DATE: __/__/____
PAGE _____

UNIT-2 Regular Expression

$L = L(M)$; If any language is regular in nature then it will be able to draw the machine.

$L_1 \cap L_2$ (both are regular)
$L_1 \cup L_2$
$L_1 \cdot L_2$
L_1^*

Defⁿ: A regular expression is recursively defined as follows:

1.) ϕ is a regular expression denoting empty language.
2.) ε -(Epsilon) is a regular expression denoting empty language string.
3.) 'a' is a regular expression denoting language containing only $\{a\}$.
4.) If α and β are regular expressions, then
   a.) $\alpha + \beta$ is also a regular expression corresponding to $L_\alpha \cup L_\beta$.
   b.) $\alpha \cdot \beta$ is also regular expression corresponding to $L_\alpha \cdot L_\beta$
   c.) α^* is also a regular expression to L_α^* (Kleene star / or closure)

NOTE: Nothing else is a Regular Expression.

Ex: The language $\{\Lambda, 00, 01, 000, 001, 010, 011, 0000, \ldots\}$ which consists all binary strings is regular.

$L = \{0\} \cdot \{0\}^* \cdot \{1\}^*$

$L = \{0\} \cdot (\{0\} \cdot \{1\})^*$

GOOD WRITE

To interpret the regular expression

$\Sigma^* = \{\lambda\}$

$\Sigma^+ \neq \{\lambda\}$

DATE: __/__/____

PAGE _____

Regular Expressions	Meaning
$(a+b)^*$ →	Set of string of a's and b's any length including NULL string.
$(a+b)^* abb$ →	Set of string of a's and b's any length only ending with abb.
$ab(a+b)^*$ →	Set of string of a's and b's any length only starting with ab.
$(a+b)^* aa (a+b)^*$ →	Set of string of a's and b's any length containing substring aa.
$a^* b^* c^*$ →	Set of string a, b and c or may be a NULL string.
$a^+ b^+ c^+$ →	Set of string a, b, c may be or may not.
$aa^* bb^* cc^*$ →	Set of string of a's and b's any length including may or may not
$(a+b)^* (a+bb)$ →	Set of string of a's and b's ending with either a or bb.
$(aa)^* (bb)^* b$ →	Set of string of a's and b's ending or start with b or even no. of a's ~~and odd~~.

Q: R.E accepting a language consisting of strings of a's and b's of even length?

Ans: $(aa+ab+ba+bb)^*$ = Regular Expression

$L(R) = \{(aa+ab+ba+bb)^n \mid n \geq 0\}$

For odd length:

$RE = (aa+ab+ba+bb)^* (a+b) \;\big|\; (a+b)(aa+ab+ba+bb)^*$

GOOD WRITE — Even — either a or b

Q: R.E over the alphabet $\Sigma = \{a, b\}$

a) All strings that don't end with aa?

(a) ~~$(a + ab + ba + bb)^* (b)$~~

$(\varepsilon + (a) + (b))(ab + ba + bb)^* (b)$

$(\varepsilon + (a) + (b))(ab + ba \quad \varepsilon + a + b + (a+b)^* (ab + ba + bb)$

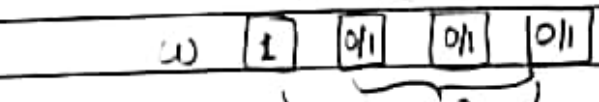
b) All strings that contain an even number of b's?

$a^* (ba^* ba^*)^*$

$L = \{\lambda, a, aa, \ldots, bb, ababa, \ldots\}$

23/02/18 Q: Obtain a R.E for all the strings made by $\{0,1\}^+$ whose 4th digit from the end is 1.

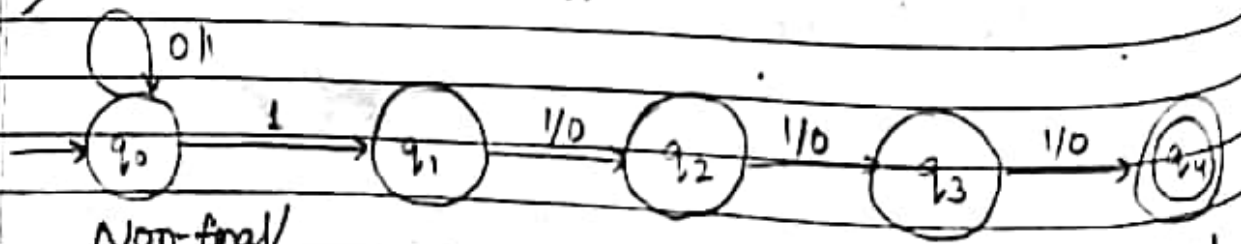
Ans



$L = \{1000, 1100, 1101, 1110, 1111, \ldots\}$

$RE = \{0,1\}^* \; 1 \{0,1\}^3$

NFA



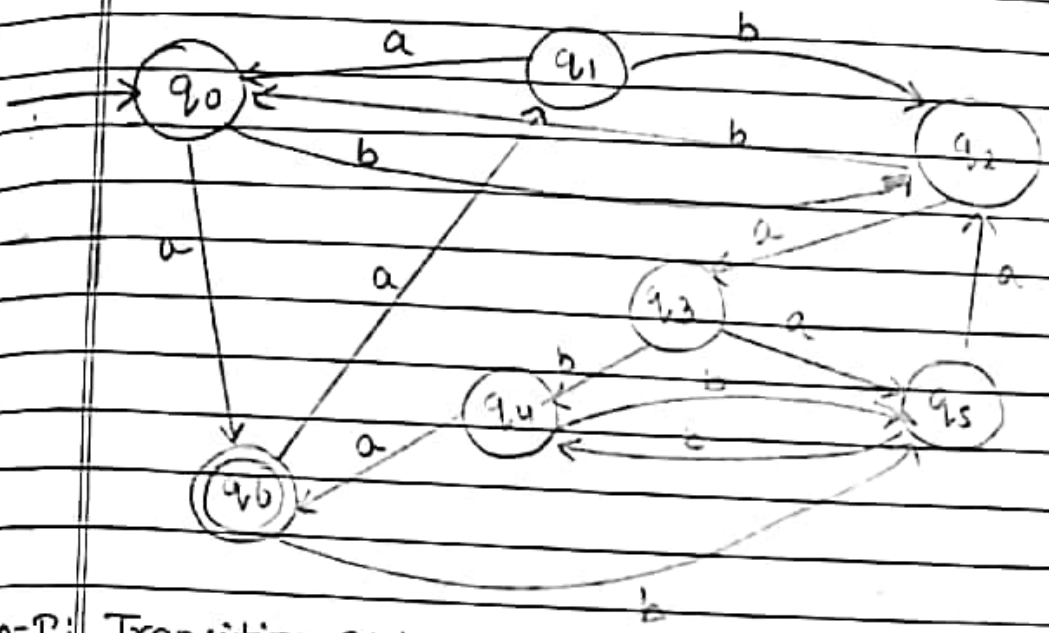
Non-final

Initial

Final

GOOD WRITE

Q:) Minimize DFA=?



Step-I: Transition State

	a	b
→q0	*q6	q2
q1	→q0	q2
q2	q3	→q2
q3	q5	q4
q4	*q6	q5
q5	q2	q4
q6	q1	q5

Step-II:

Final	Non-final
q6	{q0, q1, q2, q3, q4, q5}

Step-III: $\Pi_0 \rightarrow \{q_6\}\{q_0, q_1, q_2, q_3, q_4, q_5\}$

$q_0 \xrightarrow{a} *q_6$ ✗ $\quad q_0 \xrightarrow{b} q_2$ ✓ Doesn't belong to same set ✓

$q_1 \xrightarrow{a} q_0$ $\quad q_1 \xrightarrow{b} q_2$

$q_0 \xrightarrow{a} q_6$ ✗ $\quad q_0 \xrightarrow{b} q_2$ ✓ Doesn't belong to same set ✓

GOOD WRITE

$q_2 \xrightarrow{a} q_3$ $\quad q_2 \xrightarrow{b} q_2$

$\Pi_1 \rightarrow \{q_6\}\ \{q_0, q_1, q_4\}\ \{q_2, q_3, q_5\}$

$q_1 \xrightarrow{a} q_0$ ✗ $\quad q_1 \xrightarrow{b} q_1$ ✗ doesn't belongs to same set
$q_2 \xrightarrow{a} q_3$ $\quad q_1 \xrightarrow{b} q_0$

$q_1 \xrightarrow{a} q_0$ ✗ $\quad q_1 \xrightarrow{b} q_2$ ✗ doesn't belong to same set ✓
$q_3 \xrightarrow{a} q_5$ $\quad q_3 \xrightarrow{b} q_4$

$q_1 \xrightarrow{a} q_0$ ✗ $\quad q_1 \xrightarrow{b} q_2$ ✗
$q_5 \xrightarrow{a} q_2$ $\quad q_3 \xrightarrow{b} q_4$

$\Pi_2 \rightarrow \{q_6\}\ \{q_0, q_4\}\ \{q_1\}\ \{q_2, q_3, q_5\}$

$q_2 \xrightarrow{a} q_3$ ✓ $\quad q_2 \xrightarrow{b} q_0$ ✗ doesn't belongs to same set ✓
$q_3 \xrightarrow{a} q_5$ $\quad q_3 \xrightarrow{b} q_4$

$\Pi_3 \rightarrow \{q_6\}\ \{q_1\}\ \{q_0, q_4\}\ \{q_2, q_3, q_5\}$

A B C D

Step 4: Now, Replace $q_0 \rightarrow \{q_0, q_4\}$
$q_4 \rightarrow \{q_0, q_4\}$

and $\{q_2, q_3, q_5\}$, table is drawn as:

	a	b
$\rightarrow\{q_0, q_4\}$		
$\{q_1\}$		
$\{q_2, q_3, q_5\}$		

GOOD WRITE $\{q_2, q_3\ A_5\}$ q_4

DATE: __/__/____
PAGE _____

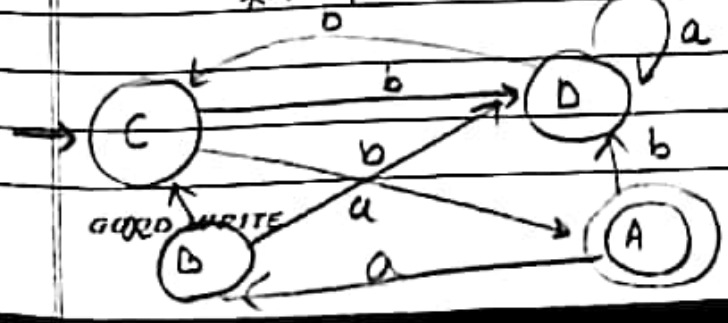
	a	b
→ {q0,q4}	*q6	{q2,q3,q5}
{q1}	→ {q0,q4}	{q2,q3,q5}
{q2,q3,q5}	{q2,q3,q5}	→ {q0,q4}
{q2,q3,q5}	{q2,q3,q5}	{q0,q4}
{q0,q4}	*q6	{q2,q3,q5}
{q2,q3,q5}	{q2,q3,q5}	{q0,q4}
{q6}	{q1}	{q2,q3,q5}

Step-V: Map the assumption into the above transition table after applying partition method

	a	b
→C	*A	D
B	→C	D
D	D	→C
D	D	C
C	*A	D
D	D	C
A	B	D

Now, Replace / Remove 2 similar states

	a	b
→C	*A	D
B	→C	D
D	D	C
*A	B	D



Q. Write Regular expressions for the following languages over the alphabet {a,b}

a) All strings that don't end with ba
b) All the strings that contain an even number of a's
c) All strings that don't contain the substring of ba

a) $\epsilon + a + b + (a+b)^*(ab + aa + bb)$

b) [illegible]

c) $a^* b^*$

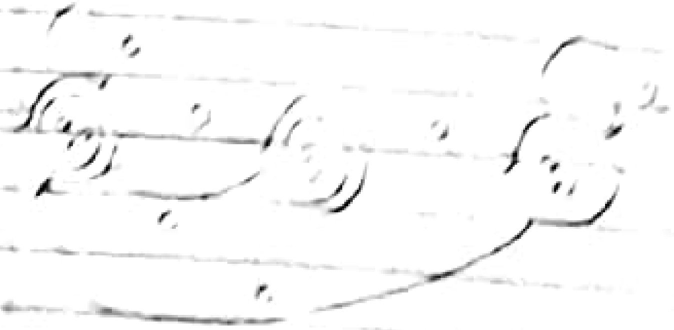
[illegible]

2) for odd no. of a's

[illegible]

* With the help of R.E. draw DFA:

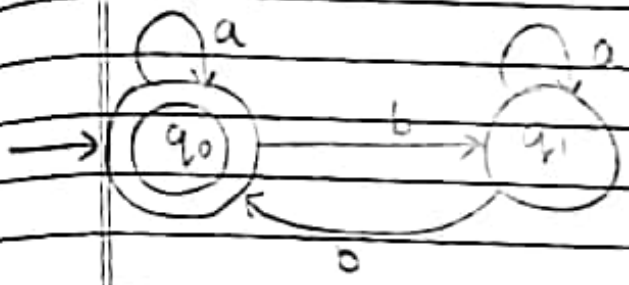
a) DFA: $\epsilon + a + b + (a+b)^*(ab + ba + bb)$



DATE: __ __ ____
PAGE: _____

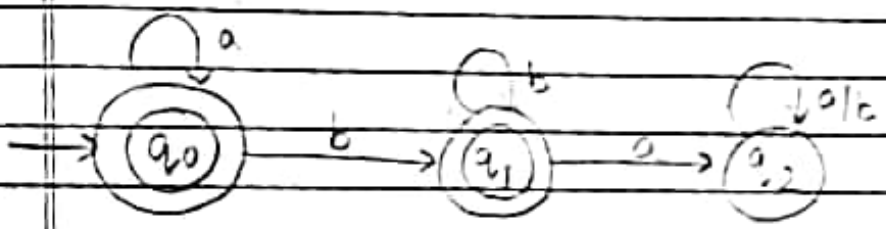
b) DFA: $a^*(ba^*ba^*)^*$

$L = \{ \epsilon, a, bb, abb, abbab, \ldots \}$



c) DFA: a^*b^*

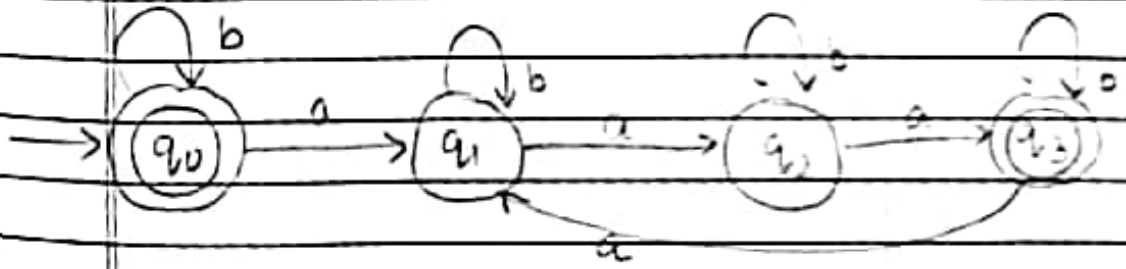
$L = \{ \epsilon, a, b, ab, aa, bb, aaa, aab, \ldots \}$



d) All strings in Σ^* whose number of a's is divisible by 3.

