DFA (Deterministic Finite Automata)

In DFA on each input there is only one state which the automata can transition from its current state.

A DFA consist of 5-tuples $(Q, \Sigma, \delta, q_0, F)$

Q : Finite non-empty set of states

Σ : Finite non-empty set of input

δ : Transition Function $Q \times \Sigma \rightarrow Q$

eg: $\delta(q_0, a) \rightarrow q_1$

q_0 : Initial state

F : set of final states (also cld acceptance state)

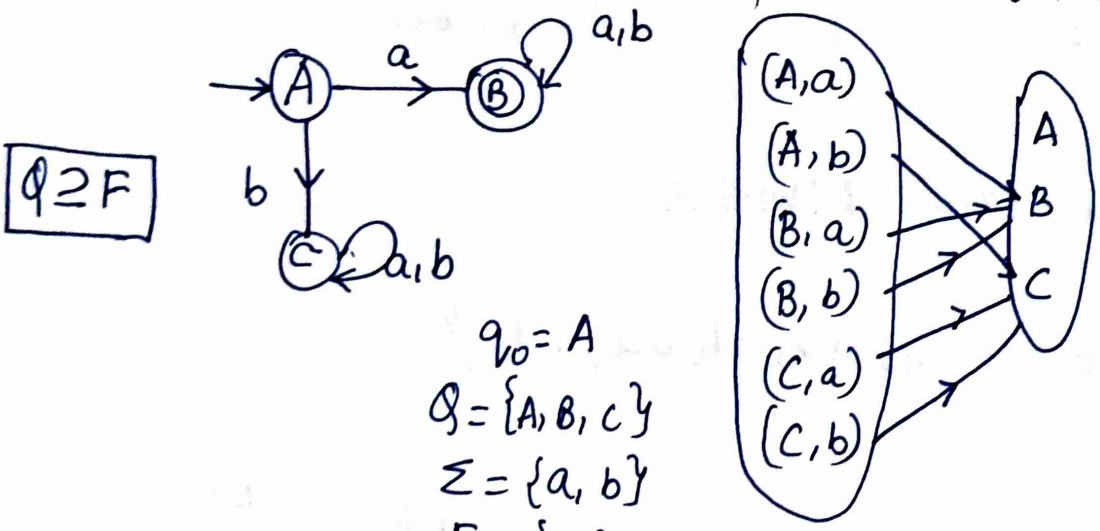
DFA (Deterministic Finite Automata)

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

- Q: Finite set of states
- Σ: input alphabet
- q_0: start state
- F: set of final states

$\delta : Q \times \Sigma \longrightarrow Q$

$\{A, B, C\} \times \{a, b\}$



$q_0 = A$

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

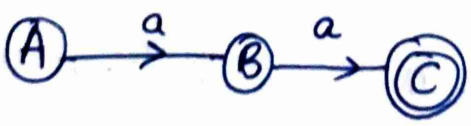
$\Sigma = \{a, b\}$

$F = \{B\}$

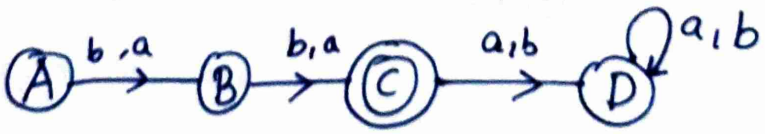
Q Construct DFA, that accepts set of all strings over $\{a, b\}$ of length 2.

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

$L = \{aa, ab, ba, bb\}$



[start constructing with minimum length input]



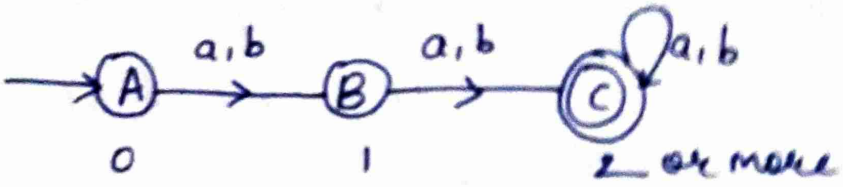
Ans.

String Accept: scan the entire string if we reach a final state from initial state.

Language Accept: A FA is said to accept a language if all the strings in the language are accepted and all the strings not in the language are rejected.

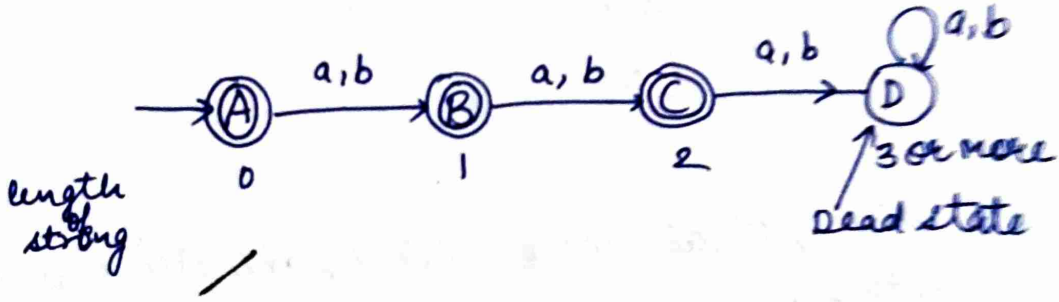
Q. construct DFA that accepts all string $w \in \{a,b\}$ & $|w| \geq 2$

