11/7/18 1. Basic Concepts & Finite Automata

Page No.
Date 11/7/18

* Symbols :-

Symbols are user defined entities.
For eg :- letters (A-Z, a-z), digits (0-9), special characters such as +, -, /, % etc.

* Alphabets (Σ) :-

Alphabets are finite set of symbols.
For eg :- $\Sigma = \{A, B, C, D\}$
$\Sigma = \{0, 1, 2 \ldots\}$

* Strings :-

Strings are various combinations of symbols.
For eg :- ab, ba, 01, 0011, abbc, bba, ...

* Length of a string :-

Length of a string counts the position of the symbols in a string.
Strings with length '0' are called as empty strings & are denoted by ϵ.

~~* Power of a alphabet :-~~

* Power of an alphabet :-

Power of an alphabet, Σ^k contains or ~~is~~ a set of strings of length k.
For eg :- $\Sigma = \{a, b\}$ $\quad \Sigma^0 = \epsilon$
$\Sigma^1 = \{a, b\}$
$\Sigma^2 = \{aa, ab, ba\}$
$\Sigma^3 = \{aba, bba, abb\}$

16/7/18

* Closure of an alphabet (Σ^*) :-
It is a finite set of all the strings over an alphabet
for eg :- $\Sigma^* = \{ \epsilon, a, b, ab, ba, bb, aa, abb, aba, abba, \ldots \}$

* Positive closure of an alphabet (Σ^+) :-
It is the finite set of non-empty strings
For eg :- $\Sigma^+ = \{a, b, ab, ba, bb, abb, aba, abba\}$

Hence, $\boxed{\Sigma^* = \epsilon \cup \Sigma^+}$

Σ^* → Closure → set of all the strings

Σ^+ → Positive closure → set of non-empty sets.

* Language (L) :-
It is a subset of Σ^*, set of strings all of which are choosen from Σ^* where Σ is a ~~af~~ alphabet
for eg :- Set of all the strings of even numbers over $\Sigma = \{0, 1, 2, \ldots, 9\}$
$L = \{0, 2, 4, 6, 8\}$

16/7/18

* Finite State Machine (FSM) :-

A FSM consists of finite set of states (S) which alters on receiving the input set (I) to produce output set (O).

FSM defines two functions :-

1) State Function (STF) = $S \times I \rightarrow S$

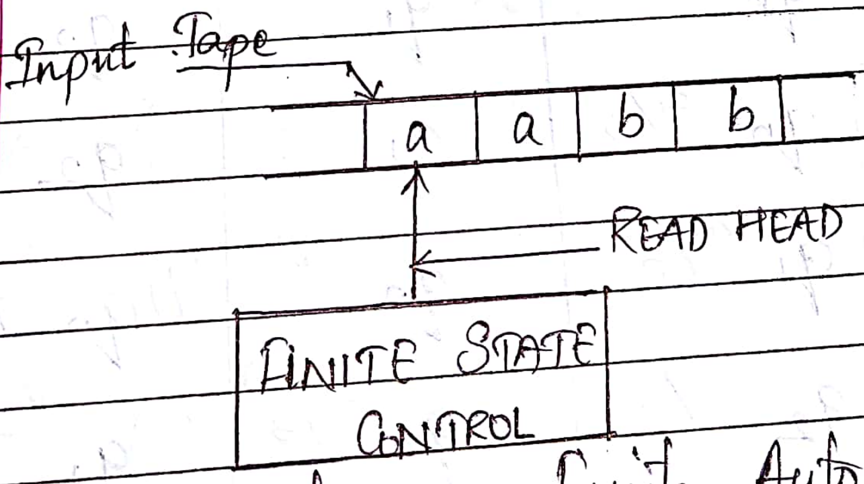
2) Machine Function (MTF) = $S \times I \rightarrow O$

* Finite Automata :-

A finite automata is considered to be a mathematical model of a machine.

* Components of Finite Automata :-

1) Finite set of input (input tape) 2) finite set state & a read head.



Model of an Finite Automata

* Working of FA :- Depending on the state & the input symbol.

1) FA can either change the state or remain in the same state.

2) FA moves the read head to the right of the current cell.

Q1. Design an FA to check whether a given decimal number is divisible by 3.

Step 1 → Theory :- defination of FA

Step 2 → logic :-

$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$O = \{Y, N\}$

$S = \{q_s, q_0, q_1, q_2\}$

Step 3 → Implimentation State Function

	S\I	{0, 3, 6, 9}	{1, 4, 7}	{2, 5, 8}
start state →	q_s	(Y) q_0	(N) q_1	(N) q_2
3	rem. $0 \to q_0^*$	(Y) q_0 30/3	(N) q_1 34/3	(N) q_2 38/3
4	$1 \to q_1$	(N) q_1 46/3	(N) q_2 41/3	(Y) q_0 45/3
5	$2 \to q_2$	(N) q_2 53/3	(Y) q_0 54/3	(N) q_1 52/3

$STF = S \times I \to S$

Step 4 →

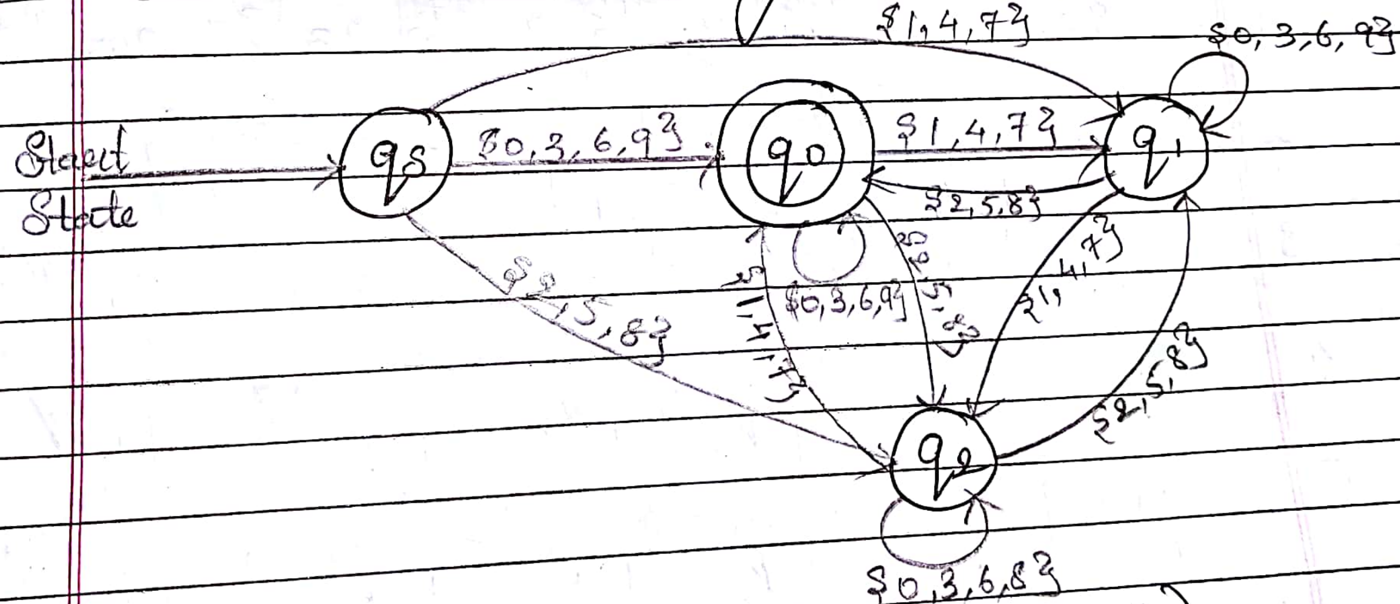
Step

Step 4 → Machine Function

S \ I	{0,3,6,9}	{1,4,7}	{2,5,8}
q_s	Y	N	N
q_0	Y	N	N
q_1	N	N	Y
q_2	N	Y	N

$$MTF = S \times I \to O$$

Step 5 → State Transition Diagram :



eg:- ~~$(q_s, 376)$~~
$(q_0, 76)$
$(q_1, 6)$
(q_1, ϵ)

$(q_s, 372)$
$(q_0, 72)$
$(q_1, 2)$
$(q_0, \epsilon) \to$ accepted

H.W
Design an FA to check whether a given decimal no. is divisible by 5.

Step 1 → Def:-

Step 2 → Logic:-

$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$O = \{Y, N\}$
$\delta = \{q_s, q_0, q_1, q_2, q_3, q_4\}$

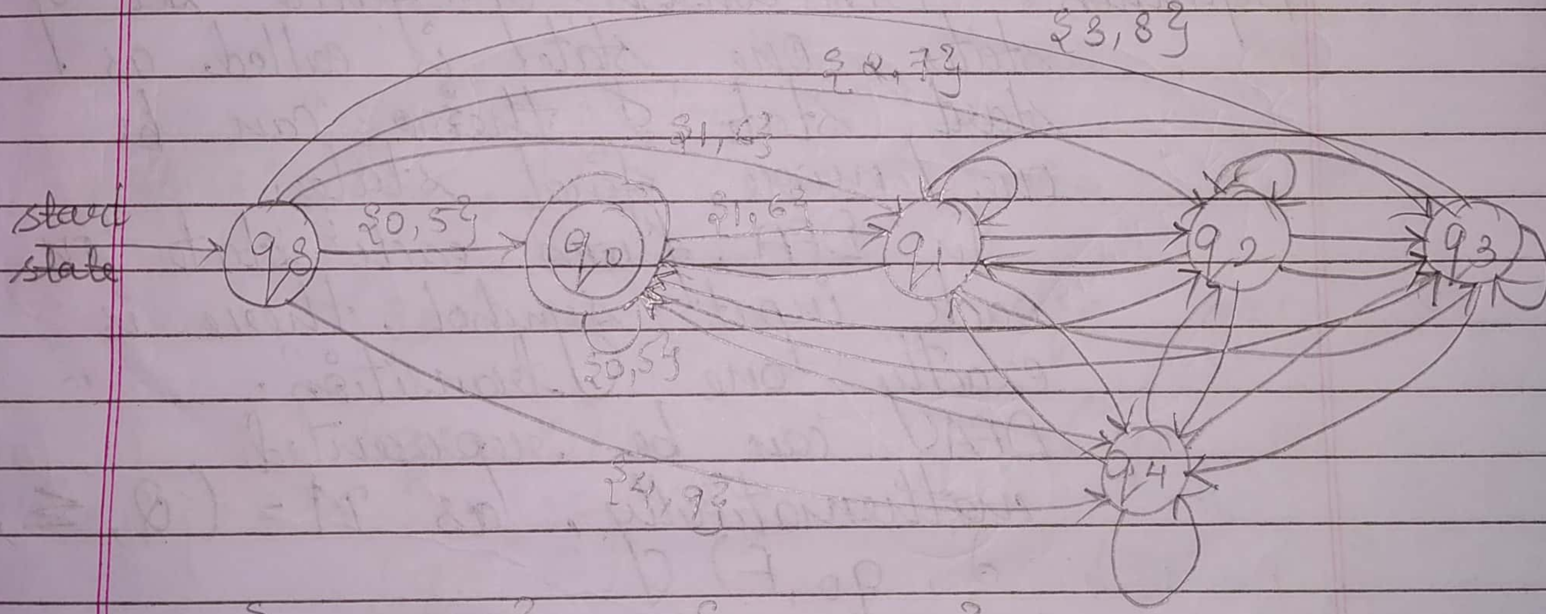
Step 3 → Implementation

	$\delta \backslash I$	{0, 5}	{1, 6}	{2, 7}	{3, 8}	{4, 9}
start state →	q_s	q_0	q_1	q_2	q_3	q_4
1	q_0^*	q_0 10/5	q_1 11/5	q_2 17/5	q_3 13/5	q_4 14/5
2	q_1	q_0 25/5	q_1 21/5	q_2 27/5	q_3 28/5	q_4 29/5
3	q_2	q_0 35/5	q_1 36/5	q_2 32/5	q_3 33/5	q_4 34/5
4	q_3	q_0 40/5	q_1 46/5	q_2 42/5	q_3 48/5	q_4 44/5
5	q_4	q_0 55/5	q_1 56/5	q_2 52/5	q_3 58/5	q_4 59/5

$STF = \delta \times I \longrightarrow \delta$

$\delta \setminus \Sigma$	{0, 5}	{1, 6}	{2, 7}	{3, 8}	{4, 9}
q_s	Y	N	N	N	N
q_0	Y	N	N	N	N
q_1	Y	N	N	N	N
q_2	Y	N	N	N	N
q_3	Y	N	N	N	N
q_4	Y	N	N	N	N

$MTP = \delta \times \Sigma \rightarrow 0$



{q_s, 479}   {q_s, 570}

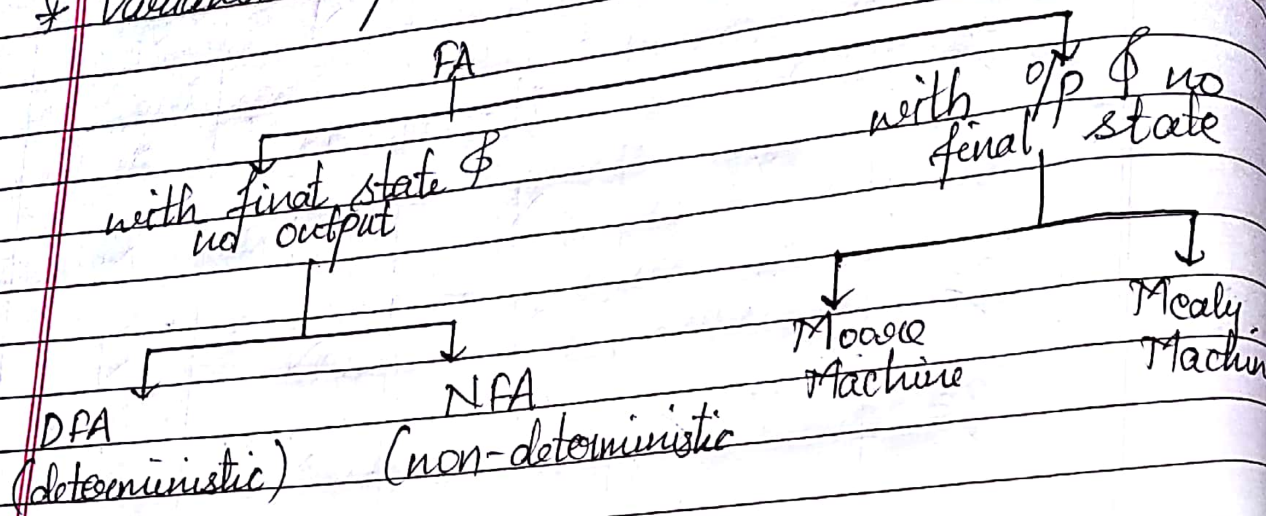
{q_4, 79}   {q_0, 70}

{q_2, 9}   {q_2, 0}

{q_4, ϵ}   {q_0, ϵ} → accepted

17/7/18

* Variations of Finite Automata (FA):-



+ Deterministic FA:-

Definition:- DFA consists of finite set of states, one state is called as start state & there can be one / more final states.

In DFA, from each state on each input symbol, there is exactly one transition.

DFA can be represented mathematically as $M = (Q, \Sigma, \delta, q_0, F)$

where,

Q = finite set of states.

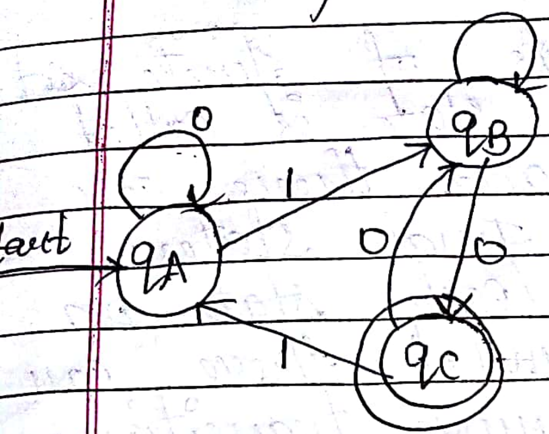
Σ = input alphabet.

δ = transition function

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

q_0 = start state $q_0 \in Q$

F = finite set of final state $F \subseteq Q$



$Q = \{q_A, q_B, q_C\}$

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

$q_0 = q_A$ $F = \{q_C\}$

δ :-

$Q \backslash \Sigma$	0	1
q_A	q_A	q_B
q_B	q_C	q_B
q_C	q_B	q_A

Ex-1 :- $(q_A, 110)$

$(q_B, 10)$

$(q_B, 0)$

(q_C, ϵ)

Accept

Ex-2 :- $(q_A, 1010)$

$(q_B, 010)$

$(q_C, 10)$

$(q_A, 0)$

(q_A, ϵ)

Reject.

Date

* Non-Deterministic Finite Automata :-

Definition :- NFA consists of finite set of states, one state is called start state and there can be one / more final states.

In NFA, from each state on each input symbol there can be 0, 1 or more transition.

NFA can be represented mathematically,

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

where,

Q = finite set of states

Σ = input alphabet

δ = transition function

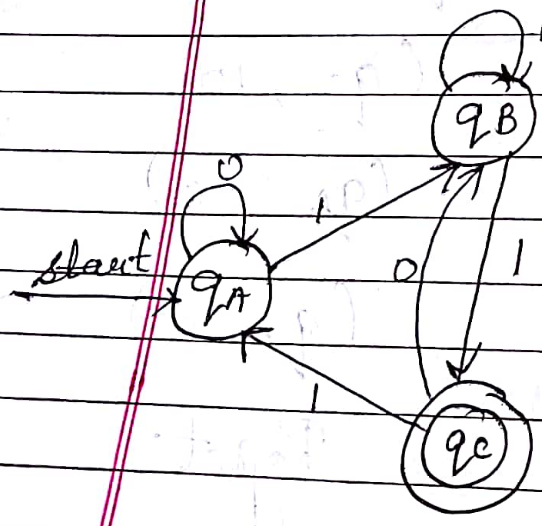
$\delta :- Q \times \Sigma \rightarrow 2^Q$

q_0 = start state $q_0 \in Q$

F = finite set of final state

$F \subseteq Q$

imp



$Q = \{q_A, q_B, q_C\}$ $q_0 = q_A$

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

δ :-

Q \ Σ	0	1
q_A	q_A	q_B
q_B	$\{\}$	$\{q_B, q_C\}$
q_C	q_B	q_A

Ex-3:- $(q_A, 111)$

$(q_B, 11)$

$(q_B, 1)$ $(q_C, 1)$

(q_B, ϵ) (q_C, ϵ) Accept (q_A, ϵ)

Ex-4:- $(q_A, 1000)$

$(q_B, 000)$

Reject.

imp

* Deterministic FA	Non-deterministic FA
1) In DFA, from each state on each i/p symbol there is exactly one transition.	- In NFA, from each state on each i/p symbol there can be 0 or more transition.
2) In DFA, the transition function is defined as $\delta :- Q \times \Sigma \rightarrow Q$	- In NFA, the transition function is defined as $\delta :- Q \times \Sigma \rightarrow 2^Q$
3) The implementation of DFA with the help of ~~DFA~~ ~~with~~ computer program is simple.	- The implementation of NFA with the help of computer program is difficult because its non-deterministic nature.
4) Every DFA is always an NFA.	- An NFA may or may not be DFA.
5) It is not a possibilistic machine.	- It is a possibilistic machine.

DFA → Divisibility check, String ending, String starting, String containing

Prob: Construct an FA to check whether a binary no. divisible by 3.

Step 1 → def of FA

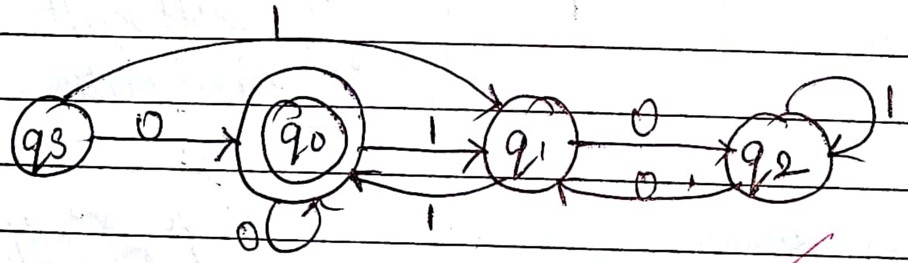
Step 2 → Logic :-

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

$O = \{Y, N\}$

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

	Q \ Σ	0	1		
Start state →	q_s	q_0	q_1	0/3	1/3
3	q_0^*	q_0	q_1	00110/3	0111/3
4	q_1	q_2	q_0	01000/3	01001/3
5	q_2	q_1	q_2	1010/3	1011/3



$(q_s, 1100)$
$(q_1, 100)$
$(q_0, 00)$
$(q_0, 0)$
(q_0)

Q \ Σ	0	1
q_s	Y	N
q_0	Y	N
q_1	Y	N
q_2	N	Y

Page No.
Date

Sum Design a DFA in which the input is valid if it ends in "100" over $\Sigma = \{0, 1\}$

Step1 → def of DFA

$\underline{\underline{100}}$

Step2 → Logic :-

$I = \{0, 1\}$

$O = \{Y, N\}$

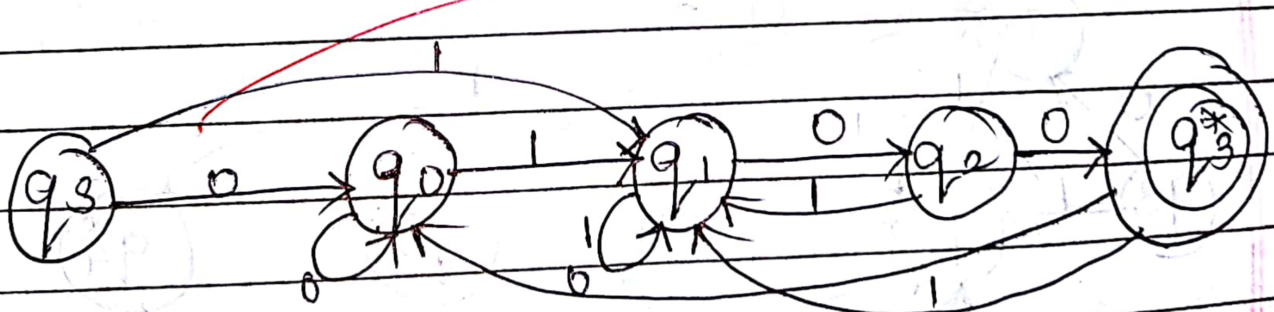
$Q = \{q_s, q_0, q_1, q_2, q_3\}$

Ends in	$Q \backslash \Sigma$	0	1
→	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3^*	q_1
100	q_3^*	q_0	q_1

0	1
~~0~~0	~~1~~0
10	~~1~~1
100	~~10~~1
~~100~~0	~~100~~1

$STF = S \times I \to S$

$Q \backslash \Sigma$	0	1
q_s	N	N
q_0	N	N
q_1	N	N
q_2	Y	N
q_3	N	N



0100

Date

H.W * Design a DFA to check whether if the i/p is valid if it ends in "AAB" over $\Sigma = \{A, B\}$

* Design a DFA in which the i/p is valid if it ends either in "100" or in '101' over $\Sigma = \{0, 1\}$

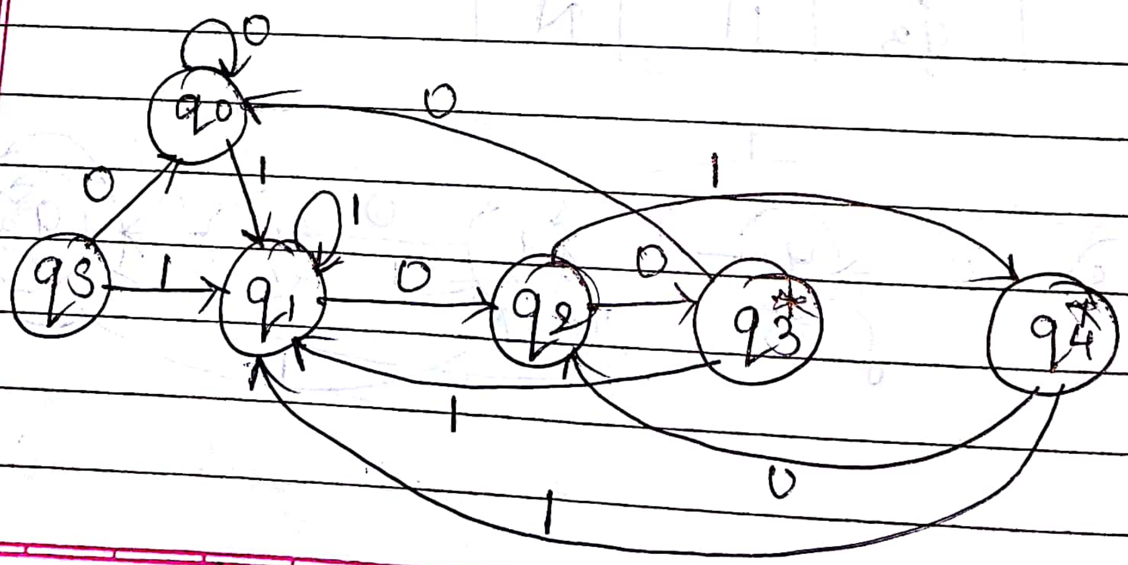
→ $\Sigma = \{0, 1\}$ $O = \{Y, N\}$ $Q = \{q_s, q_0, q_1, q_2, q_3 \dots$

100 101

	Q\Σ	0	1
	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3	q_4
100	q_3*	q_0	q_1
101	q_4*	q_2	q_1