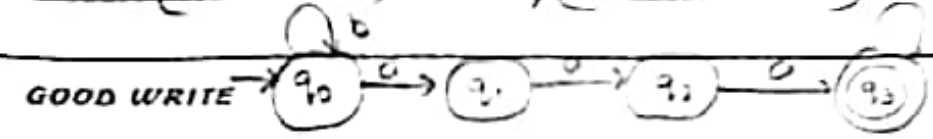
$(ab^*ab^*ab^*)^*b^*$

$L = \{ b, bb, aaa, baaa, aaab, \ldots \}$



e) All strings in Σ^* with exactly one occurrence of the substring aaa

$(b+ab+abb)^*(aaa)(b+bb+abb)^*$



GOOD WRITE

DATE 9/3/18
PAGE

→ Arden's theorem:

Let two regular expressions P and Q defined over alphabet Σ.

If P ∉ ε (null string), then

$$R = Q + RP$$

has an unique soln

$$R = QP^*$$

PROOF: $R = Q + RP$

Put the value of R in R.H.S

$$R = Q + (Q + RP)P$$ put value

$$R = Q + QP + RP^2$$

again

$$R = Q + QP + QP^2 + RP^3$$

.
.
.

$$R = Q + QP + QP^2 + \cdots + QP^n + RP^{n+1}$$

.
.
.

$$R = Q(P^0 + P^1 + P^2 + \cdots)$$
$$= Q(\varepsilon + P^1 + P^2 + \cdots) \quad (\because \alpha^* = \alpha^0 + \alpha^1 + \alpha^2 + \cdots)$$

$$R = QP^*$$

DATE: __/__/____
PAGE _____

* Use of Arden's Theorem to find Regular expression of a DFA:

There are certain assumptions which are made regarding the transition system:

1) The transition diagram should not have 'ε' transition.
2) It must have only a single initial state.
3) Its vertices are $q_1, q_2, \cdots, q_n$.
4) Its final q_i is the final state.
5) w_{ij} denotes the R.E. representing the set of labels ↑ of edges from q_i to q_j. We can get the following set of equation in $q_1, q_2, \cdots, q_n$.

$$q_1 = q_1 w_{11} + q_2 w_{21} + \cdots + q_n w_{n1} + \varepsilon \quad \text{(1)}$$

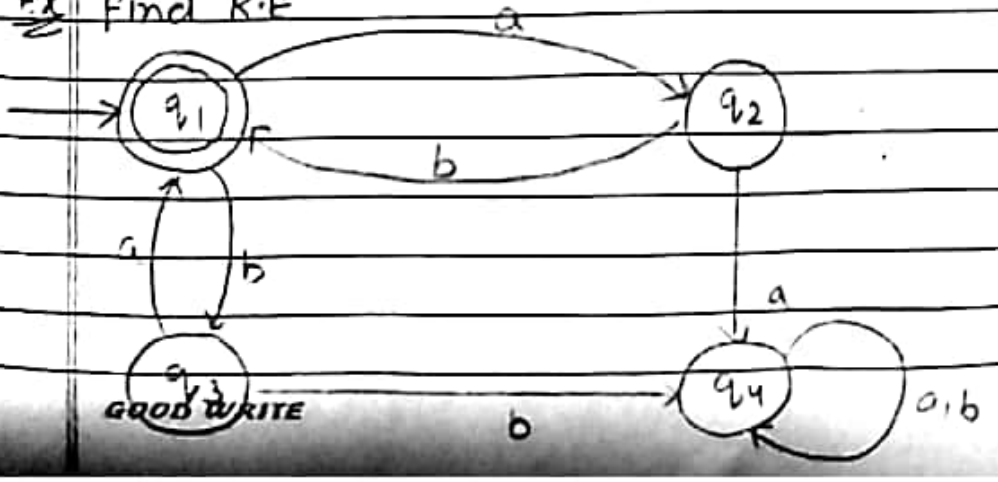
$$q_2 = q_1 w_{12} + q_2 w_{22} + \cdots + q_n w_{n2} + \varepsilon \quad \text{(2)}$$

.
.
.

$$q_n = q_1 w_{1n} + q_2 w_{2n} + \cdots + q_n w_{nn} \quad \text{(n)}$$

We solve these equations for q_i in terms of w_{ij} and it will be the required regular expression.

Ex Find R.E



$q_1 = q_2 b + q_3 a + 0 + \varepsilon = q_2 b + q_3 a + \varepsilon$

$q_2 = q_1 a + 0 + 0 + 0 = q_1 a$

$q_3 = q_1 b + 0 + 0 + 0 = q_1 b$

$q_4 = q_1 a + q_2 a + q_3 b + q_4 a + q_4 b$

$= q_2 a + q_3 b + q_4 a + q_4 b$

Now, final q_1 state

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

$= q_1 ab + q_1 ba + \varepsilon$

Then,

$q_1 = \varepsilon + q_1 ab + q_1 ba$

$q_1 = \varepsilon + q_1 (ab + ba)$

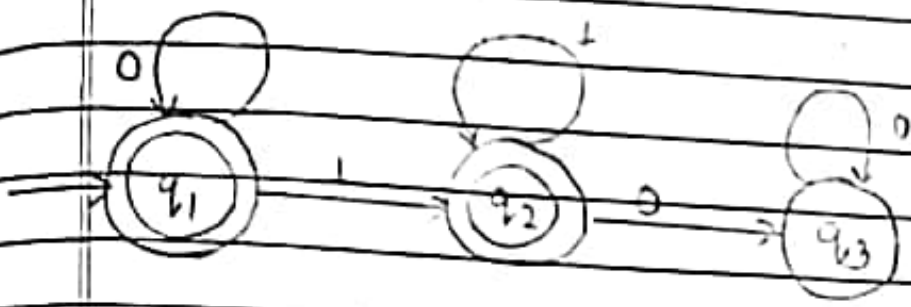
R Q R P

$q_1 = \varepsilon (ab + ba)^*$ $\quad (\because \varepsilon R = R\varepsilon = R)$

$\downarrow$

$q_1 = (ab + ba)^*$

Q: Describe in English the set of strings accepted by the following finite automata:



Ans: Now,

$$q_1 = q_1 0 + q_2 \times 0 + q_3 \times 0 \text{ if } = 0 q_1 + \varepsilon = q_1 0 + \varepsilon$$

$$q_2 = q_1 \times 1 + q_2 1 + q_3 0 = 1 q_1 + 1 q_2 = q_1 1 + q_2 1$$

$$q_3 = q_1 \times 0 + q_2 0 + q_3 0 + q_3 1$$

$$= q_2 0 + q_3 (0+1)$$

Now, we all know that in a given DFA there are two final state then,

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

Then,

$$q_1 = \varepsilon + q_1 0 \Rightarrow q_1 = \varepsilon (0)^*$$

(R = Q + R P)

$$q_1 = 0^*$$

$$q_2 = q_1 1 + q_2 1 \quad \text{---} (1)$$

putting the value of 'q_1' in q_2

GOOD WRITE

$$q_2 = 0^* 1 + q_2 1 \Rightarrow q_2 = 0^* 1 (1)^*$$

(P = Q + P 1)

So, the required regular expression is

$= q_1 + q_2 = 0^* + 0^* 1^*$ (if no symbol is present with the string take ε)

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

Using an identity,

$$\varepsilon + RR^* = R^*$$

Then,

$$q_1 + q_2 = 0^* (\varepsilon + 1\ 1^*)$$

$$\downarrow \quad \downarrow$$

$$R \quad R^*$$

$$= 0^* 1^*$$

→ Set of all strings with any number of 0's followed by any no. of 1's.

* Identities (I):

1.) $\phi + R = R$
2.) $\phi R = R\phi = \phi$
3.) $\varepsilon R = R\varepsilon = R$
4.) $\varepsilon^* = \varepsilon$
5.) $R + R = R$
6.) $R^* R^* = R^*$
7.) $RR^* = R^* R$
8.) $(R^*)^* = R^*$
9.) $\varepsilon + RR^* = R^*$
10.) $(PQ)^* P = P(QP)^*$
11.) $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

GOOD WRITE

12) $(P+Q)R = PR+QR$

13) $R(P+Q) = RP+RQ$

→ PROVE THAT:

$(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

PROOF: L.H.S

$(1+00^*1)[\varepsilon + (0+10^*1)^*(0+10^*1)]$ Using 12th property

$= (1+00^*1)[\varepsilon + (0+10^*1)(0+10^*1)^*]$ (Using 7th property)

Now apply I_4

$= (1+00^*1)(0+10^*1)^*$

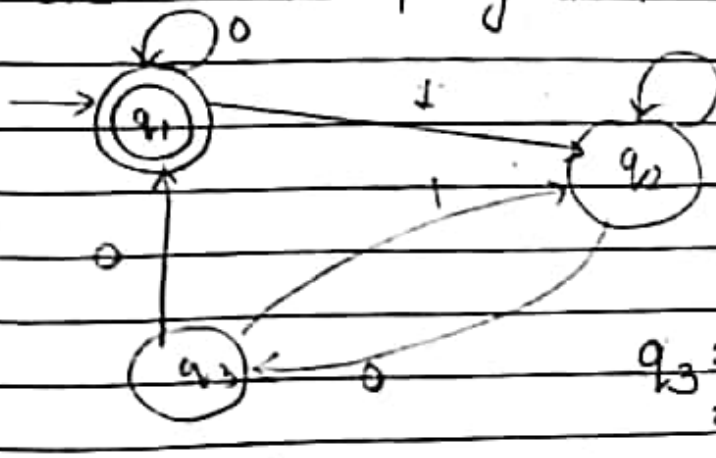
$= (\varepsilon + 00^*)1(0+10^*1)^*$ Using I_1, I_2,

$= 0^*1(0+10^*1)^*$ Using I_4

= R.H.S

Hence proved

→ Construct R.E for given FA:



Now,

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

$= q_1 0 + q_3 0 + \varepsilon$ — ①

$q_2 = q_1 1 + q_2 1 + q_3 1$ — ②

$q_3 = q_1 x 0 + q_2 0 + q_3 x 0$

$= q_3 q_2 0$ — ③

GOOD WRITE

DATE __/__/____
PAGE _____

Now, we have 1 final state, then

$$q_1 = \varepsilon + q_1 0 + q_3 0$$
$$= \varepsilon + 0(q_1 + q_3) \quad \text{—(1)}$$

$q_3 = q_2 0$ put this value in eqn (2)

$$q_2 = q_1 1 + q_2 1 + q_2 01 \qquad \text{Using } I_{1_2}$$
$$= q_1 1 + q_2(1 + 01) \Rightarrow q_2 R = q_1 1 (1+0)^*$$
$$R \quad Q \quad R \quad P$$

putting value of q_3 & q_2 in eqn (1)

$$q_1 = \varepsilon + 0(q_1 + q_2 0)$$
$$= \varepsilon + 0(q_1 + q_1 10 +$$

$$q_1 = \varepsilon + q_1 0 + q_3 0$$
$$= \varepsilon + q_1 0 + q_2 00 = \varepsilon + q_1 0 + q_1 100 + q_2(1+a)\infty$$
$$= \varepsilon + q_1 0 + q_1 100 + q_2(1+01)00$$
$$= \varepsilon + q_1 0 + q_1 100 + q_1 1(1+01)$$

$$q_1 = \varepsilon + q_1 0 + q_1 1(1+01)^* 0 \qquad \text{Using } I_{1_2}$$
$$q_1 = \varepsilon + q_1(0 + 1(1+01)^* 0)$$
$$R \quad Q \quad R \quad P$$

$$R = QP^* \Rightarrow q_1 = \varepsilon(0 + 1(1+01)^* 0)^*$$

GOOD WRITE

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

Union → break the states
Intersection → Add an intermediate state

DATE: 12/03/18
PAGE

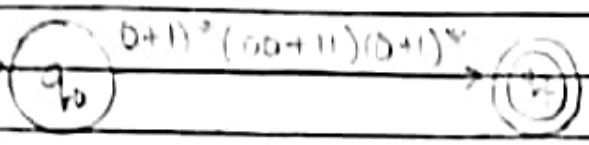
Finite Automata of a Regular Expression

Q: Construct an FA corresponding to the following R.E:

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

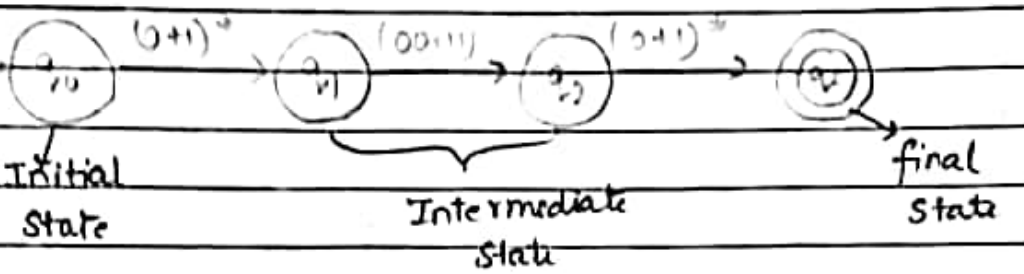
R.E

Step-I

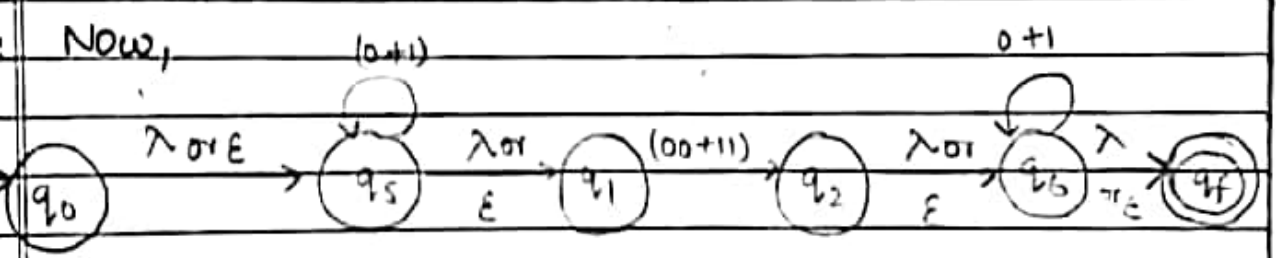


Break the states into intermediate states

Step-II

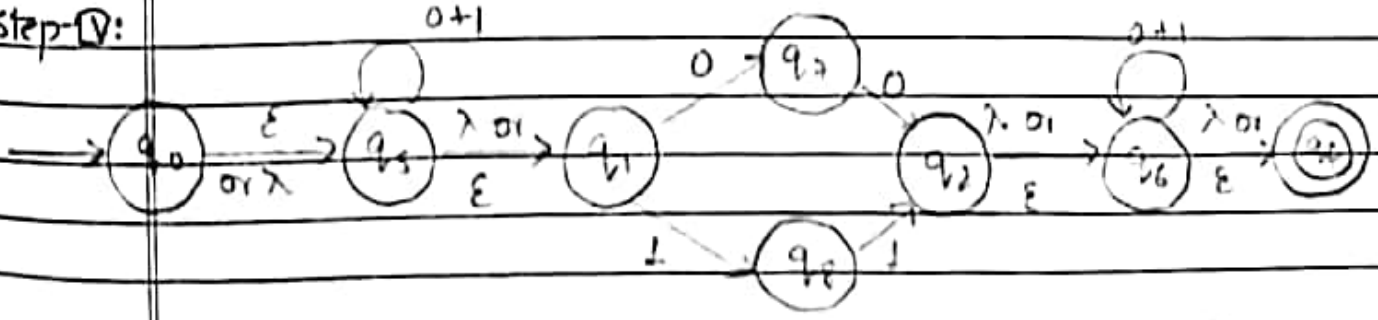


Step-III: Now,



ε: w/o reading the i/p symbol move to another state.