Sol. $L = \{aa, ab, ba, bb, aaa, \ldots bbb, \ldots\ldots\}$



Q. DFA $w \in \{a,b\}$ & $|w| \leq 2$

Sol. $L = \{\epsilon, a, b, aa, ab, ba, bb\}$



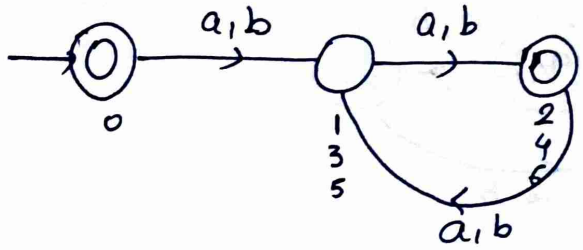
Language ⇒ many DFA's ← can have only 1 minimal DFA

Note :⇒	$\|w\| = n$	$\|w\| \geq n$	$\|w\| \leq n$
Total states	$(n+2)$	$(n+1)$	$(n+2)$

Q: Construct minimal DFA, $w \in \{a,b\}$ s.t. $|w| \bmod 2 = 0$

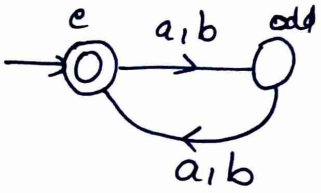
sol: $L = \{\epsilon, aa, ab, ba, bb, aaaa, \ldots\ldots, bbbb, \ldots\ldots\}$

✗



} ← This is DFA but not minimal

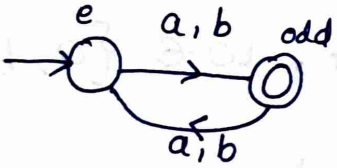
✓



} This is minimal DFA.

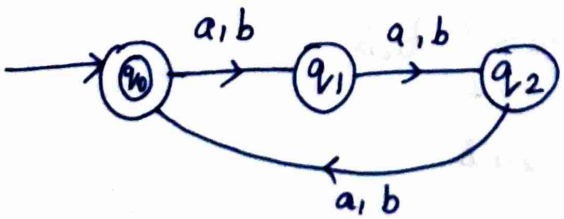
Q. Construct minimal DFA, $w \in \{a,b\}$ s.t. $|w| \bmod 2 = 1$

sol:



Q. Construct minimal DFA, $w \in \{a,b\}$ s.t. $|w| \bmod 3 = 0$

sol: $L = \{\epsilon,$ aaa, aab, bbb, ⋮ , aaaaaa, aaaaab, ⋮ , ⋯⋯⋯ $\}$



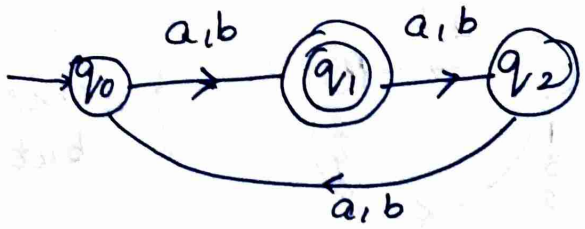
[we have taken 3 states bcoz when any no. is divided by 3 then remainder is either 0, 1 or 2.

Q. Construct DFA, $|w| \cong 1 \bmod 3$

or

$|w| \bmod 3 = 1$

Sol.



NOTE: $\boxed{|w| \bmod n = 0}$

"n"

3 ⇒ 3

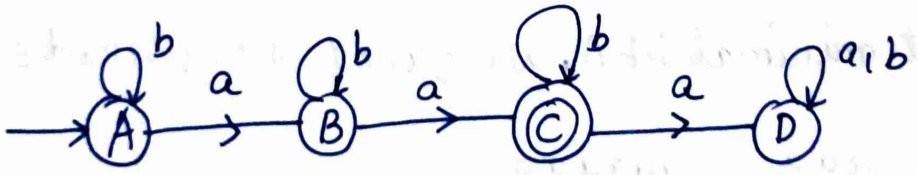
4 ⇒ 4

⋮

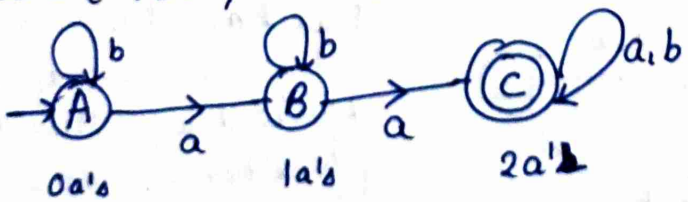
Divisible by n ⇒ n states

Q. Construct minimal DFA, $w \in \{a,b\}^*$, $n_a(w) \geq 2$

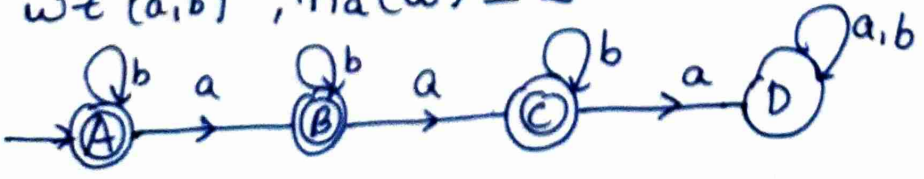
$L = \{aa, baa, aba, aab, bbaa, \ldots\ldots\}$



Q. $w \in \{a,b\}^*$, $n_a(w) \geq 2$

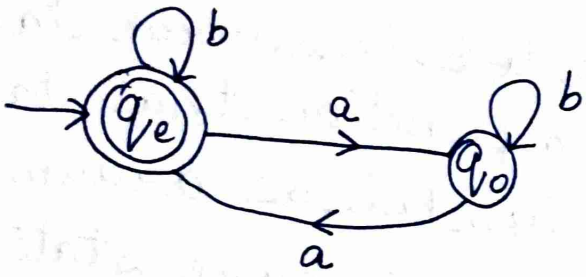


Q. $w \in \{a,b\}^*$, $n_a(w) \leq 2$



Q. Construct minimum DFA, $w \in \{a,b\}^*$
$n_a(w) \bmod 2 = 0$ or $n_a(w) \cong 0 \bmod 2$

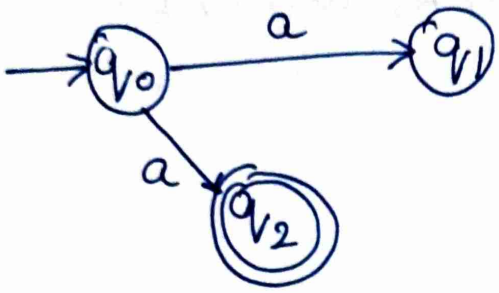
Sol.



