAGE NO.
DATE : / / 20

Q) $S \to aSa \mid bSb \mid c$

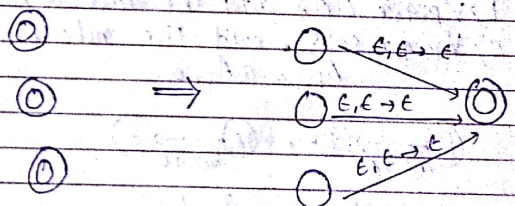


Dt : 17th Sept'18

ii) Convert PDA to CFG :

→ The given PDA should satisfy following cond^n :

a) A single accept state.

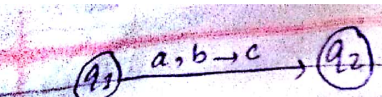


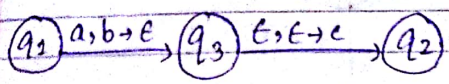
b) The stack should be empty accepting the string i.e it starts with an empty stack & finishes with an empty stack.

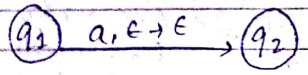
c) Each transition is either a push operation or a pop operation.

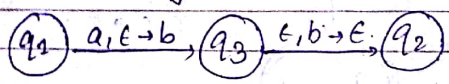
pg no

Teacher's Signature









* Two Step process :

i) Simplify the PDA.

ii) Build the CFG.

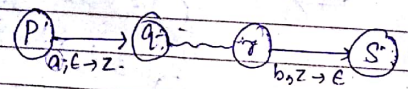
Let $P(Q, \Sigma, \Gamma, \delta, q_0, F)$ be the given PDA.

a) $\forall p \in Q$, add the rule $A_{pp} \to \epsilon$.

b) $\forall p, q, r \in Q$, add the rule $A_{pr} = A_{pq} A_{qr}$.

c) $\forall p, q, r, s \in Q$, add the rule.

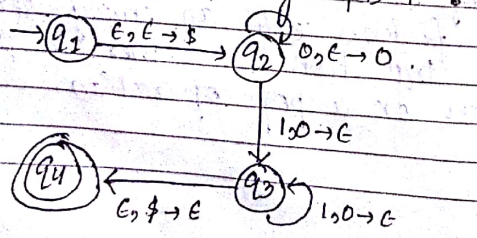
$A_{ps} = a A_{qr} b$.



$A_{ps} \to a A_{qr} b$.

d) Start variable will be $A_{q_{start} q_{accept}}$.

Q) Consider following PDA :



Sol: Its equivalent CFG :

$A_{q_1 q_2} \to \epsilon$

$A_{q_2 q_2} \to \epsilon$

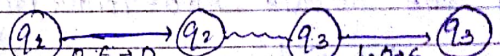
$A_{q_3 q_3} \to \epsilon$

$A_{q_4 q_4} \to \epsilon$

$A_{q_2 q_1} \to A_{q_2 q_2} A_{q_2 q_3} \mid A_{q_2 q_3} A_{q_3 q_3}$

$A_{q_1 q_2} \to$

$A_{q_2 q_3} \to$



$A_{q_2 q_3} \to 0 A_{q_2 q_3} 1$

Start state = $A_{q_1 q_4}$. (Add rule $S \to A_{q_1 q_4}$)

* Non-Context Free Language :-

① Regular Grammar generates, Regular Language

↓ recognized by

Finite Automata (DFA/NFA)

② CFG ——→ Context Free Language

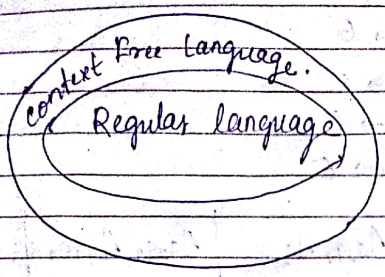
↓ recognized by

Push down Automata (PDA)

→ Every regular language is a context free language. | False |.

PAGE NO.
DATE: / 128

$L = \{0^n 1^n \mid n \geq 0\}$ → Eg of context free lang. i.e not regular



→ The language that is not recognised by any pDA is K/a Non-Context Free language.