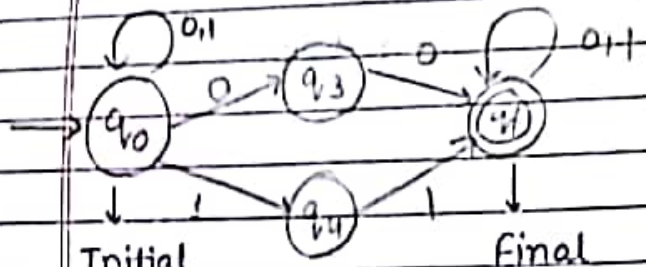
Step-IV:



Break the i/p into two states such as (00 or 11).

GOOD WRITE

Step-2: Remove all λ or ε move, we get the FA:



NFA

00011101

Step-3: Draw a DFA with the help of NFA:

→ Step-1: Transition table for NFA:

S:

State / Symbol	0	1
→q_0	$\{q_0, q_3\}$	$\{q_0, q_4\}$
q_3	$\{q_f\}$	$\{\}q_f$
q_4	$\{\}$	$\{q_f\}$
*q_f	$\{q_f\}$	$\{q_f\}$

→ Step-2: Transition table for DFA:

S:

State / Symbol	0	1
→ q_0	$\{q_0, q_3\}$	$\{q_0, q_4\}$
$\{q_0, q_3\}$	$\{q_0, q_3, q_f\}$	$\{q_0, q_4\}$
$\{q_0, q_3, q_f\}$	$\{q_0, q_3, q_f\}$	$\{q_0, q_4, q_f\}$
$\{q_0, q_4\}$	$\{q_0, q_3\}$	$\{q_0, q_4, q_f\}$
$\{q_0, q_4, q_f\}$	$\{q_0, q_3, q_f\}$	$\{q_0, q_4, q_f\}$

GOOD WRITE

Q: $10 + \underbrace{(0+11)\,0^*}_{R.E}\,1$

DATE: __/__/____
PAGE _____

For $\{q_0, q_3\}$ on 0, the next possible state is

$\delta(q_0, 0) \cup \delta(q_3, 0) = \{q_0, q_3\} \cup \{q_f\}$

$\delta(q_0, 1) \cup \delta(q_3, 1) = \{q_0, q_4\}$

$\delta(q_0, 0) \cup \delta(q_3, 0) \cup (q_f, 0) = q_0, q_3, q_f, q_f$

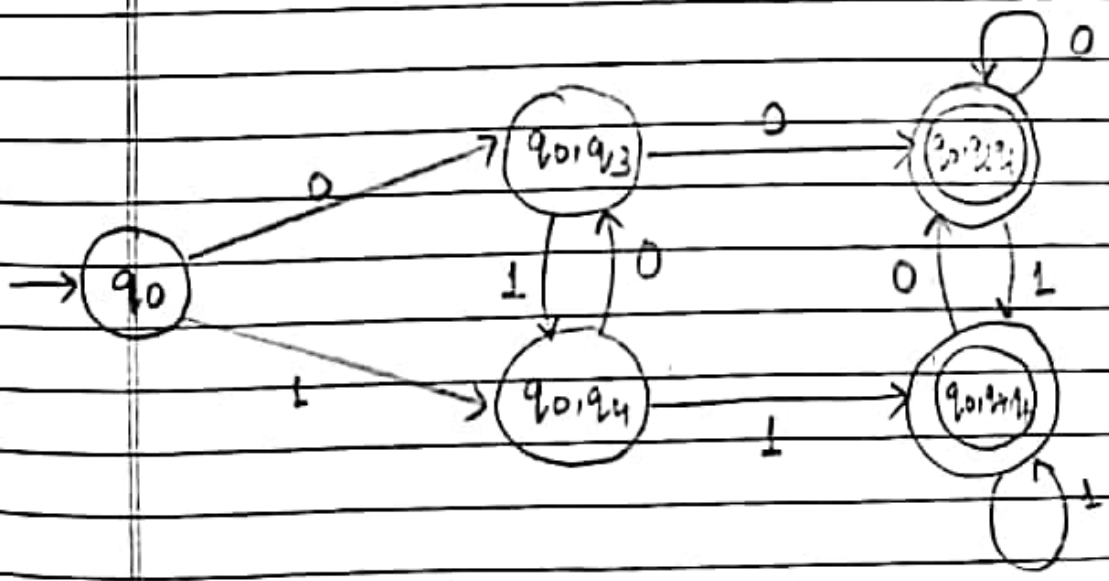
$\delta(q_0, 1) \cup \delta(q_3, 1) \cup \delta(q_4, 1) = q_0, q_4, q_f$

$\delta(q_0, 0) \cup \delta(q_4, 0) = \{q_0, q_3\}$

$\delta(q_0, 1) \cup \delta(q_4, 1) = \{q_0, q_4, q_f\}$

$\delta(q_0, 0) \cup \delta(q_4, 0) \cup \delta(q_f, 0) = \{q_0, q_3, q_f\}$

$\delta(q_0, 1) \cup \delta(q_4, 1) \cup \delta(q_f, 1) = \{q_0, q_4, q_f\}$



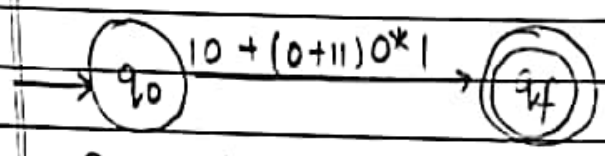
DFA

DATE: __/__/__
PAGE

Q: Construct a fA corresponding to the following R.E

$$\underbrace{10 + (0+11)\,0^*1}_{RE}$$

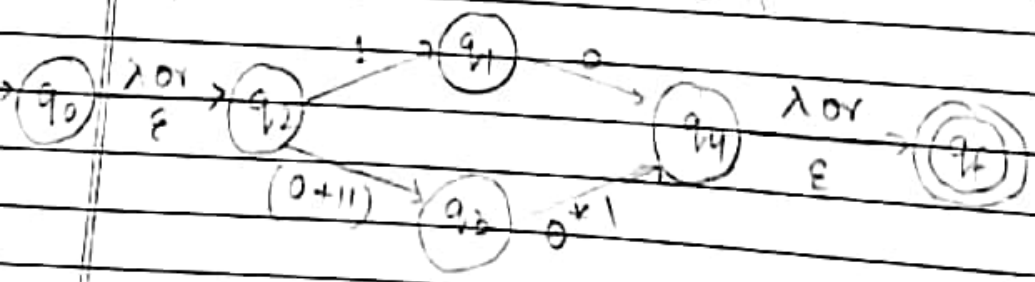
Ans: Step - I:



Break the states into intermediate states

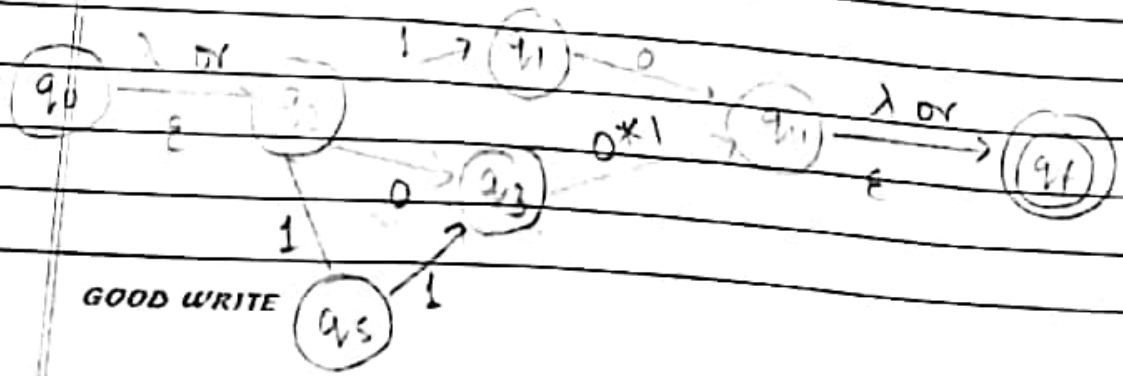
Step-II:

Step-III:



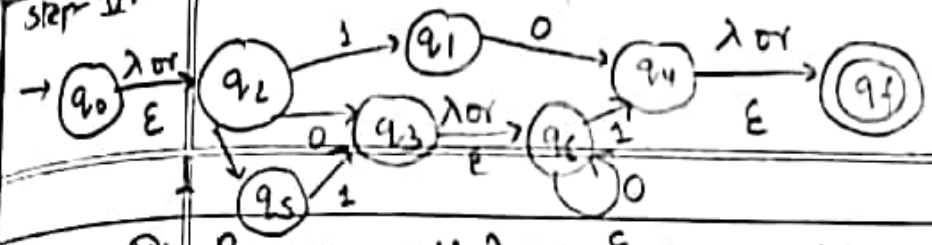
Break the state into sub-states such as

Step-IV:

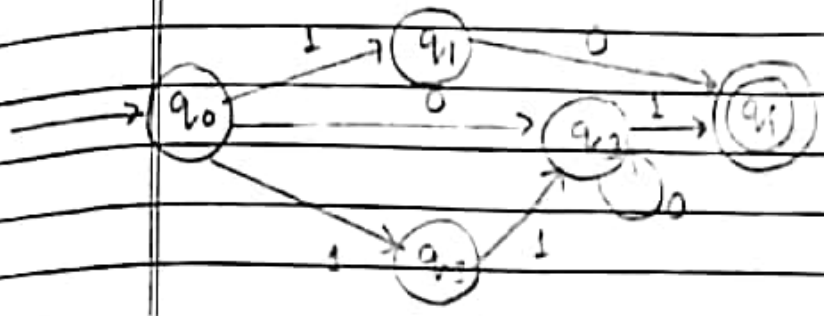


GOOD WRITE

Step-V:



Step-VI: Remove all λ or ε move, we get the FA:



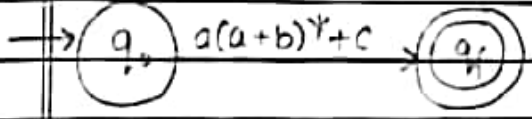
NFA

Q: Regular expression to FA:

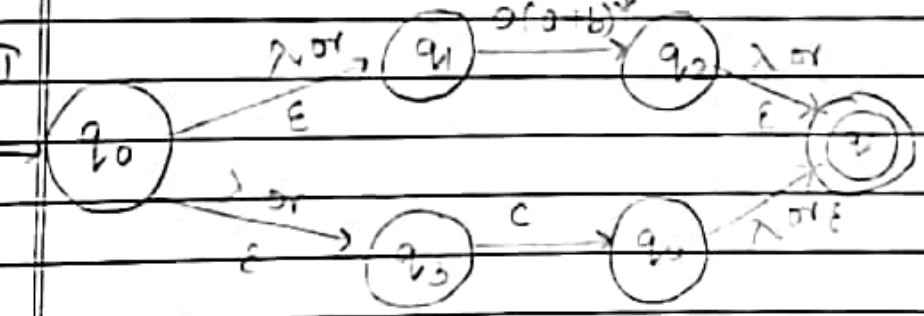
R.E = a(a+b)*+c

* Precedence: ()
*
+

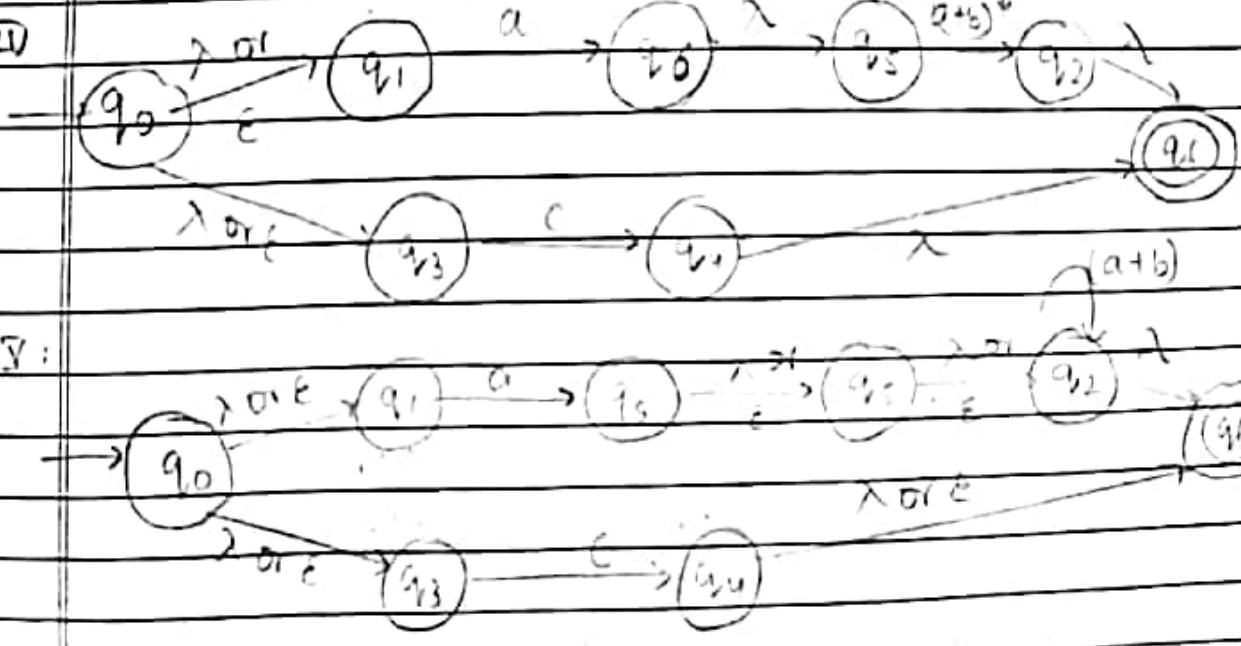
Step - I



Step - II



Step - III



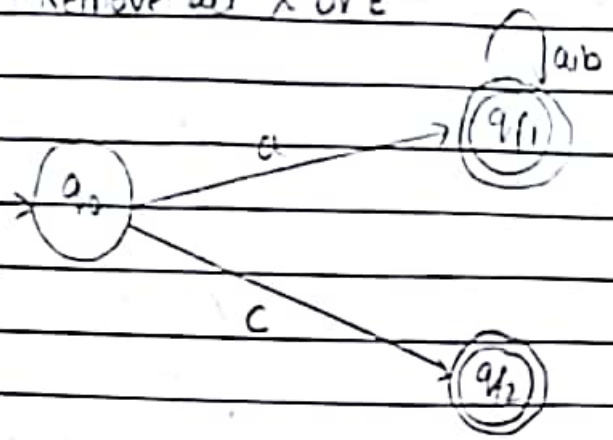
Step - IV:

GOOD WRITE

0, 5, 10, 15, 25 ---

DATE: __/__/____
PAGE _____

Step - VI: Remove all λ or ε



13/03/18

Q: i) $L = \{ \omega \in (a,b)^* : n_a(\omega) \bmod 3 = 0 \}$

ii) $L = \{ \omega : |\omega| \bmod 5 = 0, \omega \in (a,b)^* \}$

Ans: i)

$L = \{ \lambda, aaa, a, bb, bbb, aaab, aabb, \dots \}$

$(a\,b^*\,a\,b^*\,a\,b^*)^*\,b^*$

ii) $L = \{ \lambda, aaaaa, bbbbb, aaabb, \dots \}$

$((a+b)^5)^*$

* Algebraic laws for Regular Expressions:

i) Associative:

$L+(M+N) = (L+M)+N$

$L\cdot(M\cdot N) = (L\cdot M)\cdot N$

ii) Commutative: $L+M = M+L$ However,

GOOD WRITE

+ → union
. → concatenation

$L \cdot M \neq M \cdot L$ in general

for eg. → $L = aab$, $M = ba$ ⇒ $L \cdot M \neq M \cdot L$

$aabba \neq baaab$

(iii) Identity:

$\phi + L = L + \phi = L$ (empty language)

$\varepsilon L = L \cdot \varepsilon = L$ (Empty string)

(iv) Annihilator:

$\phi \cdot L = L \cdot \phi = \phi$

(v) Distributive:

$L(M+N) = LM + LN$ (Left distribution)

$(M+N)L = ML + NL$ (Right distribution)

(vi) Idempotent law:

$L + L = L$

(vii) Closure property:

$\rightarrow (L^*)^* = L^*$

$\rightarrow \phi^* = \varepsilon$

$\rightarrow \varepsilon^* = \varepsilon$

$\rightarrow L L^* = L^+ \Rightarrow L^+ = L L^*$

$\rightarrow L^* = L^+ + \varepsilon$

(viii) De morgan's law:

$(L \cdot M)^* = L^* + M^*$

$(L+M)^* = L^* \cdot M^*$

GOOD WRITE

16/03/18

If we are able to find out the expression/pattern for that particular language then it will be ——— Regular language.

DATE: __/__/____
PAGE _____

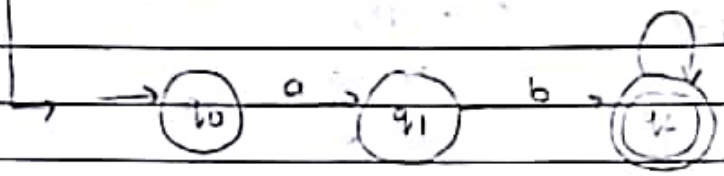
* Non-Regular language:

Some established theorem

Pumping lemma:

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

$L = \{a^p \mid p \text{ is prime no.}\}$ Non-Regular



(Here to include a self-loop or we have to increase the length by one step)

Let L is regular defined by some DFA having a constant 'n' (which depends on L) such that for every string w in L, $|w| \geq n$, we can break w into three strings, $w = xyz$, such that:

i) $|y| > 0$ i.e., $y \neq \varepsilon$

ii) $|xy| \leq n$

iii) $\forall\ i \geq 0$, the string xy^iz is also in L.

(For any language if it doesn't hold and not a Regular Expression)

* Method to prove a language L is not regular:

→ At first, we have to assume that L is regular.

→ So, the pumping lemma should hold for L.

→ Use the pumping lemma to obtain a contradiction-

1) Select w such that $|w| \geq n$.
2) Select y such that $|y| \geq 1$
3) Select x such that $|xy| \leq n$
4) Assign remaining string to z.
5) Select ~~suitably~~ suitably to show that resulting

GOOD WRITE

String is not in L.
Hence L is not regular.

DATE: __/__/____
PAGE _____

* Regular language:

Defn: Let M = (Q, Σ, q0, F, δ) be a DFA. The language L is regular if there exists a machine M such that L = L(M).

How to check a language is Regular or not?
How to answer the question?

Ans: If the language is finite then it is a Regular language.

Ex: L = {aa, ab, ba, bb} is a regular language because it is a finite language.
But infinite language can be regular or non-regular.

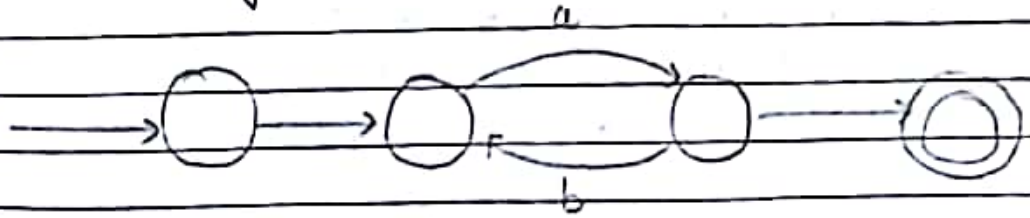
Ex: L = {ab, abab, ababab, ...} check regular or not?

Soln: In the infinite language there is some particular pattern which can be generated by applying some loop. Then we can say that the language is regular.

Like in above language ab is a kind of pattern which can be repeated by using a loop to generate such infinite language.

And if there is no such pattern available in the given language then, we can't able to generates its finite automata or we can say the language is not regular.
This is what Pumping Lemma says.

It means if we are not able to find some pattern then we can't design finite automata. On the other hand,

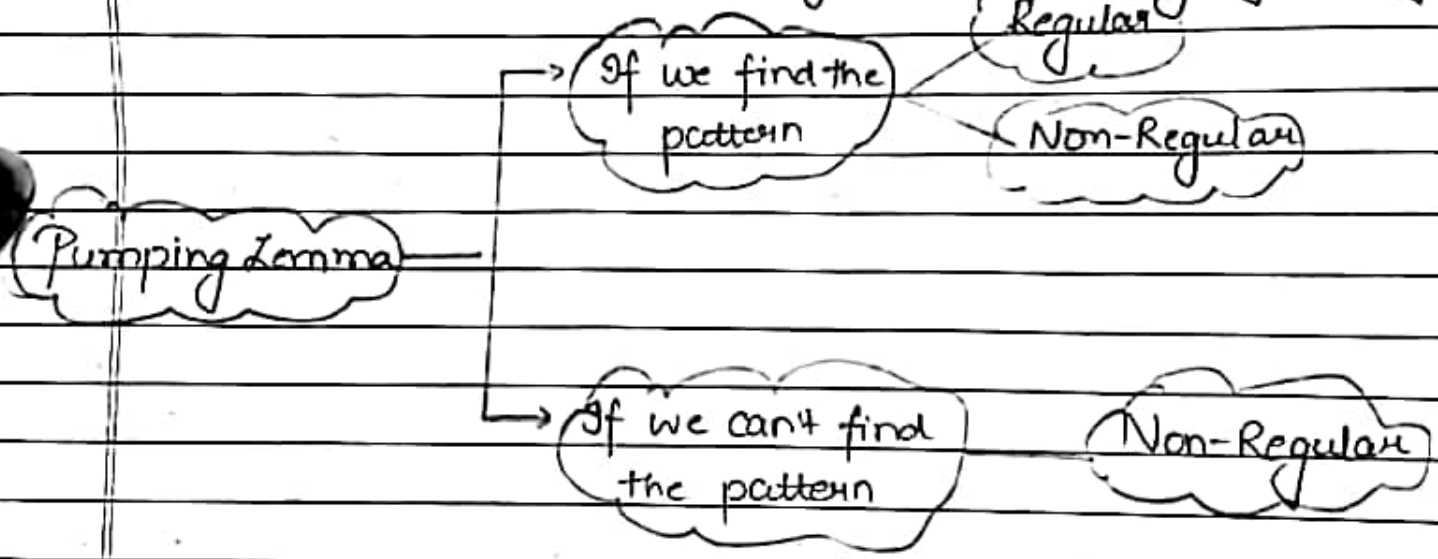


GOOD WRITE

If the infinite language has to be accepted by a finite automata then, there has to be a loop inside the automata, and if there is no such string or pattern which can be used with loop to generate infinite language then this is impossible to generate finite automata for such languages.

It means pumping lemma is a negative test.

If we don't find such pattern in the language then we can say that the ~~lemma~~ language is not regular, but finding such pattern doesn't assure you that the language is regular.



Hence, pumping lemma is -ve test.

Ex: $L = \{ a^p \mid p \text{ is prime} \}$

$p = 2, 3, 5, 7, 11, 13, 17, 19, \ldots$

No pattern in the language

Hence, Language is not - regular

* Pumping Lemma:

Let L be a regular language. Then there exists a constant (n) (which depends on L) such that for every string w in L, such that $|w| \geq n$, we can break w into three strings, $w = xyz$, such that:

1. $|y| > 0$ i.e., $y \neq \varepsilon$
2. $|xy| \leq n$
3. for all $i \geq 0$, the string xy^iz is also in L.

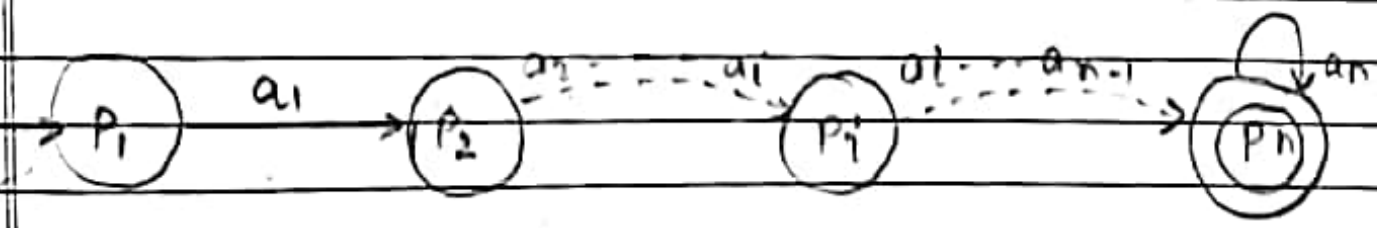
PROOF: Let L is a regular defined by some DFA, M having 'n' states. Let $w = a_1, a_2, a_3, \cdots a_n$ is in L.

$|w| = n \geq n$. Let the start state be P_1.

Let $w = xyz$, where $x = a_1, a_2, a_3, \cdots, a_{n-1}$

$y = a_n$

$z = \varepsilon$



$\delta(P_1, a_1) = P_2$
$\delta(P_2, a_2) = P_3$
.
.
.
$\delta(P_n, a_n) = P_{n+1}$

But there are only 'n' states
⇒ There must be a loop.
Let there be a loop in P_n state.
Let $x = a_1, \cdots, a_{n-1}$
$y = a_n$
$z = \varepsilon$

Therefore, $xy^iz = a_1 \cdots a_{n-1} (a_n)^i \varepsilon$

for $i = 0$, $a_1 \cdots a_{n-1}$ is accepted
for $i = 1$, $a_1 \cdots a_n$ is accepted
for $i = 2$, $a_1 \cdots a_{n+1}$ is accepted
.
.
.

for $i = 10$, $a_1 \cdots a_{n+9}$ is accepted and so on.

* **Uses of pumping lemma:**
Pumping lemma is to be applied to show that certain languages are not regular. It should never be used to show that some languages is regular.
→ If L is regular, it satisfies pumping lemma otherwise L is not regular.

* Prove that $L = \{a^n b^n \mid n \geq 1\}$ is not regular.
PROOF: At first, we assume that L is regular and n is the number of states.
→ Let $w = a^n b^n$. Thus $|w| = 2n > n$
Consider

$$w = \overbrace{aa\cdots a}^{n}\ \overbrace{bb\cdots b}^{n}$$
$$\vdash xy \dashv \vdash z \dashv$$

'xy' contain only a's. (Because $|xy| \leq n$)
Let $|y| = l$, where $l > 0$. (Because $|y| \geq 1$)

Then, the break up of x, y and z can be as follows

$$w = \underbrace{a^{n-l}}_{x}\ \underbrace{a^{l}}_{y}\ \underbrace{a^{n}}_{z}$$

from the definition of Pumping lemma,
$w = xy^i z$, where $i = 0, 1, 2, \cdots, \infty$, should $\in L$

GOOD WRITE

That is $a^{n-1}(a^1)^i b^n \in L$, $\forall\ i = 0, 1, 2, \ldots, \infty$,

Putting $i = 0$, we get

$a^{n-1} b^n \notin L$ (contradiction)

Hence, the language L is not regular.