0	1
~~00~~	~~01~~
10	~~11~~
100	101
~~1000~~	1001
~~1010~~	~~1011~~

Q\Σ	0	1
q_s	N	N
q_0	N	N
q_1	N	N
q_2	Y	Y
q_3	N	N
q_4	N	N



Page No. | Date

* Design a DFA to check whether it is valid if 2nd last symbol is "a" over $\Sigma = \{a, b\}$

→

$I = \{a, b\}$

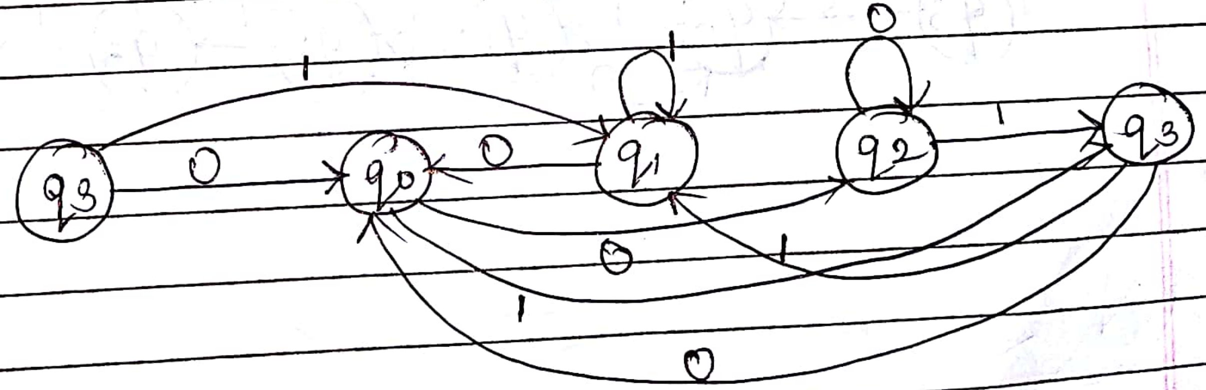
$O = \{Y, N\}$

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

	$Q \backslash \Sigma$	a	b
	q_s	q_0	q_1
a	q_0	q_2^*	q_3^*
b	q_1	q_0	q_1
aa	q_2^*	q_2^*	q_3^*
ab	q_3^*	q_0	q_1

a	b
aa	ab
ba	bb
~~x~~aa	~~x~~ab
~~xb~~a	~~xb~~b

$Q \backslash \Sigma$	a	b
q_s	N	N
q_0	Y	Y
q_1	N	N
q_2	Y	Y
q_3	N	N



* Design a DFA to check if it is valid if it contains "1011" over $\Sigma = \{0, 1\}$

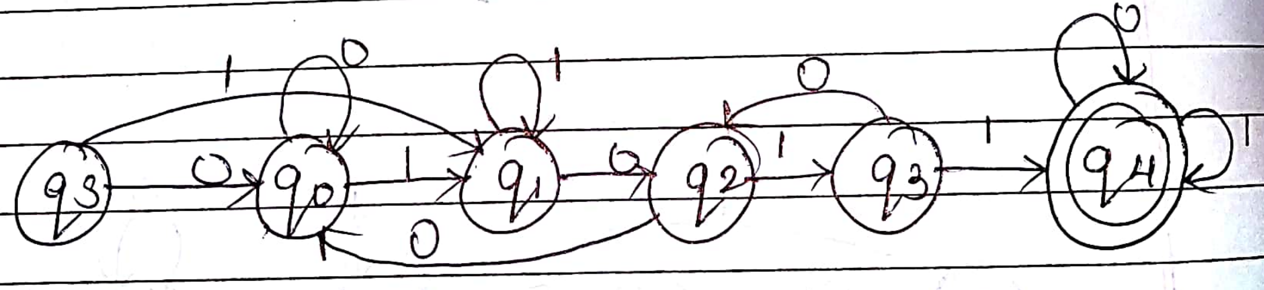
$I = \{0, 1\}$
$O = \{Y, N\}$
$Q = \{q_s, q_0, q_1, q_2, q_3, q_4\}$

→ 1011

	$Q \backslash \Sigma$	0	1
	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_0	q_3
101	q_3	q_2	q_4^*
1011	q_4^*	q_4^*	q_4^*

← TRAP STATE → contains → occurance

$Q \backslash \Sigma$	0	1
q_s	N	N
q_0	N	N
q_1	N	N
q_2	N	N
q_3	N	Y
q_4	Y	Y



* Valid if it does not contains 3 consecutive b's over $\Sigma = \{a, b\}$

→ $I = \{a, b\}$

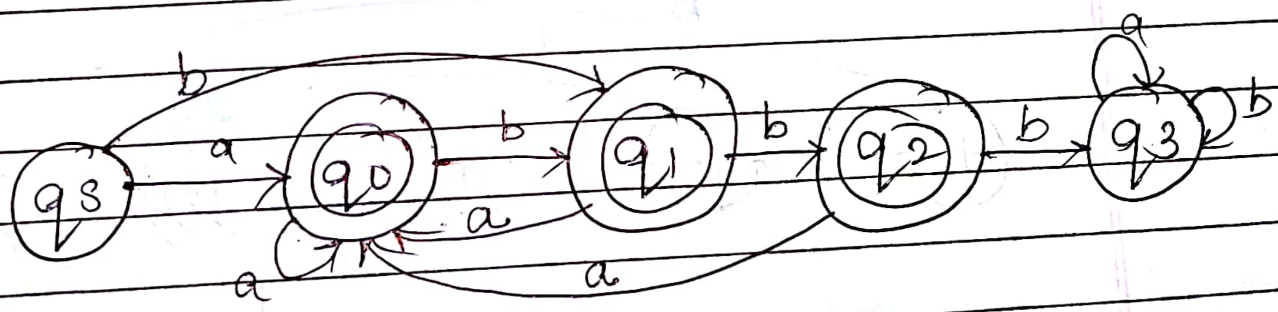
$O = \{Y, N\}$

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

	$Q \backslash \Sigma$	a	b
	q_s	q_0^*	q_1^*
a	q_0^*	q_0^*	q_1^*
b	q_1^*	q_0^*	q_2^*
bb	q_2^*	q_0^*	q_3
bbb	q_3	q_3	q_3

← DEAD STATE

$Q \backslash \Sigma$	a	b
q_s	Y	Y
q_0	Y	Y
q_1	Y	Y
q_2	Y	N
q_3	N	N



* Valid if it starts with 3 consecutive "a's" over $\Sigma = \{a, b\}$

$\rightarrow$ $I = \{a, b\}$ $\quad Q = \{q_s, q_0, q_1, q_2, q_3\}$

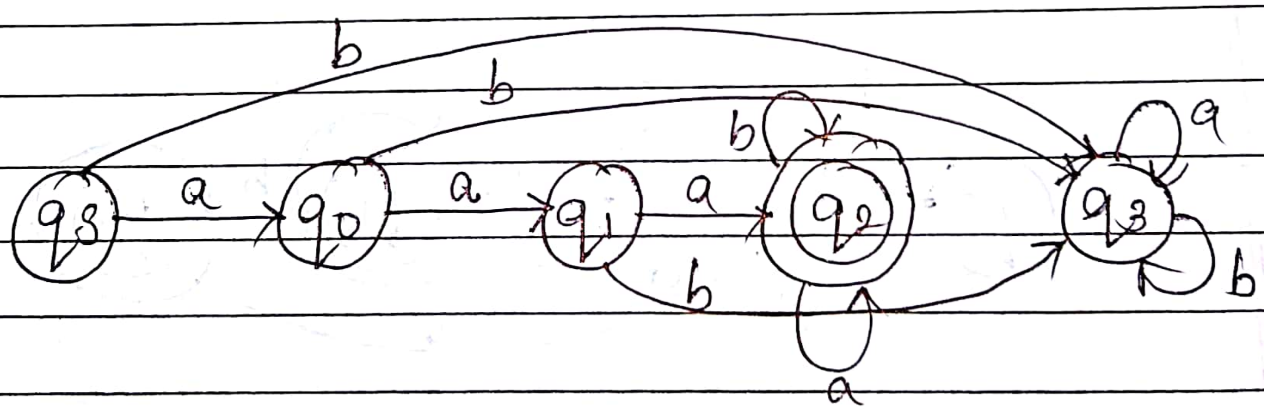
$O = \{Y, N\}$

δ	ε
a	aa
aa	a
aaa	–

Rejecting state

Q \ Σ	a	b
q_s	q_0	q_3
q_0	q_1	q_3
q_1	q_2^*	q_3
q_2^*	q_2^*	q_2^*
q_3	q_3	q_3

Q \ Σ	a	b
q_s	N	N
q_0	N	N
q_1	Y	N
q_2	Y	Y
q_3	N	N



* Points to remember :-

1) There is always one start state.
2) There can be more than one final states.
3) Start state can also be a final state. (in such a case "blank input" is also valid)
4) Dead state is a trap state which is non-final.
5) Every dead state is a trap state but a trap state may or may not be a dead state.
6) Start state ⟷ initial state.
Final state ⟷ accepting state.
Non-final state ⟷ rejecting state.
7) Dead state is always a rejecting state.

18/7/18

* Regular Expressions :-

- Definition :- Regular expression is used for specifying the strings of regular languages & is defined as follows.

1) 'φ' = φ is a regular expression for specifying { } set

2) 'ε' = ε is a regular expression for specifying {ε}

3) 'a' = a is a regular expression for specifying {a}

4) Let 'R' & 'S' be the two regular expressions for specifying LR & LS respectively.

- i) (R)/(S) is a regular expression for specifying LR ∪ LS

- ii) (R)(S) is a regular expression for specifying LR·LS

- iii) $(R)^*$ is a regular expression for specifying LR^*

ex :-

r	$L(r)$
a	$\{a\}$
b	$\{b\}$
$a+b$ or a/b	$\{a\} \cup \{b\} = \{a,b\}$
$a \cdot b$	$\{a\} \cdot \{b\} = \{ab\}$
a^*	$\{\epsilon, a, aa, aaa, \ldots\}$
a^+	$\{a, aa, aaa, \ldots\}$
$(ab)^*$	$\{\epsilon, ab, abab, \ldots\}$
$(a+b)^*$	$\{\epsilon, a, b, aa, ab, ba, bb \ldots\}$
$(00)^*$	$\{\epsilon, 00, 0000, \ldots\}$
$0(00)^*$	$\{0, 000, 00000, \ldots\}$
$(00)^*0$	$\{0, 000, 00000, \ldots\}$
$(000)^*$	$\{\epsilon, 000, 000000, \ldots\}$
$a \cdot a^*$	$\{a, aa, aaa, aaaa, \ldots\}$

Note :-
$$0 \cdot \epsilon = \epsilon \cdot 0 = 0$$
$$1 \cdot \epsilon = \epsilon \cdot 1 = 1$$
$$a \cdot \epsilon = \epsilon \cdot a = a$$
$$a^+ = a \cdot a^*$$
$$L^+ = L \cdot L^*$$
$$L^* = L^+ + \epsilon$$
$$(L^*)^* = L^*$$
$$\phi^* = \epsilon$$
$$\epsilon^* = \epsilon$$

Scanned by CamScanner

23/7/18

Q1. Write RE for the following:-

i) Set of all the strings that begin with 'a' over $\Sigma = \{a, b\}$

ii) Set of all the strings that ends either in 0 or 11 over $\Sigma = \{0, 1\}$

iii) Set of all the strings that starts with a & ends with b.

iv) Set of all the strings that starts with x & ends with xy.

→ i) $r = a(a+b)^*$

$L(r) = a \cdot \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}$

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

→ ii) $(0+1)^*0 + (0+1)^*11$

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

$L(r) = \{\epsilon, 0, 1, 00, 11, 01, 10, 110, 001 \ldots\} (0+11)$

$= \{0, 00, 10, 000, 110, 11, 011, 111, 0011, 1111, \ldots\}$

→ iii) $a(a+b)^*b$

$L(r) = \{ab, aab, abb, aabb, abab, \ldots\}$

→ iv) $x(x+y)^*xy + xy$

$L(r) = \{xy, xxy, xxxy, xyxy, xxyyxy, \ldots\}$

Scanned by CamScanner

23/7/18

Date

Q1. Set of all the strings that starts with "abb" & ends with "bba" over $\Sigma = \{a, b\}$

abb___bba

$r = abb(a+b)^* bba + abba + abbba$

$L(r) = \{abba, abbba\}, \{abb \cdot \{\epsilon, a, b, aa, bb, ab, ba, \dots\} \cdot bba\}$

$L(r) = \{abba, abbba, abbbba, abbabba, abbbbba, abbaabba, abbbbbba, abbabbba, \dots\}$

Q2. Set of all the strings that start & ends with different letters over $\Sigma = \{x, y\}$

$r = x(x+y)^* y + y(x+y)^* x$

$L(r) = \{xy, yx, xxy, xyy, yxx, yyx, \dots\}$

Q3. Set of all the strings that start & ends with same symbol over $\Sigma = \{0, 1\}$

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

$L(r) = \{00, 11, 000, 010, 1101, 1111, 0010, 1011, \dots 0, 1 \dots\}$

Q4. Set of all the strings that contain atleast 1 occurrence of "aa" over $\Sigma = \{a, b\}$

$r = (a+b)^* aa (a+b)^*$

$L(r) = \{aaa, aab, aaab, aaba, aaaa, aabb, abaa, bbaa, aa, baaa, \dots\}$

Q5: Set of all the strings that contains atleast 2 or more than 2 'a's.' over $\Sigma = \{a, b\}$

$r = a(a+b)^*a \quad (a+b)^*a$

$L(r) = \{a \cdot \{\epsilon, a, b, aa, ab, ba, bb \ldots\} a \cdot \{\epsilon, a, b, aa, ab, ba, bb, \ldots\} a$

$= \{aa, aaa, aba, aaaa, aaba, abaa, abba, \ldots\}$

Q6. Set of all the strings that contains exactly 2 a's over $\Sigma = \{a, b\}$

$r = (b)^* a (b^*) a (b)^*$ aabb abba

$L(r) = \{baa, aab, aabb, abba, aba, aa, \ldots\}$

Q7. Set of all the strings that contains atmost 2 a's over $\Sigma = \{a, b\}$

$r = b^* + b^*ab^* + b^* ab^* ab^*$

$L(r) = \{b, bb, bbb, ab, abb, ba, bba, aba, abba, aab, \ldots\}$

Q8: Set of all the strings that contains atleast 1 x over $\Sigma = \{x, y\}$

$r = x(x+y)^*$

$L(r) = \{x, xx, xy, xxy, \ldots\}$

Q9) Set of all the string that contains atleast 1 x & 1 y over $\Sigma = \{x, y\}$

$r = x(x+y)^* y$

$L(r) = x\{\epsilon, x, y, xx, xy, yx, yy \dots\}y$

$L(r) = \{xy, xxy, xyy, xxxy, xxyy, xyxy, xyyy, \dots\}$

Q10) Set of all the strings that contain atleast 1 x, atleast 1 y & atleast 1 z over $\Sigma = \{x, y, z\}$

$r = x(x+y+z)^* y(x+y+z)^* z(x+y+z)^*$

$L(r) = \{\epsilon, xy, xz, xx, yx, yy, yz, zx, zy, zz \dots\} xyz$

$= \{xyz, xyxyz, xyzxz, xxxyz, xyzyx, yyxyz, yzxyz, xyzzx, xyzzy, zzxyz \dots\}$

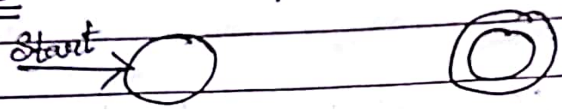
24/7/18

* RE to NFA (Regular Expression to NFA) :-

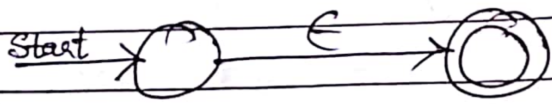
1) Divide the given RE into smaller sub expressions & create NFA for each using rule 1, 2 & 3.

2) Combine the above NFA's using rule 4.

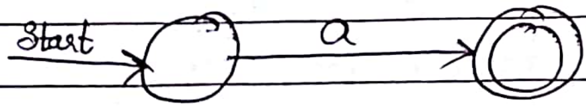
Rule 1 :- $r = \phi$



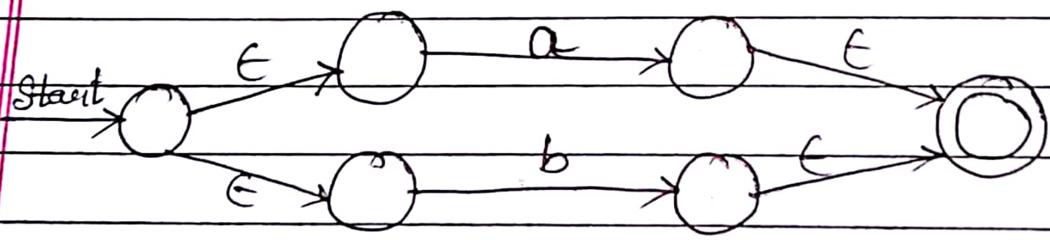
Rule 2 :- $r = \epsilon$



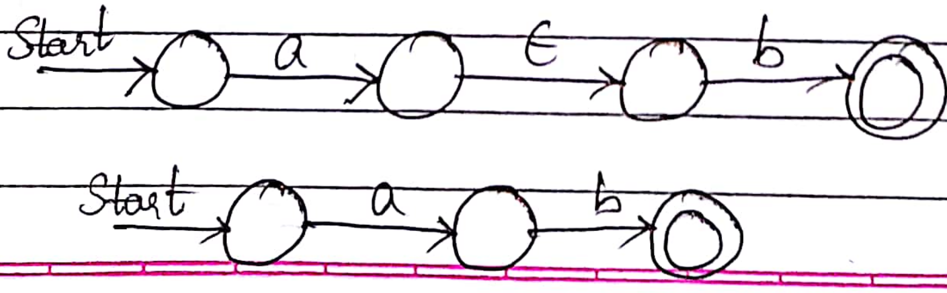
Rule 3 :- $r = a$



Rule 4.1 :- $r = (R)/(S) = a+b$

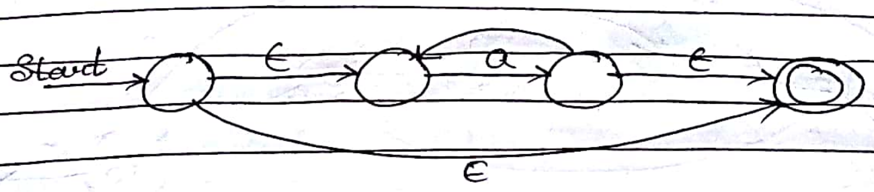


Rule 4.2 :- $r = (R)\cdot(S) = a \cdot b$

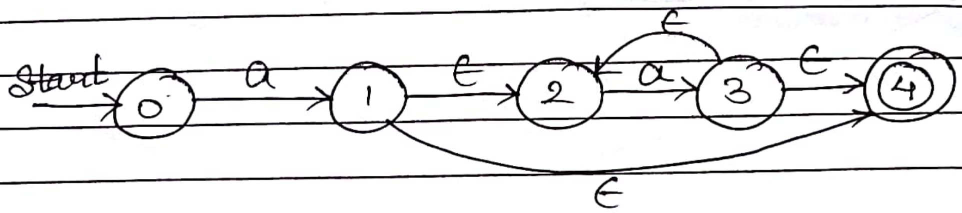


Page No.
Date

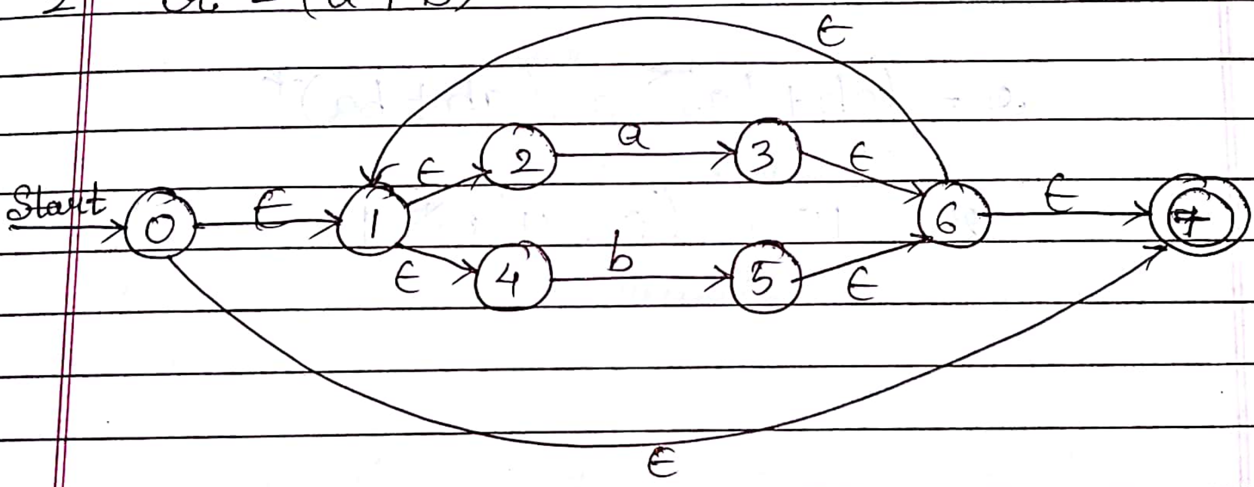
Rule 4.3 :- $r = R^* = a^*$



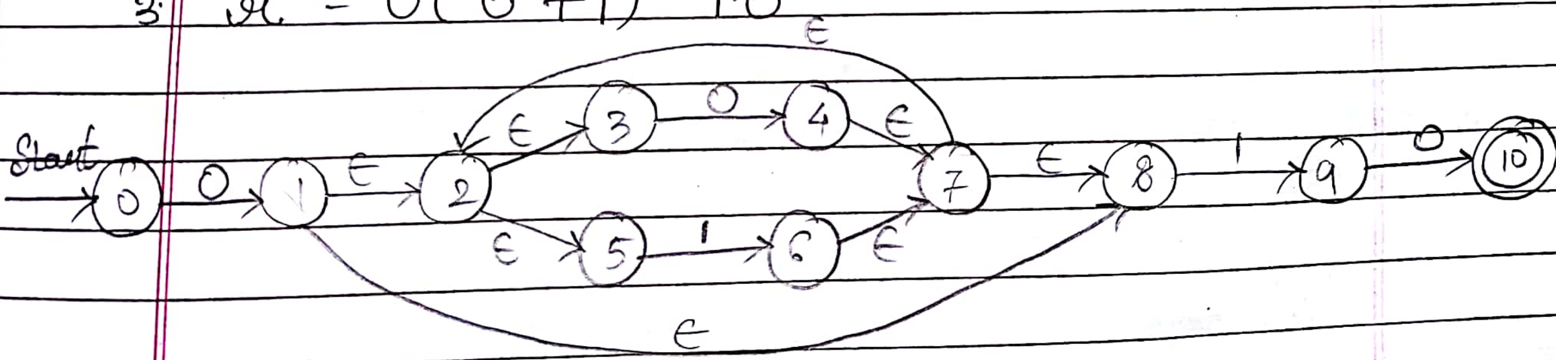
Prob. 1. $r = a \cdot a^*$



2. $r = (a+b)^*$



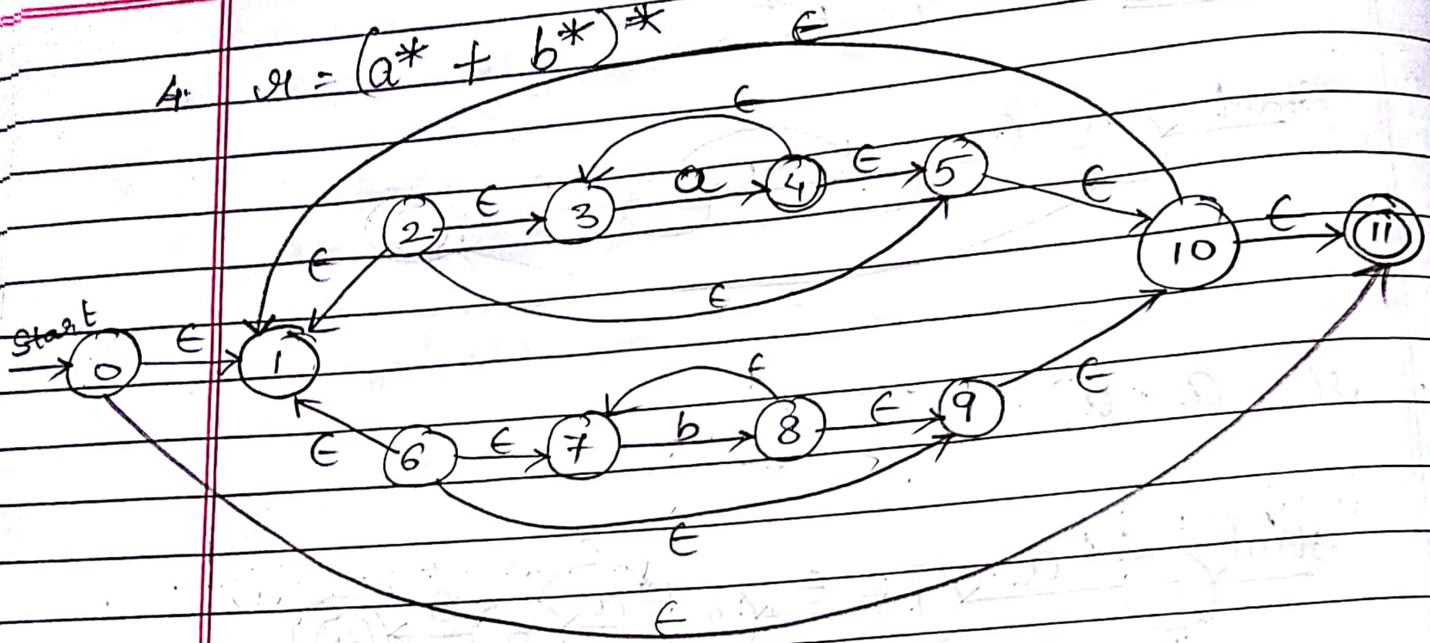
3. $r = 0(0+1)^* 1 \cdot 0$



Scanned by CamScanner

Date

4. $r = (a^* + b^*)^*$



Tutorial 3

~~m. imp~~

1) $r = (a + bb)^*$

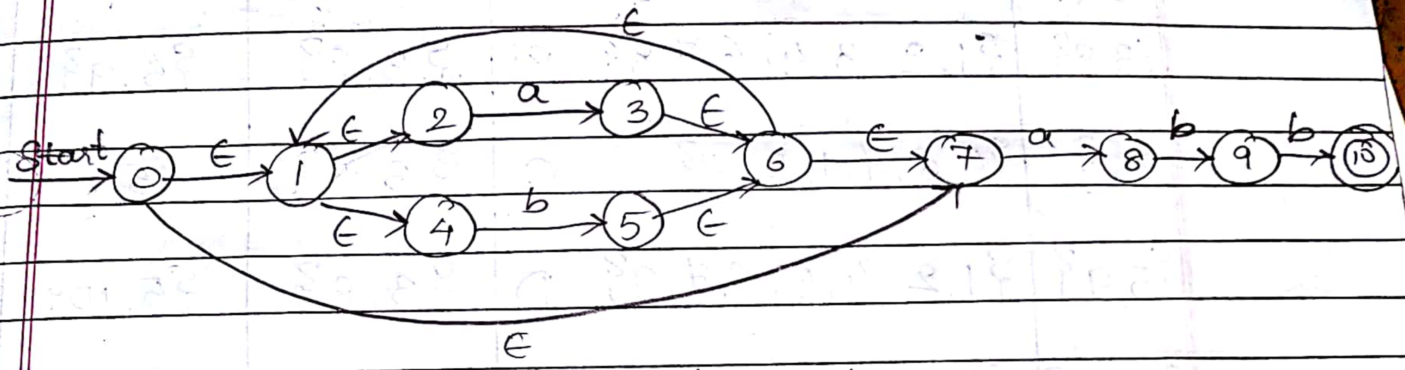
2) $r = (ab + ba)^* \cdot aa\,(ab + ba)^*$

3) $r = 10 + (0 + 11)0^*1$

Scanned by CamScanner

* Conversion of NFA to DFA.