$L = \{0^n 1^n \mid n \geq 0\}$

Non-CFL - $L = \{a^n b^n c^n \mid n \geq 0\}$

$\{a^m b^m c^n d^m \mid n, m \geq 0\}$.

Dt: 19th sept' 18

* Pumping lemma for Context free language:

→ It states that every CFL has a special value called the pumping length, such that the strings of length at least pumping length can be pumped.

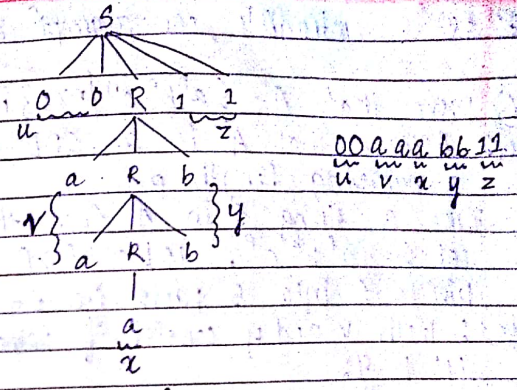
→ It means that the string can be divided into 5-parts such that the second & fourth part can be repeated any no. of times and the resulting string still is in the language.

$S \rightarrow 00R11$

$R \rightarrow aRb/a/b$

P.T.O

PAGE NO.
DATE: / 129



* Pumping lemma:

→ If A is a CFL then there is a number P (pumping length), where if s is any string in A of length at least P, then s can be divided into 5 parts, $s = uvxyz$ satisfying the following cond^n:

i) for each $i \geq 0$, $uv^i xy^i z \in A$.

ii) $|vy| > 0$.

iii) $|vxy| \leq p$.

imp

* Steps to prove that a language is non-CFL using pumping lemma (proof by contradiction):

Step 1: Assume that A is a context free language.

2: According to pumping lemma, it has a pumping length p.

3: Take a string s in A such that $|s| \geq P$.

4: Divide s into five parts.

5: Show that for some i, $uv^i xy^i z \notin A$.

6: Consider the diff. ways of dividing string & satisfy all of the cond^{n}s. (such that it satisfy)

7: If any of the cond^n is not satisfied then s cannot be pumped.

Teacher's Signature

PAGE NO.
DATE :

∴ it contradicts our assumptn & its Non-C

Imp Q. Prove that $A = \{a^n b^n c^n \mid n \geq 0\}$ is non-CFL.

Proof: Assume that A is a CFL and it has a pumping length p.

Take a string S in A such that $|s| \geq p$.

let $S = a^p b^p c^p$, $|s| = 3p \geq p$.

Divide S into 5 parts i.e $S = uvxyz$.

Case 1: Both v and y consist of same symbol.

Let $p = 4$,

$S = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{b}_{x}\underbrace{bb}_{y}\underbrace{bbcccc}_{z}$

for $i = 2$, $uv^2xy^2z = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{aa}_{v}\underbrace{b}_{x}\underbrace{bb}_{y}\underbrace{bb}_{y}\underbrace{bcc}_{z}$

$= a^6b^6c^4 \notin A$.

Case 2: Either v or y contain different type of symbol.

Let $p = 4$, $S = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{bb}_{x}\underbrace{bb}_{y}\underbrace{cccc}_{z}$

for $i = 2$, $uv^2xy^2z = \underbrace{aaa}_{u}\underbrace{ab}_{v}\underbrace{ab}_{v}\underbrace{b}_{x}\underbrace{bb}_{y}\underbrace{bb}_{y}\underbrace{cccc}_{z}$

$= a^4bab^4c^4 \notin A$.

None of these cases satisfy 1st condn of pumping lemma. So it contradicts our assumptn. Hence A is not a CFL.

Q. Prove that $B = \{ww \mid w \in \{0,1\}^*\}$ is not a CFL using pumping lemma.

Proof: Assume that B is a CFL.

So it has a pumping length p acc. to pumping lemma.

PAGE NO.
DATE :

Take a string S in A with that $|s| \geq p$

Let $S = 0^p1^p0^p1^p$, $|s| = 4p \geq p$.

$S = uvxyz$.

Case 1: vxy does not straddle a boundary.

Let $p = 5$.