* Prove that $L = \{w \mid w \text{ is a palindrome on } \{a,b\}^*\}$ is not regular.

PROOF: Given $L = \{aabaa, aba, abbbba, \ldots\}$

Let L be regular and n is a constant (number of states)

Let $w = a^n b a^n$, such that $|w| = 2n+1 > n$ and L is regular it must satisfy pumping lemma.

Consider,

$$w = \underbrace{aa\ldots a}_{n}\ b\ \underbrace{aa\ldots a}_{n}$$
$$\vdash xy \dashv\vdash z \dashv$$

xy contains only a's. (Because $|xy| \leq n$)

Let $|y| = 1$, where $1 > 0$. (Because $|y| > 0$)

Then, the break up of x, y and z can be as follows

$$w = \underbrace{a^{n-1}}_{x}\ \underbrace{a^1}_{y}\ \underbrace{ba^n}_{z}$$

from the definition of pumping lemma,

$w = xy^iz$, where $i = 0, 1, 2, \ldots, \infty$, should $\in L$.

That is, $a^{n-1}(a^1)^i b a^n \in L$, $\forall\ i = 0, 1, 2, \ldots, \infty$.

Putting $i = 0$, we get

$a^{n-1} b a^n \notin L$. (Because it is not a palindrome)

Contradiction, hence the language is not regular.

17/03/18

* Closure properties of Regular languages:

1) The union of two regular language is regular - Closure under union

Theorem: If L and M are two regular language then so L∪M is also regular.

PROOF: $L_1 = \{a, a^3, a^5, ---\}$

$L_2 = \{a^2, a^4, a^6, ---\}$

$L_1 \cup L_2 = \{a, a^2, a^3, ---\}$

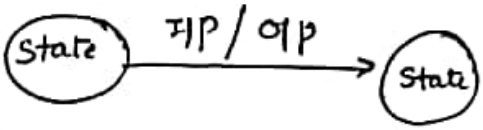
$R.E = a(a^*) = a^+ \quad \{\overline{L} = \Sigma^* - L\}$

$L_1 \cap L_2 = \phi \qquad \Sigma = \{a\}$

$\overline{A} = U - A \qquad \overline{L} = \{\lambda, a, a^2, a^3, \ldots\}$

2.) Intersection is also a R.E.

3.) The complement of a regular language is also regular closure under intersection.



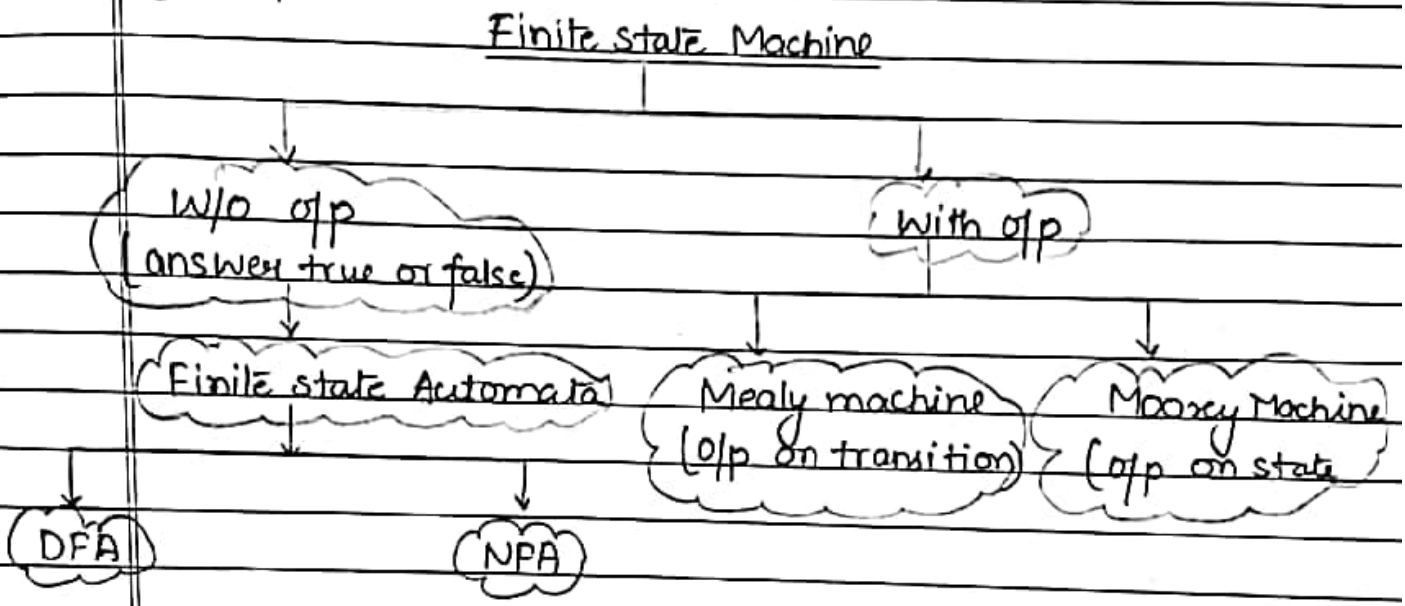
DATE: __/__/____
PAGE _____

19/3/18

* What is a FSM?

A finite state machine is a machine that has some many states and has a logical way of changing from one state to another state under guiding rules.

* Types of FSM



* Mealy Machine: A Mealy Machine is an FSM whose o/p depends upon the present state as well as present i/p. In a mealy machine, every transition for a particular i/p symbol has a fixed o/p.

It can be described by a 6-tuple:

$M_e = (Q, \Sigma, O, \delta, X, q_0)$

* Q is a finite set of states
Σ is a finite set of symbols called I/p alphabet.
O is a finite set of symbols called o/p alphabet.
δ is the i/p transition fn where $\delta : Q \times \Sigma \to Q$
X is the o/p transition fn where $X : Q \times \Sigma \to O$
q_0 is the Initial state ($q_0 \in Q$)

GOOD WRITE

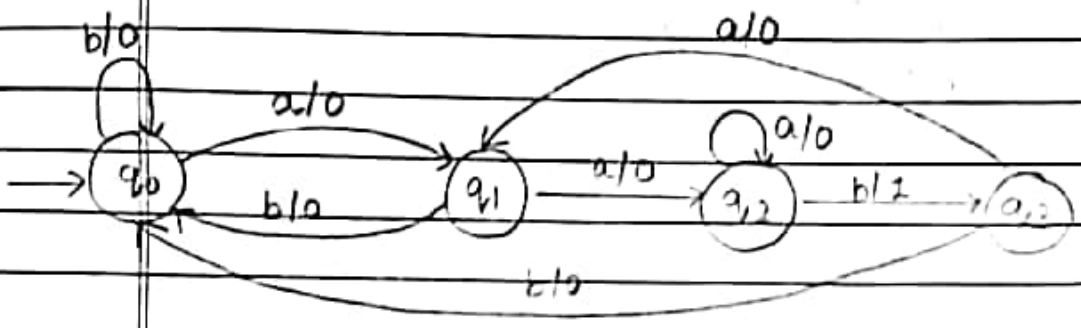
-> No final state is present in it

DATE: __/__/____
PAGE _____

* For each state and an i/p alphabet, the state to go to next and the character that is o/p (printed).

Input / Output

Q: Now, $\Sigma = \{a, b\}$, $O = \{0, 1\}$



Draw the transition table :

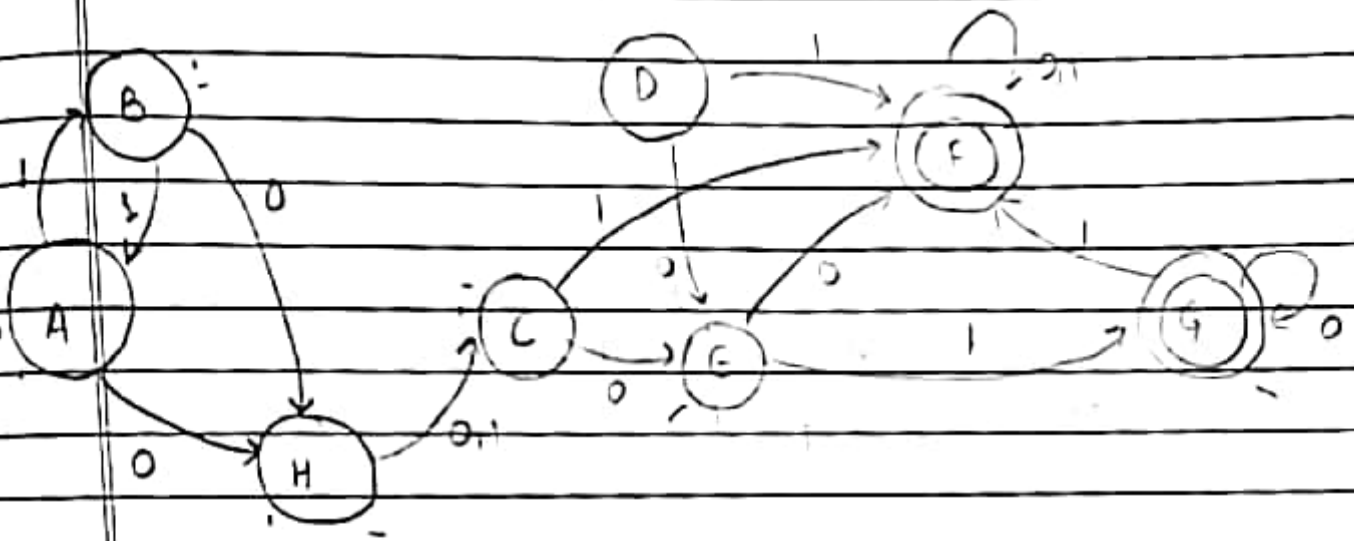
States	For input a		For input b	
	NEXT State	Output	NEXT State	Output
→ q_0	q_1	0	q_0	0
q_1	q_2	0	q_0	0
q_2	q_2	0	q_3	1
q_3	q_1	0	q_0	0

* Test String :-

a a b
↓ ↓ ↓
0 0 1

$\delta(a,0) = h$ → Blank $\delta(c,0) = E$ → Blank

$\delta(a,1) = B$ → Blank $\delta(c,1) = E$ → non-Blank



	a	b	c	d	e	f	g	h
a								
b								
c	1	1						
d	1	1	1					
e	0	0	0	0				
f	E	E	E	E	E			
g	E	E	E	E	E			
H			1	1	0	E	E	

At a, b both are same the block will be b.

$\delta(a,0) = E$, $(a,0) = H$

$\delta(a,1) = F$, $(a,1) = B$

$H,1) = F$

$E,0) = F$, $(B,0)$

$E,1)$

Length must be equal of o/p & i/p string

DATE: __/__/____
PAGE _____

Q: Design a Mealy Machine which prints 1's compliment of i/p bit string over alphabet $\Sigma = \{0,1\}$

Ans: $M_e = (Q, \Sigma, O, \delta, \lambda, q_0)$

bit string = 0 1 0 1 1 1 0 1 (ω)

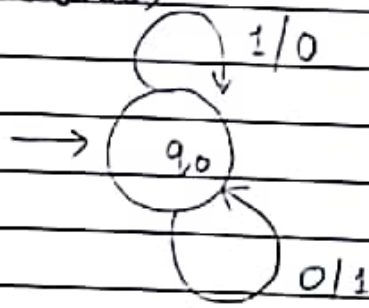
1's complement = 1 0 1 0 0 0 1 0 (ω')

$Q = \{q_0\}$, $\Sigma = \{0,1\}$, $O = \{0,1\}$

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

$\lambda : Q \times \Sigma \to O$

q_0 (initial state)



Transition Table:

Present State	Next state			
	For i/p 0		For i/p 1	
	State	Output	State	Output
→ q_0	q_0	1	q_0	0

DATE: __/__/____
PAGE _____

id → Address

id(x)
95526304 → Address of x

id("hello")
4598 2752

>>> a = 10
>>> id(a)
Address of a

if the value of the variable is same then the address will be same

>>> b = 10
>>> id(b)
Address of b

→ variable is the identifier of the memory

FUNCTION

Scanf → input (use here)