Q1. Convert the following RE into NFA & also design the equivalent DFA for the same.
$r = (a+b)^* abb$



NFA for $r = (a+b)^* abb$
$L(r)$ = Set of all the strings that ends in abb

ϵ-closure of a state :-
It is defined as the set of states that are reachable from that state by waiting on ϵ-transitions only (including the state).
For eg:-
ϵ-closure of {0} = {0, 1, 7, 4, 2}
ϵ-closure of {5} = {5, 6, 7, 1, 2, 4}

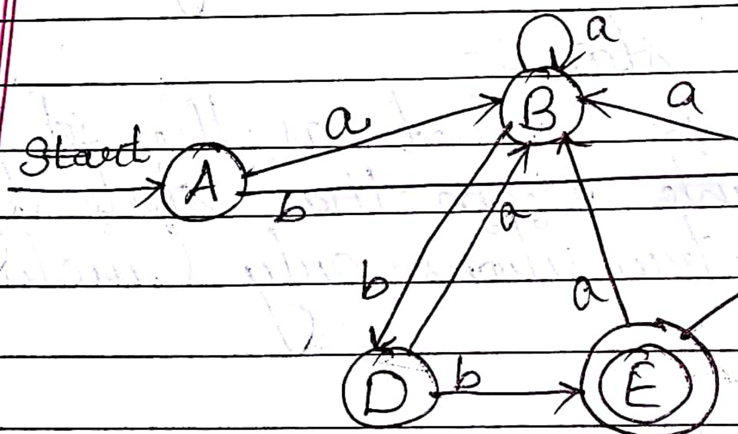
ϵ-closure of set of states :-
It is defined as the union of ϵ-closures of each state of the set.
for eg:- ϵ-closure of {3, 8}
= ϵ-closure (3) ∪ ϵ-closure (8)

$= \{3, 6, 7, 1, 2, 4\} \cup \{8\}$
$= \{1, 2, 3, 4, 6, 7, 8\}$

To convert NFA to DFA

x	$y = \epsilon\text{-closure}(x)$		$\delta(y, a)$	$\delta(y, b)$
0	$\{0, 1, 2, 4, 7\}$	$\rightarrow$ A	$\{3, 8\}$	$\{5\}$
$\{3, 8\}$	$\{1, 2, 3, 4, 6, 7, 8\}$	B	$\{3, 8\}$	$\{5, 9\}$
$\{5\}$	$\{5, 6, 7, 1, 2, 4\}$	C	$\{3, 8\}$	$\{5\}$
$\{5, 9\}$	$\{1, 2, 4, 5, 6, 7, 9\}$	D	$\{3, 8\}$	$\{5, 10\}$
$\{5, 10\}$	$\{1, 2, 4, 5, 6, 7, 10\}$	E*	$\{3, 8\}$	$\{5\}$

Q \ Σ	a	b
Start → A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C



DFA for $r = (a+b)^* abb$

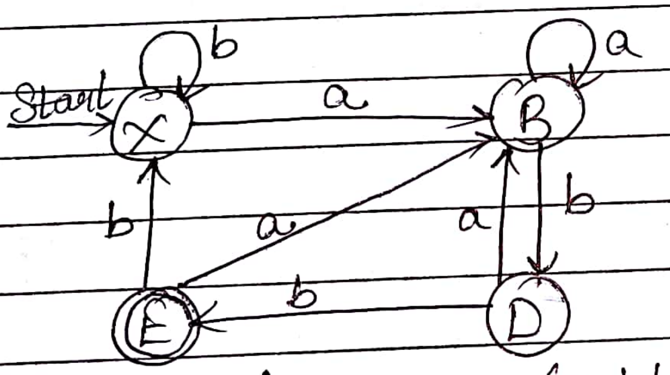
* Conversion of DFA to minimised DFA (classical method)

Rule: State can be merged

if (all states have same transition) AND (all are final OR all are non-final)

Q\Σ	a	b
Start→ x	B	x
B	B	D
D	B	E
E*	B	x

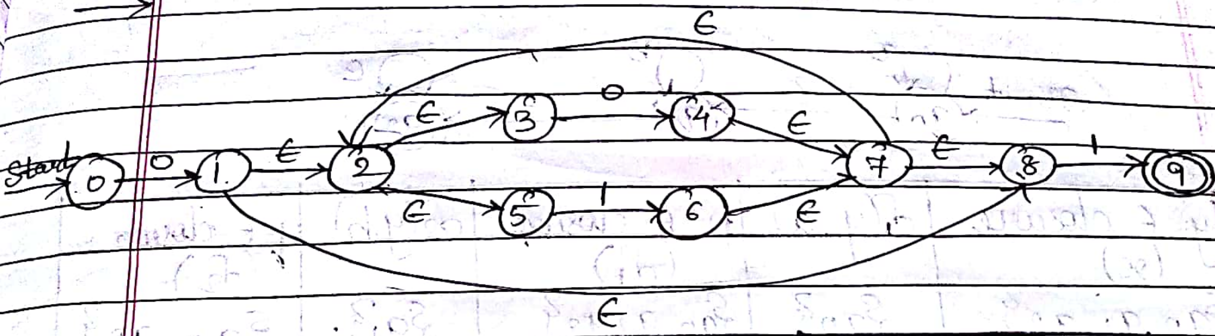
(x, abbbabb)
(B, bbbabb)
(D, bbabb)
(E, babb)
(x, abb)
(B, bb)
(D, b)
(E, ∈)



Min. DFA for $r = (a+b)^* abb$

31/7/18

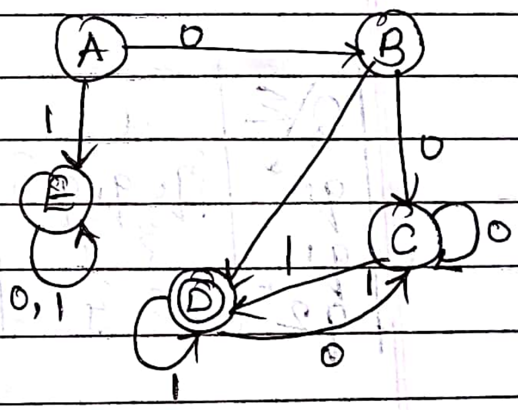
Construct NFA for $r = 0(0+1)^*1$ & convert it into minimized DFA.



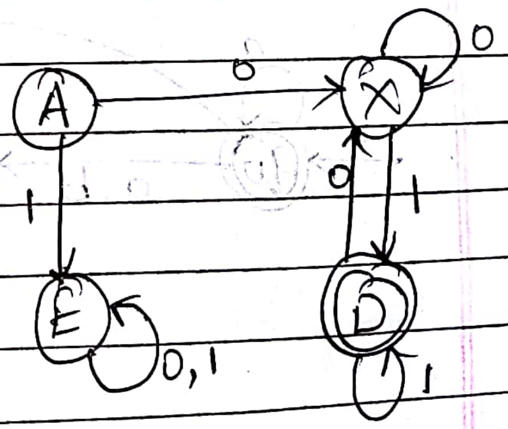
x	$y = \epsilon$-closure (x)		$\delta(y,0)$	$\delta(y,1)$
{1}	{1, 2, 3, 5, 8}	B	{4}	{6, 9}
{4}	{4, 7, 8, 2, 3, 5}	C	{4}	{9, 6}
{6, 9}	{6, 7, 8, 2, 3, 5, 9}	D*	{4}	{9, 6}
0	{0}	→ A	{1}	{ }
{ }	{ }	E	{ }	{ }

Q \ Σ	0	1
A	B	E
B	C	D
C	C	D
D*	C	D
E	E	E

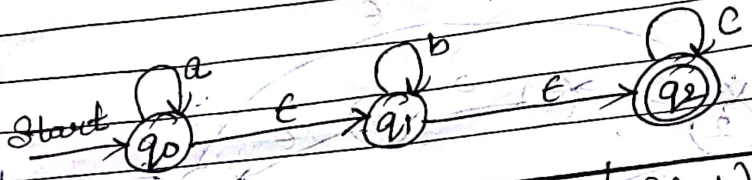
→ X



Q \ Σ	0	1
A	X	E
X	X	D
D*	X	D
E	E	E



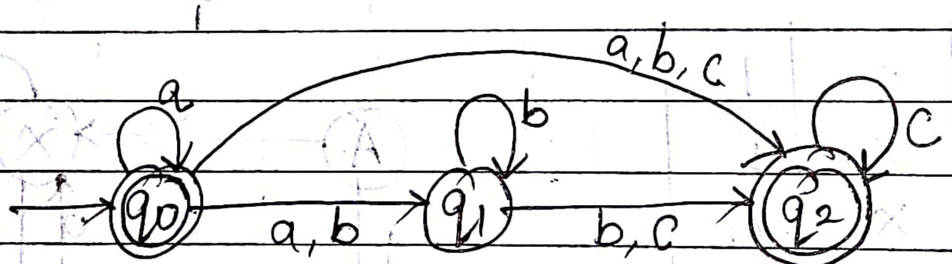
* Conversion of NFA with ϵ-transition to NFA without ϵ-transition.



x	$y = \epsilon$ closure (x)	$\delta_1(y, a)$	$p = \epsilon$-closure (δ_1)	$\delta_2(y, b)$	$q = \epsilon$-closure (δ_2)
q_0	$\{q_0, q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$
q_1	$\{q_1, q_2\}$	$\{\ \}$	$\{\ \}$	$\{q_1\}$	$\{q_1, q_2\}$
q_2	$\{q_2\}$	$\{\ \}$	$\{\ \}$	$\{\ \}$	$\{\ \}$

$\delta_3(y, c)$	$r = \epsilon$-closure (δ_3)
$\{q_2\}$	$\{q_2\}$
$\{q_2\}$	$\{q_2\}$
$\{q_2\}$	$\{q_2\}$

$Q \backslash \Sigma$	a	b	c
q_0^*	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{\ \}$	$\{q_1, q_2\}$	$\{q_2\}$
q_2^*	$\{\ \}$	$\{\ \}$	$\{q_2\}$



Convert the following NFA with ε-transition to NFA without ε-transition.

1) The no. of states remain the same.

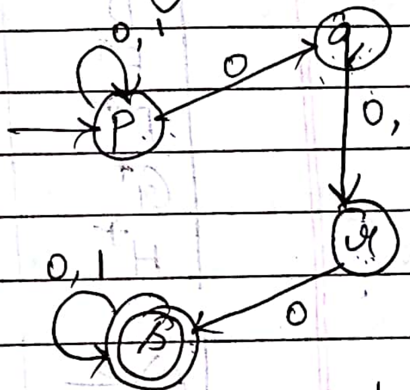
2) The start state & the final state also remains the same.

3) If the ε-closure of the start state contains a final state then make that start state also a final state.

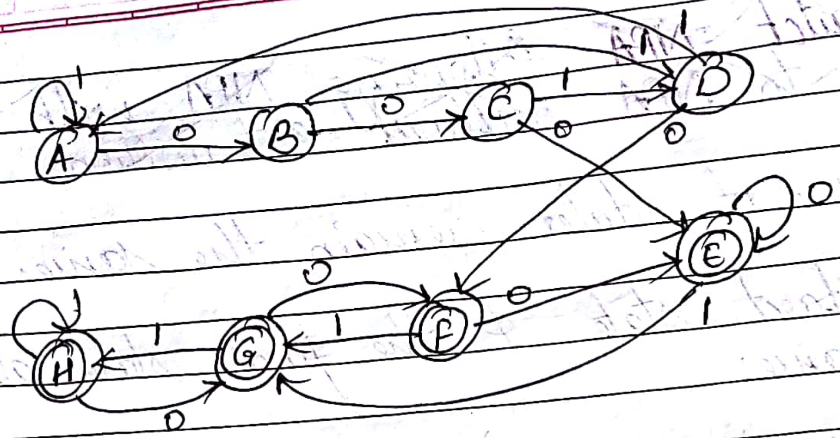
* Construct a DFA equivalent to the given NFA.

δ :-

Q\Σ	0	1
p	p,q	p
q	r	r
r	s	—
s*	s	s



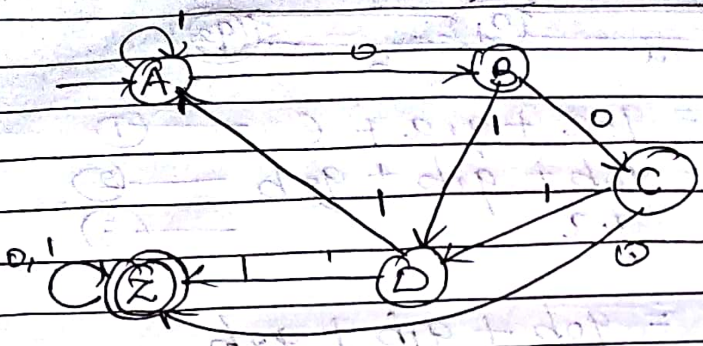
x	y = ε-closure (x)		δ(y, 0)	δ(y, 1)
p	{p}	A	{p, q}	{p}
{p, q}	{p, q}	B	{p, q, r}	{p, r}
{p, q, r}	{p, q, r}	C	{p, q, r, s}	{p, r}
{p, r}	{p, r}	D	{p, q, s}	{p}
{p, q, r, s}	{p, q, r, s}	E*	{p, q, r, s}	{p, r, s}
{p, q, s}	{p, q, s}	F*	{p, q, r, s}	{p, r, s}
{p, r, s}	{p, r, s}	G*	{p, q, s}	{p, s}
{p, s}	{p, s}	H*	{p, q, s}	{p, s}



Q\Σ	0	1	
A	B	A	
B	C	D	
C	E	D	
D	F	A	
E*	F	G	X
F*	E	G	
G*	F	H	Y
H*	F	H	

Q\Σ	0	1	
A	B	A	
B	C	D	
C	X	D	
D	X	A	
X*	X	Y	Z
Y*	X	Y	

Q\Σ	0	1
A	B	A
B	C	D
C	Z	D
D	Z	A
Z*	Z	Z



* Conversion of DFA to regular expression:

- Steps for finding regular expressions from the given transition diagram

1) Write the state equation by looking at the incoming transitions. (Add ϵ to the start state equation).

2) Perform any combination of substitution, rearrangement & apply Arden's Theorem until we are able to get the final state's equation in terms of alphabets.

* **Arden's Theorem:-**

Let P & Q be any two RE & if P is ϵ free then the following equation in R.

$$\boxed{R = Q + RP}$$

has a unique solution given by

$$\boxed{R = QP^*}$$

11/8/18

Step 1 :- $q_0 = q_0 a + q_1 a + \epsilon$ —— (1)

$q_1 = q_0 b + q_1 b + q_2 b$ —— (2)

$q_2 = q_1 a$ —— (3)

Step 2 :- $q_1 = q_0 b + q_1 b + q_2 b$ (S)

$q_1 = q_0 b + q_1 b + q_1 ab$ (Re)

$q_1 = q_0 b + q_1 (b + ab)$

$R = Q + R \; P$

Eqn (1) by Arden's Theorem

$q_1 = q_0 b (b+ab)^*$ —— (4)

$q_0 = q_0 a + [q_0 b (b+ab)^*] a + \epsilon$ (S)

$= q_0 a + q_0 ab(b+ab)^* + \epsilon$

$q_0 = q_0 (a + ab(b+ab)^*) + \epsilon$

$q_0 = \epsilon + q_0 (a + ab(b+ab)^*)$ (RE)

$R = Q + R \; P$

$q_0 = (a + ab(b+ab)^*)^*$ --- By Arden's Theorem

Sub q_0 in eq. (4)

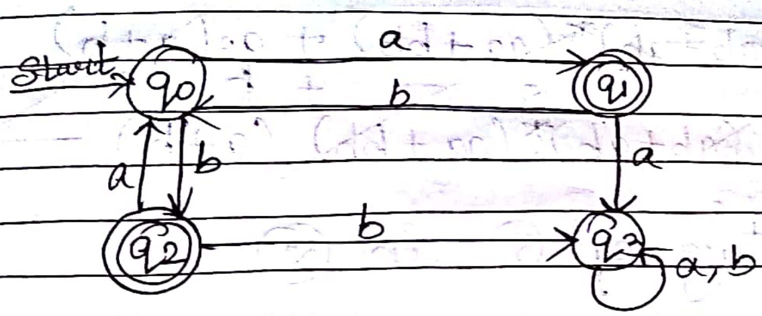
$q_1 = (a + ab(b+ab)^*)^* b (b+ab)^*$ —— (5)

Sub eq. (5) in eq. (3)

$q_2 = a[(a+ab(b+ab)^*)^* b (b+ab)^*$ ⟶ form exp.

1/8/18

*



Find the RE for the above FA.

→ Step 1 :- $q_0 = q_1 b + q_2 a + \epsilon$ —— ①

$q_1 = q_0 a$ —— ②

$q_2 = q_0 b$ —— ③

$q_3 = q_1 a + q_2 b + q_3 a + q_3 b$ —— ④

Step 2 :- $q_0 = q_0 ab + q_0 ba + \epsilon$ (S)

$q_0 = q_0 (ab + ba) + \epsilon$

$q_0 = \epsilon + q_0 (ab + ba)$ (Re)

$R = Q + RP$

Eq. ① by Arden's Theorem

$q_0 = (\epsilon (ab+ba))^*$

$\boxed{q_0 = (ab+ba)^*}$ —— ⑤

$q_3 = q_1 a + q_2 b + q_3 a + q_3 b$

$q_3 = q_0 aa + q_0 bb + q_3 a + q_3 b$

$q_3 = (ab+ab)^* aa + (ab+ab)^* bb + q_3 a + q_3 b$

$q_3 = (ab+ab)^* (aa+bb) + q_3 (a+b)$

$q_3 = q_3 (a+b) + (ab+ba)^* (aa+bb)$

$R = Q + RP$

$q_3 = (q_3 (a+b)(aa+bb))^*$ —— ⑥

-- By Arden's Theorem

Substituting eqⁿ ⑤ in ②

$q_1 = (ab+ab)^* b$ —— ⑦

6/8/18

$\cancel{q_3 = (ab+ab)^*(aa+bb) + q_3(a+b)}$ — ⑥

$\cancel{R = (ab+ab)^*(aa+bb)(a+b)}$

$q_3 = (ab+ab)^*(aa+bb)$

Substituting eq' ⑤ in ②

$q_1 = q_0 a$

$\boxed{q_1 = (ab+ba)^* a}$ — ⑦ $\longrightarrow r_1$

Substituting eq' ⑤ in ③

$q_2 = q_0 b$

$\boxed{q_2 = (ab+ba)^* b}$ — ⑧ $\longrightarrow r_2$

1/8/18

$r = r_1 + r_2$

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

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

6/8/18

* Moore Machine

Defination :-

It is a finite automata with no final state & it produces output sequence for the given input sequence.

In Moore machine, the output symbol is associated with each state (i.e; state gives the output).

Mathematical representation :-

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where Q = finite set of states.

Σ = input alphabet

Δ = output alphabet

δ = transition function

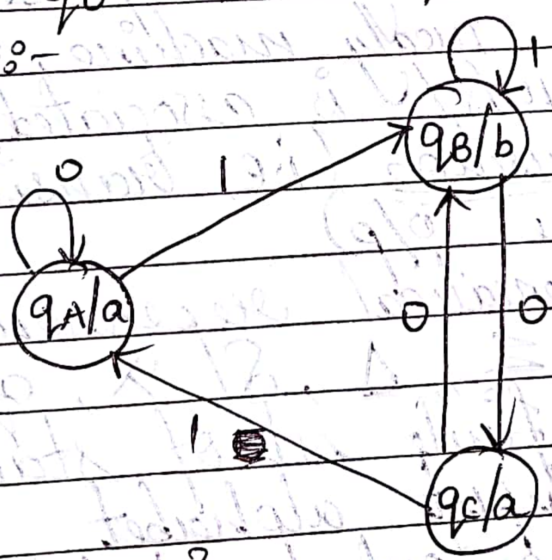
$\delta :- Q \times \Sigma \rightarrow Q$

λ = output mapping

$\lambda :- Q \rightarrow \Delta$

q_0 = start state.

for eg:-



$Q = \{q_A, q_B, q_C\}$

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

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

$q_0 = q_A$

δ :-

$Q \backslash \Sigma$	0	1
q_A	q_A	q_B
q_B	q_C	q_B
q_C	q_B	q_A

$\lambda(q_A) = a$
$\lambda(q_B) = b$
$\lambda(q_C) = a$

* $(q_A, 1010)$ abaa
$(q_B, 010)$
$(q_C, 10)$
$(q_A, 0)$
(q_A)

$(q_A, 01001)$ abaab
$(q_A, 1001)$
$(q_B, 001)$
$(q_C, 01)$
$(q_B, 1)$
(q_B, ϵ)

* Mealy Machine

Defination :- It is a FA with no final state & it produces o/p sequence for the given i/p sequence.
In mealy machine, the o/p symbol is associated with each transition (i.e; transition gives the o/p)

Mathematical representation.

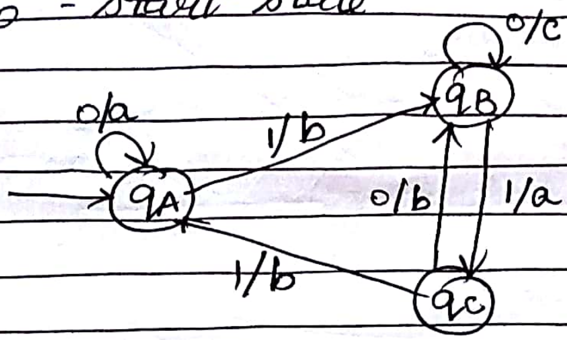
$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where Q = finite set of states
Σ = input alphabet
Δ = output alphabet
δ = transition function
δ :- $Q \times \Sigma \rightarrow Q$

λ = output mapping

$\lambda :- Q \times \Sigma \longrightarrow \Delta$

q_0 = start state



$Q = \{q_A, q_B, q_C\}$

$q_0 = q_A$

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

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

δ :-

$Q \backslash \Sigma$	0	1
q_A	q_A	q_B
q_B	q_B	q_C
q_C	q_B	q_A

Δ :-

$Q \backslash \Sigma$	0	1
q_A	a	b
q_B	c	a
q_C	b	b

$(q_A, 1010)$ bcab

$(q_B, 010)$

$(q_B, 10)$

$(q_C, 0)$

$(q_B$

7/8/18

v.imp.

* Difference between moore & mealy machine

⇒

	Moore Machine	Mealy Machine
1)	In moore machine, O/P symbol is associated with each state.	1) In mealy machine, O/P symbol is associated with each transition.
2.)	In moore machine, the O/P is dependent on the state.	2) In this, O/P is dependent on the state & the i/p.
3.)	In moore m/c, the O/P mapping is defined as $\lambda :- Q \rightarrow \Delta$	4) In mealy m/c, if length of the i/p sequence is 'n' then the length of the O/P sequence is also 'n'.
4)	In moore m/c, if length of the i/p sequence is 'n' then the length of the O/P sequence is 'n+1'	3.) In mealy m/c the O/P mapping is defined as $\lambda :- Q \times \Sigma \rightarrow \Delta$
5)	In moore m/c, we can get the output on ϵ.	5) In mealy m/c, we cannot get the O/P on ϵ
6)	Eg:-	Eg:-

v.v.imp

Page No. Date

* Minimization of Moore Machine :-

States can be merged only if they satisfy the following 2 conditions.