$\underbrace{00000}_{u}|\underbrace{11111}_{v}|\underbrace{00000}_{xy}|\underbrace{11111}_{z}$

for $i = 1$,

$uv^2xy^2z = 0^5\underbrace{11}_{v}\underbrace{11}_{v}\underbrace{1}_{x}\underbrace{1111}_{yy}\underbrace{0^5}_{z}1^5$

$= 0^5 1^9 0^5 1^5 \notin B$.

Case 2: vxy straddles the first boundary.

$\underbrace{00000}_{u}|\underbrace{1}_{v}|\underbrace{11}_{x}\underbrace{11}_{y}\underbrace{00000}_{z}11111$.

for $i = 2$,

uv^2xy^2z

$= 000\underbrace{00}_{v}\underbrace{00}_{v}1\underbrace{11}_{y}\underbrace{11}_{y}11\,00000\,11111$

$= 0^7 1^7 0^5 1^5 \notin B$.

Case 3: vxy straddles the mid-point.

$p = 5$,

$00000\,1\underbrace{1}_{v}|\underbrace{0}_{x}\underbrace{00}_{y}00\,11111$

for $i = 2$, uv^2xy^2z

$= 0^5 1^3 10 1 0^5 0^5 0^5 11111 \notin B$.

So none of these cases satisfy 1st condn of pumping lemma so it contradicts our assumptn i.e v is a non-CFL.

P.T.O.

Teacher's Signature

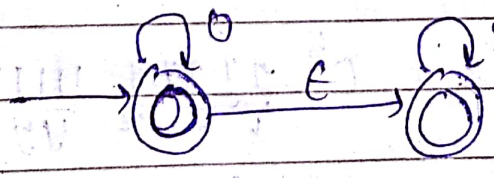
PAGE NO.
DATE : / /20

Q) prove that $L = \{w \mid w$ has equal no. of 0s & 1s$\}$ is not a regular language.

$L \cap 0^*1^* = 0^n1^n$

$\{0011, 0101, 1010, 1100\} \cap \{\epsilon, 0, 1, 01, 0011, 001, 011, 0111, 0001, \ldots\} = 0011$

$= 0^2 1^2$



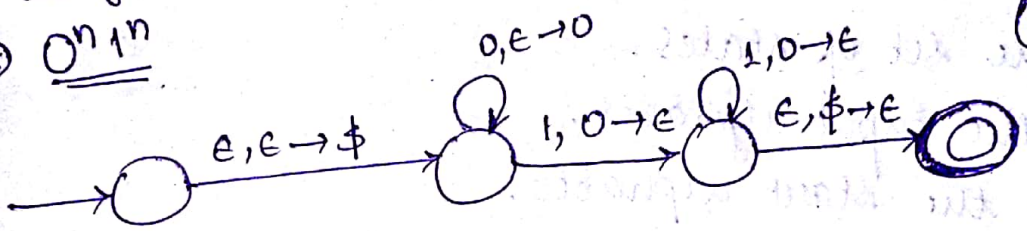
Dt: 22nd Sept'18

22/9/18 Deterministic PDA

PDA

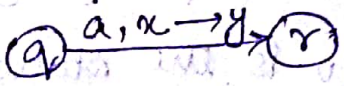
- Deterministic PDA (DPDA) — every transition there is a defined next state.
- Non-deterministic PDA (NPDA)

(*) $0^n 1^n$ (DPDA)

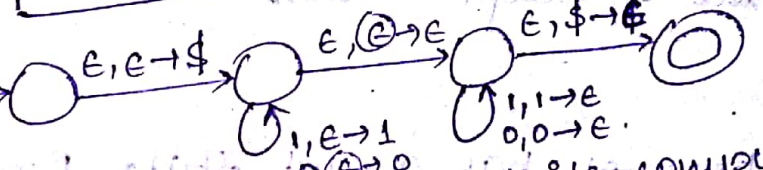


(*) $\delta(q,a,x)$ has a single move or ϕ

i.e., $\delta(q,a,x) = \{(r,y)\ \text{or}\ \phi\}$



(NPDA) WWR.



(*) in DPDA, at each step of its computation it has atmost one way to proceed for every transition.