Q. Construct minimum DFA, $w \in \{a,b\}^*$
$n_a(w) \bmod 3 =$

(NDFA / NFA) Non-Deterministic Finite Automata

A NFA has the power to be in several states at once for NFA 'δ' is a function that takes a state and input symbol as argument but return to zero, one or more state.



NFA consists of 5 tuples $(Q, \Sigma, \delta, q_0, F)$

Q : set of finite states

Σ : input alphabet

q_0 : start state

F : set of final states

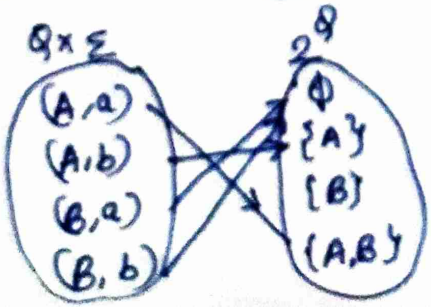
$\delta : Q \times \Sigma \rightarrow 2^Q$

Example

$L = \{$ ends with 'a' $\}$, $\Sigma = \{a, b\}$

$L = \{ a, aa, ba , aaa , aba , baa , bba , \ldots\ldots \}$

$Q = \{A, B\}$, $\Sigma = \{a, b\}$

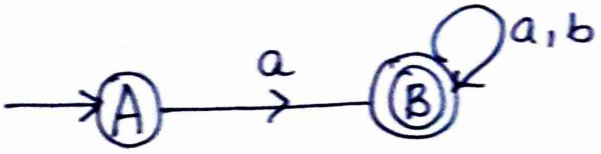


$\delta : Q \times \Sigma \rightarrow 2^Q$

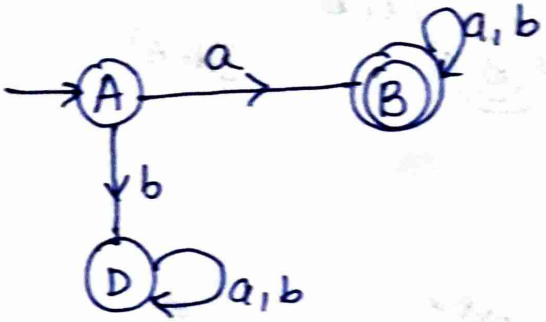
→ Every DFA is a NFA.

NFA Examples

Q.1 Construct NFA for language L_1 starts with 'a' !

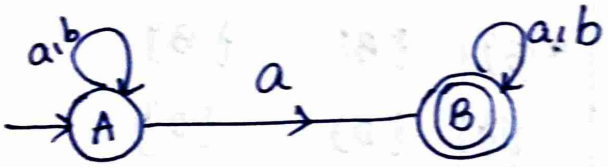


[NFA]

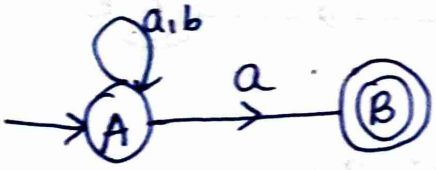


[DFA]

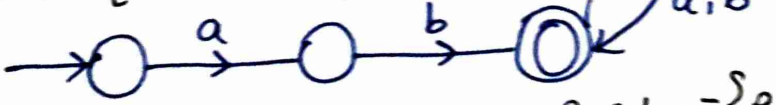
Q.2 L_2 = {contains 'a'}



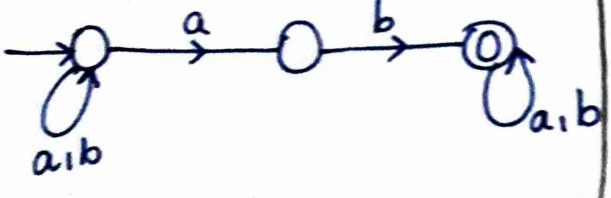
Q.3 L_3 = {ends with 'a'}



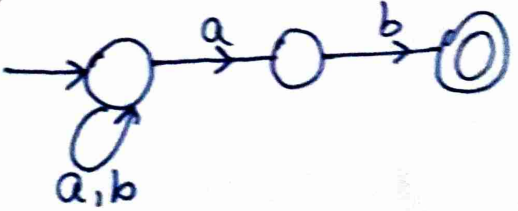
Q.4 L_4 = {starts with 'ab'}



Q.5 L_5 = {contains 'ab'}



Q.6 L_6 = {ends with 'ab'}



NFA → DFA [we need to convert NFA to DFA]

DFA → NFA [we know every DFA is an NFA]

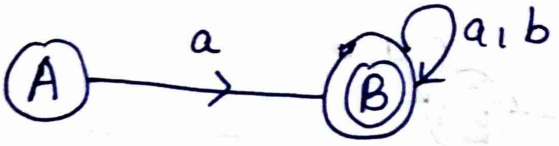
NFA ≅ DFA

Conversion of NFA to DFA

Q.1 $\Sigma = \{a, b\}$

$L_1 = \{$ starts with 'a' $\}$

NFA :