1) All the states must have same transition &

2) Output symbols along with the states are also same.

v. v. imp

*1) Design Moore Machine to o/p 'A' if i/p ends in "101", 'B' if i/p ends in '101', 'C' otherwise over $\Sigma = \{0, 1\}$

→ Step 1 :- Theory of Moore M/C

Step 2 :- Logic

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

$\Delta = \{A, B, C\}$

$Q = \{q_s, q_0, q_1, q_2, q_3, q_4, q_5\}$

Step 3 :- Implementation

	Q \ Σ	0	1
	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_4
10	q_2	q_0	q_3
101	q_3	q_2	q_4
11	q_4	q_5	q_4
110	q_5	q_0	q_3

$\lambda(q_s) = C$ — q_0

$\lambda(q_0) = C$

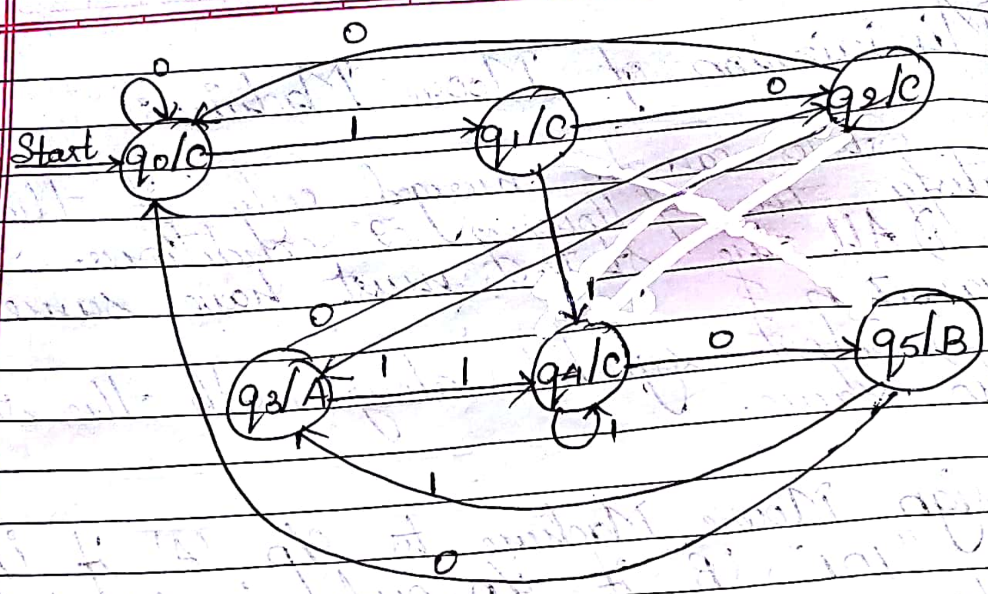
$\lambda(q_1) = C$

$\lambda(q_2) = C$

$\lambda(q_3) = A$

$\lambda(q_4) = C$

$\lambda(q_5) = B$



$(q_0, 1101)$
$(q_1, 101) \longrightarrow C$
$(q_4, 01) \longrightarrow C$
$(q_5, 1) \longrightarrow B$
$(q_3, \epsilon) \longrightarrow A$

Design Moore m/c for the follo process "print residue for modulo 4 of binary nos."

2) ~~Binary Number divisible by 4~~

Step 1 :- Defination

Step 2 :- logic

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

$\Delta = \{0, 1, 2, 3\}$

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

~~Q\Σ 0 1~~

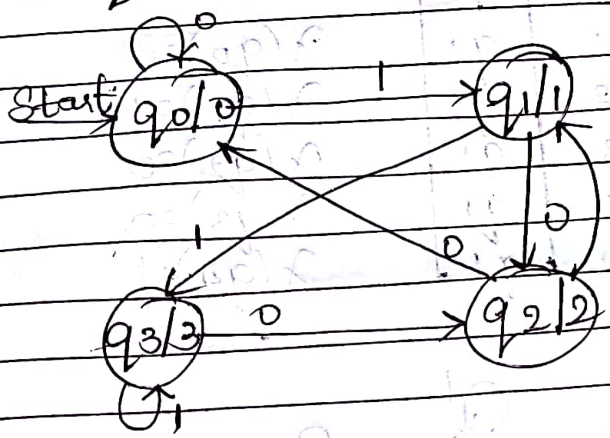
~~q_s~~

~~000~~

8

Step 3 :- Implementation

	Q\Σ	0	1	
	q8	q0	q1	λ(q8)=0 — q0
000	0 q0	q0	q1	λ(q0)=0 ~~0~~000/4 0001/4
001	1 q1	q2	q3	λ(q1)=1 0010/4 0011/4
010	2 q2	q0	q1	λ(q2)=2
011	3 q3	q2	q3	λ(q3)=3



(q0, 1101)
(q1, 101) ⟶ 1
(q3, 01) ⟶ 3
(q2, 1) ⟶ 2
(q1, ∈) ⟶ 1

v.v. imp

*3) Design Moore m/c to change each occurance of "1000" to "1001" over Σ = {0, 1}

→ Step 1 :- Def

Step 2 :- logic

Σ = {0, 1}

Δ = {0, 1} ~~0,1,0,1~~

Q = {q8, q0, q1, q2, q3, q4}

	δ Q\Σ	0	1
	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3	q_1
100	q_3	q_4	q_1
1000	q_4	q_0	q_1

q_0

λ :-

Q\Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_3	q_1
q_3	q_4	q_1
q_4	q_0	q_1

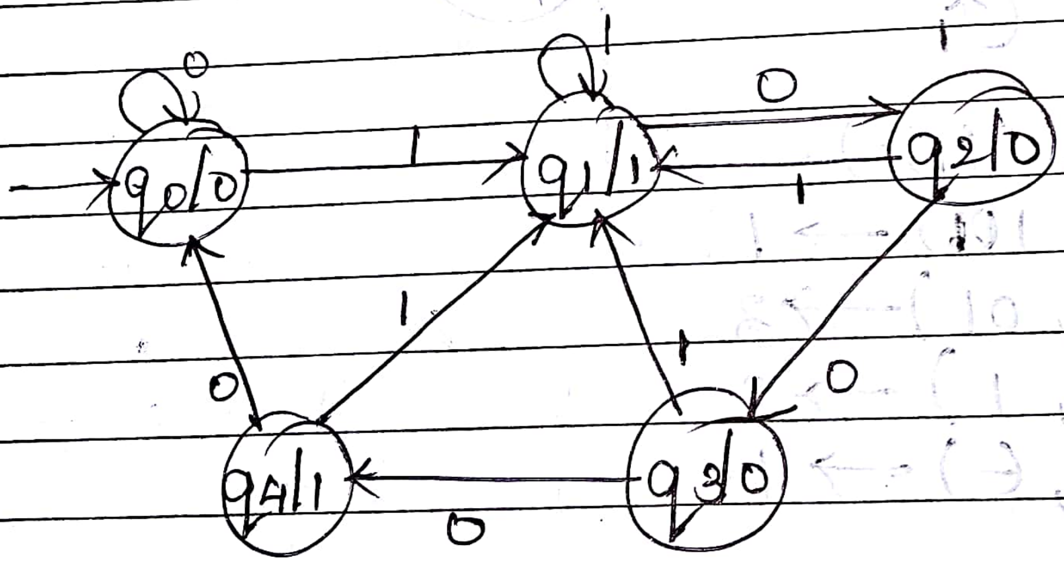
$\lambda(q_0)=0$

$\lambda(q_1)=1$

$\lambda(q_2)=0$

$\lambda(q_3)=0$

$\lambda(q_4)=1$



Unary
0 - ε
1 - 0
2 - 00
3 - 000
4 - 0000

v.imp 4) Design a moore m/c to count each occurrence of "AAB" over $\Sigma = \{A, B\}$

→ Step 1:- def.

Step 2:- logic

$$\Sigma = \{A, B\}$$
$$\Delta = \{\epsilon, 0\}$$
$$Q = \{q_s, q_0, q_1, q_2, q_3\}$$

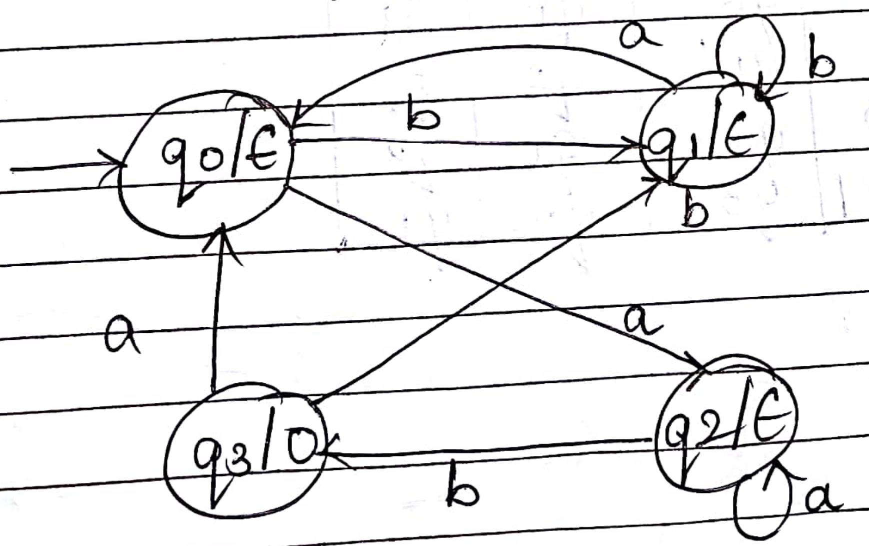
Step 3:-

δ:-	$Q \backslash \Sigma$	A	B
	q_s	q_0	q_1
A	q_0	q_2	q_1
B	q_1	q_0	q_1
~~AA~~	q_2	q_2	q_3
~~AAB~~	q_3	q_0	q_1

q_0
— ε
— ε
— ε
— ε
— 0

λ:- $Q \backslash \Sigma$	A	B
q_0	q_2	q_1
q_1	q_0	q_1
q_2	q_2	q_3
q_3	q_0	q_1

$\lambda(q_0) = \epsilon$
$\lambda(q_1) = \epsilon$
$\lambda(q_2) = \epsilon$
$\lambda(q_3) = 0$



* Minimization of Mealy Machine :-

States can be merged only if they satisfy the following 2 conditions.

1) All the states have same transition &

2) O/p symbol along with the transitions are also same.

Q1) Design mealy m/c to o/p "x" if the i/p ends in "101" & "y" otherwise over $\Sigma = \{0,1\}$

⟹ Step 1:- Theory

Step 2 :- Logic

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

$\Delta = \{x, y\}$

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

Step 3:- Implementation

δ :-

	Q\Σ	0	1
	q_s	q_0	q_1
0	q_0	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_0	q_3
101	q_3	q_2	q_1

Scanned by CamScanner

λ :-

Q\Σ	0	1	
qs	y	y	q0
q0	y	y	
q1	y	y	q1
q2	y	x	
q3	y	y	

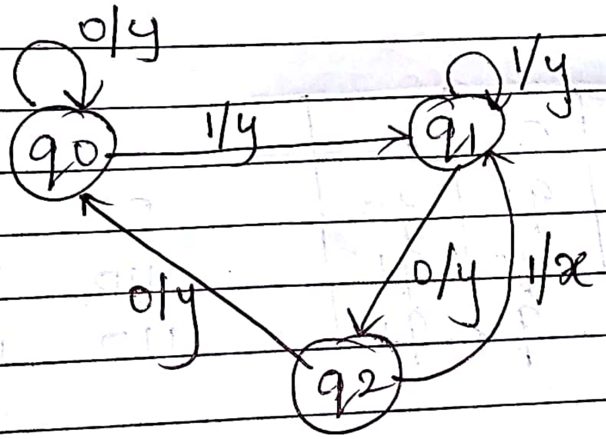
Step 4 :- Minimization

δ :-

Q\Σ	0	1
q0	y	y
q1	y	y
q2	y	x

λ :-

Q\Σ	0	1
q0	q0	q1
q1	q2	q1
q2	q0	q1



Page No.	
Date	

2) Construct mealy m/c to o/p even & odd number depending on the number of 1's encountered are even & odd over $\Sigma = \{0, 1\}$

$\rightarrow$

Step 1 :- Theory

Step 2 :- Logic

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

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

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

Step 3 :-

δ :-

$Q \backslash \Sigma$	0	1
q_s	q_1	q_0
O q_0	q_0	q_1
E q_1	q_1	q_0

0	1
1110	11
110	111

λ :-

$Q \backslash \Sigma$	0	1
q_s	E	O
q_0	O	E
q_1	E	O

— q_1

Step 4 :-

δ :-

$Q \backslash \Sigma$	0	1
q_0	q_0	q_1
q_1	q_1	q_0

λ :-

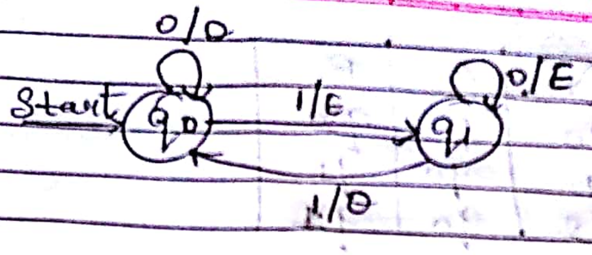
$Q \backslash \Sigma$	0	1
q_0	O	E
q_1	E	O

8/8/18

3) Co...

oc...

$\Rightarrow$



8/8/18

3) Construct mealy machine to change the occurance of "ABB" to ABA over $\Sigma = \{A, B\}$

$\Rightarrow$

Step 1 :- Theory

Step 2 :- Logic

$\Sigma = \{A, B\}$

$\Delta = \{A, B\}$

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

Step 3 :-

δ :-

		$Q \backslash \Sigma$	A	B
$\lambda(q_s) = a$		q_s	q_0	q_1
$\lambda(q_0) = a$	A	q_0	q_0	q_2
$\lambda(q_1) = b$	B	q_1	q_0	q_1
$\lambda(q_2) = b$	AB	q_2	q_0	q_3
$\lambda(q_3) = a$	ABB	q_3	q_0	q_1

q_1

AA	~~B~~AB	a
~~B~~A	~~B~~B	b
ABA	ABB	b
ABBA	ABBB	a

δ :-

$Q \backslash \Sigma$	A	B
q_1	q_0	q_1
q_0	q_0	q_2
q_2	q_0	q_3
q_3	q_0	q_1

q_1

δ :-

$Q \backslash \Sigma$	A	B
q_0	q_0	q_2
q_1	q_0	q_1
q_2	q_0	q_1

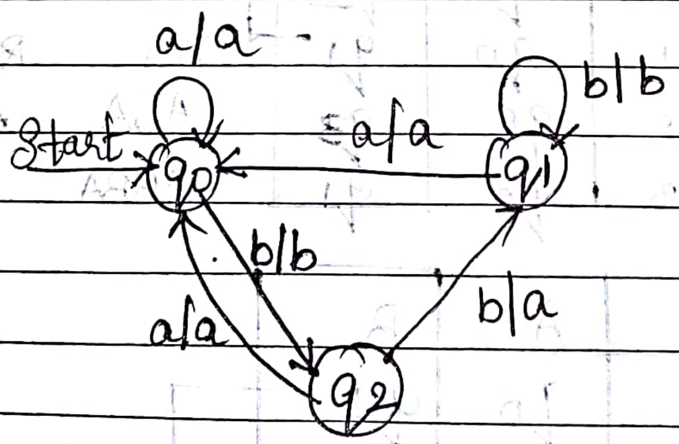
λ :-

$Q \backslash \Sigma$	A	B
q_s	a	b
q_0	a	b
q_1	a	b
q_2	a	a
q_3	a	b

q_1

λ :-

$Q \backslash \Sigma$	A	B
q_0	a	b
q_1	a	b
q_2	a	a



4) Construct mealy machine for recognizing $r = (0+1)^*(00+11)$ over $= \{0,1\}$

→ Step 1:- Theory

Step 2:- Logic:-

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

$\Delta = \{a, R\}$

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

Step 3:-

$\lambda(q_s) = R$

$\lambda(q_0) = R$

$\lambda(q_1) = R$

$\lambda(q_2) = A$

$\lambda(q_3) = A$

	δ:- Q\Σ	0	1
	q_s	q_0	q_1
0 R	q_0	q_2	q_1
1 R	q_1	q_0	q_3
0 0 A	q_2	q_2	q_1
1 1 A	q_3	q_0	q_3

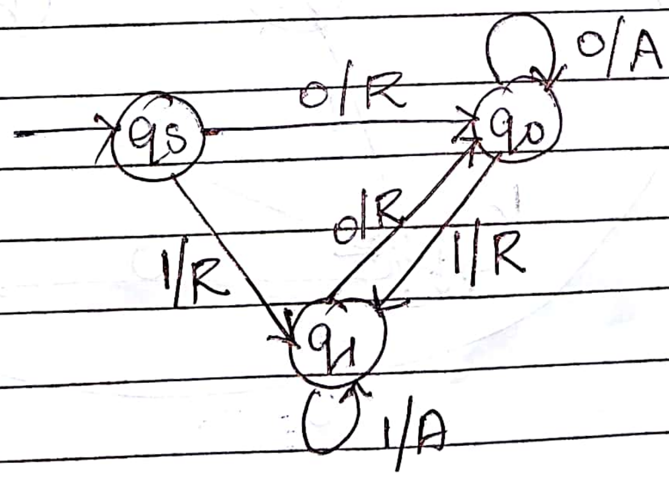
q_0

q_1

Q\Σ	0	1
q_s	R	R
q_0	A	R
q_1	R	A
q_2	A	R
q_3	R	A

δ:- Q\Σ	0	1
q_s	q_0	q_1
q_0	q_0	q_1
q_1	q_0	q_1

Q\Σ	0	1
q_s	R	R
q_0	A	R
q_1	R	A



Scanned by CamScanner

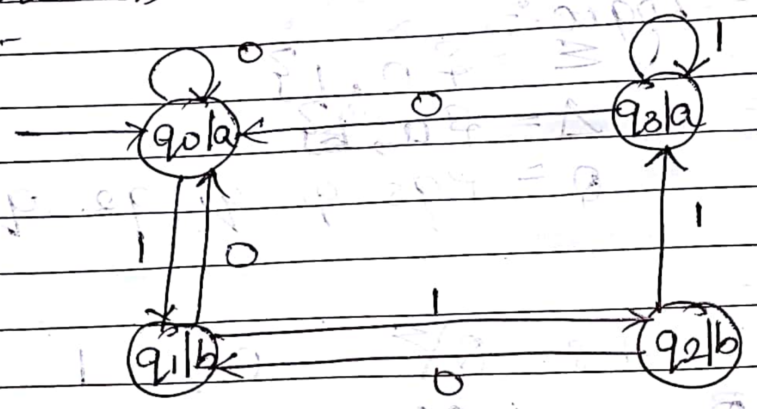
21/8/18

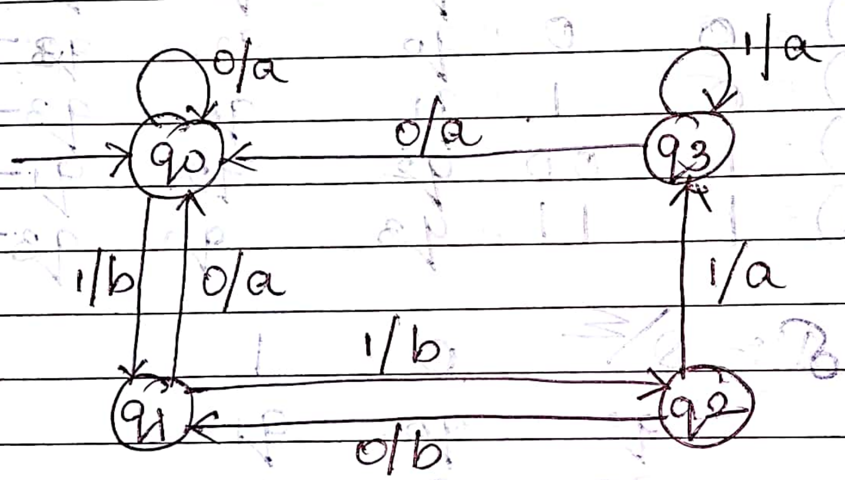
* Moore Machine to Mealy Machine

Assign the output symbol along with the state to all of its incoming transition

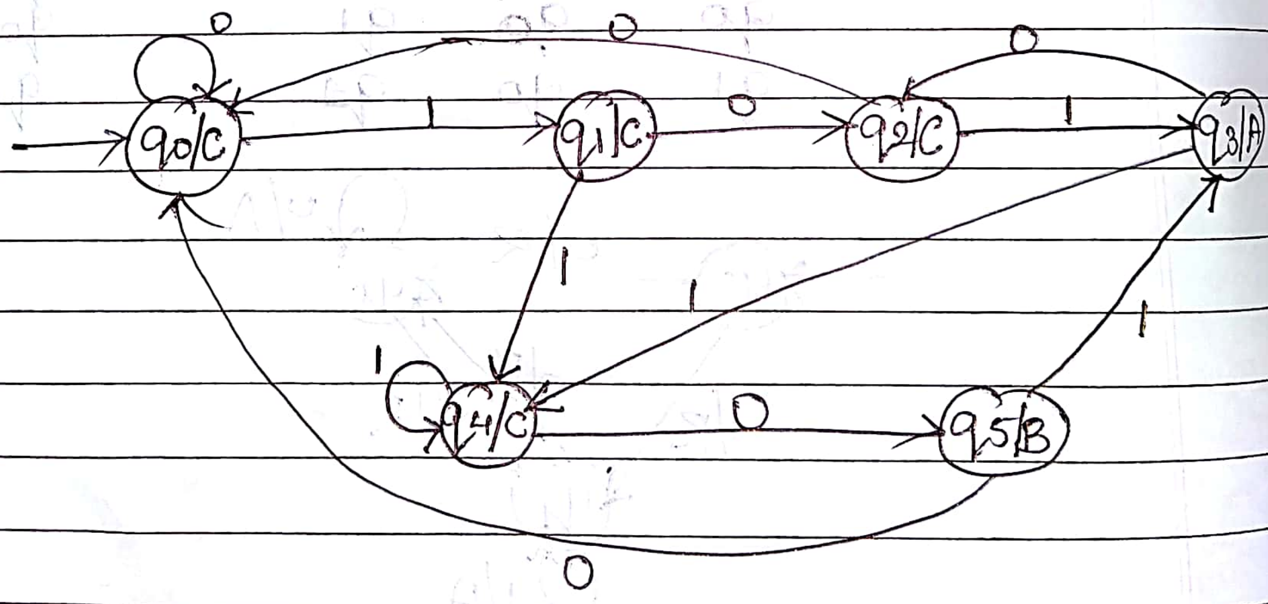
eg:-

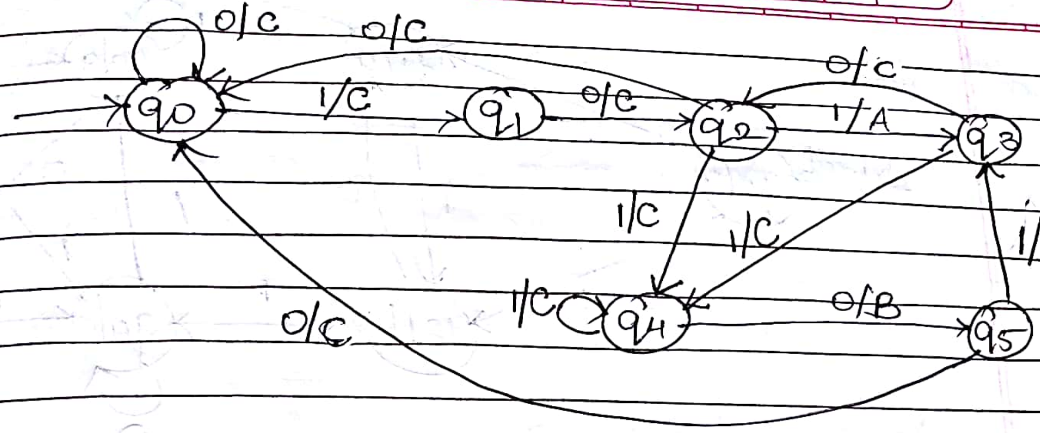
1)





2)



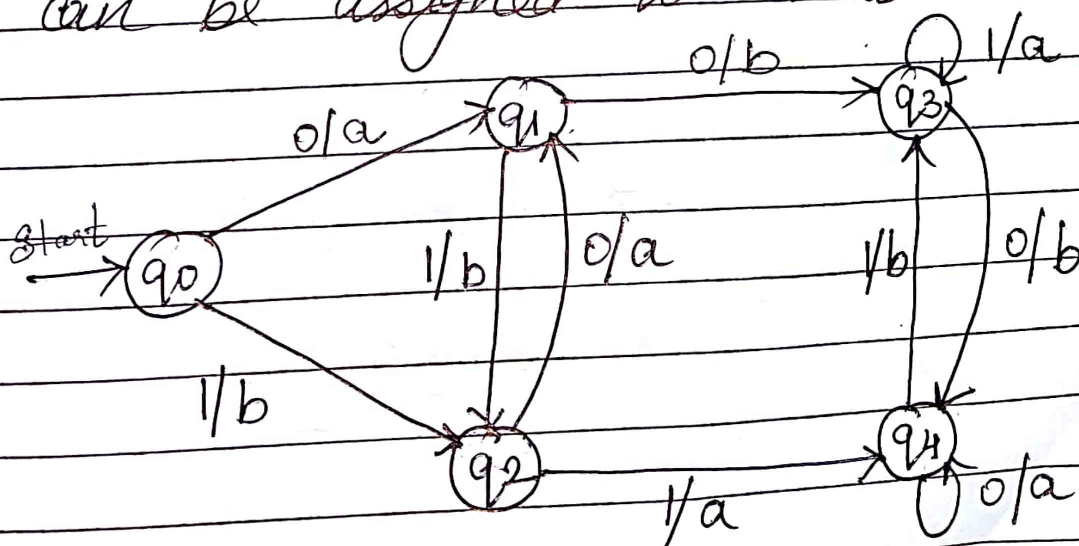


* Mealy Machine to Moore Machine :-

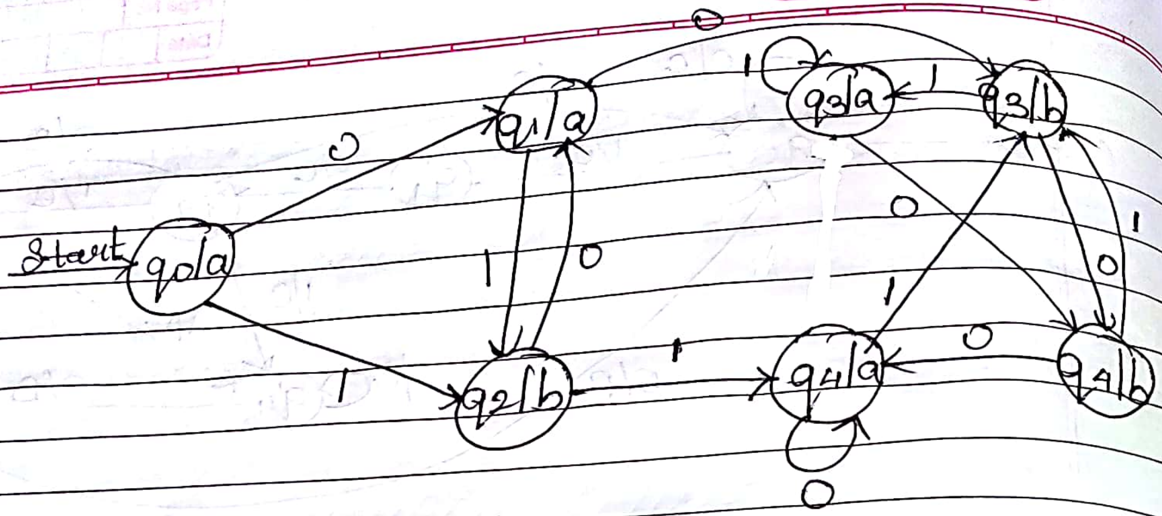
1) If the output symbol with the incoming transitions are same then assign that symbol to that state.