Text editor:

```
def add(a,b):
    a=8
    b=9
    c=a+b
    print()
```

IDLE:

```
>>> add(8,9)
17
```

GOOD WRITE

Q: Odd string {a,b}

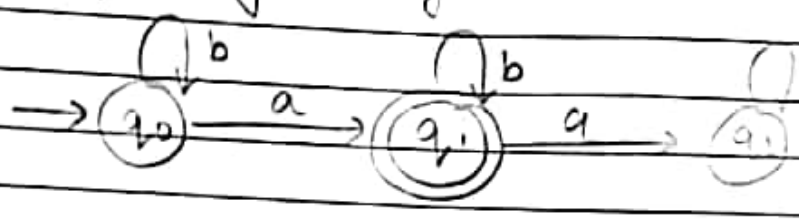
Q: 011

Q: at least two 0's

Q: Draw a F.A → accept string of 0's and 1: ending with the string 011

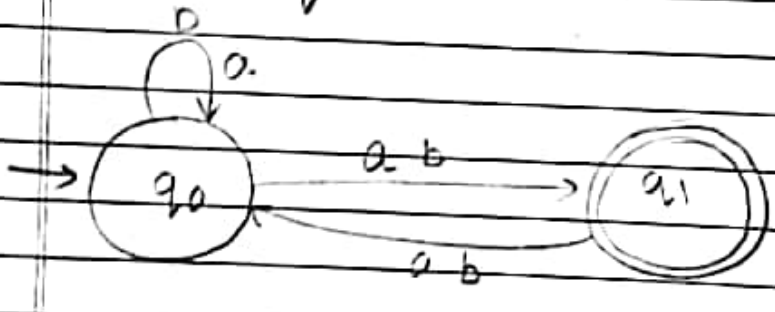
Q: Σ = {a, b} Accept substring aa

Q: string having exactly one a

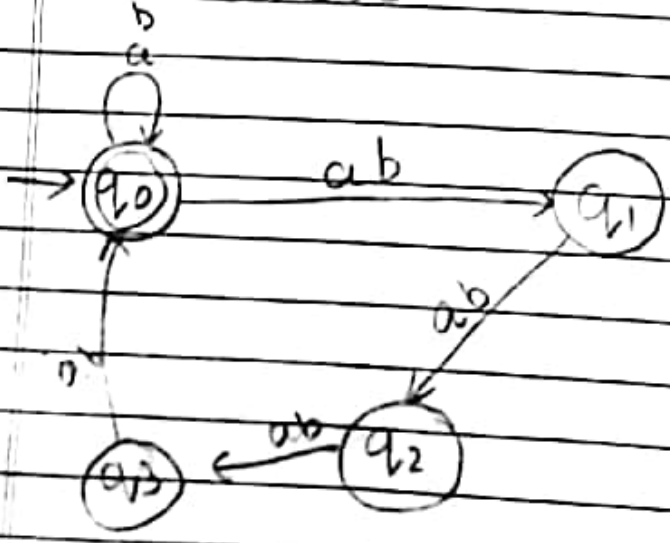


abb

Q: Even no. of a's and b's



aabb abab

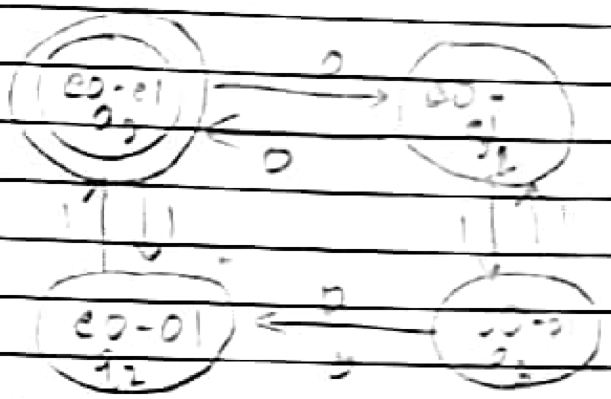


a a a a a a

GOOD WRITE

~~ab aaa bbb~~

0011

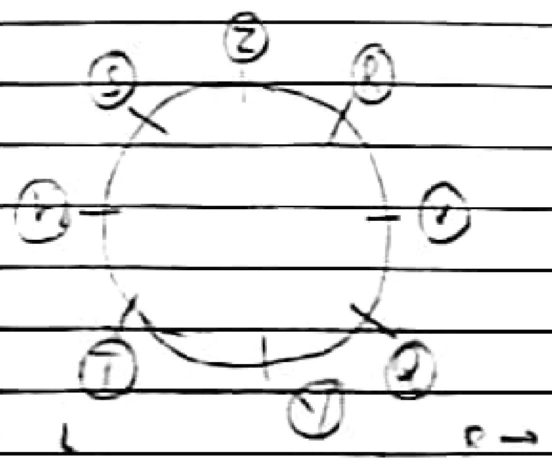


000111 01

Q. Persons — Q, R, S, T, V, W, Z sitting around a circular table, facing the centre.
3 - males, 5 - females in the group.
No two males are immediate neighbours.
V sits second to the right of his wife. S sits third to the right of V. W sits second to right of her husband Z. Z is not an immediate neighbor of V's wife. T is a male and it is not immediate neighbor of V. R sits second to the right of Q.

T → M
V → M
Z → M

3 M → +
5 F → ⊖



5 females

R related to T

GOOD WRITE

Class - user-defined data type, collection of similar type of object.
Class containing the method, variable, DATE: __/__/__ PAGE: _____

Option → General → open edit window

→ a=10
→ b=10
print (id(a), id(b))

To know the address id() is used
list / modify / mutable
tuple / modify / immutable

The range b/w -5 to 256 value will be same

→ if True:
    print ('True')
else:
    print ('False')

str = "python"
print (str * 3)

→ Getting input from user:
x = input ('Enter the name:')
print (x)

→ num = int (input ("Enter the number:"))
if num > 10:
    print ("hello python")
else:
    print ("not greater than 10")

→ num = int (input ("Enter the number:"))
a, b = 0, 1
while b < num:
    print (b)
    a, b = b, a+b

→ str = "hello"
print (str[-3])
l

→ str = "hello"
print (str[:])
"hello"

→ GOOD WRITE
str = "hello"
print (str[2:4])

[starting index : ending index]

DATE: __/__/____
PAGE _____

```
str = "python"
for i in str:
    print(i)

for i in range(1,10):
    print(i, end = "g")
```

13/2/18

DFA Minimization (DFA)

The task of DFA minimization then, is to automatically transform a given DFA into a state-minimized DFA.

→ Several algorithms and variants are known
→ Note that this also in effect can minimize a NFA (Equivalence relation of NFA & DFA).

→ DFA minimization Algorithm:

1.) Recall that a DFA $M = (Q, \Sigma, \delta, q_0, F)$
2.) Two states p and q are distinct if
→ p in F and q not in F or vice-versa.
→ For some α in Σ, $\delta(p, \alpha) = q'$ and $\delta(q, \alpha) = q''$ are distinct.

$q' \neq q'' \in Q$

3.) Using this inductive definition, we can calculate which states are distinct.

Algorithm

1) Create lower-triangular table Distinct (Name), initially blank.

⇒ for every pair of states (p,q):
1.) if p is final and q is not or vice versa
Distinct (p,q) = ε

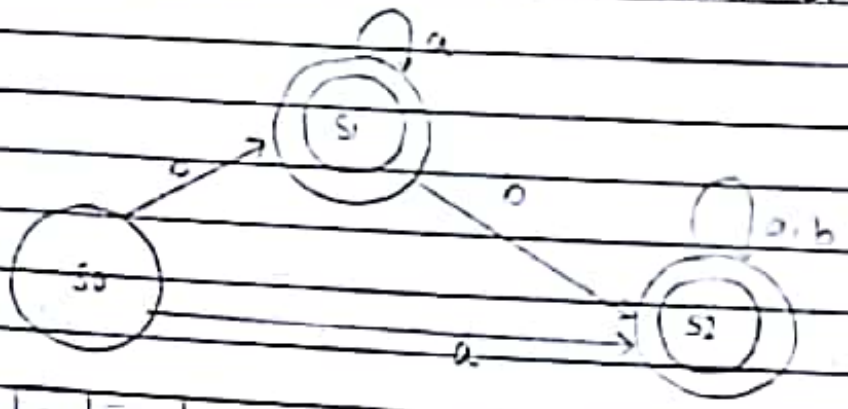
2) Loop until no change for an iteration:
1.) For every pair of states (p,q) and each symbol

GOOD WRATE

→ if Distinct (p,q) is blank and
DISTINCT $(\delta(p,\alpha), \delta(q,\alpha))$ is not ~~block~~ blank
DISTINCT $(p,q) = \alpha$

⇒ Combine all states that are not distinct.

Eg



S_0			
S_1			
S_2			
	S_0	S_1	S_2

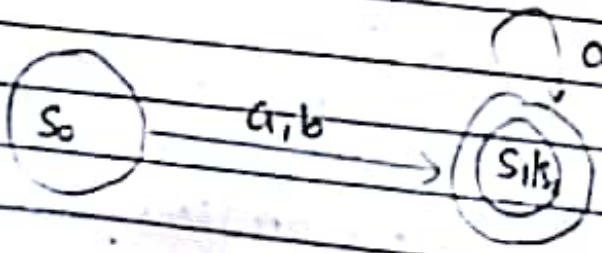
$S_1 \in F$
$S_0 \notin F$

S_0			
S_1	ε		
S_2	ε		
	S_0	S_1	S_2

→

S_0			
S_1	ε		
S_2	ε		
	S_0	S_1	S_2

Now, merge (S_1, S_2) we get,



GOOD WRITE

Q. Design a mealy machine, which increments a binary bit string by 1.

Sol: $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0,1\}$, $O = \{0,1\}$

$\delta : Q \times \Sigma \to Q$ $\quad q_0 \to$ initial state

$\lambda : Q \times \Sigma \to O$

CASE I: I/p string → 01001110 → Right most bit if 0 = 1

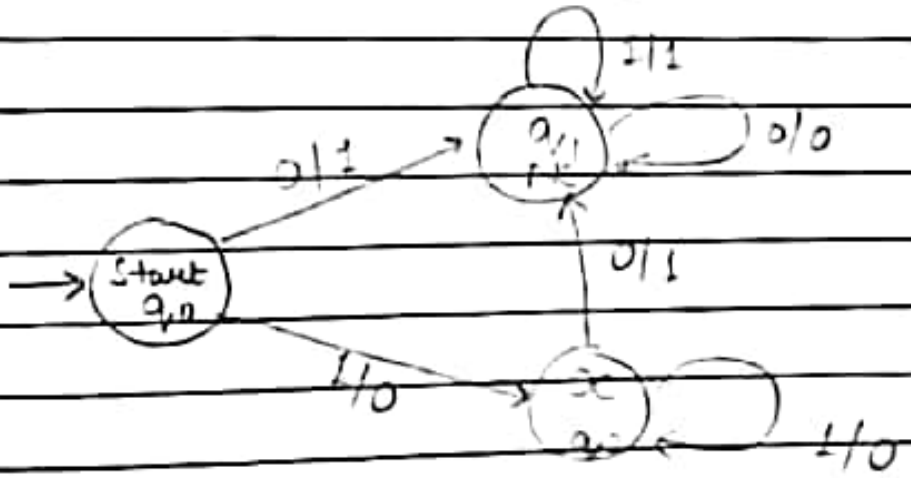
+ 1

O/p string → (01001111)

CASE - II: I/p string → 10110111 — ω

+ 1

(10111000) — ω°

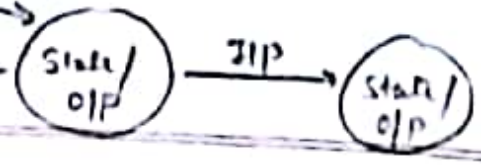


10001 (ω)

+ 1

(10010)

(10010)



DATE: __/__/____
PAGE: _____

* MOORE MACHINE:

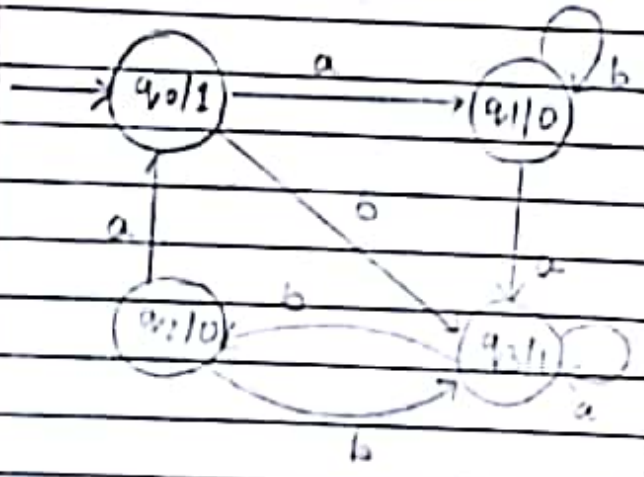
The value of the o/p fn is depend on the present state only.

$$\lambda : Q \to O$$

Ex: $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0,1\}$, $O = \{0,1\}$

Transition Table:

Present State	O/p by the present state	Next State	
		After the i/p a	After i/p b
→ q_0	1	q_1	q_2
→ q_1	0	q_3	q_1
→ q_2	0	q_0	q_2
→ q_3	1	q_3	q_2



Test String:

w		a	b	a	b
State	q_0	q_1	q_1	q_3	q_2
O/P	1	0	0	1	0

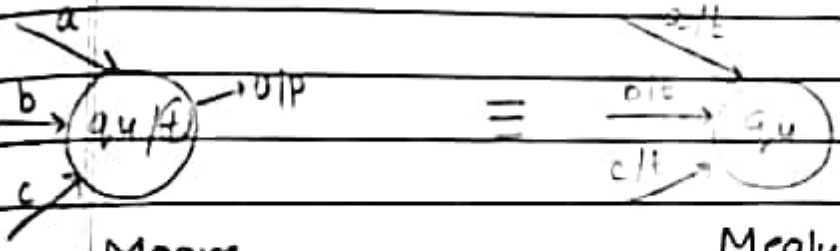
GOOD WRITE

* Equivalence of Machines:

→ Moore to Mealy conversion:

Defn: Let M_o be a Moore Machine that prints x in the start.
Let M_e be a Mealy Machine.
The two machines are equivalent if for every input, whenever the o/p from M_e is w, the output from M_o is xw.

Theorem: By constructive algorithm,

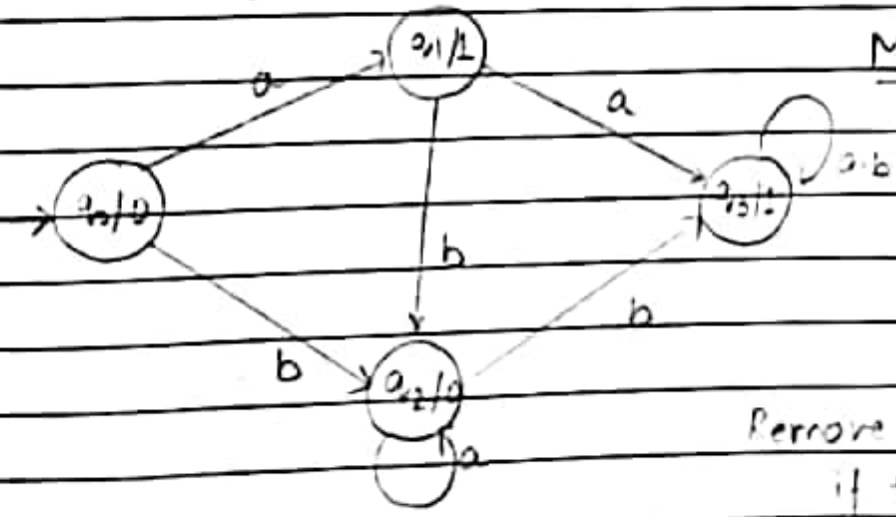


Moore Machine

Mealy Machine

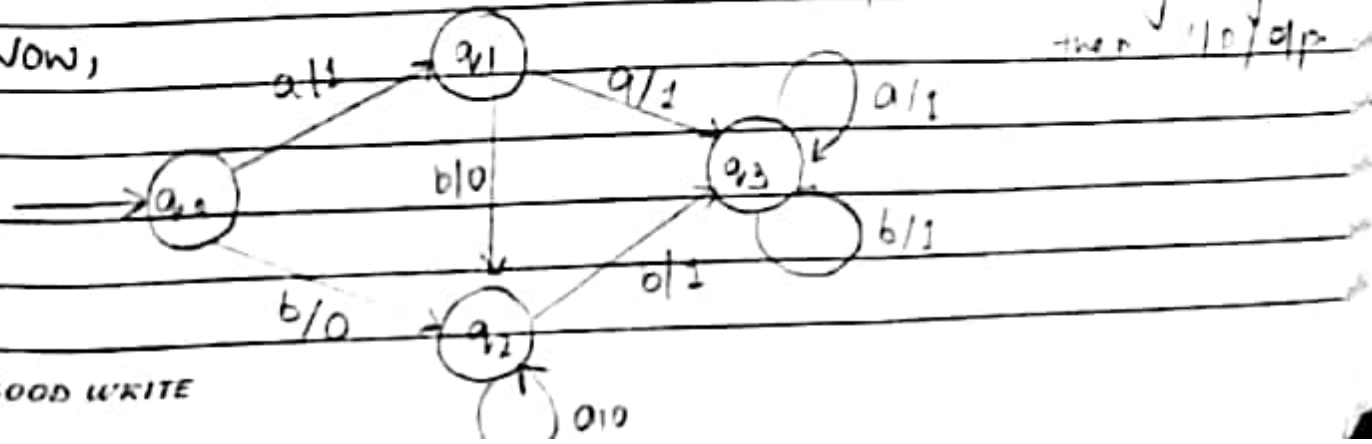
Eg

MOORE MACHINE



Remove the starting o/p but if there is any edge then 1/0/o/p

Now,



GOOD WRITE

* Moore to Mealy Conversion:

1.) Input: Moore Machine
2.) Output: Mealy Machine
3.) Step-I: Take a blank Mealy machine transition table format.
4.) Step-II: Copy all the Moore machine transition states into this table format.
5.) Step-III: Check the present states and their corresponding o/ps in the Moore Machine.
State table, if for state Q_i o/p is m, copy it into the o/p columns of the mealy machine state table wherever Q_i appears in the next state.