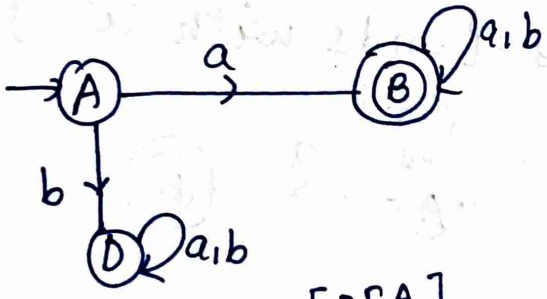


[NFA]

state Transition Table :-

NFA	a	b
→ {A}	{B}	{ }
* {B}	{B}	{B}

DFA	a	b
→ {A}	{B}	{D}
* {B}	{B}	{B}
{D}	{D}	{D}

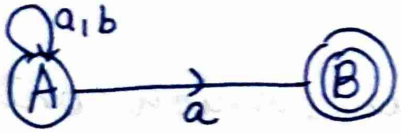


[DFA]

Q2 Conversion of NFA to DFA.

L_1 = { ends with 'a' }

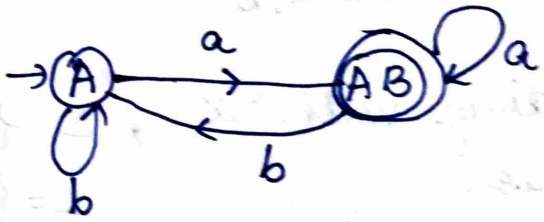
sol:



State Transition Table :-

	a	b
→ {A}	{A, B}	{A}
* {B}	{ }	{ }

	a	b
[A]	[A B]	[A]
[A B] *	[A B]	[A]



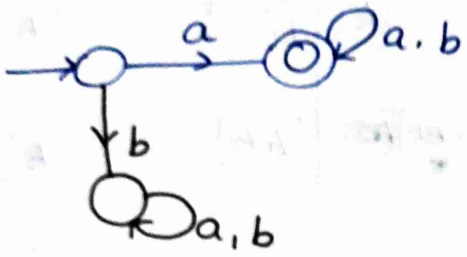
Tutorial Sheet – I

SOLUTIONS

Q.1 Design DFA for the language which contains all the string –

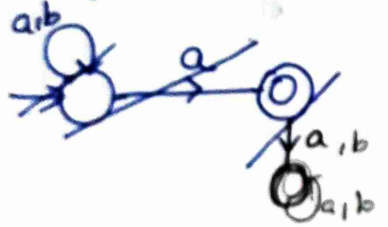
(i) starting with 'a' ; $\Sigma = \{a, b\}$

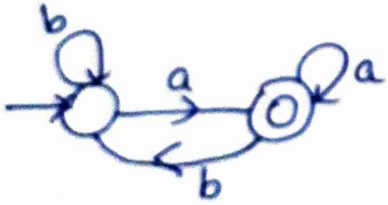
$L = \{a, aa, ab, aaa, aab, abb, \ldots\}$



(ii) ending with 'a' ; $\Sigma = \{a, b\}$

$L = \{a, aa, ba, aaa, aba, baa, bba, \ldots\}$

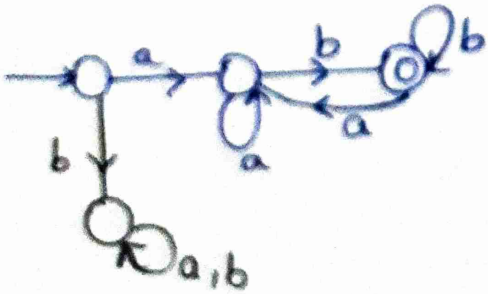




[DFA]

(iii) starting with 'a' and ending with 'b'

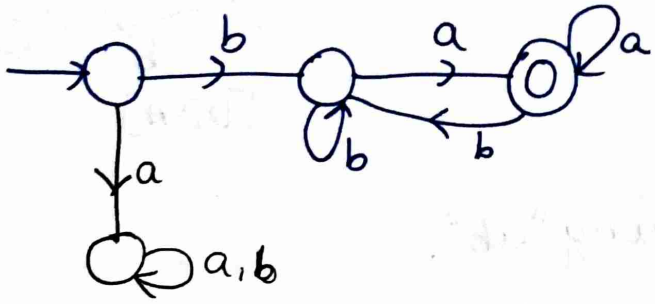
$L = \{ab, aab, abb, aaab, aabb, \ldots\}$



(iv) not starting with 'a' and not ending with 'b'.

$L = \{ba, b\_a, b\_\_a, b\_\_\_a, \ldots\ldots\}$

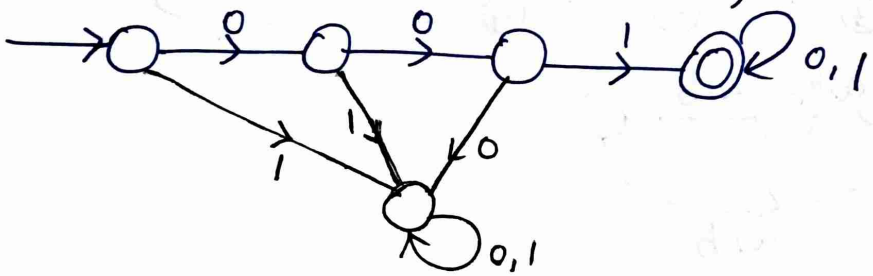
$= \{ba, baa, bba, baaa, baba, bbba, bbaa, \ldots\ldots\}$



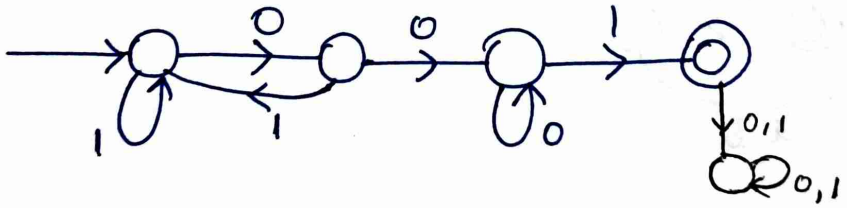
(v) starting with '001'.