2) If the o/p symbols with the incoming transitions are not same, then split the state as many times as o/p symbols with each state producing a different o/p

3) If there are no incoming transitions to a state then any o/p symbol can be assigned to that state.



Page No.
Date



21/8/18

21/8/18 4. Grammers

Page No. Date

* Defination :- Grammer is used for specifying the syntax of a language & is defined as $G = (V, T, P, S)$
where, V = finite sets of variables or non-terminals.
T = finite set of terminals.
P = finite set of productions
S = ~~fun~~ start variable.

* Variable :- It is a symbol that takes part in the derivation of a sentence, but it not the part of a derived sentence. (Capital Letters).

* Terminals :- It is a symbol that is the part of derived sentence. (Small letters or Operators).

* Context Free Grammer :- Grammer is context free grammer (CFG) if all the productions are of the form

$$\boxed{A \longrightarrow \alpha}$$

where, A = variable & α = sentential form, i.e; any combinations of variable & terminals

eg:- $G = (\{S, A\}, \{a, b\}, P, S)$

$P = S \longrightarrow aSb \mid aA \mid a$

$A \longrightarrow Ab \mid a$

Date

* Derivation :- It is a process to check whether the given grammar (G) can derive or generate the given sentence using any combination of production rules starting from start variable production.

* Left-most derivation (LMD) :- Derivation is LMD, if at every step we select & replace the left-most variable.

* Right-most derivation (RMD) :- A derivation is RMD, if we select & replace the right-most variable.

Q1. $S \rightarrow aAS / a$ Derive "aabbaa" using
$A \rightarrow SbA / SS / ba$ LMD & RMD.

$\rightarrow$

LMD $\Rightarrow$ $S \underset{lm}{\Rightarrow} aAS$

$\underset{lm}{\Rightarrow} aSbAS$ using $A \rightarrow SbA$

$\underset{lm}{\Rightarrow} aabAS$ using $S \rightarrow a$

$\underset{lm}{\Rightarrow} aabbaS$ using $A \rightarrow ba$

$\underset{lm}{\Rightarrow} aabbaa$ using $S \rightarrow a$

RMD $\longrightarrow S \underset{rm}{\Longrightarrow} aAS$

$\underset{rm}{\Longrightarrow} aAa$ using $S \longrightarrow a$

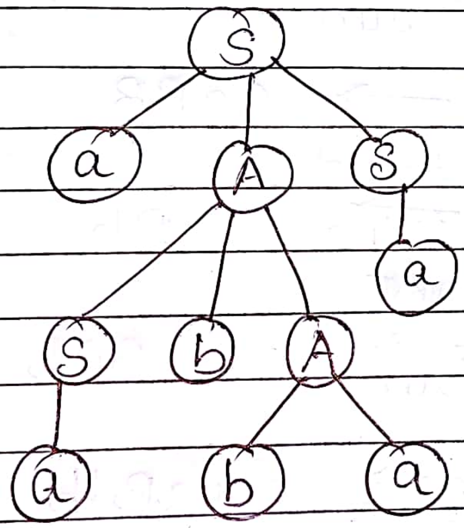
$\underset{rm}{\Longrightarrow} aSbAa$ using $A \longrightarrow SbA$

$\underset{rm}{\Longrightarrow} aSbbaa$ using $A \longrightarrow ba$

$\underset{rm}{\Longrightarrow} aabbaa$ using $S \longrightarrow a$

* Derivation / Rule / Parse Tree :-

It is the graphical representation of derivation process.



$S \longrightarrow aB / bA$

$A \longrightarrow a / aS / bAA$

$B \longrightarrow b / bS / aBB$

Derive the following using LMD & RMD

1. aaabbb
2. bbaaba
3. aaba

① aaabbb

LMD $\Longrightarrow S \underset{lm}{\Longrightarrow} aB$

$\underset{lm}{\Longrightarrow} aaBB$

$\underset{lm}{\Longrightarrow} aaaBBB$

$\underset{lm}{\Longrightarrow} aaabBB$

$\underset{lm}{\Longrightarrow} aaabbB$

$\underset{lm}{\Longrightarrow} aaabbb$

RMD $\Longrightarrow S \underset{rm}{\Longrightarrow} \cancel{bA}\ aB$

$\underset{rm}{\Longrightarrow} aaBB$

$\underset{rm}{\Longrightarrow} aaBb$

$\underset{rm}{\Longrightarrow} aaaBBb$

$\underset{rm}{\Longrightarrow} aaaBbb$

$\underset{rm}{\Longrightarrow} aaabbb$

② bbaaba

LMD $\Rightarrow S \xRightarrow[lm]{} bA$

$\xRightarrow[lm]{} bbAA$

$\xRightarrow[lm]{} bbaSA$

$\xRightarrow[lm]{} bbaaBA$

$\xRightarrow[lm]{} bbaabA$

$\xRightarrow[lm]{} bbaaba$

RMD $\Rightarrow S \xRightarrow[rm]{} bA$

$\xRightarrow[rm]{} bbAA$

$\xRightarrow[rm]{} bbAaS$

$\xRightarrow[rm]{} bbAabA$

$\xRightarrow[rm]{} bbAaba$

$\xRightarrow[rm]{} bbaaba$

③ aaba

LMD $\Longrightarrow S \xrightarrow[lm]{} aB$

$\xrightarrow[lm]{} aaBB$

$\xrightarrow[lm]{} aabB$

It is not derivable

RMD $\Longrightarrow S$

Note :- If the sentence is derivable then it can be derived using LMD & RMD both

Scanned by CamScanner

/8/18

* Example of CFG (Context Free Grammar):-

G → CFG	RE	L(G) → CFL
S → a	r = a	{a}
S → b	r = b	{b}
S → a/b	r = a/b	{a} ∪ {b} = {a, b}
S → a·b	r = ab	{a}·{b} = {ab}
S → aS/ε	$r = a^*$	{ε, a, aa, aaa, ---}
S → aS/a	$r = a^+$	{a, aa, aaa, aaaa, --}
S → abS/ε	$r = (ab)^*$	{ε, ab, abab, -----}
S → aS/bS/ε	$r = (a+b)^*$	{ε, a, b, aa, bb, ab, ba ---}

* Write CFG to generate the following:

Q1.) Set of all the strings that starts with a over Σ = {a, b}

⟹

P → S → ax
x → ax / bx / ε
G = ({S, x}, {a, b, ε}, P, S}
L(G) = {a, aa, ab, aab, aba ---}

Q2.) Set of all the strings that start & end with a different symbol over Σ = {0, 1}

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

⟹

S → 0x1 / 1x0
x → 0x / 1x / ε
G → ({S, x}, {0, 1, ε}, P, S}
L(G) → {01, 10, 001, 100, ---}

Q3) Set of all the string that starts & ends with same letter over $\Sigma = \{x, y, z\}$

$x(x+y+z)^* x + y(x+y+z)^* y + z(x+y+z)^* z$

$\Rightarrow$

$S \rightarrow xMx \mid yMy \mid zMz \mid x \mid y \mid z$

$M \rightarrow xM \mid yM \mid zM \mid \epsilon$

$G \rightarrow (\{S, M\}, \{x, y, z, \epsilon\}, P, S)$

$L(G) \rightarrow \{xx, yy, zz, xxyzx, yxyzy, zxyzz, x, y, z, \ldots\}$

Q4) Set of all the strings that contains atleast one occurance of "00" over $\Sigma = \{0, 1\}$

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

$\Rightarrow$

$S \rightarrow X00X$

$X \rightarrow 0X \mid 1X \mid \epsilon$

$G \rightarrow (\{S, X\}, \{0, 1, \epsilon\}, P, S)$

$L(G) \rightarrow \{00, 0000, 1001, 0100, 0010, 001, \ldots\}$

Q5) Set of all the strings that contains atleast 2 0's over $\Sigma = \{0, 1\}$

$\Rightarrow$

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

$S \rightarrow X0X0X$

$X \rightarrow 0X \mid 1X \mid \epsilon$

$G \rightarrow (\{S, X\}, \{0, 1, \epsilon\}, P, S)$

$L(G) \rightarrow \{00, 0001, 0000, \ldots\}$

Page No.
Date

Q6) Set of all the strings that contains exactly 2 0's over $\Sigma = \{0, 1\}$

$\Rightarrow$

$\underbrace{1^* 0 1^* 0 1^*}_{x}$

$S \to x0x0x$

$x \to 1x / \epsilon$

$G \to (\{S, x\}, \{0, 1, \epsilon\}, P, S)$

$L(G) \to \{00, 10101, 100, 001, ----\}$

Q7) Set of all the strings that contains atmost 2 0's over $\Sigma = \{0, 1\}$

$\Rightarrow$

$\underbrace{1^* + 1^* 0 1^* + 1^* 0 1^* 0 1^*}_{x}$

$S \to x / x0x / x0x0x$

$x \to 1x / \epsilon$

$G \to (\{S, x\}, \{0, 1, \epsilon\}, P, S)$

$L(G) \to \{1, 00, 0, 10, 01, 1010, ----\}$

Q8) $L = \{a^n b^n / n \geq 1\}$

$\Rightarrow$

$n = 1 \to ab$

$n = 2 \to aabb$

$n = 3 \to aaabbb$

$S \to ab / aSb$

OR

$S \to axb$

$x \to axb / \epsilon$

$G \to (\{S, x\}, \{a, b, \epsilon\}, P, S)$

$L(G) \to \{ab, aabb, aaabbb, ----\}$

*** $\boxed{n = 0 \quad \epsilon \qquad S \to aSb / \epsilon, \; n \geq 0}$

28/8/18

Q9) $L = \{a^n b^{3n} / n \geq 0\}$

$n=0 = \epsilon$
$n=1 = abbb$ ×
$n=2 = aabbbbbb$

$\Rightarrow$

$S \rightarrow axbbb$
$x \rightarrow axbbb / \epsilon$
$G \rightarrow (\{S, x\}, \{a, b, \epsilon\}, P, S)$
$L(G) \rightarrow \{abbb, aabbbbbb, aaabbbbbbbbb \ldots\}$

Q10) $L = \{a^n b^{n+1} / n \geq 1\}$

$n=1 = abb$
$n=2 = aabbb$
$n=3 = aaabbbb$

$\Rightarrow$

$S \rightarrow axbb$
$x \rightarrow axb / \epsilon$
$G \rightarrow (\{S, x\}, \{a, b, \epsilon\}, P, S)$
$L(G) \rightarrow \{abb, aabbb, aaabbbb \ldots\}$

Q11) $L = \{a^m b^{m-3} / m \geq 3\}$

$\Rightarrow$

$n=3 = aaa$
$n=4 = aaaab$
$n=5 = aaaaabb$

$S \rightarrow aaax$
$x \rightarrow axb / \epsilon$
$G \rightarrow (\{S, x\}, \{a, b, \epsilon\}, P, S)$
$L(G) \rightarrow \{aaa, aaaab, aaaaabb, \ldots\}$

Q12) $L = \{a^i b^j / i, j \geq 1\}$

$\Rightarrow$

$S \rightarrow AB$
$A \rightarrow aA / a$
$B \rightarrow bB / b$
$G \rightarrow (\{S, A, B\}, \{a, b\}, P, S)$
$L(G) \rightarrow \{ab, abb, aabb, aaabbb \ldots\}$

$i=1, j=1 \rightarrow ab$
$i=2, j=2 \rightarrow aabb$
$i, j = 3 \rightarrow aaabbb$

Q13)

Q14)

Q13) $L = \{a^i b^j c^k \mid i,j,k \geq 1\}$

$\Rightarrow$

$S \to ABC$

$A \to aA/a$

$B \to bB/b$

$C \to cC/c$

$L(G) = \{abc, aabbcc, aaabbbccc \ldots\}$

$G \to (\{S,A,B,C\},\{a,b,c\},P,S)$

Q14) $L = \{a^i b^j c^k \mid \underline{i=j},\ i,j,k \geq 1\}$

$\Rightarrow$

$(ab)^+ c^+$

$\underbrace{a^i b^j}_{X} \ \underbrace{c^k}_{Y}$

$i=j=1, k=2 \to abcc$

$i=j=2, k=3 \to aabbccc$

$S \to XY$

$X \to aXb/ab$

$Y \to cY/c$

or

$S \to aXbC$

$X \to aXb/\epsilon$

$C \to cC/c$

$G \to (\{S, X, C\}, \{a, b, c, \epsilon\}, P, S)$

$L(G) \to \{abc, aabb, c, abcc, aabbccc \ldots\}$

Q15) $L = \{a^i b^j c^k \mid j=k,\ i,j,k \geq 1\}$

$\Rightarrow$

$a^+ (bc)^+$

$S \to AbXC$

$A \to aA/a$

$X \to bXc/\epsilon$

$G \to (\{S, A, X\}, \{a, b, c, \epsilon\}, P, S)$

$L(G) \to \{abc, aabc, aaabbcc, \text{---}\}$

Q16) $L = \{a^i b^j c^k \mid j=k+i,\ i,k \geq 1\}$

$\Rightarrow$

$S \to XY$

$X \to aXb/ab$

$Y \to bXc/bc$

$G \to (\{S, X, Y\}, \{a, b, c\}, P, S)$

$L(G) \to \{abbc, aabbbbcc, \ldots\ldots\}$

$a^i b^{k+i} c^k$

$b^k \quad b^i$

$a^i b^i \quad b^k c^k$

$\downarrow \quad \downarrow$

$X \quad Y$

$\downarrow \quad \downarrow$

$ab \quad bc$

Scanned by CamScanner

Q17) $L = \{a^i b^j c^k \mid i = j \text{ OR } j = k,\ i, j, k \geq 1\}$

$\{(ab)^+ c^+\}\{a^+(bc)^+\}$ → Q20

$\Rightarrow$ $i = j$

$S \to aXbC$

$X \to aXb / \epsilon$

$C \to cC / c$

$G \to (\{S, X, C\}, \{a, b, c, \epsilon\}, P, S)$

or $j = k$

$S \to A\,bXc$

$A \to aA / a$

$X \to bXc / \epsilon$

$G \to (\{S, A, X\}, \{a, b, c, \epsilon\}, P, S)$

Q18)

$\underbrace{a^i b^j}_{X}\ \underbrace{c^k}_{Y}$

$X \to axb/ab$ $\quad Y \to cY/c$

$\underbrace{a^i}_{A}\ \underbrace{b^j c^k}_{B}$

$A \to aA/a$ $\quad B \to bBc / bc$

$S \to XY / AB$

$X \to axb/ab$

$Y \to cY/c$

$A \to aA/a$

$B \to bBc/bc$

Q18) Even palindrome over $\Sigma = \{a, b\}$

Q19) Odd palindrome over $\Sigma = \{a, b\}$

Q20) Palindrome over $\Sigma = \{a, b\}$

$\Rightarrow$ Q18)

$S \to \epsilon / aSa / bSb$

$G \to (\{S\}, \{a, b, \epsilon\}, P, S)$

$L(G) \to \{\epsilon, aa, bb, abba, abbbba, baab, baaaab \ldots\}$

ϵ
aa
bb
abba
abbbba
baab
baaaab

$\Rightarrow$ Q19)

$S \to aSa / bSb / a / b$

$G \to (\{S\}, \{a, b\}, P, S)$

$L(G) \to \{aaa, bbb, a, b, aba, bab \ldots\}$

a
b
aba
bab

Q20.)

$S \rightarrow \epsilon / aSa / bSb / a / b$

$G \rightarrow (\{S\}, \{a, b, \epsilon\}, P, S\}$

$L(G) \rightarrow \{aa, a, abba, b, bb, aaa, bbb \ldots\}$

29/8/18

V.V. Imp.

(10 marks)

* CHOMSKY'S HIERARCHY / TYPES OF GRAMMAR

As per Chomsky there are four different types of grammars which have been classified based on the restrictions placed on the production rules.

Types	Grammar	Language	Restrictions	Examples
0	Unrestricted Grammar	Recursively Enumerable Grammar	$\alpha \rightarrow \beta$	$S \rightarrow AB$ $AB \rightarrow AC$ $A \rightarrow a$ $B \rightarrow b$ $C \rightarrow a$
1.	Context Sensitive Grammar	Context Sensitive Language (CSL)	$\lvert\alpha\rvert \leq \lvert\beta\rvert$	$S \rightarrow aA$ $aA \rightarrow aaA$ $aA \rightarrow bA$ $A \rightarrow a$
2.	Context Free Grammar	Context Free Language (CFL)	$A \rightarrow \alpha$	$S \rightarrow aSb \mid aA \mid a$ $A \rightarrow Ab \mid b$
.	Regular Grammar	Regular Language	A → any no. of T's atmost 1 V	LLG :- $S \rightarrow Aa \mid b$ $A \rightarrow ba \mid Ab \mid a$ RLG :- $S \rightarrow aB \mid b$ $B \rightarrow bB \mid ab \mid a$

LLG :- Left Linear Grammar

RLG :- Right Linear Grammar

Page No.
Date

* Unambiguous & Ambiguous Grammar :-

1) Unambiguous Grammar :-

Grammar is unambiguous if it can derive all sentences using exactly one LMD or RMD

2) Ambiguous Grammar :-

Grammar is ambiguous if it can derive atleast one sentence using more than one LMD or RMD.

For eg :-

$$E \rightarrow E+E \mid E*E \mid id$$

Derive " id + id * id using LMD

1) $S \rightarrow E+E$
$\rightarrow id + E$ using $E \rightarrow id$
$\rightarrow id + E * E$ using $E \rightarrow E*E$
$\rightarrow id + id * E$ using $E \rightarrow id$
$\rightarrow id + id * id$ using $E \rightarrow id$

2) $S \rightarrow E * E$
~~$\rightarrow E\ id * E$~~
~~$\rightarrow$ id~~
$\rightarrow E + E * E$ using $E \rightarrow E+E$
$\rightarrow id + E * E$ using $E \rightarrow id$
$\rightarrow id + id * E$ using $E \rightarrow id$
$\rightarrow id + id * id$ using $E \rightarrow id$

Since G can derive the sentence using more than 1 LMD, G is ambiguous. To eliminate ambiguity we rewrite the grammar as

$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow id$

$id + id * id$

$S \rightarrow E + T$	using $E \rightarrow T$
$\rightarrow T + T$	using $T \rightarrow F$
$\rightarrow F + T$	using $T \rightarrow T * F$
$\rightarrow F + T * F$	using $F \rightarrow id$
$\rightarrow id + T * F$	using $T \rightarrow F$
$\rightarrow id + F * F$	using $F \rightarrow id$
$\rightarrow id + id * F$	using $F \rightarrow id$
$\rightarrow id + id * id$	using $F \rightarrow id$

* Simplification of Context Free Grammar

I) Elimination of useless productions.

- **Def. of useless variable :-**
  A variable 'X' is useful if
  $S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$ else X is useless.

- **Def. of useless production :-**
  A production where useless variable is present is called useless production.

- Elimination Procedure :-

Given CFG, $G = (V, T, P, S)$.

Define $G' = (V', T', P', S)$.

Be the CFG that does not contains any useless productions such that $L(G') = L(G)$.

Step 1 :- initialize P' to P

Step 2 :- find useless variables.

2.1 :- variable is useless if it cannot derive any sentence (i.e; any combination of terminals)

2.2 :- variable is useless if it is not reachable from 'S'

Step 3 :- delete all useless productions.

Q. Eliminate the useless production from the given grammar :-

$S \rightarrow aSb \mid b \mid aAB$

$A \rightarrow aA \mid b$

$B \rightarrow bB \mid cB$

$\Rightarrow$

Given $G = (\{S, A, B\}, \{a, b, c\}, P, S)$

$P \rightarrow S \rightarrow aSb \mid b \mid aAB$

$A \rightarrow aA \mid b$

$B \rightarrow bB \mid cB$

4/9/18

Define $G' = (V', T', P', S)$ be the CFG, that does not contain any useless production such that $L(G') = L(G)$

$P' = S \rightarrow aSb \mid b \mid \cancel{aAB}$

$A \rightarrow aA \mid b$

$B \rightarrow \cancel{bB} \mid \cancel{cB}$

$\because$ B is not deriving any sentence, it is useless.

$S \rightarrow aSb \mid b$

$A \rightarrow \cancel{aA} \mid \cancel{b}$

$\because$ A is not reachable from start variable, it is useless.

$P' = S \rightarrow aSb \mid b$

$\therefore G' = (\{S\}, \{a, b\}, P', S)$

29/8/18

4/9/18

(i) Elimination of Unit Productions :-

* Def. of Unit Productions :-

A production of the form "X → Y" where X & Y are variables is called unit productions.

* Def. of Non-Unit Productions :-

A production of the form "X → Y" where X is a variable, Y is the combination of V's & T's is called non-unit production.

* Elimination Procedure :-

Given CFG, G = (V, T, P, S).
Define G' = (V', T', P', S)
Be the CFG, that does not contain any unit productions such that L(G') = L(G)

Step 1 :- initialize P' to P.

Step 2 :- find unit productions.

* Say "X → Y" is a unit production of X, then add to X, the non-unit productions of Y not present in X.

Step 3:- delete all unit productions.

Q) Eliminate the unit productions from the given grammar. $S \to aSb/a/A$
$A \to Ab/a/b$

⟹ Given CFG, $G = (V, T, P, S)$.
Define $G' = (V', T', P', S')$ be the CFG, that does not contain any unit production such that $L(G') \subseteq L(G)$

Production	New Production
	—
$A \to b$	—
$A \to a$	—
$A \to Ab$	$S \to b$, $S \to Ab$
$S \to A$	—
$S \to a$	—
$S \to aSb$	

Production + New Production

$S \to aSb / a / \cancel{A} / b / Ab$
$A \to Ab / a / b$

After deleting the unit production

$P' = S \to aSb / a / b / Ab$
$A \to Ab / a / b$

2)
$S \to aS / A / D$
$A \to a$
$B \to aa$
$D \to aDb$

⇒ Given CFG, $G = (V, T, P, S)$.
Define $G' = (V', P', T', S)$ ..... $L(G') = L(G)$

Production	New Production
$D \to aDb$	—
$B \to aa$	—
$A \to a$	—
$S \to D$	$S \to aDb$
$S \to A$	$S \to a$
$S \to aS$	—

Production + New Production
$S \to aS / \cancel{A} / \cancel{D} / aDb / a$
$A \to a$
$B \to aa$
$D \to aDb$

After deleting the unit Production

$P' = S \to aS / \cancel{aDb} / a$
$\left.\begin{array}{l} \cancel{A \to a} \\ \cancel{B \to aa} \end{array}\right\}$ useless
$\cancel{D \to aDb}$

Here A & B are not reachable by starting variable S, it is useless

$P' \rightarrow S \rightarrow aS / a / \cancel{aDb}$

$D \rightarrow \cancel{aDb}$

Here D can not ~~be~~ derive any sentence, it is useless.

$P' = S \rightarrow aS / a$

$G' = (\{S\}, \{a\}, P', S)$

III.) Elimination of Null Productions :-

* Def. of null production :-

A production of the form "$X \rightarrow \epsilon$" OR "$X \rightarrow \lambda$" where X is a variable, is called null production.

* Def. of nullable variable :-

A variable X is nullable if "$X \rightarrow \epsilon$" or "$X \xRightarrow{*} \epsilon$".

* Elimination procedure :-

Given CFG, $G = (V, T, P, S)$

Define $G' = (V', T', P', S)$ be the CFG that does not contains null productions such that $L(G') = L(G) - \{\epsilon\}$ (ϵ free CFL)

Step 1 :- initialize P' to P

Step 2 :- find nullable variables.