Ex:

MOORE TO MEALY

Present State	Output	Next State	
		After i/p a=0	After i/p b=1
→ q_0	1	q_3	q_1
q_1	0	q_0	q_3
q_2	0	q_2	q_2
q_3	1	q_1	q_0

Mealy Machine → Transition Table

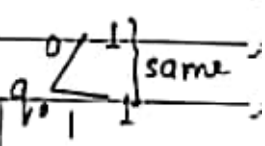
Present State	For a = 0 NEXT STATE		For b=1	
	State	Output	State	Output
→ q_0	q_3	1	q_1	0
q_1	q_0	1	q_3	1
q_2	q_2	0	q_2	0
q_3	q_1	0	q_0	1

GOOD WRITE

* Mealy Machine to Moore Machine:

1. Input: Mealy Machine
2. Output: Moore Machine
3. Step-I: Take a blank format Moore Machine transition table format.
4. Step-II: Calculate the number of different o/ps for each state (Q_i) that are available in the state table of the Mealy Machine.
5. Step-III: If all the o/ps of Q_i are same, copy state Q_i. If it has 'n' distinct o/ps, break Q_i into n states as Q_i^n, where n = 0, 1, 2, ...
6. Step-IV: If the o/p of the initial state is 1, insert a new initial state at the beginning which gives 0 o/p.

Ex: Let us consider a Mealy Machine.



Present State	For 0		For 1	
	N.S	O/p	N.S	O/p
→ q_0	q_3	0	q_1	1
q_1	q_0	1	q_3	0
q_2	q_2	1	q_2	0
q_3	q_1	0	q_0	1

PRESENT STATE	FOR 0		FOR 1	
	N.S	O/p	N.S	O/p
→ q_0	q_3	0	q_1	1
q_1	q_0	1	q_3	0
q_{1_0}	q_0	1	q_3	0
q_{1_1}	q_2	1	q_2	0
q_2	q_2	1	q_2	0
q_3	q_0	0	q_0	1

GOOD WRITE

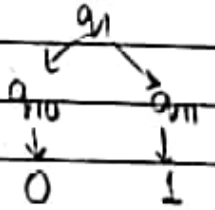
* MOORE MACHINE:

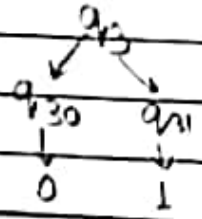
Present State	O/p	For '0' NS	For '1' NS
$\rightarrow q_0$	1	q_3	q_1
q_1	0	q_0	q_3
q_2	1	q_0	q_3
q_3	0	q_2	q_2
	1	q_2	q_2
	0	q_0	q_0

→ Mealy Machine:

Q:

Present State	For '0' NS	For '0' O/p	For '1' NS	For '1' O/p
$\rightarrow q_0$	q_1	0	q_3	0
q_1	q_3	1	q_2	0
q_2	q_4	1	q_0	0
q_3	q_0	0	q_4	1
q_4	q_2	0	q_1	1





Present State	For '0' NS	For '1' NS	O/p
$\rightarrow q_0$	q_{10} | 0	q_{30} | 0	0
$\rightarrow q_{10}$	q_3 | 1	q_2 | 0	0
$\rightarrow q_{11}$	q_3 | 1	q_2 | 0	1
$\rightarrow q_2$	q_4 | 1	q_0 | 0	0
$\rightarrow q_{30}$	q_0 | 0	q_4 | 1	0
0000 WRITE	q_0 | 0	q_1 | 1	1
$\rightarrow q_4$	q_2 | 0	q_{11} | 1	1

UNIT

G → Grammar

DATE 29/03/18
PAGE

CONTEXT-FREE GRAMMAR

To generate a language with the help of some grammer rules.

→ 4 Tuple (or) a quadruples, NTPS

Defn: A Context-free grammer (CFG) consisting a finite set of grammer rules is a quadruple.

$G = (N, T, P, S)$, where

* N: is a finite set of non-terminal symbols (usually represented by capital letters A, B, S ...)
* T: is a finite set of terminal symbols (usually represented by small letters (a, b, 0, 1, ... ε, etc))
* P: is a set of rules or productions in CFG.

$$P: N \rightarrow (N \cup T)^*$$

* S: Starting non-terminal symbol, called start symbol. ($S \in N$) of the grammer.

$N \cap T = NULL$

Ex: The Grammer $(\{A\}, \{a, b, c\}, P, A)$

$P: A \rightarrow aA, \quad A \rightarrow abc$

$N = \{A\}$

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

$S = \{A\}$

but when we include another cond^n, $a \rightarrow Ba$

it doesn't satify the above mentioned cond^n

Not a CFG.

Ex: The Grammar $(\{S, F\}, \{0, 1\}, P, S)$

$P: S \to 00S \mid 11F$, $F \to 00F \mid \varepsilon$

Sol: $N = \{S, F\}$

$T = \{0, 1\}$

$S = \{S\}$

There are 4 productions such as

$P: S \to 00S$

$P: S \to 11F$

$P: F \to 00F$

$P: F \to \varepsilon$

Hence, it is CFG.

* Language of a CFG:

If $G = (N, T, P, S)$ is a CFG, the language of 'G' denoted by $L(G)$ is the set of terminal strings that have derivations from the start symbol.

$$L(G) = \{ \omega \in T \mid S \overset{*}{\underset{G}{\Rightarrow}} \omega \}$$

If a language L is a language of some context-free grammar, then L is said to be a context-free language (CFL).

Q: The Grammar $(\{S\}, \{a, b\}, P, S)$

$P: S \to aSb, S \to ab$

Sol: $N = \{S\}$

$T = \{a, b\}$

GOOD WRITE

DATE : __/__/____
PAGE _____

We try to generate a string 'ab'. $S \Rightarrow ab$ (we will use it $(n-1)$ times)

$a^n b^n$

$S \Rightarrow a\textcircled{S}b$ — ①

$S \Rightarrow aabb$ — ② $(S \to asb)$

$S \Rightarrow a^2b^2$ $(S \to ab)$

Now,

$S \Rightarrow aaSbb$ — ① $(S \to asb)$

$S \Rightarrow aaabbb$ $(S \to ab)$

$S \Rightarrow a^3b^3$

⋮

$S \Rightarrow a^{n-1} S b^{n-1}$ $(S \to asb)$

$S \Rightarrow a^{n-1} ab\, b^{n-1}$ $(S \to ab)$

$S \Rightarrow a^n b^n$

$L = \{a^n b^n \mid n \geq 1\}$. This is the obtained CFL.

Q: $G = (\{A\}, \{a, b, c\}, P, A)$

$P: A \to aA$

$A \to abc$

$S \Rightarrow \omega$

Derivation,

$A \Rightarrow aA$

$A \Rightarrow aabc$

$A \Rightarrow a^2bc$

Now,

$A \Rightarrow aabc$

$A \Rightarrow aA$ $(A \to aA)$

$A \Rightarrow aaA$ $(A \to aA)$

$A \Rightarrow aaaA$ $(A \to aA)$

GOOD WRITE

$\Rightarrow a^{n-1} A$

$\Rightarrow a^{n-1} abc$

$\Rightarrow a^n abc$

Now,

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

Q: $G = (\{S, F\}, \{0, 1\}, P, S)$

P: $S \rightarrow 00S \mid 11F$, $F \rightarrow 00F \mid \varepsilon$

Ans

$S \rightarrow 00S$

$S \rightarrow 11F \quad (F \rightarrow \varepsilon)$

$S \rightarrow 11\varepsilon$

$S \rightarrow 11$

$S \Rightarrow \omega$

G $S \rightarrow 11F \quad (F \rightarrow 00F)$

$S \rightarrow 1100F \quad (F \rightarrow \varepsilon)$

$S \rightarrow 1100\varepsilon$

$S \rightarrow 11(00)^{n-1} F \quad (F \rightarrow \varepsilon)$

$S \rightarrow 11(00)^{n-1}$

$S \rightarrow 00(11)^{n-1}\varepsilon$

$S \rightarrow 00S$

$S \rightarrow 0000S \quad (S \rightarrow 00S)$

$S \rightarrow 000011F \quad (S \rightarrow 11F)$

$S \rightarrow 00001100\varepsilon \quad (F \rightarrow \varepsilon)$

$S \rightarrow 000011$

$S \rightarrow 0011(00)^{n-1}$

GOOD WRITE

* Derivation Tree:

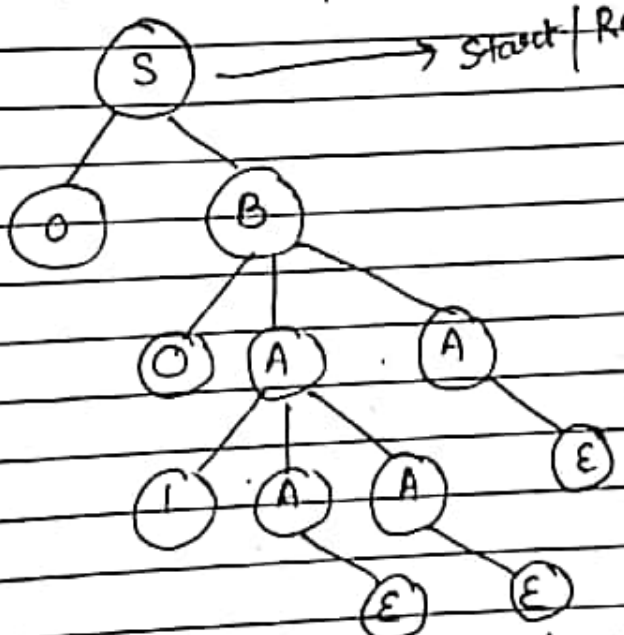
- Root vertex : S → start symbol
- Vertex → Non-Terminal symbols {A, B, S, F, ...}
- Leaves → Terminal symbols (a, b, a, 1, ..., ε)

Ex: $G = (N, T, P, S)$

$N = \{A, B, S\}$

$T = \{0, 1\}$

Ex: P: S → OB , A → 1AA | ε , B → 0AA

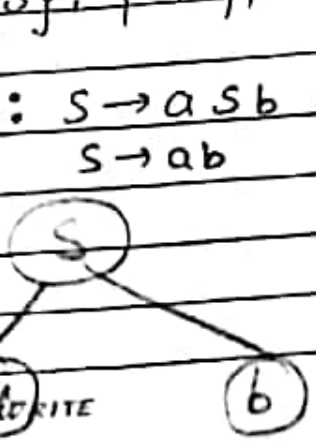


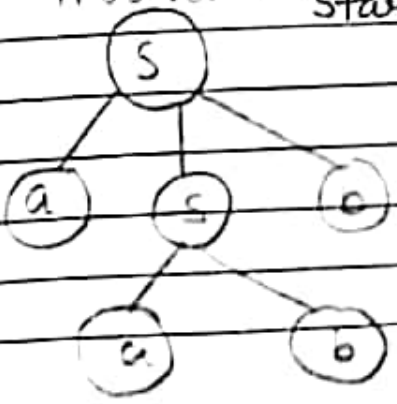
We do this as per the rule not allows to place non-terminal symbols.