(*) δ (state, input symbol, stack top element) has a single move

ε-moves are allowed in DPDA.

(*) ε- ~~stack~~ moves are of 2 types:

(i) ε-input move → $\delta(q,\varepsilon,x)$

(ii) ε-stack move → $\delta(q,a,\varepsilon)$

⊛ In DPDA either of

$\delta(q,a,x)$, $\delta(q,\epsilon,x)$, $\delta(q,a,\epsilon)$ $\delta(q,\epsilon,\epsilon)$

has a next state.

Formal Definition.

The DPDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

i) Q is the set of states.

ii) Σ is the input alphabet.

iii) Γ is the stack alphabet.

iv) $\delta : (Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \longrightarrow (Q \times \Gamma_\epsilon) \cup \{\phi\}$

v) q_0 is the start state.

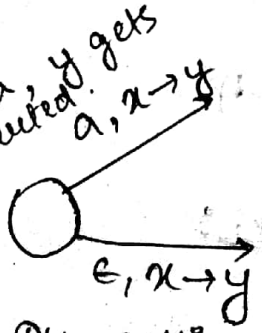
vi) F is the set of accept states.

~~In DPDA either of~~ has a next state for each

$q \in Q$, $a \in \Sigma_\epsilon$, $x \in \Gamma_\epsilon$ $\quad |\delta(q,a,x)| = 1$.

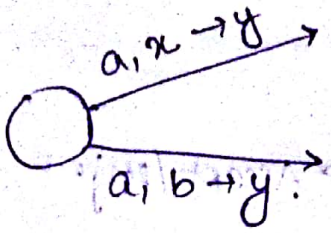
exactly one of $\delta(q,a,x)$, $\delta(q,\epsilon,x)$, $\delta(q,a,\epsilon)$, $\delta(q,\epsilon,\epsilon)$ is not ϕ.

e.g



on getting ϵ, y gets inserted.

not possible in DPDA - but possible in NPDA.

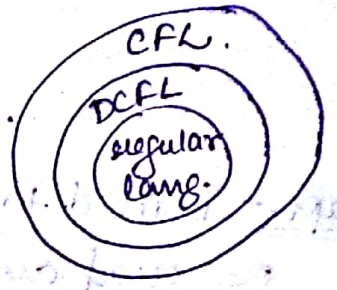
eg. 

allowed, on reading a goes in that direction, on reading b goes in that stack direction.

* The languages determined by deterministic PDA ~~the DPDA~~ is known as deterministic context free languages (DCFL).

* NPDA more powerful than DPDA.

DPDA and NPDA not ~~eq~~ equal (same).



* DPDA – $L = \{0^n 1^n \mid n \geq 0\}$
NPDA – $L = \{wwR \mid w \in \{0,1\}^*\}$.

DCFL are closed under complementation operation.

* Turing Machine :-

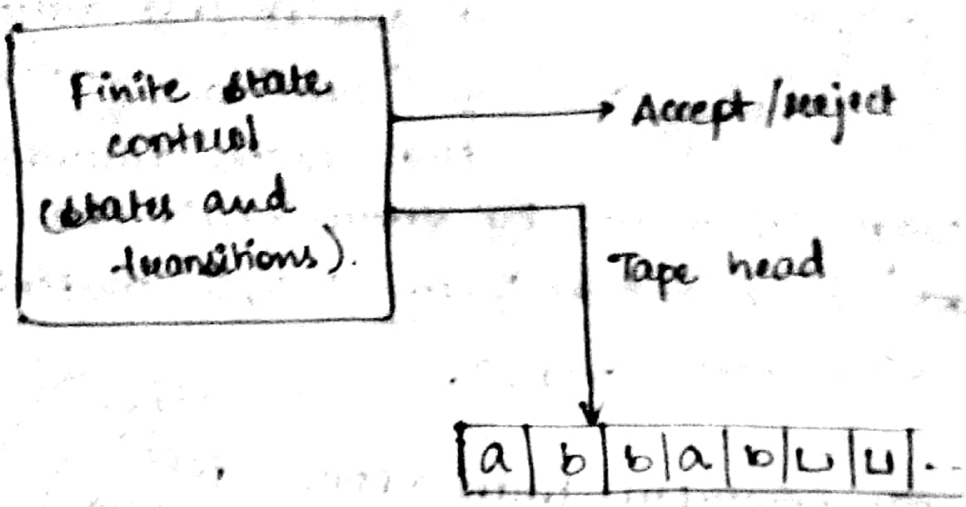
Computational Models.

i) Finite Automata – states and transition with limited memory.
(DFA, NFA)

ii) Pushdown Automata – states and transitions with unlimited but restrictive memory (stack).

iii) Turing machine – states and transitions with unlimited and unrestrictive memory (tape).