Say "$X \to \alpha$" is the production of X such that α contains some nullable variables, then add to X, the productions obtained by deleting all possible subsets of nullable variables from α. (if not present in X)

Step 3 :- delete all null productions.

Q1) Eliminate null productions from the given grammar.

$S \to aSb \mid a \mid aAB$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$

$\Rightarrow$

Given CFG, $G = (V, T, P, S)$.
Define $G' = (V', T', P', S)$ be the CFG that does not contains null production such that $L(G') = L(G) - \{\epsilon\}$

Production	New Production
$B \to \epsilon$	—
$B \to bB$	$B \to b$
$A \to \epsilon$	—
$A \to aA$	$A \to a$
$S \to aAB$	$S \to aB$, $S \to aA$
$S \to a$	—
$S \to aSb$	—

Production + New Production

$S \to aSb \,/\, a \,/\, aAB \,/\, aB \,/\, aA$

$A \to aA \,/\, a \,/\, \cancel{\epsilon}$

$B \to bB \,/\, b \,/\, \cancel{\epsilon}$

After deleting null production

$S \to aSb \,/\, a \,/\, aAB \,/\, aA \,/\, aB$

$A \to aA \,/\, a$

$B \to bB \,/\, b$

* Normal Forms :-

Norm forms implies the rewriting of CFG as per the specified conditions.

I) Chomsky Normal Form (CNF)

* Def :- Any ϵ- free CFL can be generated

by CFG in which all the productions are of the form "A → BC" or "A → a", where A B C are variables & a is terminal. Such a CFG is ~~said~~ set to be in CNF

* Conversion procedure from CFG to CNF.

Step 1:- Perform elimination of null, unit, useless productions

Step 2:- Add to the solutions the productions that are already in CNF.

Step 3:- For the remaining Non-CNF productions
- replace the terminals by some variables
- limit the number of variables on the RHS to 2.

Q1.) Express CFG in CNF

$S \to aB / bA$

$A \to a / aS / aBB$

$B \to b / bS / aAAB.$

⇒

Step 1:- done

Step 2 & 3 :-

5/9/1

Production	New Production
$A \to a$	–
$B \to b$	–
$B \to bS$	$C_1 \to b$
$B \to C_1S$	–
$B \to aAAB$	$C_2 \to a$
$B \to C_2AAB$	$C_3 \to AAB$
$B \to C_2C_3$	–
$A \to aS$	$C_2 \to a$
$A \to C_2S$	–
$A \to aBB$	$C_2 \to a$
$A \to C_2BB$	$C_4 \to BB$
$A \to C_2C_4$	–
$S \to bA$	$C_1 \to b$
$S \to C_1A$	–
$S \to aB$	$C_2 \to a$
$S \to C_2B$	–
$C_1 \to b$	–
$C_2 \to a$	–
$C_3 \to AAB$	$C_5 \to AB$
$C_3 \to AC_5$	–
$C_4 \to BB$	–
$C_5 \to AB$	–

$S \to C_2B / C_1A$

$A \to a / C_2S / C_2C_4$

$B \to b / C_1S / C_2C_3$

$C_1 \to b$

$C_2 \to a$

$C_3 \to AC_5$

$C_4 \to BB$

$C_5 \to AB$

5/9/18

1) $S \to 00S \mid 11S \mid 0 \mid 1 \mid \lambda \backslash \wedge \backslash \epsilon$

2) $S \to \sim S$
$S \to [S \supset S]$
$S \to p$
$S \to q$

1) $S \to 00S \mid 11S \mid 0 \mid 1 \mid \lambda \backslash \wedge \backslash \epsilon$

Given CFG $G = (V, T, P, S)$ - - - - $L(G') = L(G) - \{\epsilon\}$

Step 1:-

Production	New Production
$S \to \lambda$	—
$S \to 1$	—
$S \to 0$	—
$S \to 11S$	$S \to 11$
$S \to 00S$	$S \to 00$

$P + NP = S \to 00S \mid 11S \mid 0 \mid 1 \mid \lambda \mid 00 \mid 11$
$P' = S \to 00S \mid 11S \mid 0 \mid 1 \mid 00 \mid 11$

Step 2 & 3:-

Production	New Production
$S \to 0$	—
$S \to 1$	—
$S \to 00$	$C_1 \to 0$
$S \to C_1C_1$	—
$S \to 11$	$C_2 \to 1$
$S \to C_2C_2$	—
$S \to 11S$	$C_2 \to 1$
$S \to C_2C_2S$	$C_3 \to C_2C_2$
$S \to C_3S$	—

	$C_1 \to 0$
$S \to 00S$	$C_4 \to C_1C_1$
$S \to C_1C_1S$	–
$S \to C_4S$	–
$C_1 \to 0$	–
$C_2 \to 1$	–
$C_3 \to C_2C_2$	–
$C_4 \to C_1C_1$	

CFG in CNF

$S \to 0/1 / C_1C_1 / C_2C_2 / C_3S / C_4S$

$C_1 \to 0$

$C_2 \to 1$

$C_3 \to C_2C_2$

$C_4 \to C_1C_1$

Q.2) $S \to \sim S$

$S \to [S \supset S]$

$S \to p$

$S \to q$

$\Rightarrow$ Step 1 :- —

Step 2 :-

Production	New Production
$S \to p$	–
$S \to q$	–
$S \to \sim S$	$C_1 \to \sim$
$S \to C_1S$	
$S \to [S \supset S]$	$C_2 \to [$, $C_3 \to \supset$, $C_4 \to]$
$S \to C_2SC_3SC_4$	$C_5 \to C_2SC_3S$
$S \to C_5C_4$	–
$C_1 \to \sim$	—

$C_2 \to [$		
$C_3 \to)$		
$C_4 \to]$		
$C_5 \to C_2SC_3S$	$C_6 \to C_2SC_3$	
$C_5 \to C_6S$		
$C_6 \to C_2SC_3$	$C_7 \to C_2S$	
$C_6 \to C_7C_3$		
$C_7 \to C_2S$		

CFG $\to$ CNF :-

$S \to p/q/C_1S/C_5C_4$
$C_1 \to \sim$
$C_2 \to [$
$C_3 \to)$
$C_4 \to]$
$C_5 \to C_6S$
$C_6 \to C_7C_3$
$C_7 \to C_2S$

* $S \to bA/aB$
$A \to bAA/aS/a$
$B \to aBB/bS/b$

$\Rightarrow$

Step 1 :- —

Step 2 :-

Production	New Production
$B \to b$	
$B \to bS$	$C_1 \to b$
$B \to C_1S$	
$B \to aBB$	$C_2 \to a$
$B \to C_2BB$	$C_3 \to C_2B$

	–
$B \to C_3B$	–
$A \to a$	$C_2 \to a$
$A \to aS$	–
$A \to C_2S$	$C_1 \to b$
$A \to bA\cancel{A}$	$C_4 \to C_1A$
$A \to C_1AA$	
$A \to C_4A$	$C_2 \to a$
$S \to aB$	
$S \to C_2B$	$C_1 \to b$
$S \to bA$	
$S \to C_1A$	
$C_1 \to b$	
$C_2 \to a$	
$C_3 \to C_2B$	
$C_4 \to C_1A$	

CFG $\to$ CNF

$S \to C_2B \,/\, C_1A$

$A \to a \,/\, C_2S \,/\, C_4A$

$B \to b \,/\, C_1S \,/\, C_3B$

$C_1 \to b$

$C_2 \to a$

$C_3 \to C_2B$

$C_4 \to C_1A$

Scanned by CamScanner

* $S \to ABC / BaB$
  $A \to aA / BaC / aaa$
  $B \to bBb / a / D$
  $C \to CA / AC$
  $D \to \epsilon$

Step 1 :- Def. of null production.
To eliminate null productions.

Production	New Production
$D \to \epsilon$	—
$C \to CA$	-
$C \to AC$	—
$B \to D$	$B \to \epsilon$
~~$B \to \epsilon$~~	—
$B \to a$	—
$B \to bBb$	—

P+NP:-

$S \to ABC / BaB$
$A \to aA / BaC / aaa$
$B \to bBb / a / \cancel{D} / \epsilon$
$C \to CA / AC$
~~$D \to \epsilon$~~

After eliminating ϵ production

$S \to ABC / BaB$
$A \to aA / BaC / aaa$
$B \to bBb / a / \epsilon$
$C \to CA / AC$

Production	New Production
$C \to AC$	–
$C \to CA$	–
$B \to \epsilon$	–
$B \to a$	–
$B \to bBb$	$B \to bb$
$A \to aa$	–
$A \to BaC$	$A \to aC$
$A \to aA$	–
$S \to BaB$	$S \to Ba$, $S \to aB$
$S \to ABC$	$S \to AC$

P+NP:-

$S \to ABC / AC / BaB / Ba / aB$
$A \to aA / BaC / aC / aaa$
$B \to bBb / a / \cancel{\epsilon} / bb$
$C \to CA / AC$

After eliminating ϵ Production

$S \to \cancel{ABC} / \cancel{AC} / BaB / Ba / aB$
$A \to aA / \cancel{BaC} / \cancel{aC} / aaa$
$B \to bBb / a / bb$
$\cancel{C} \to \cancel{CA} / \cancel{AC}$

Eliminate useless production. C is useless
After elimination

$S \to BaB / Ba / aB$
$\cancel{A} \to \cancel{aA} / \cancel{aaa}$
$B \to bBb / bb / a$

'A' is not reachable from start variable.
So, A is useless production.
Eliminating useless production:

$S \rightarrow BaB / Ba / aB$
$B \rightarrow bBb / bb / a$
} — CFG

Step 2:-

Production	New Production
$B \rightarrow a$	-
$B \rightarrow bb$	$C_1 \rightarrow b$
$B \rightarrow C_1C_1$	-
$B \rightarrow bBb$	$C_1 \rightarrow b$
$B \rightarrow C_1BC_1$	$C_2 \rightarrow C_1B$
$B \rightarrow C_2C_1$	-
$S \rightarrow aB$	$C_3 \rightarrow a$
$S \rightarrow C_3B$	-
$S \rightarrow Ba$	$C_3 \rightarrow a$
$S \rightarrow BC_3$	-
$S \rightarrow BaB$	$C_3 \rightarrow a$
$S \rightarrow BC_3B$	$C_4 \rightarrow BC_3$
$S \rightarrow C_4B$	-
$C_1 \rightarrow b$	-
$C_2 \rightarrow C_1B$	-
$C_3 \rightarrow a$	-
$C_4 \rightarrow BC_3$	-

Final Production:-

$S \rightarrow C_3B / BC_3 / C_4B$
$B \rightarrow a / C_1C_1 / C_2C_1$
$C_1 \rightarrow b$
$C_2 \rightarrow C_1B$
$C_3 \rightarrow a$
$C_4 \rightarrow BC_3$

* GREIBACH NORMAL FORM (GNF):-

Def:- Any ϵ-free CFL can be generated by CFG in which all the productions are of the form $\boxed{A \rightarrow a\gamma}$ → (Gamma)

where A - variable
a - Terminal
γ - String of variables (can be empty)

Such a CFG is said to be in GNF

* Conversion ~~proced~~ procedure from CFG to GNF

Step 1:- Perform elimination of unit & useless productions.

Step 2:- Perform any combinations of rule 1 & rule 2 to obtain the given CFG in GNF

- Rule 1:- Let $A \rightarrow B\alpha$ be some A-production &

Let $B \rightarrow \beta_1 | \beta_2 | \cdots \beta_n$ be B-productions

then we can write A-productions as
$A \rightarrow \beta_1\alpha | \beta_2\alpha | \cdots | \beta_n\alpha$

- Rule 2:- Let $A \rightarrow A\alpha_1 | A\alpha_2 | \cdots | A\alpha_r$ be A-productions &

Let $B \to \beta_1 / \beta_2 / \dots / \beta_s$ be remaining A-productions

then introduce a variable B

B- productions $B \to \alpha_i / \alpha_i B$ $(1 \le i \le r)$

A - productions $A \to \beta_i / \beta_i B$ $(1 \le i \le s)$

* Expression the CFG in GNF

1) $S \to AA / AB / BA$

$A \to aA / b$

$B \to bB / a$

$\Rightarrow$

$B \to a$

$B \to bB$

$A \to b$

$A \to aA$

$S \to BA$

$A \to B \quad \alpha$ R_1

$S \to bBA / aA$

$S \to AB$

$A \to B \quad \alpha$ R_1

$S \to aAB / bB$

$S \to AA$

$A \to B \quad \alpha$ R_1

$S \to aAA / bA$

$R_1 \to X \to Y\alpha$

2) $A_1 \to A_2A_2 / a$
$A_2 \to A_1A_1 / b$

$\Longrightarrow$

$A_2 \to A_1 A_1 / b$
$X \to Y \alpha \to R_1$

$A_2 \to A_2 \underbrace{A_2 A_1} / aA_1 / b$
$A \to A \alpha_i \beta_1 \beta_2 \to R_2$

$B \to \alpha_i / \alpha_i B$
$B \to A_2A_1 / A_2A_1B$
GNF $\to$ $A_2 \to aA_1 / aA_1B / b / bB$
$\beta_1 \quad \beta_1 B \quad \beta_2 \quad \beta_2 B$
R_2

Substituting A_2 in A_1 & B

$A_1 \to aA_1A_2 / aA_1BA_2 / bA_2 / bBA_2 / a$
$B \to aA_1A_1 / aA_1BA_1 / bA_1 / bBA_1 / aA_1A_1B / aA_1BA_1B / bA_1B / bBA_1B$

3) $S \to AA / 0$
$A \to SS / 1$

$\Longrightarrow$

$A \to SS / 1$
$X \to Y \alpha$ R_1

$A \to A\underbrace{AS} / 0S / 1$ R_2
$A \to A \alpha_i \beta_1 \beta_2$

$B \to \alpha_i \,/\, \alpha_i B$

$B \to AS \,/\, ASB$

$A \to 0S \,/\, 0SB \,/\, 1 \,/\, 1B$ } (R2)

Substitution of A in S & B

$S \to 0SA \,/\, 0SBA \,/\, 1A \,/\, 1BA \,/\, 0$

$B \to 0SS \,/\, 0SSB \,/\, 0SBS \,/\, 0SBSB \,/\, 1S \,/\, 1SB \,/\, 1BS \,/\, 1BSB$

4) $S \to aSa \,/\, bSb \,/\, c$

$S \to aSa \,/\, bSb \,/\, c$

$C_1 \to a \quad C_2 \to b$

$S \to aSC_1 \,/\, bSC_2 \,/\, c$

$C_1 \to a$

$C_2 \to b$

{ Selective Replacement }

5) $S \to SS \,/\, aSb \,/\, ab$

$\quad \alpha \quad \alpha \;\; \alpha$

$S \to SS \,/\, aSb \,/\, ab$

$A \quad A \;\; \alpha_i \quad \beta_1 \quad \beta_2$

$A = S$

$\alpha_i = S$

$\beta_1 = aSb$

$\beta_2 = ab$

$B \to \alpha_i \,/\, \alpha_i B$

$B \to S \,/\, SB$

$S \to aSb \,/\, aSbB \,/\, ab \,/\, abB$

(Selective Replacement)

$b \to C_1$

$A \to aSC_1 \,/\, aSC_1B \,/\, aC_1 \,/\, aC_1B \to$ GNF

$B \to aSC_1 \,/\, aSC_1B \,/\, aC_1 \,/\, aC_1B \,/\, aSC_1B \,/\, aSC_1BB \,/\, aC_1BB$

* Rules for GNF :-

1.) Check that the variable on the LHS is not repeated as the first character of the RHS production.

2.) If rule 1 is true then use rule 1 of GNF.

If rule 1 is false (variable is repeated on RHS) then use rule 2 of GNF.

3) If all the productions of a Grammar starts with a terminal & are followed by the combination of variable & terminals then use selective replacement.

* Generate and reduce the following into CNF and GNF

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

$\Longrightarrow$

ab

aabb

aaabbb

$S \to aSb / ab$

~~$X \to aXb / \epsilon$~~

~~$G \to (\{S, X\}$~~, $\{a, b\}$, ~~$\{S\}, P, S)$~~

Production	New Production
$S \to ab$	$C_1 \to a$
$S \to C_1 b$	$C_2 \to b$
$S \to C_1 C_2$	
$S \to aSb$	
$S \to C_1 S b$	$C_1 \to a$
$S \to C_1 S C_2$	$C_2 \to b$
	$C_3 \to C_1 S$

Page No.
Date

$S \rightarrow C_3C_2$	–	
$C_1 \rightarrow a$	–	
$C_2 \rightarrow b$	–	
$C_3 \rightarrow C_1S$	–	

P+NP =

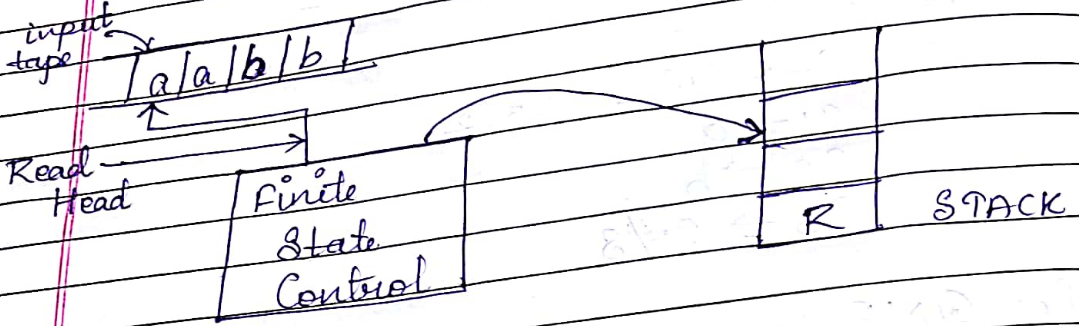
$S \rightarrow C_1C_2 \,/\, C_3C_2$
$C_1 \rightarrow a$
$C_2 \rightarrow b$
$C_3 \rightarrow C_1S$

For GNF :-
by selective replacement:

$S \rightarrow ab \,/\, aSb$
$C_1 \rightarrow b$
$S \rightarrow aC_1 \,/\, aSC_1$
$C_1 \rightarrow b$

8/9/18 Chp 5. Push-Down Automata (PDA)

PDA is used for recognizing context free language which is generated by CFG



MODEL OF PDA

* Components of PDA :-

PDA consists of finite set of states, input tape, read head & a stack.

- Working of PDA :-

Depending on the state, input symbol & stack top symbol

1) PDA can change the state or remain in the same state.

2) PDA moves the read head to the right of the current cell.

3) PDA can perform some stack operations.

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

where

Q = Finite set of states

Σ = Input alphabet

Γ = Stack alphabet

δ = Transition function

$\delta :- Q \times \Sigma \cup \{\epsilon\} \times \Gamma \longrightarrow$ subset of $Q \times \Gamma^*$

q_0 = Start state

$q_0 \in Q$

R = Initial Stack top symbol

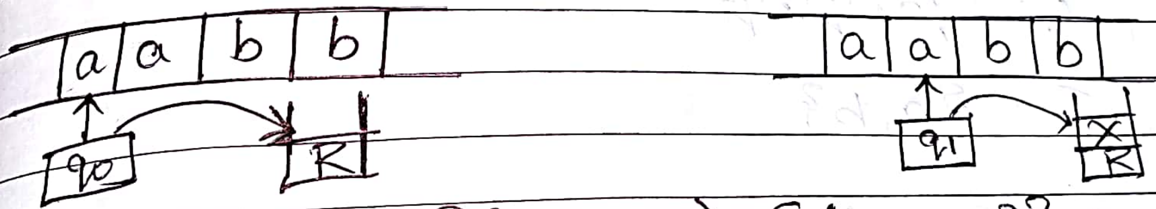
$R \in \Gamma \quad (Z_0 = R)$

F = Finite set of final state.

Some examples of transition functions :-

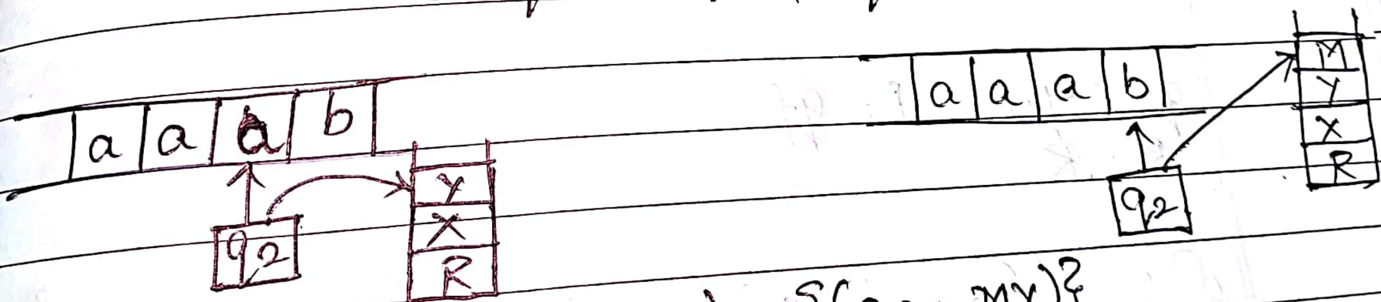
Combination   Action

1.



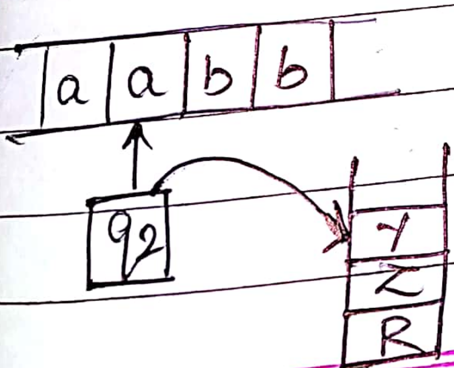
$\delta(q_0, a, R) = \{(q_1, XR)\}$

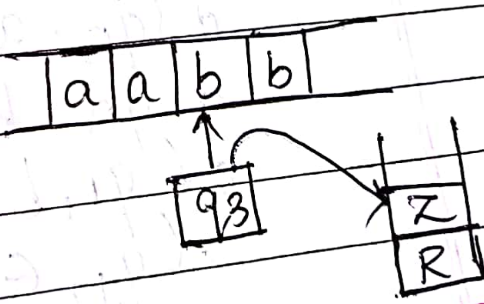
2.



$\delta(q_2, a, Y) = \{(q_2, MY)\}$

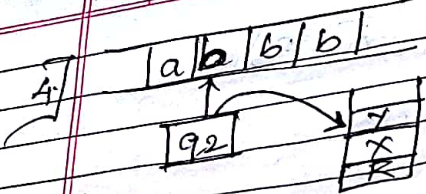
3.

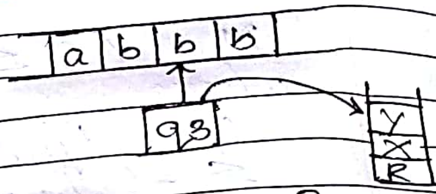




$\delta(q_2, a, Y) = \{(q_3, \epsilon)\}$

4)





$\delta(q_2, b, Y) = \{(q_3, Y)\}$

Design a PDA for recognizing $L = \{a^n b^n \mid n \geq 1\}$

Step 1 = Theory

Step 2 :- Logic :- For each 'a' push 1X
For each 'b' pop 1X

Step 3 :- Implementation

$M = (Q, \Sigma, \Gamma, q_0, R, F)$

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

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

$\Gamma = \{R, X\}$

n=2 aabb

$q_0 = q_0$ $F = q_f$

$R = R$

δ :- $\delta(q_0, a, R) = \delta(q_0, XR)$

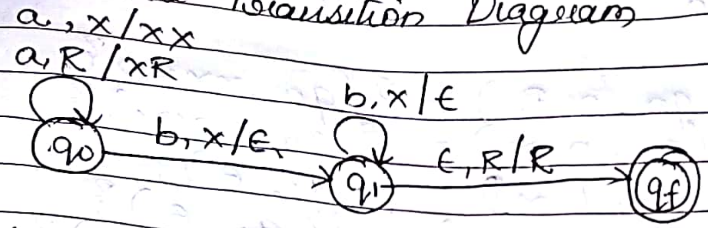
$\delta(q_0, a, X) = \delta(q_0, XX)$

$\delta(q_0, b, X) = \delta(q_1, \epsilon)$

$\delta(q_1, b, X) = \delta(q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = \delta(q_f, R)$

State Transition Diagram



Step 4:- Eg (i) $(q_0, aaabbb, R)$
$(q_0, aabbb, xR)$
$(q_0, abbb, xxR)$
$(q_0, bbb, xxxR)$
(q_1, bb, xxR)
(q_1, b, xR)
(q_1, ϵ, R)
(q_f, R)

(ii) $(q_0, bbaa, R)$
Rejected.