Ex: $G = (\{S\}, \{a, b\}, P, S)$ if their is a starting symbol is involved then 'Start'/Root

P: S → aSb

S → ab



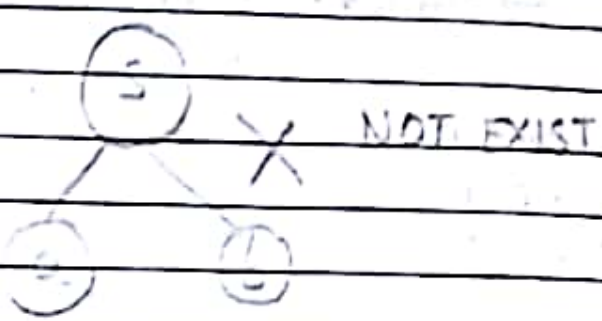


Otherwise,

$S \rightarrow aAb$

$A \rightarrow ab$

then



Q: $G = (N, T, P, S)$

P: $S \rightarrow a$

$S \rightarrow aAS$

$A \rightarrow bS$

Obtain derivation tree for the string word,

$\omega = abaabaa$

Ans

$S \Rightarrow aAS$ $(S \rightarrow aAS)$

$S \Rightarrow abSS$ $(A \rightarrow bS)$

$S \Rightarrow abaASS$ $(S \rightarrow aAS)$

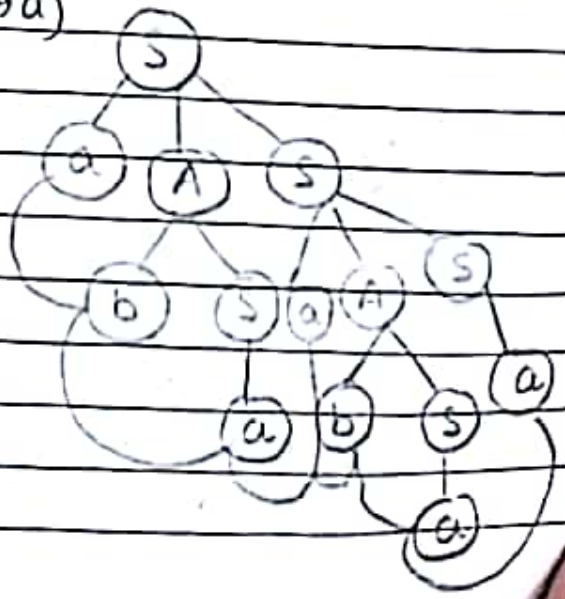
$S \Rightarrow abaS$ $(S \rightarrow aAS)$

$S \Rightarrow abaaAS$ $(A \rightarrow bS)$

$S \Rightarrow abaabSS$ $(S \rightarrow a)$

$S \Rightarrow abaabaS$ $(S \rightarrow a)$

$S \Rightarrow abaabaa$



GOOD WRITE

DATE 27 03 18
PAGE

Types of Derivation Tree

Left Derivation Tree

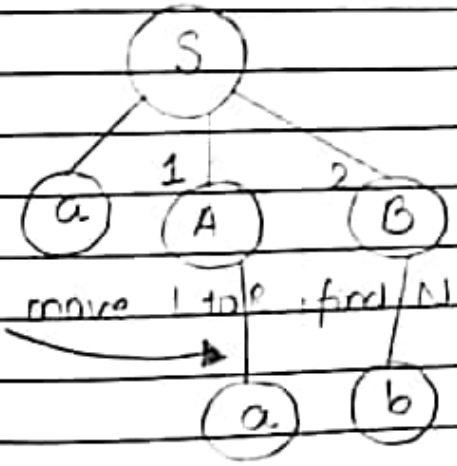
Obtained by applying productions to the leftmost variable in each step.

Right Derivation Tree

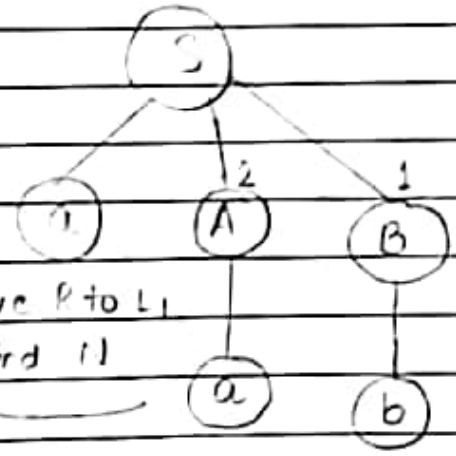
Obtained by applying productions to the rightmost variable in each step.

Q: P: $S \rightarrow aAB$, $A \rightarrow a$, $B \rightarrow b$

LDT



RDT



Eg: $S \rightarrow aAS \mid aSS \mid \varepsilon$ $A \rightarrow SbA \mid ba$

Obtain LDT and RDT for the string $\omega = aabaaa$.

$S \Rightarrow aSS$ $(S \rightarrow aAS)$

$\Rightarrow aaASS$ $(A \rightarrow ba)$

$\Rightarrow aabaSSS$ $(S \rightarrow aSS)$

$\Rightarrow aabaaSSS$ $(S \rightarrow \varepsilon)$

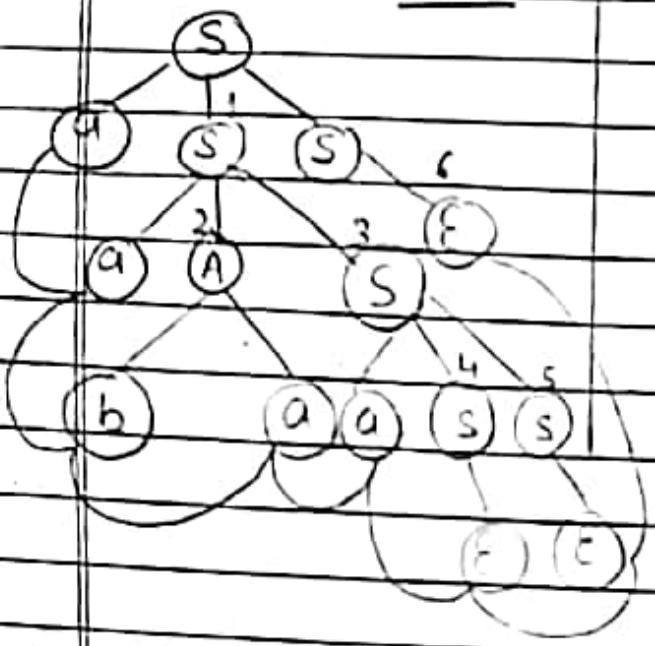
$\Rightarrow aabaaESS$ $(S \rightarrow \varepsilon)$

$\Rightarrow aabaaEES$ $(S \rightarrow \varepsilon)$

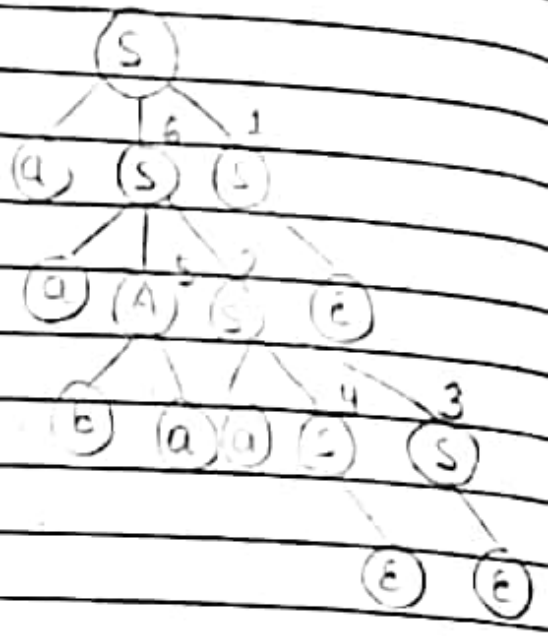
$\Rightarrow aabaaEEE$

GOOD WRITE $\rightarrow aabaa$

LDT



RDT



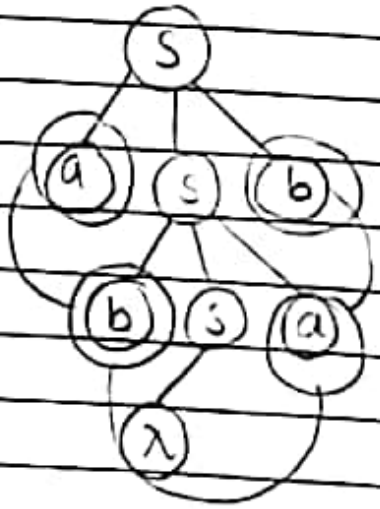
* Ambiguous Grammar:

A grammar is said to be ambiguous if there exists two or more derivation tree for a string 'w', otherwise non-ambiguous.

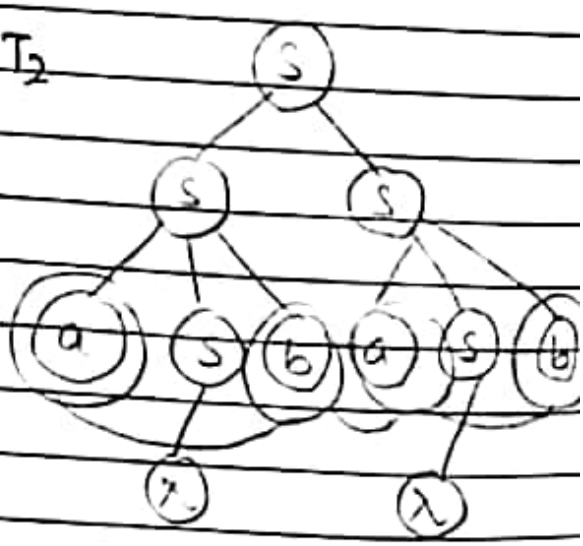
Ex: $G=(\{S\}, \{a,b\}, (S \to aSb \mid bSa \mid SS \mid \lambda), S)$

$w =$ "abab"

T_1



, T_2



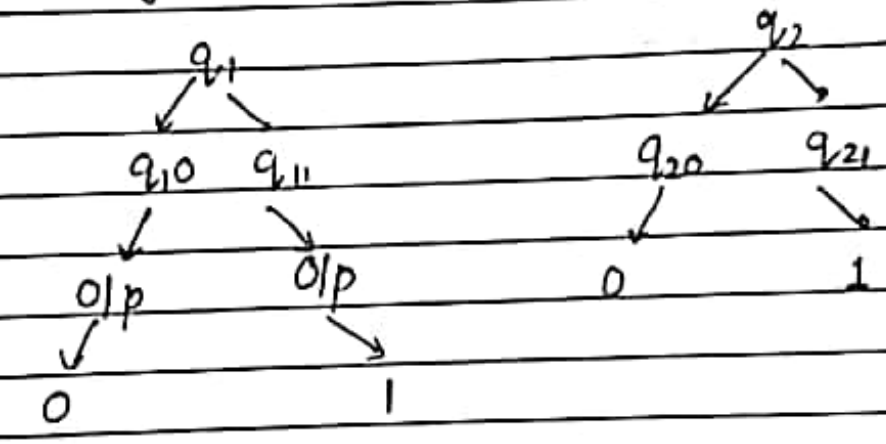
for string "abab" there are more than one distinct derivation tree exists. So, the given grammer is Ambiguous.

GOOD WRITE

* Conversion of Mealy to Moore:

Present State	For 'a'		For 'b'	
	State	o/p	State	o/p
→ q_0	q_3	0	q_1	1
q_1	q_0	1	q_3	0
q_2	q_2	1	q_2	0
q_3	q_1	0	q_0	1

q_0 & q_3 are giving same o/p



Moore Machine:

Present State	For 'a' Next state	For 'b' Next state	o/p
→ q_0	$q_3/0^-$	$q_{11}/1^*$	1
q_{10}	$q_0/1^-$	$q_3/0^-$	0
q_{11}	$q_0/1^-$	$q_3/0$	1
q_{20}	$q_{21}/1^-$	$q_{20}/0^*$	0
q_{21}	$q_2/1^-$	$q_{20}/0^-$	1
q_3	$q_{10}/0^-$	$q_0/1^-$	0

Imp

* Chomsky Normal form: CNF

A CFG in which productions are of the form:

- $A \rightarrow BC$
  OR Exactly two non-terminals
- $A \rightarrow a$
  Single terminal symbol

Ex: $S \rightarrow 1A \mid 0B$
$A \rightarrow 1AA \mid 0S \mid 0$
$B \rightarrow 0BB \mid 1$
in each of the following which is in CNF form

Q2 $A \rightarrow 0$ is in CNF form (A/c to the production form)
$B \rightarrow 1$ is in CNF form

Rewrite:

$S \rightarrow 1A$
$S \rightarrow C_1A$ } CNF
$C_1 \rightarrow 1$

$S \rightarrow 0B$
$S \rightarrow C_0B$ } CNF
$C_0 \rightarrow 0$

$A \rightarrow 1AA$
$A \rightarrow C_1C_A$ } CNF
$C_1 \rightarrow 1$
$C_A \rightarrow AA$

$A \rightarrow 0S$
$A \rightarrow C_0S$ } CNF
$C_0 \rightarrow 0$

$B \rightarrow 0BB$
$B \rightarrow C_0C_b$ } CNF
$C_0 \rightarrow 0$
$C_b \rightarrow BB$

DATE __/__/____
PAGE _____

Simplification:

$S \to C_1A \mid C_0B$

$A \to C_1C_a \mid C_0S \mid 0 \Rightarrow A \to C_1C_a \mid C_0S \mid 0$

$B \to C_0C_b \mid 1 \Rightarrow B \to C_0C_b \mid 1$

Ex: $L = \{a^n \mid n >= 1\}$, Find CFG and convert it into CNF.

$L = \{a, aa, aaa, aaaa, \ldots\}$

Ex: $L = \{a^{4n} \mid n >= 1\}$, Find CFG and convert it into CNF

Sol: $n = 1$, $L = \{aaaa\}$

$n = 2$, $L = \{aaaa\,aaaa\}$

$L = \{aaaa, \ldots\}$

$N = \{S\}$, $T = \{a\}$

P: $N \to (NUT)^*$

p: $S \to aaaaS$

$S \to aaaa$

* Rewrite:

CNF form:

$S \to AR_1$

$R_1 \to AR_2$

$R_2 \to AR_3$

$R_3 \to AS$

$A \to a$

CNF:

$S \to AR_4$

$R_4 \to AR_5$

$R_5 \to AA$

$A \to a$

GOOD WRITE

DATE: __/__/____
PAGE _____

Q: Convert into CNF:

$S \to bA \mid aB$

$S \to bA$

$S \to C_bA$ } CNF

$C_b \to b$

$S \to aB$

$S \to C_aB$ } CNF

$C_a \to a$

$A \to bAA \mid as \mid a$

$A \to bAA$

$A \to C_bC_A$ } CNF

$C_b \to b$

$C_A \to AA$

$A \to aS$

$A \to C_aS$ } CNF

$C_a \to a$

$A \to a$ (CNF)

* $B \to aBB \mid bS \mid b$

$B \to aBB$

$B \to C_aC_B$ } CNF

$C_a \to a$

$C_B \to BB$

$B \to bS$

$B \to C_bS$ } CNF

$C_b \to b$

$B \to b$ (CNF)

* Greibach Normal form:

$A \to a\alpha \mid a$

α → string of none or more variable (NT)

↳ Exactly one terminal

Ex: $A \to aB$ Greibach NF (GNF)

GOOD WRITE

SPE DATE: __/__/____
PAGE _____

$A \rightarrow aBCA$ (GNF)

$A \rightarrow abAa$ (Not a GNF)

Q: $S \rightarrow AB$ (Not a GNF)

$A \rightarrow aA / bB / b$ (Yes, a GNF)

$B \rightarrow b$ (GNF)

$S \rightarrow AB$

$S \rightarrow aAB \Rightarrow S \rightarrow aAB$ } GNF

~~aBA~~ $A \rightarrow aA$

OR

$S \rightarrow AB$ }

$S \rightarrow bBB$ } GNF

$A \rightarrow bB$ }

OR

$S \rightarrow AB$

$A \rightarrow b$ } GNF

$S \rightarrow bB$ }