$L = \{001, 001\_, 001\_\_, 001\_\_\_, \ldots\ldots\}$

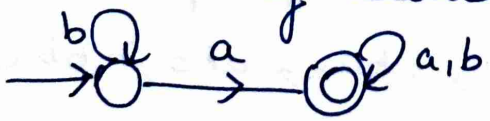
$= \{001, 0010, 0011, 00101, 00110, \ldots\ldots\}$



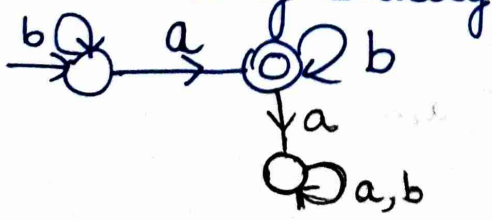
(vi) ending with '001'.



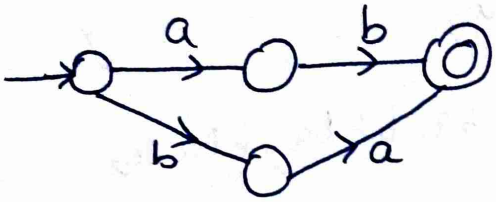
(vii) containing atleast one 'a'.



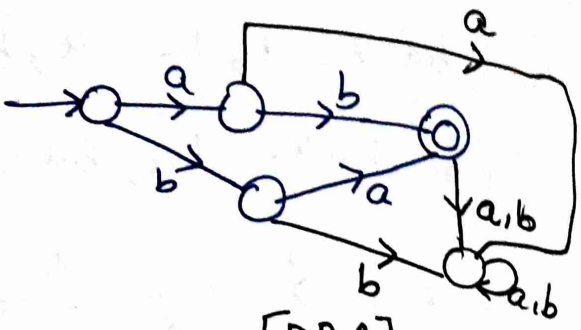
(viii) containing exactly one 'a'.



(ix) $L = \{ab, ba\}$

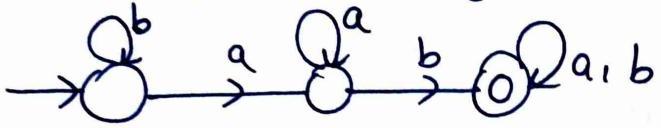


[FA]



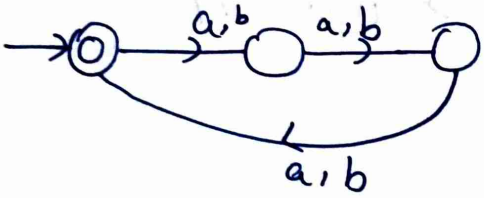
[DFA]

(x) containing substring "ab".



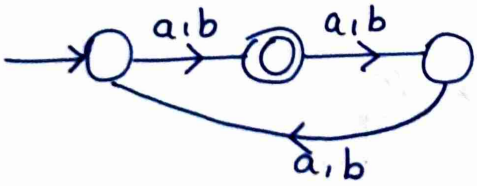
(xi) $w \in \{a,b\}$, $|w| \bmod 3 = 0$, where $|w|$ is length of string.

$L = \{\epsilon, \underset{3}{---}, \underset{6}{------}, \underset{9}{---------}, \ldots\}$



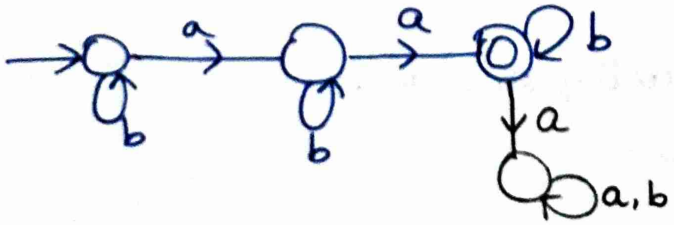
(xii) $|w| \bmod 3 = 1$

$L = \{1, 4, 7, 10, \ldots\}$



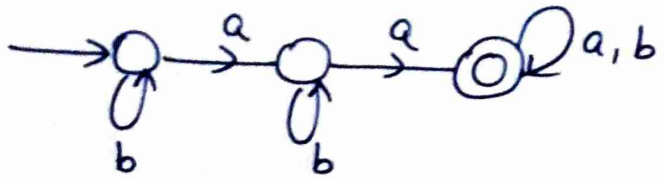
(xiii) $w \in \{a,b\}$, s.t. $n_a(w) = 2$, where n_a means total number of 'a' occuring in the string.

$L = \{aa, baa, aab, aba, bbaa, aabb, abba, \ldots\}$



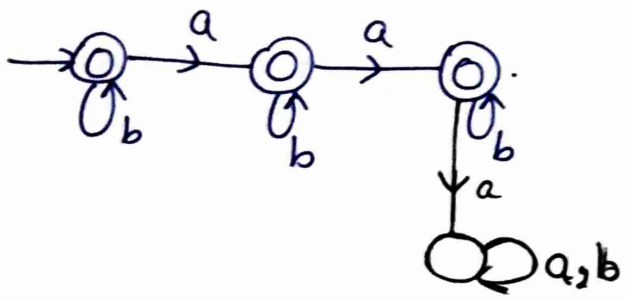
(xiv) $w \in \{a,b\}$, s.t. $\eta_a(w) >= 2$

$L = \{aa, baa, aba, aab, aaa, baaa, \ldots\ldots\}$



(xv) $w \in \{a,b\}$, such that $\eta_a(w) <= 2$

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



Minimization of Automata

Q.