* Turing machine is a computational model that consists of states and transitions along with unlimited and unrestrictive memory (tape).



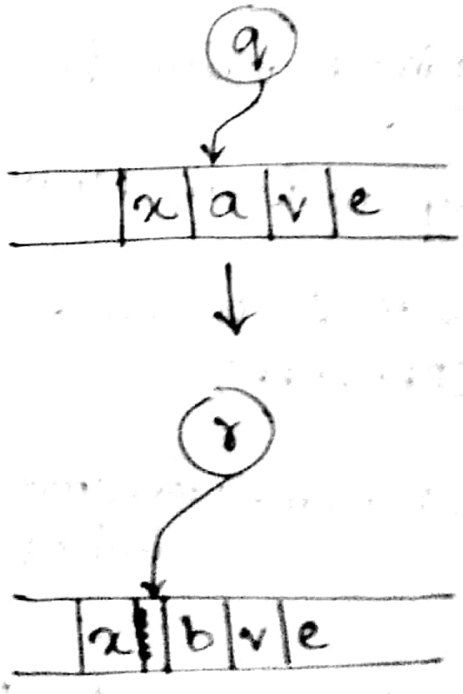
tape - is used to store i/p strings one read and write the i/p symbols in the same place.

The Turing machine has a designated accept and reject state, so the output of a turing machine can be

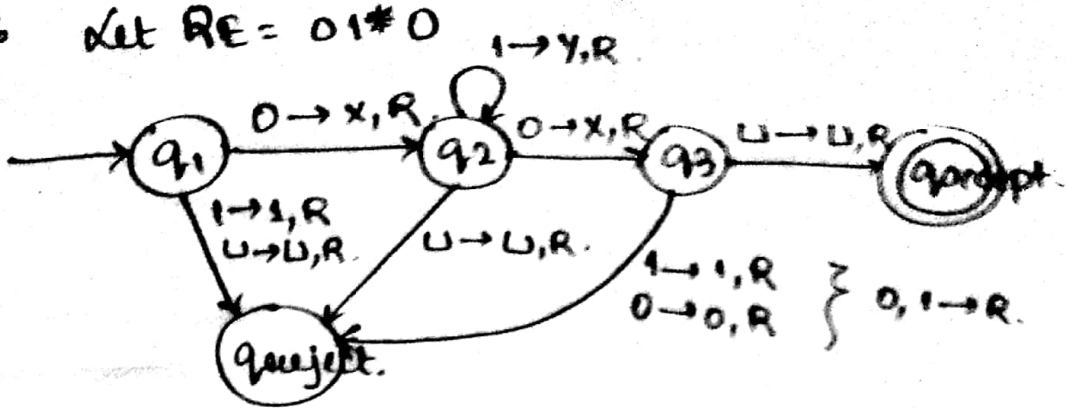
(i) Accept & halt
(ii) Reject & halt
(iii) loop.

Transitions

$$\delta(q, a) = (r, b, L)$$



Ex: Let RE = 01*0



Formal Definition

A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where.

(i) Q is the set of states

(ii) Σ is the input alphabet.

(iii) Γ is the tape alphabet that includes $\sqcup$.

(iv) δ is the transition function defined as.