(iii) $(q_0, ababab, R)$
$(q_0, babab, xR)$
$(q_1, abab, R)$
Reject

Q) $L = \{0^n 1^{2n} \mid n \geq 1\}$

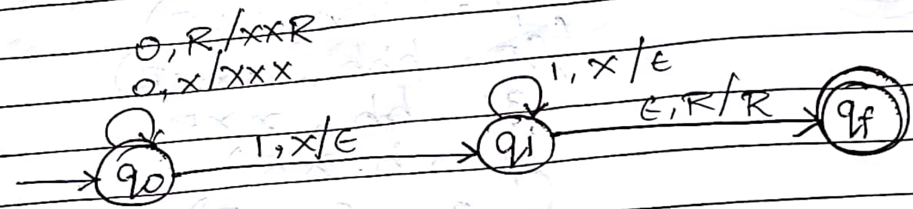
Step 1:- Theory
Step 2:- Logic:-
For each '0' push 2x
For each '1' pop 1x

Step 3:- Implementation
$M = (Q, \Sigma, \Gamma, \delta, q_0, R, F)$

$Q = \{q_0, q_1, q_f\}$
$\Sigma = \{a, b\}$
$\Gamma = \{R, x\}$

$q_0 = q_0 \quad F = q_f \quad R = R$

$\delta :- (q_0, 0, R) = \{(q_0, XXR)\}$
$(q_0, 0, X) = \{(q_0, XXX)\}$
$(q_0, 1, X) = \{(q_1, \epsilon)\}$
$(q_1, 1, X) = \{(q_1, \epsilon)\}$
$(q_1, \epsilon, R) = \{(q_f, R)\}$



Step4:-

eg:- $(q_0, 000111111, R)$
$(q_0, 00111111, XXR)$
$(q_0, 0111111, XXXXR)$
$(q_0, 111111, XXXXXXR)$
$(q_1, 11111, XXXXXR)$
$(q_1, 1111, XXXXR)$
$(q_1, 111, XXXR)$
$(q_1, 111, XXXR)$
$(q_1, 11, XXR)$
$(q_1, 1, XR)$
(q_1, ϵ, R)
(q_f, R)

3) $L = \{(ab)^n c^n \mid n \geq 1\}$

Step 1 :- Theory

Step 2 :- Logic :-

For each 'a' do nothing
" " 'b' push 1X
" " 'c' pop 1X

Step 3 :- $M = (Q, \Sigma, \Gamma, q_0, R, F)$

$Q = (q_0, q_1, q_2, q_f)$

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

$\Gamma = \{R, x\}$

$q_0 = q_0$, $R = R$, $F = q_f$

δ:- $(q_0, a, R) = \{(q_0, R)\}$

δ:- $(q_0, b, R) = \{(q_1, xR)\}$

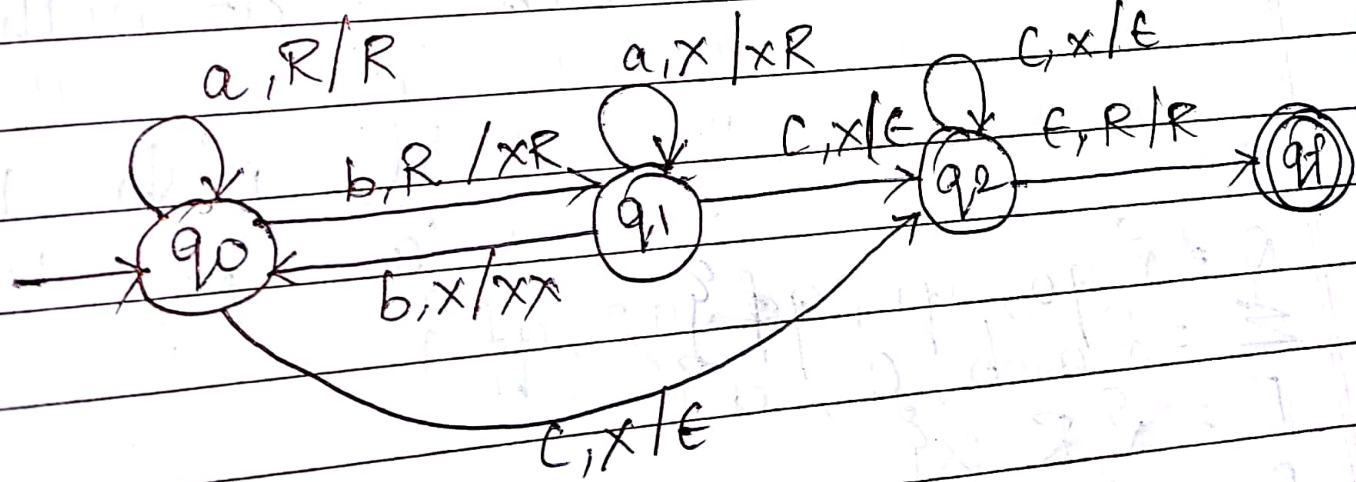
$(q_1, a, x) = \{(q_1, xR)\}$

$(q_1, b, x) = \{(q_1, xx)\}$

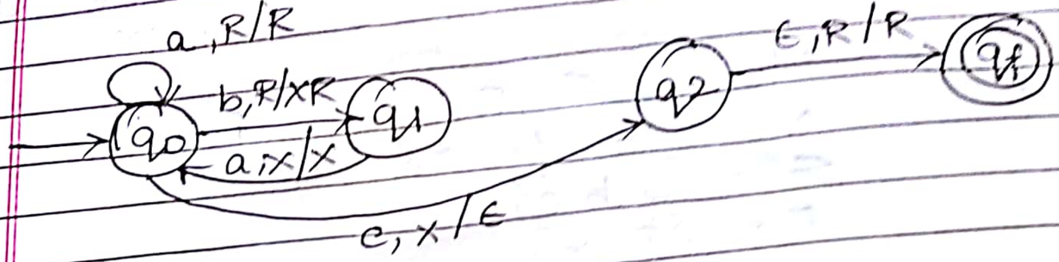
$(q_0, c, x) = \{(q_2, \epsilon)\}$

$(q_2, c, x) = \{(q_2, \epsilon)\}$

$(q_2, \epsilon, R) = \{(q_f, R)\}$



δ :- $(q_0, a, R) = \{(q_0, R)\}$
$(q_0, b, R) = \{(q_1, XR)\}$
$(q_1, a, X) = \{(q_0, X)\}$
$(q_0, c, X) = \{(q_2, \epsilon)\}$
$(q_2, \epsilon, R) = \{(q_f, R)\}$



4) $L = \{(bcd)^n a^n \mid n \geq 1\}$

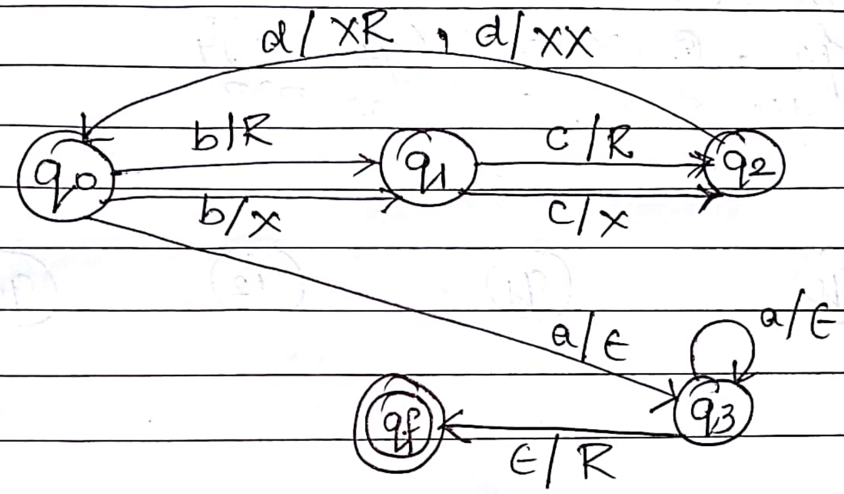
Step 1 = Theory

Step 2 :- Logic

For each 'b' do nothing
" " 'c' " "
" " 'd' push 1X
" " 'a' pop 1X

$Q = \{q_0, q_1, q_2, q_3, q_f\}$
$\Sigma = \{a, b, c, d\}$
$\Gamma = \{R, X\}$
$F = q_f$

$\delta(q_0, b, R) = \{(q_1, R)\}$
$\delta(q_1, c, R) = \{(q_2, R)\}$
$\delta(q_2, d, R) = \{(q_0, XR)\}$
$\delta(q_0, b, X) = \{(q_1, X)\}$
$\delta(q_1, c, X) = \{(q_2, X)\}$
$\delta(q_2, d, X) = \{(q_0, XX)\}$
$\delta(q_0, a, X) = \{(q_3, \epsilon)\}$
$\delta(q_3, a, X) = \{(q_3, \epsilon)\}$
$\delta(q_3, a, X) = \{(q_3, \epsilon)\}$
$\delta(q_3, \epsilon, R) = \{(q_f, R)\}$



5) $L = \{(bdac)^n \, a^n \mid n \geq 1\}$

→

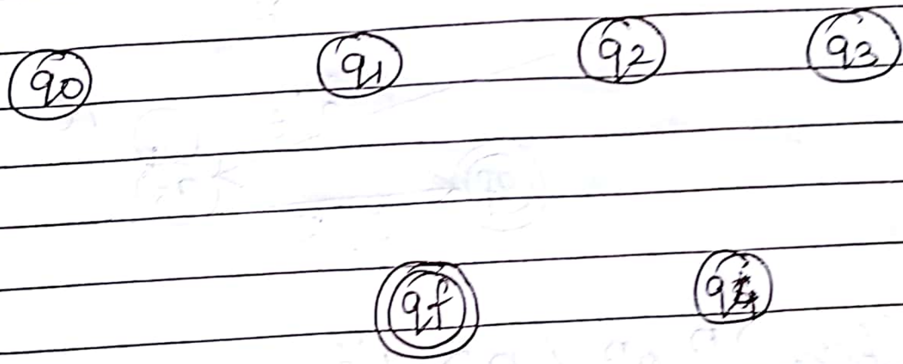
Step 1 :- Theory
Step 2 :- Logic

For each 'b' do nothing
" " 'd' " "
" " 'a' " "
" " 'c' push 1X
" " 'a' pop 1X

$Q = \{q_0, q_1, q_2, q_3, q_4, q_f\}$
$\Sigma = \{a, b, c, d, e\}$

$\Gamma = \{R, x\}$

$\delta(q_0, b, R) = \{(q_1, R)\}$
$\delta(q_1, d, R) = \{(q_2, R)\}$
$\delta(q_2, a, R) = \{(q_3, R)\}$
$\delta(q_3, c, R) = \{(q_0, xR)\}$
$\delta(q_0, b, x) = \{(q_1, x)\}$
$\delta(q_1, d, x) = \{(q_2, x)\}$
$\delta(q_2, a, x) = \{(q_3, x)\}$
$\delta(q_3, c, x) = \{(q_0, xx)\}$
$\delta(q_0, \epsilon, x) = \{(q_4, \epsilon)\}$
$\delta(q_4, \epsilon, x) = \{(q_4, \epsilon)\}$
$\delta(q_4, \epsilon, R) = \{(q_f, R)\}$



5) $L = \{a^n (bc)^n \mid n \geq 1\}$

⇒ Step 1 :- Theory

Step 2 :- Logic

For 'a' push 1X
For 'b' do nothing
For 'c' pop 1X

$Q = \{q_0, q_1, q_2, q_f\}$
$\Sigma = \{a, b, c\}$

$\Gamma = \{R, x\}$

$$\delta(q_0, a, R) = \{(q_0, xR)\}$$
$$\delta(q_0, a, x) = \{(q_0, xx)\}$$
$$\delta(q_0, b, x) = \{(q_1, x)\}$$
$$\delta(q_1, c, x) = \{(q_2, \epsilon)\}$$
$$\delta(q_2, b, x) = \{(q_1, x)\}$$
$$\delta(q_1, c, x) = \{(q_2, \epsilon)\}$$
$$\delta(q_2, \epsilon, R) = \{(q_f, R)\}$$

q_0 q_1 q_2 q_f

6) $L = \{c^n (abd)^n \mid n \geq 1\}$

Step 1 :- Theory

Step 2 :- Logic

For 'a' do nothing

" 'b' " "

" 'c' push 1X

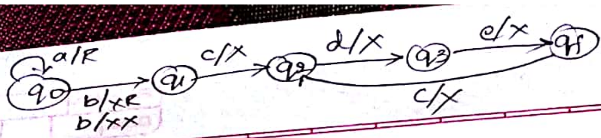
" 'd' pop 1X

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

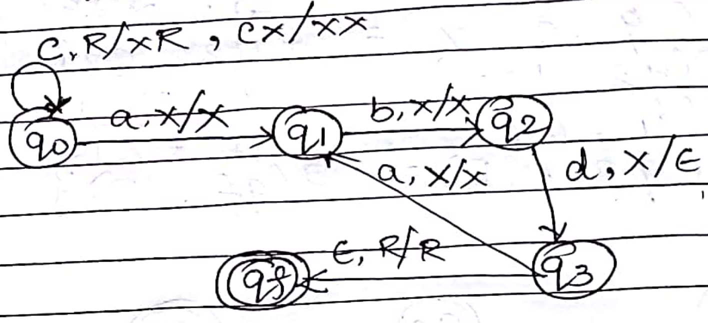
$\Sigma = \{a, b, c, d\}$

$\Gamma = \{R, x\}$

$$\delta(q_0, c, R) = \{(q_0, xR)\}$$
$$\delta(q_0, c, x) = \{(q_0, xx)\}$$
$$\delta(q_0, a, x) = \{(q_1, x)\}$$
$$\delta(q_1, b, x) = \{(q_2, x)\}$$
$$\delta(q_2, d, x) = \{(q_3, \epsilon)\}$$



$\delta(q_3, a, x) = \{(q_1, x)\}$
$\delta(q_1, b, x) = \{(q_2, x)\}$
$\delta(q_2, d, x) = \{(q_3, \epsilon)\}$
$\delta(q_3, \epsilon, R) = \{(q_f, R)\}$



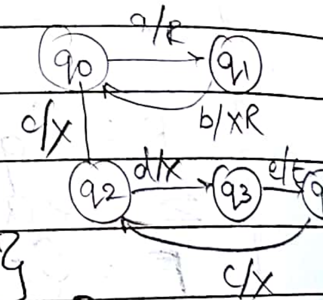
7) $L = \{(ab)^n (cde)^n / n \geq 1\}$

$\implies$ Step1 :- Theory

Step2 :- Logic

For 'a' do nothing
For 'b' push 1X
For 'c' do nothing
For 'd' " "
" 'e' pop 1X

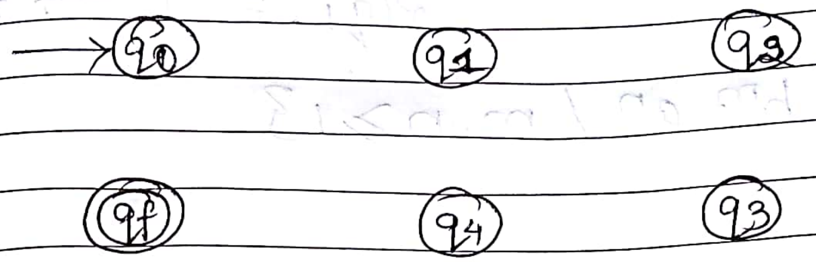
$Q = \{q_0, q_1, q_2,$
$\Sigma = \{a, b, c, d, e\}$
$\Gamma = \{R, x\}$



$\delta(q_0, a, R) = \{(q_1, R)\}$
$\delta(q_1, b, R) = \{(q_0, XR)\}$
$\delta(q_0, a, X) = \{(q_1, X)\}$
$\delta(q_1, b, X) = \{(q_1, XX)\}$
$\delta(q_1, c, X) = \{(q_2, X)\}$
$\delta(q_2, d, X) = \{(q_3, X)\}$
$\delta(q_3, e, X) = \{(q_1, \epsilon)\}$

Date

$\delta(q_1, c, x) = \{(q_2, x)\}$
$\delta(q_2, d, x) = \{(q_3, x)\}$
$\delta(q_3, e, x) = \{(q_4, \epsilon)\}$
$\delta(q_4, \epsilon, R) = \{(q_f, R)\}$



* PDA Design Method :-

1) PDA by Final State Method (FSM)

$(q, w, R) \vdash^* (q_f, \epsilon, R) \quad q \in F$
$R \in \Gamma$

2) PDA by Null / Empty Store Method (NSM/ESM)

$(q, w, R) \vdash^* (q, \epsilon, \epsilon) \rightarrow (q, \epsilon)$
$q \in Q, \ F = \{\ \}$

1) $L = \{a^n b^{n+1} / n \geq 1\}$

$\Rightarrow$ Logic :-
for each 'a' push 1X
Bypass 1st 'b'
For each 'b' pop 1X

$\delta(q_0, a, R) = \{(q_0, xR)\}$

$\delta(q_0, a, x) = \{(q_0, xx)\}$

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

$\delta(q_1, b, x) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, R) = \{(q_f, R)\} \longrightarrow$ FSM

$\{(q_1, \epsilon)\} \longrightarrow$ NSM

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

24/9/18

3) $L = \{a^n b^m c^m d^n / m, n \geq 1\}$

⇒ Logic :-

For each 'a' push 1X
For each 'b' push 1Y
For each 'c' pop 1Y
For each 'd' pop 1X

$$\delta(q_0, a, R) = \{(q_0, XR)\}$$
$$\delta(q_0, a, X) = \{(q_0, XX)\}$$
$$\delta(q_0, b, X) = \{(q_1, YX)\}$$
$$\delta(q_1, b, Y) = \{(q_1, YY)\}$$
$$\delta(q_1, c, Y) = \{(q_2, YX)\}$$
$$\delta(q_2, c, Y) = \{(q_2, XX)\}$$
$$\delta(q_2, d, X) = \{(q_3, XR)\}$$
$$\delta(q_3, d, X) = \{(q_3, R)\}$$
$$\delta(q_3, \epsilon, R) = \{(q_f, R)\} \rightarrow FSM$$
$$\{(q_3, \epsilon)\} \rightarrow NSM$$

4) $L = \{x / na(x) = nb(x)\}$

⇒ Logic :-

(ab)
for a push X
for b pop X

(abab)
For a push X
for b pop X
for a push X
for b pop X

(abba)
for a push X
for b pop X
for b push Y
for a pop Y

(aabb)
For a push X
for b pop X

(aabbba)
push X for a
pop X for b
push Y for b
pop Y for a

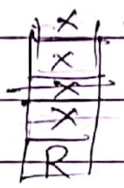
(ababab)

$\delta(q_0, a, R) = \{(q_0, xR)\}$
$\delta(q_0, b, x) = \{(q_1, \epsilon)\}$
$\delta(q_1, a, R) = \{(q_1, xR)\}$
$\delta(q_1, b, x) = \{(q_2, \epsilon)\}$
$\delta(q_2, a, R) = \{(q_2, xR)\}$
$\delta(q_2, b, x) = \{(q_3, \epsilon)\}$
$\delta(q_3, \epsilon, R) = \{(q_f, R)\} \rightarrow$ FSM
$\{(q_3, \epsilon)\} \rightarrow$ NSM

* Design a PDA for recognizing well-formedness of parenthesis (( ) (( )))

$\Rightarrow$ Logic :-

For every '(' push x
For every ')' pop x



$\delta(q_0, (, R) = \{(q_0, xR)\}$
$\delta(q_0, (, x) = \{(q_0, xxR)\}$
$\delta(q_0,), x) = \{(q_1, xR)\}$
$\delta(q_1,), x) = \{(q_2, \epsilon)\}$
$\cancel{\delta(q_1, (, xx) = \{(q_1, xxxR)\}}$
$\cancel{\delta(q_1,), xx) = \{(q_2, xxR)\}}$
$\cancel{\delta(q_2,), x) = \{(q_3, xR)\}}$
$\cancel{\delta(q_3,), x) = \{(q_4, R)\}}$
$\delta(q_2, \epsilon, R) = \{(q_f, R)\} \rightarrow$ FSM
$\{(q_2, \epsilon)\} \rightarrow$ NSM

1) c $w \notin w^R \in \emptyset$
$\Gamma = R$
2) aca $\Gamma = x$
3) bcb $\Gamma = y$

26/9/18

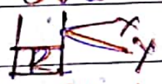
$\underbrace{aaba}_{w}\ \underbrace{c}_{c}\ \underbrace{abaa}_{w^R}$

2) $L = \{ wcw^R \mid w \in \{a,b\}^* \ \&\ w^R = \text{Reverse of } w \}$

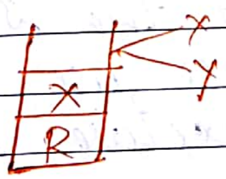
⇒ Logic :- For 'a' push x
" 'b' " y

$\delta(q_0, a, R) = \{(q_0, xR)\}$
$\delta(q_0, a, x) = \{(q_0, xx)\}$
$\delta(q_0, b, x) = \{(q_0, yx)\}$
$\delta(q_0, a, y) = \{(q_0, xy)\}$
$\delta(q_0, c, R) = \{(q_1, R)\}$
$\delta(q_0, c, x) = \{(q_1, x)\}$
$\delta(q_0, c, y) = \{(q_1, y)\}$
$\delta(q_1, a, x) = \{(q_1, \epsilon)\}$
$\delta(q_1, b, y) = \{(q_1, \epsilon)\}$
$\delta(q_1, \epsilon, R) = \{(q_f, R)\} / \{(q_1, \epsilon)\}$

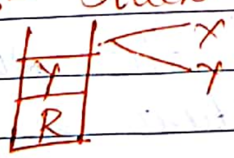
Case 1 :- Stack empty



Case 2 :- Stack top is x



Case 3 :- Stack top is y



Case 4 :- 4.1) c $w \notin w^R \in \emptyset$
$\Gamma = R$
4.2) aca $\Gamma = x$
4.3) bcb $\Gamma = y$

$\delta(q_0, a, R) = \{(q_0, XR)\}$
$\delta(q_0, b, R) = \{(q_0, YR)\}$
$\delta(q_0, a, X) = \{(q_0, XX)\}$
$\delta(q_0, b, X) = \{(q_0, YX)\}$
$\delta(q_0, a, Y) = \{(q_0, XY)\}$
$\delta(q_0, b, Y) = \{(q_0, YY)\}$
$\delta(q_0, c, R) = \{(q_1, R)\}$
$\delta(q_0, c, X) = \{(q_1, X)\}$
$\delta(q_0, c, Y) = \{(q_1, Y)\}$
$\delta(q_1, a, X) = \{(q_1, \epsilon)\}$
$\delta(q_1, b, Y) = \{(q_1, \epsilon)\}$
$\delta(q_1, \epsilon, R) = \{(q_f, R)\} \longrightarrow$ FSM
$= \{(q_1, \epsilon)\} \longrightarrow$ NSM

* $L = \{ww^R \mid w \in \{a, b\}^*$ & w^R = Reverse of $w\}$

$\Rightarrow$ This is a case of <u>NON-DETERMINISTIC PDA</u> (NPDA) design. Whenever there is double letter, 2 cases are possible.

<u>case 1</u> :- The middle of the string is not reached & hence we continue to push the respective symbols on the stack.

<u>case 2</u> :- The middle of the string is reached & hence we start to pop the respective symbols from the stack.

$\delta(q_0, \epsilon, R) = \{(q_1, R)\}$
$\delta(q_0, a, R) = \{(q_0, XR)\}$
$\delta(q_0, b, R) = \{(q_0, YR)\}$
$\delta(q_0, a, X) = \{(q_0, XX)\} \,/\, \{(q_0, \epsilon)\}$
$\delta(q_0, b, X) = \{(q_0, YX)\}$
$\delta(q_0, a, Y) = \{(q_0, XY)\}$
$\delta(q_0, b, Y) = \{(q_0, YY)\} \,/\, \{(q_0, \epsilon)\}$
$\delta(q_0, a, X) = \{(q_1, \epsilon)\}$
$\delta(q_0, b, Y) = \{(q_1, \epsilon)\}$
$\delta(q_1, \epsilon, R) = \{(q_f, R)\} \rightarrow$ FSM
$= \{(q_1, \epsilon)\} \rightarrow$ NSM

Eg:- $(q_0, abba, R)$
(q_0, bba, XR)
(q_0, ba, YXR)

$(q_0, a, YYXR)$
$(q_0, \epsilon, XYYXR)$

(q_0, a, XR)

(q_0, ϵ, XXR)

(q_0, ϵ, R)
(q_f, R) /
(q_1, ϵ)

Page No.
Date

* CFG to PDA :-

Given CFG, $G = (V, T, P, S)$.

Define PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, R, F)$

for recognizing CFL define PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, R, F)$ for recognizing CFL generated by CFG.

Step 1 :- Express CFG in GNF.