	0	1
→ q_0	q_1	q_3
q_1	q_6	q_2
(final) q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Sol. $\pi_0 = \underbrace{\{q_2\}}_{\text{Final}\ x} \quad \underbrace{\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}}_{\text{Non-Final states}\ y}$

$\pi_1 = \underbrace{\{q_2\}}_{a} \underbrace{\{q_0, q_4, q_6\}}_{b} \underbrace{\{q_1, q_7\}}_{c} \underbrace{\{q_3, q_5\}}_{d}$

$\pi_2 = \{q_2\}\{q_0, q_4\}\{q_6\}\{q_1, q_7\}\{q_3, q_5\}$

$\pi_3 = \{q_2\}\{q_0, q_4\}\{q_6\}\{q_1, q_7\}\{q_3, q_5\}$

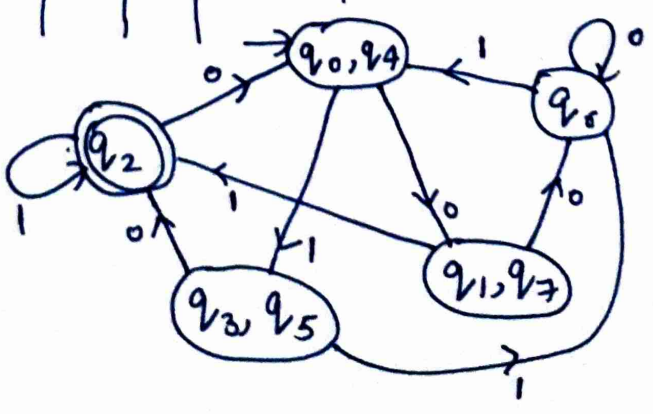
$\boxed{\pi_2 = \pi_3}$

	0	1
q_0	y	y
q_1	y	x
q_3	x	y
q_4	y	y
q_5	x	y
q_6	y	y
q_7	y	x

	0	1
q_0	c	d
q_4	c	d
q_6	b	b

	0	1
q_1	b	a
q_7	b	a

	0	1
q_3	a	b
q_5	a	b



Q.

	0	1
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
(q_3)	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

$\pi_0 = \{q_3\}_x \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}_y$

$\pi_1 = \{q_3\}_a \{q_0, q_1, q_5, q_6\}_b \{q_2, q_4\}_c \{q_7\}_d$

$\pi_2 = \{q_3\} \{q_0, q_6\} \{q_1, q_5\} \{q_2, q_4\} \{q_7\}$

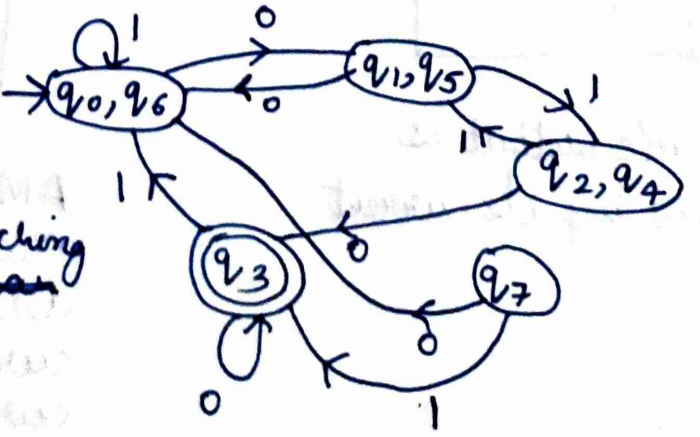
$\pi_3 = \{q_3\} \{q_0, q_6\} \{q_1, q_5\} \{q_2, q_4\} \{q_7\}$

	0	1
q_0	y	y
q_1	y	y
q_2	x	y
q_4	x	y
q_5	y	y
q_6	y	y
q_7	y	x

$\pi_2 = \pi_3$

	0	1
q_0	b	b
q_1	b	c
q_5	b	c
q_6	b	b

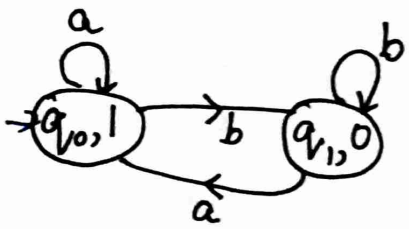
	0	1
q_2	a	b
q_4	a	b



In fig. q_7 is the dead state as it is not reaching the final state ∴ it ~~can~~ should be removed.

Finite Automata with Output

MOORE Machine



$\lambda : Q \rightarrow \Delta$

$q_0 \rightarrow 1$, $q_1 \rightarrow 0$

I/p : ab

$q_0 \rightarrow q_0 \rightarrow q_1$

1 1 0

'n' input (n+1) o/p

6-Tuples

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

Q - Finite set of states

Σ - Input alphabet

δ : Transition Function

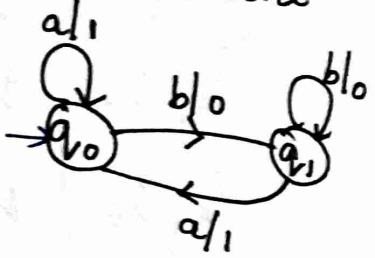
$Q \times \Sigma \rightarrow Q$

q_0 - Initial state

Δ - O/P Alphabet

λ - Output Function

MEALY Machine



$\lambda : Q \times \Sigma \rightarrow \Delta$

$(q_0, a) \rightarrow 1$

$(q_0, b) \rightarrow 0$

$(q_1, a) \rightarrow 1$

$(q_1, b) \rightarrow 0$

I/p : 'ab'