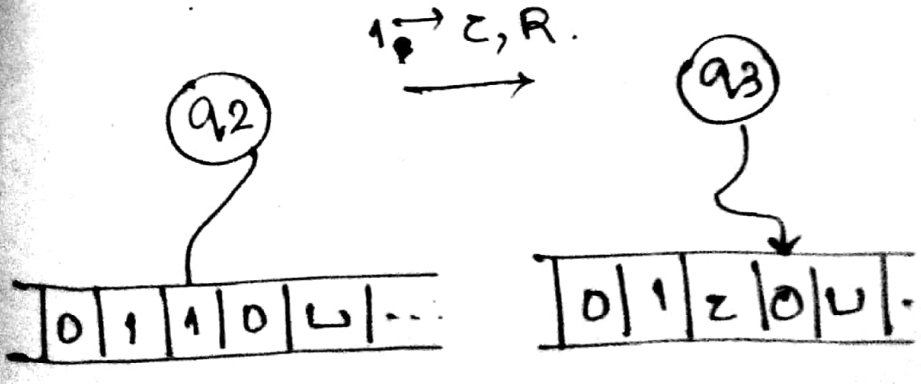
$$\delta, Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}.$$

(v) q_0 is the ~~state~~ start state

(vi) q_{accept} → accept state

(vii) q_{reject} → reject state.

$q_{accept} \neq q_{reject}$.

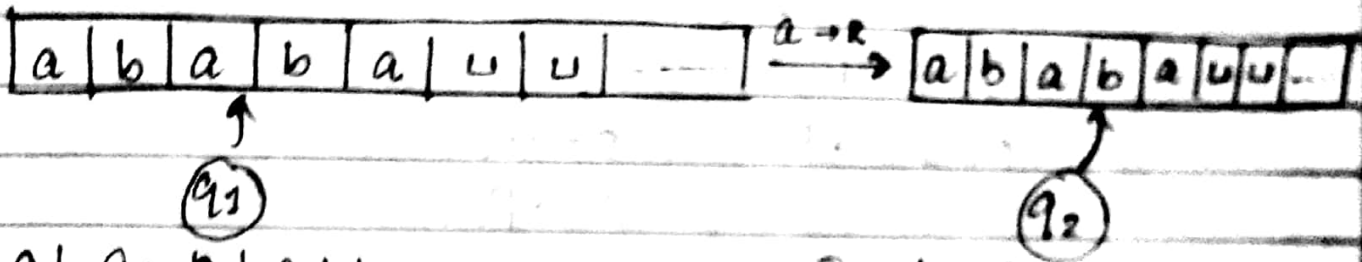


$$\delta(q_2, 1) = (q_3, z, R).$$

Dt: 24th sept '18

* Configuration :

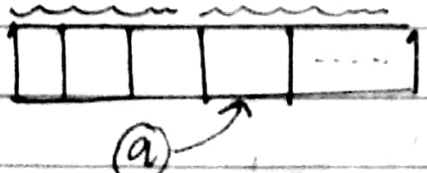
␣ → blank



① ab q_1 aba␣ ② aba q_2 ba␣

String = a b a b a

→ If the configuratⁿ of a turing machine (TM) is expressed as [uqv], then it is shown as :



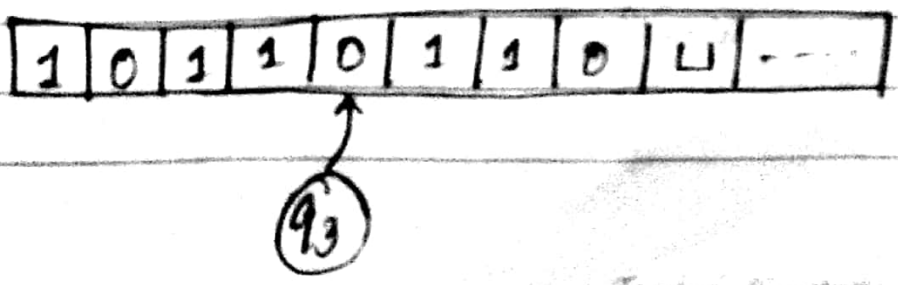
→ The configuratⁿ of a TM w.r.t an I/P string that consists of 3 components :

i) current state
ii) Current tape contents.
iii) current tape head position.

Q) Let the configuratⁿ of a TM is :

1011 q_3 0110␣.