Step 2 :- PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, R, F)$

$Q = \{q_0\}$ $\quad q_0 = q_0$ $\quad \Sigma = T'$

Z_0 or $R = \{S\}$ $\quad \Gamma = V'$

$F = \{\ \}$

δ :-

* $A \longrightarrow aBCD$

$\delta(q_0, a, A) = \{(q_0, BCD)\}$

* $A \longrightarrow a$

$\delta(q_0, a, A) = \{(q_0, \epsilon)\}$

1. Construct a PDA for the following
$S \rightarrow aSa / bSb / c$

$\Rightarrow$ **Step 1 :-** CNF $\rightarrow$ GNF

$S \rightarrow aSC_1 / bSC_2 / c$

$C_1 \rightarrow a$

$C_2 \rightarrow b$

PDA , $M = (Q, \Sigma, \Gamma, \delta, q_0, R, F)$ where

$Q = [q_0$, $q_0 = q_0$, $\Sigma = T'$, $Z_0/R = S$, $\Gamma = V'$, $F = \{\}]$

Step 2 :- $\delta - A \longrightarrow aBCD$

$$\delta(q_0, a, A) \rightarrow \{[q_0, BCD]\}$$

$S \rightarrow aSC_1$

$\delta(q_0, a, S) \rightarrow \{(q_0, SC_1)\}$

CFG $\rightarrow$ PDA

$S \rightarrow bSC_2$

$\delta(q_0, b, S) \rightarrow \{(q_0, SC_2)\}$

δ :- $A \rightarrow a$

$\delta(q_0, a, A) \rightarrow \{(q_0, \epsilon)\}$

$S \rightarrow c$

$\delta(q_0, c, S) \rightarrow \{(q_0, \epsilon)\}$

$C_1 \rightarrow a$

$\delta(q_0, a, C_1) \rightarrow \{(q_0, \epsilon)\}$

$C_2 \rightarrow b$

$\delta(q_0, b, C_2) \rightarrow \{(q_0, \epsilon)\}$

Eg:- $(q_0, abcba, S)$

C_2
S
C_1
S
S

$(q_0, bcba, SC_1)$

(q_0, cba, SC_2C_1)

(q_0, ba, C_2C_1)

(q_0, a, C_1)

(q_0, ϵ)

Q.] $S \rightarrow aAA$

$A \rightarrow aS / bS / a$

$\Rightarrow$ Step 1 :- Already in GNF

Step 2 :-

$S \rightarrow aAA$

$\delta(q_0, a, S) \rightarrow \{(q_0, AA)\}$

$A \rightarrow aS$

$\delta(q_0, a, A) \rightarrow \{(q_0, S)\}$

$A \rightarrow bS$

$\delta(q_0, b, A) \rightarrow \{(q_0, S)\}$

$A \rightarrow a$

$\delta(q_0, a, A) \rightarrow \{(q_0, \epsilon)\}$

Eg :- (q_0, aaa, S)

(q_0, aa, AA)

(q_0, a, SA)

(q_0, ϵ)

(q_0, a, SAA)

(q_0, ϵ, AAA)

* PDA to CFG :-

Given PDA, $M = (Q, \Sigma, \Gamma, q_0, Z, F)$
Define CFG, $G = (V, T, P, S)$ for generating CFL recognized by PDA machine

* $V = S + [q, A, p]$ $p, q \in Q$ $A \in \Gamma$
* $T = \Sigma$
* $S = S$
* $P = S \rightarrow [q_0, Z, p]$ $p \in Q$

$\rightarrow \delta(q, a, A) = \{(p, BC)\}$

$[q, A, r] \rightarrow a[p, B, s][s, C, r)$ $r, s \in Q$

$\rightarrow \delta(q, a, A) = \{(p, B)\}$

$[q, A, r] \rightarrow a[p, B, r]$ $r \in Q$

* Construct CFG for the following PDA:

$A = (\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi)$

$\delta =$
(i) $\delta(q_0, b, z_0) = \{(q_0, zz_0)\}$
(ii) $\delta(q_0, b, z) = \{(q_0, zz)\}$
(iii) $\delta(q_0, a, z) = \{(q_1, z)\}$
(iv) $\delta(q_1, a, z) = \{(q_1, z)\}$
(v) $\delta(q_1, b, z_0) = \{(q_1, \epsilon)\}$
(vi) $\delta(q_1, \epsilon, z_0) = \{(q_1, \epsilon)\}$

$\Rightarrow$ Step 1 :- $Q = \{q_0, q_1\}$

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

$\Gamma = \{z_0, z\}$

$\delta = \delta$

$q_0 = q_0$

$R = z_0$

$F = \phi$

$V = \{S, \overset{A}{[q_0, z_0, q_0]}, \overset{B}{[q_0, z_0, q_1]},$

$\overset{C}{[q_1, z_0, q_0]}, \overset{D}{[q_1, z_0, q_1]},$

$\overset{E}{[q_0, z, q_0]}, \overset{F}{[q_0, z, q_1]}, \overset{G}{[q_1, z, q_0]},$

$\overset{H}{[q_1, z, q_1]}\}$

$T = \{a, b\}$

$S = S$

$P = S \rightarrow [q_0, z_0, \overset{q_0 \; q_1}{p}],$

$S \rightarrow [q_0, z_0, q_0] / [q_0, z_0, q_1]$

$S \rightarrow A/B$

(i) $\delta(q_0, b, z_0) = \{[q_0, zz_0]\}$

$A = bEA$ $\quad [q_0, z_0, q_0] = b[q_0, z, q_0][q_0, z_0, q_0]$

$A = bFC$ $\quad [q_0, z_0, q_0] = b[q_0, z, q_1][q_1, z_0, q_0]$

$B = bEB$ $\quad [q_0, z_0, q_1] = b[q_0, z, q_0][q_0, z_0, q_1]$

$B = bFD$ $\quad [q_0$ …

(ii) $\delta($ …

$[q_0$ …

$[q_0$ …

(iii)

$D = bFD$ $\quad [q_0, z_0, q_1] = b[q_0, z, q_1][q_1, z_0, q_1]$

(ii) $\delta(q_0, b, z) = \{(q_0, zz)\}$

$[q_0, z, q_0] = b[q_0, z, q_0][q_0, z, q_0]$ $\quad A = bEE$

$[q_0, z, q_0] = b[q_0, z, q_1][q_1, z, q_0]$ $\quad A = bFG$

$[q_0, z, q_1] = b[q_0, z, q_0][q_0, z, q_1]$ $\quad F = bEF$

$[q_0, z, q_1] = b[q_0, z, q_1][q_1, z, q_1]$ $\quad F = bFH$

(iii) $\delta(q_0, a, z) = \{(q_1, z)\}$

$[q_0, z, q_0] = a[q_1, z, q_0]$ $\quad E = aG$

$[q_0, z, q_1] = a[q_1, z, q_1]$ $\quad F = aH$

(iv) $\delta(q_1, a, z) = \{(q_1, z)\}$

$[q_1, z, q_0] = a[q_1, z, q_0]$ $\quad G = aG$

$[q_1, z, q_1] = a[q_1, z, q_1]$ $\quad H = aH$

(v) $\delta(q_1, b, z_0) = \{(q_1, \epsilon)\}$

$\cancel{[q_1, z_0, q_0] = b[q_1, \epsilon, q_0]}$ $\quad \cancel{C = b\epsilon}$

$[q_1, z_0, q_1] = b$ $\quad D = b$

(vi) $\delta(q_1, \epsilon, z_0) = \{[q_1, \epsilon]\}$

~~$[q_1, \quad , q_0] = \epsilon[q_1, \epsilon, q_0]$~~ ~~$\epsilon = \epsilon\epsilon$~~

$[q_1, \quad , q] = \epsilon$ $D = \epsilon$

The CFG for the given PDA is

$S \rightarrow S$

$A \rightarrow bEA / bFC / bEE / bFG$

$B \rightarrow bEB / bFD$

~~$C \rightarrow bE$~~

$D \rightarrow b$

$E \rightarrow bEE / b\,FG / aG$

$F \rightarrow aH$

$G \rightarrow aG$

$H \rightarrow aH$

Scanned by CamScanner

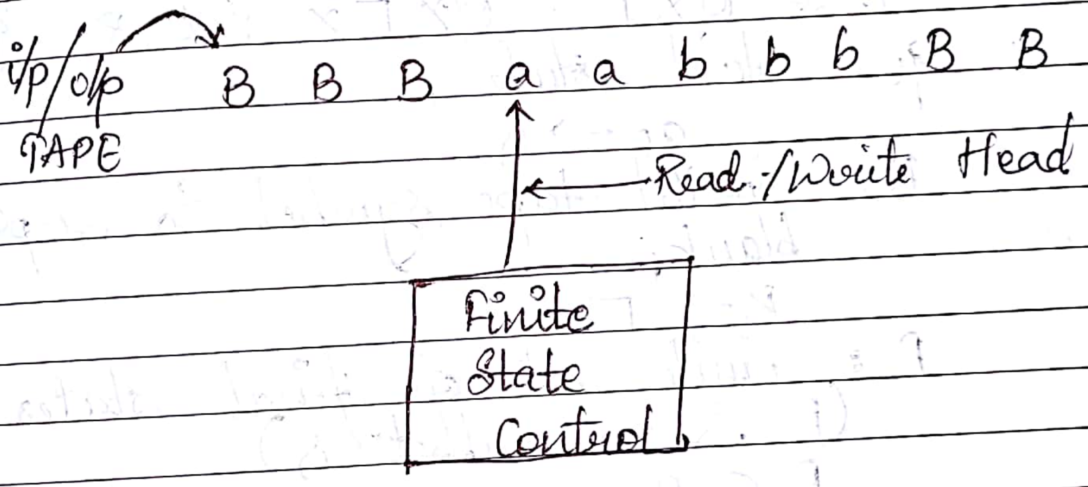
Turing Machine

Turing machine is considered to be a simple model of a computer & is the most powerful machine.

This machine was invented by Mr. Allan Turing in the year 1935.

TM can perform the following operations :-

1) Language recognization
2) Language generation
3) Computation of some functions



* Components of TM :-

TM consists of finite set of states, i/p or o/p tape & read / write head

* Working of TM :-

Depending on the state & tape symbol

1) TM can change the state or remain in the same state

2) TM can change tape symbol or keep it the same.
3) TM can move the head either $\{L, R, S\}$

TM can be mathematically represented as
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$
where,

Q : Finite set of states
Σ : I/p alphabet
Γ : Tape alphabet
δ : Transition function
$\delta :- Q \times \Gamma = Q \times \Gamma \times \{L, R, S\}$
q_0 : Start state,
$q_0 \rightarrow Q$
B : Special tape symbol to represent blank,
$B \rightarrow \Gamma$
F : Finite set of final states,
(F is a subset Q)
$F \subseteq Q$

Scanned by CamScanner

Page No. | Date

5/10/11.

Decidability & Undecidability.

1) Recursive Languages:- imp
A language 'L' is said to be recursive if there exists a TM which will accept all the strings in 'L' & reject all the strings not in 'L'.
The TM will halt every time & give an answer (accepted or rejected) for each & every string input.

2) Recursively Enumerable Language:- imp
A language 'L' is said to be a recursively enumerable language if there exists a TM which will accept (and therefore halt) for all the input strings which are in 'L'.
But may or may not halt for all input strings which are not in 'L'.

3) Decidable Language:-
A language L is decidable if it is a recursive language. All decidable languages are recursive languages & vice versa.

4) Partially decidable language:-
A language 'L' is partially decidable if 'L' is a recursively enumerable language.

Page No.
Date

5) Undecidable Language :-

A language is undecidable if it is not decidable.

An undecidable language may sometimes be partially decidable but not decidable.

If a language is not even partially decidable, then there exists no TM for the language.

Recursive language	TM will always halt.
Recursively Enumerable language	TM will halt sometimes & may not halt sometimes
Decidable Language	Recursive language
Partially decidable language	Recursively Enumerable language.
Undecidable	No TM for that language.

5/10/18

* The Post Correspondance Problem (PCP):-

The Post correspondance problem is an undecidable decision problem that was introduced by Emil Post in 1946.

eg:-

Dominos:-

$\boxed{\frac{B}{CA}}$ $\boxed{\frac{A}{AB}}$ $\boxed{\frac{CA}{A}}$ $\boxed{\frac{ABC}{C}}$

We need to find a sequence of dominos such that the top & the bottom strings are the same.

$\boxed{\frac{A}{AB}}$ $\boxed{\frac{B}{CA}}$ $\boxed{\frac{CA}{A}}$ $\boxed{\frac{A}{AB}}$ $\boxed{\frac{ABC}{C}}$ ABCAAABC
ABCAAABC

Another way of requesting the PCP:-

	A	B
1	1	111
2	10111	10
3	10	0

$\boxed{\frac{1}{111}}$ $\boxed{\frac{10111}{10}}$ $\boxed{\frac{10}{0}}$

2 1 1 3

A 10111 1 1 10

B 10 111 111 0 ✓

eg:-

	A	B
1	10	101
2	011	11
3	101	011

$\frac{10}{101}$ $\frac{011}{11}$ $\frac{101}{011}$

① $\frac{1010}{101101}$ ✗

② $\frac{10011}{10111}$ ✗

③ $\frac{1010110110101\ldots}{1010.11011011011\ldots}$ ✗

PCP is in undecidability manner

universal TM & variants of TM → v.v.v. imp.

8/10/18.

1) Design a TM for recognizing no. of a = no. of b
$L = \{a^n b^n \mid n \geq 1\}$

⟹

Step 1 :- Theory of TM

Step 2 :- Logic :-
Algorithm.

Simulation

Eg :- B B a a a b b b B B
$\uparrow q_0$

B B X a a b b b B B
$\uparrow q_1$

B B X a a b b b B B
$\uparrow q_1$

B B X a a b b b B B
$\uparrow q_1$

B B X a a Y b b B B
$\uparrow q_2$

B B X a a Y b b B B
$\uparrow q_2$

B B X a a Y b b B B
$\uparrow q_2$

B B X a a Y b b B B
$\uparrow q_2$

B B X a a Y b b B B
$\uparrow q_0$

B B X X a Y b b B B
$\uparrow q_1$

B B X X a Y b b B B
↑q_1

B B X X a Y b b B B
↑q_1

B B X X a Y Y b B B
↑q_2

B B X X a Y Y b B B
↑q_2

B B X X a Y Y b B B
↑q_2

B B X X a Y Y b B B
↑q_2

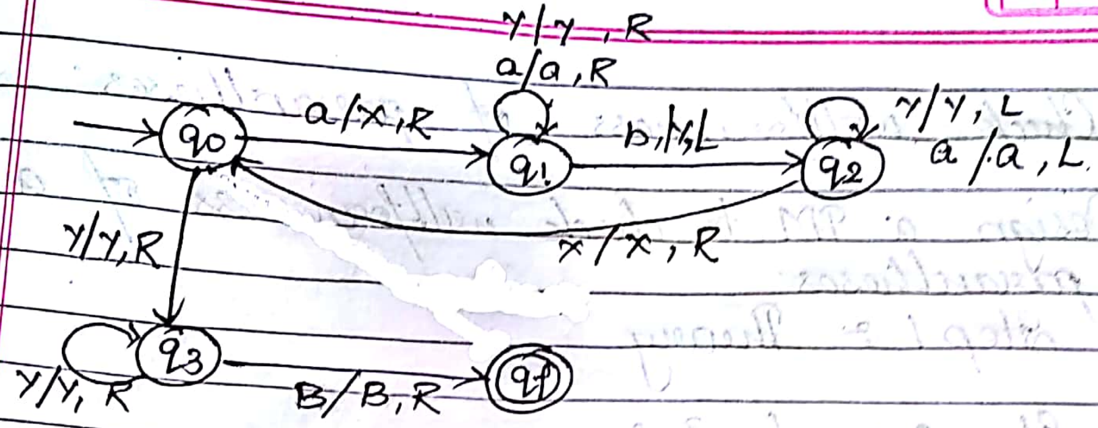
B B X X a Y Y b B B
↑q_0

B B X X X Y Y b B B
↑q_1 ↑q_1 ↑q_2

B B X X X Y Y Y B B
↑q_3 ↑q_f

Algorithm :-

1. Replace 'a' with 'x' & move R
2. Move to R till 1st b & replace it with Y
3. Repeate 1 & 2 time all the i/p are replaced.



Q\Γ	a	b	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	(q_1, a, R)	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	(q_2, a, L)	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_f, B, R)
q_f	final state			–	–

Check wellformness of paranthesis.

*. Design a TM to check wellformness of a paranthesis.

⇒ Step 1 :- Theory

Step 2 :- Logic :-

Algorithm :-

1) Move the read head to right till you find 1st ')' & replace it with x

2) Move to left till you find the 1st '(' & replace with X

Simulation :-

B B (( ) ( )) B B

↑q_0 ↑q_0 ↑q_0

B B ((X ( )) B B

↑q_1 ↑q_1

B B (X X ( )) B B

↑q_0 ↑q_0 ↑q_0

B B (X X (X) B B

↑q_1 ↑q_1

B B (X X X X) B B

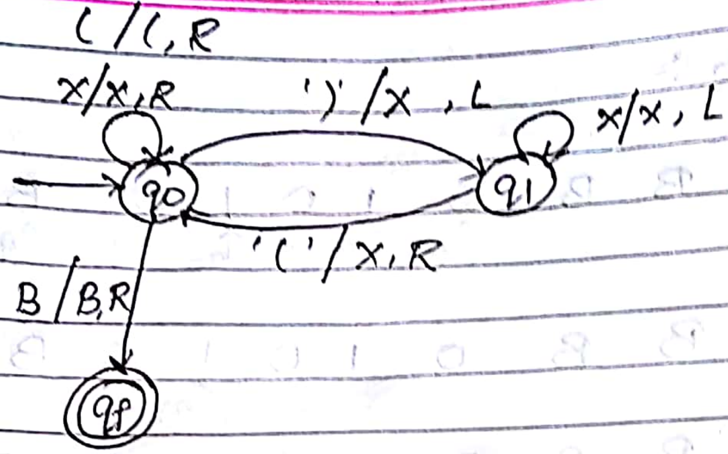
↑q_0 ↑q_0 ↑q_0

B B (X X X X X B B

↑q_1 ↑q_1 ↑q_1 ↑q_1 ↑q_1 ↑q_1

B B X X X X X X B B

↑q_f ↑q_0



* Design a TM to find 2's complement of a binary number.

⟹ Step 1:- Theory

Step 2:- Logic:-

Algorithm:-

1) Move the read head to the right till you encounter B.

2) Start moving towards left till you reach 1st '1'.

3) Then move towards left & replace each '0' by '1' & each '1' by '0' till you reach the left B.

Scanned by CamScanner

Simulation :-

B B 0 1 0 1 0 B B
$\uparrow q_0$ $\uparrow q_0$ $\uparrow q_0$ $\uparrow q_0$ $\uparrow q_0$ $\uparrow q_0$

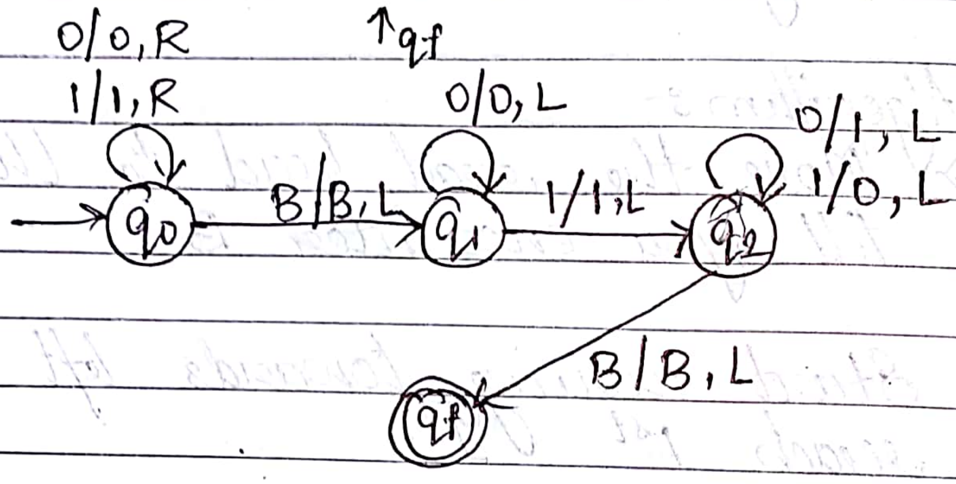
B B 0 1 0 1 0 B B
$\uparrow q_1$

B B 0 1 1 1 0 B B
$\uparrow q_2$

B B 0 0 1 1 0 B B
$\uparrow q_2$

B B 1 0 1 1 0 B B
$\uparrow q_2$

B B 1 0 1 1 0 B B
$\uparrow q_f$



* Design a TM that recognizes strings containing equal no. of 0's & 1's

⟹ Step 1:- Theory.

Step 2:- Logic:-

Algorithm:-

1) Replace the 1st 0 by x & move to the right till you reach the 1st 1 & replace 1 by x.

2) If you reach 1 1st replace 1 by x & move right till you reach 1st 0 & replace that 0 with 'x'.

3) Repeat ① & ② till you replace 0 & 1.

Simulation:-

B B 0 1 1 0 B B
↑q_0

B B x 1 1 0 B B
↑q_1 ↑q_1

B B x x 1 0 B B
↑q_0

v.v.imp

* Universal TM :-

A universal TM is capable of simulating any TM (T), if the following information is available on its tape.

1) The description of T in terms of its state function.
2) The initial configuration of T with the processing data (input string) to be fed to T.

This means that UTM should have an emitation algorithm in order to correctly interpret the rules of the operation given in the state function of the TM being simulated i.e; T.

The UTM should also have a table lookup facility & should perform following steps.

* Emitation Algorithm :-

facility should perform following lookup steps.

* The Halting Problem:-

Given a program, will it halt?

Given a TM, will it halt when run on some particular given i/p string?

Given some program written in some language (Java/C/etc), will it ever get into an infinite loop or will it always terminate?

Answer:-

- In general we can't always know.
- The best we can do is run the program & see whether it halts.
- For many program we can see that it will always halt or sometimes loop.

But for programs in general the question is undecidable.

* Pumping Lemma for regular language:-

<u>STMT</u>:- Let L be a regular language & $M = (Q, \Sigma, \delta, q_0, F)$ be a FA with n-states. Let L is accepted by M.

Let $w \in L$ & $|w| \geq n$, then w can be written as xyz,

where, $|y| > 0$

$|xy| \leq n$

$xy^iz \in L$, for all $i \geq 0$, here y^i denotes that y is repeated or pumped i times.

<u>Interpretation</u>:- (i) Let L be a regular language.

(ii) Let M be a DFA with n states, such that DFA accept the given regular language, L.

No. of states, $n = 5$ (q_0 to q_4)

(iii) Let us consider a string $w \in L / |w| \geq n$ with $|w| \geq 5$

$w = abcabcb$

(iv) To recognize the string w, the machine will transit through various states i.e; from q_0 to q_4.

q_1 repeating

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{c} q_3 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{c} q_3 \xrightarrow{b} q_4$$

Thus if $abcabcb$ is accepted by the FA then $abcabcb$ can be written as xyz, with

$x = a \quad y = bca \quad z = bcb$.

length of $abcabcb$ is $\geq n$

xy^iz for every $i \geq 0$ or $a(bca)^i bcb$ for every $i \geq 0$ will be accepted by the FA.

Proof 1 :-

Let L be a regular language

Let M be a DFA with n states, such that DFA accepts the given regular language L.

i.e; $L = L(M)$, language of M is same as given Regular language.

Let us consider a string $w \in L / |w| \geq n$

$w = a_1 \, a_2 \, a_3 \cdots a_m$ with $m \geq n$

Let us assume that states of M are given by $q_0, q_1, q_2, \ldots q_{n-1}$ with q_0 as start state and q_{n-1} as final state.
Let us assume that after feeding the first i characters of word $w = a_1 a_2 \ldots a_m$

$$\delta(q_0, a_1 a_2 \ldots a_i) = q_i$$

$q_0 \quad q_1 \quad q_2 \ldots\ldots q_{m-1} \quad q_{n-1}$ ← states

$a_1 \quad a_2 \ldots\ldots a_{m-1} \ldots a_m$ ← input

Since the length of $w \geq n$ it is not possible for the machine to move through the distinct states.
Let us assume that $q_i = q_j$ are same.
There is a loop from q_i to q_j.
The string w can be divided into three parts.

1. x = portion before loop $= a_1 a_2 \ldots\ldots a_i$
2. y = portion of loop $= a_{i+1} \; a_{i+2} \ldots\ldots a_j$
3. z = portion after loop $= a_{j+1} \; a_{j+2} \ldots\ldots a_m$

Since y is a portion relating to loop, it can repeat any number of times

$x = a_1 \; a_2 \ldots\ldots a_i$

$y = a_{i+1} \; a_{i+2} \ldots\ldots a_j$

$z = a_{j+1} \ldots\ldots a_m$

$$\rightarrow (q_0) \xrightarrow{x} (q_i = q_j)^{\circlearrowleft y} \xrightarrow{z} ((q_{m-1}))$$

Thus, it is clear that if $xyz \in L$ then xy^iz will also be accepted by the machine for every $i \geq 0$

* Rice's Theorem :

- Property of languages :- is a function P that goes from set of all languages to $\{0, 1\}$

- For a language L, if $P(L) = 1$ we say that L satisfies P and if $P(L) = 0$, we say that L does not satisfy the property P.

Eg:- of properties :-

- L has the string 0110
- L is empty.
- L has 1000 strings.

- P is said to be a non trivial property of languages of TM if there exists TM's M_1 & M_2 such that $P(L(M_1)) = 1$ & $P(L(M_2)) = 0$

Rice's Theorem :-

Let P be a non-trivial property of languages of TM.
Then the language
$L_p = \{ \langle M \rangle \mid P(L(M)) = 1 \}$ is undecidable.

Proof :-

Case 1 :-

Assume that $P(\emptyset) = 0$.
Since P is a non-trivial property of languages of TM's.
There exists of TM 'N' such that the language of N, $P(L(N)) = 1$

We will show that
$$A_{TM} \leq_m L_p$$

The reduction. f

$$\langle M, w \rangle \xrightarrow{f} \langle M' \rangle$$

i/p $\langle M, w \rangle$

1) Design a TM M' that on any i/p x does the following.

(M' has a description of N hardcoded into its description together with M & w)

- Simulate M on w. if M rejects w then it rejects.
- If M accepts w then simulate N on x
- If N accepts x then accept.
- If N rejects then reject

O/P :-

$\langle M' \rangle$

If M accepts $w \Rightarrow L(M') = L(N)$
$\Rightarrow P(L(M')) = 1$

If M does not accept $w \Rightarrow L(M') = \phi$
$\Rightarrow P(L(M')) = 0$

$\therefore A_{TM} \leq_m L_P$

Case 2 :-

Assume that $(P(\phi)) = 1$

We will reduce "this" to Case 1.

Consider the complement property $\overline{P}$

$\overline{P}(L) = 1 - P(L)$

Since, $P(\phi) = 1$, $\therefore \overline{P}(\phi) = 0$

$\therefore$ By Cas 1,

$A_{TM} \leq_m L_{\overline{P}}$

If $A_{TM} \leq_m L_{\overline{P}} \Rightarrow \overline{A_{TM}} \leq_m \overline{L_{\overline{P}}}$

Observe that $\overline{L_{\overline{P}}} = \{\langle M \rangle \mid \overline{P}(L(M)) = 0\}$

$= \{\langle M \rangle \mid P(L(M)) = 1\}$

$\therefore \overline{A_{TM}} \leq_m L_P$

L_P is undecidable