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

1 0

'n' input 'n' output

Moore M/c

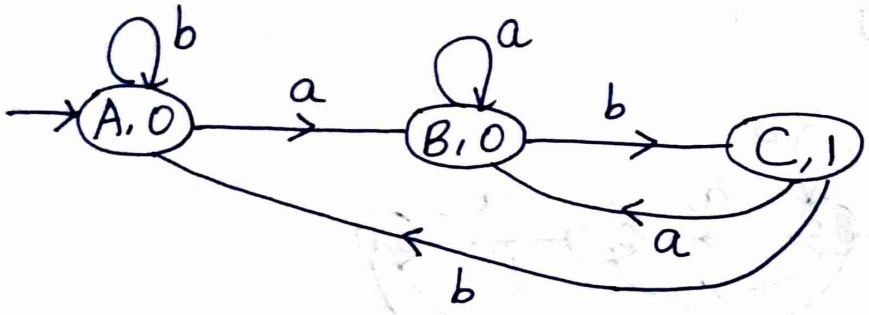
State Transition Table

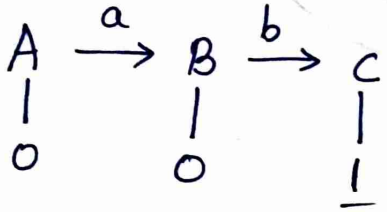
state	Input	Output

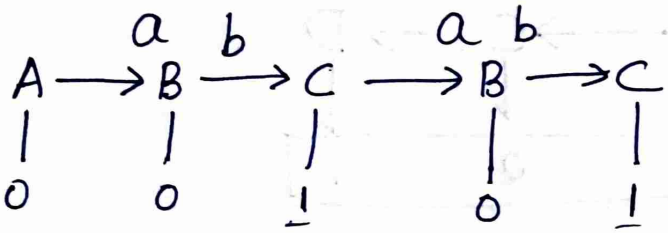
MEALY M/c

Q:- Construct a Moore Machine that takes set of all strings over {a, b} as i/p and prints '1' as o/p for every occurrence of 'ab' as a substring.

sol: $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$





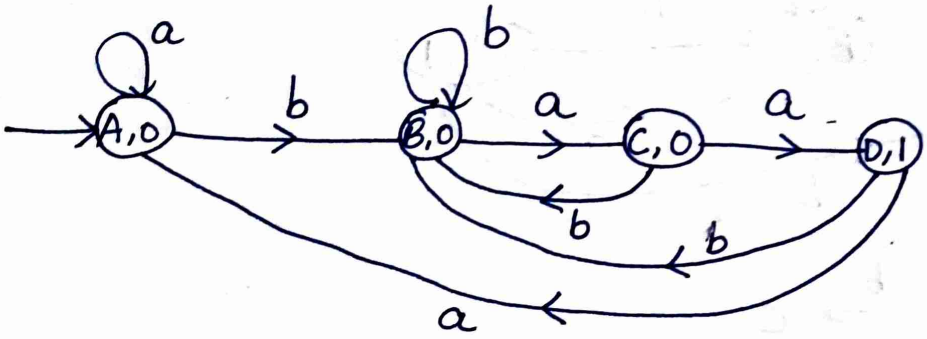


Q. Construct a moore m/c that takes set of strings over {a, b} and counts no. of occurrences of substrings 'baa'.

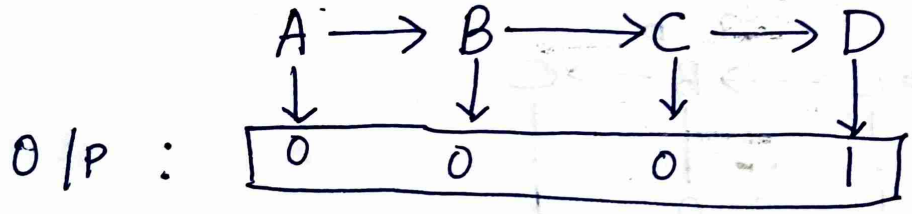
Sol:

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

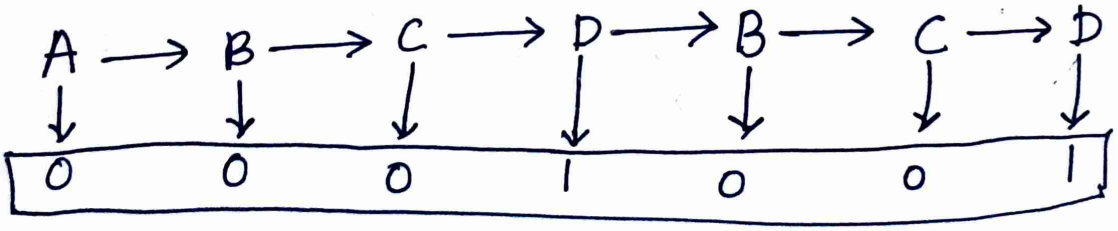
$\Delta = \{0, 1\}$



I/P : baa

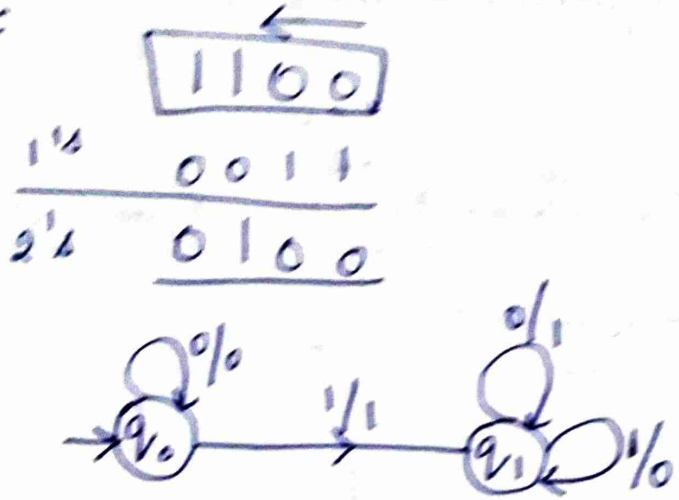


I/P : baabaa



Q. Construct a mealy m/c that takes binary number as input and produce 2's complement of that number as o/p. Assume the string is read LSB to MSB and end carry is discarded.