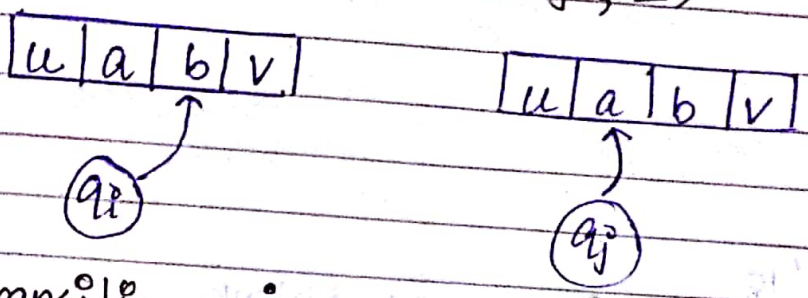
ans→



Teacher's Signature

Ex: Let ua_iq_ibv generates uaq_jabv, then the transition is represented as

$$\delta(q_i, b) = (q_j, L)$$



Transitions :

$0 \rightarrow X, R$

$0 \rightarrow R$

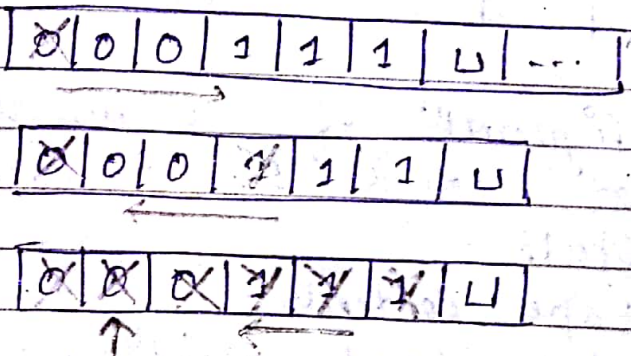
$0, 1 \rightarrow R$ $\quad 0 \rightarrow R$

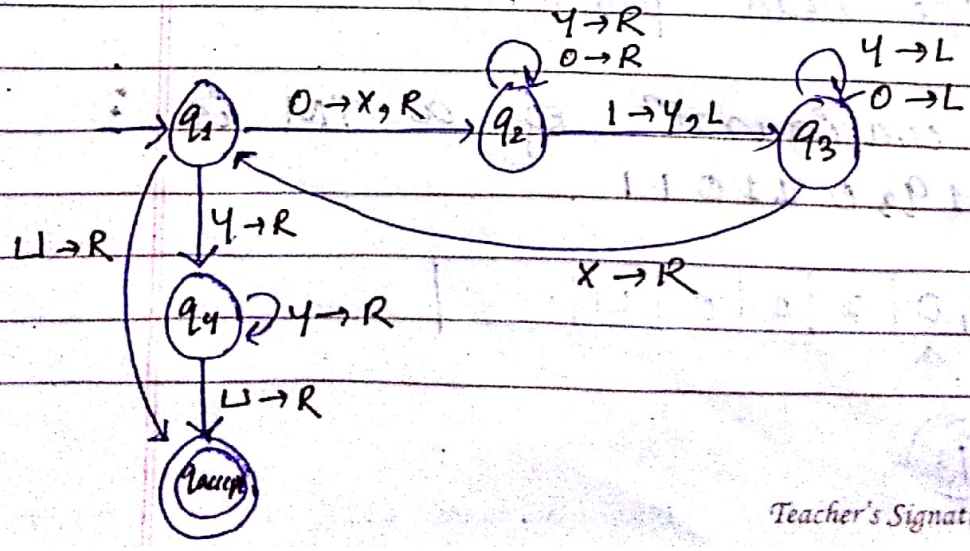
$1 \rightarrow R$

Ex: Design a Turing Machine that recognizes the language, $L = \{0^n 1^n \mid n \geq 0\}$

Design steps

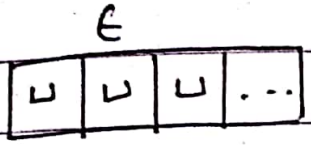
$L = 0^3 1^3$





Teacher's Signature

PAGE NO.
DATE : / /20



* STEPS :

1. Read the first 0 & replace with x.
2. Read the symbol 0 and y, move right to get the first 1.
3. Replace 1 with Y & move left.
4. Read the symbol 0 & y to move left to get the symbol X.
5. Move right & repeat from step 1.
6. If Symbol Y comes, then move right to get the blank (⊔) & accept the string.