Sol:



0 0 11

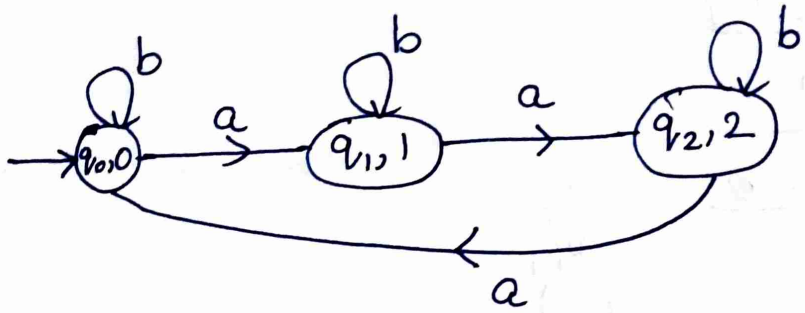
$q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_1$

0 0 1 0

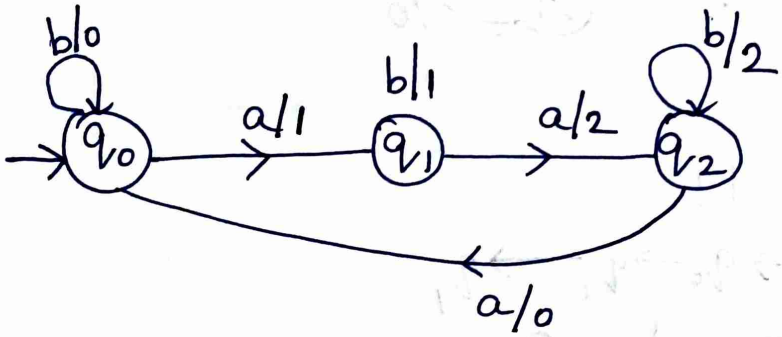
Conversion of MOORE m/c to MEALY m/c

Q. Construct a Moore M/c that count no. of a's % 3 [i.e. it prints the remainder when a is divided by 3] and convert it into its equivalent Mealy M/c.

Sol:



[MOORE M/C]



[MEALY M/C]

State Transition Table for MOORE M/C & MEALY M/C

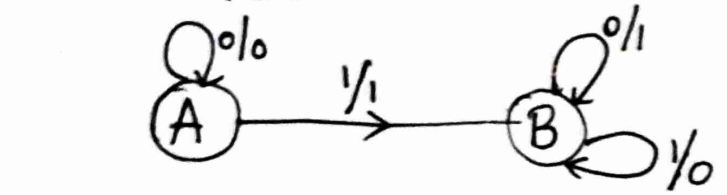
MOORE State	a	b	Δ
→ q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

MEALY State	a	b
→ q_0	$(q_1, 1)$	$(q_0, 0)$
q_1	$(q_2, 2)$	$(q_1, 1)$
q_2	$(q_0, 0)$	$(q_2, 2)$

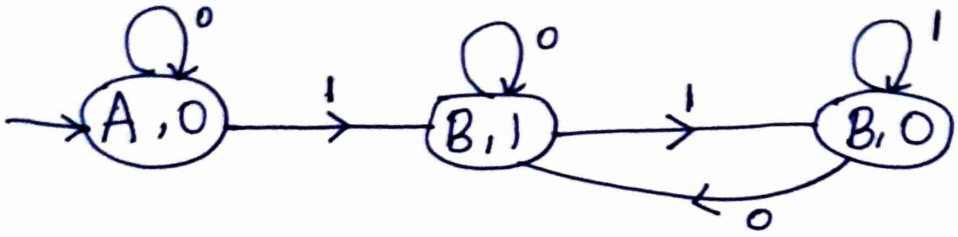
Conversion of Mealy M/c to Moore M/c

Q Convert the given Mealy M/c into its equivalent Moore M/c.

Sol:



Sol:



Note: Moore ⟶ Mealy
[no. of states doesn't change]

Mealy ⟶ Moore
[no. of state increase]

N × M ⟶ (N × M) states { where N is the i/p & M is the total o/p }

Algebraic Laws for Regular Expression

Let L, M, N are RE.

1. Associativity and Commutativity

$L+M = M+L$ → Commutative

$(L+M)+N = L+(M+N)$ → Associative

$(LM)N = L(MN)$ → Associative

2. Identities and Annihilators

$\phi + L = L + \phi = L$ ← Identities

$\epsilon L = L\epsilon = L$ ← Identities

$\phi L = L\phi = \phi$ ← Annihilator

3. Distributive Law

$L(M+N) = LM + LN$, and

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

4. Idempotent Law

$L+L = L$

5. Laws involving closure (*)

(i) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

(ii) $R^* R^* = R^*$

(iii) $R R^* = R^* R$ (iv) $(R^*)^* = R^*$

(v) $\epsilon + R R^* = R^* = \epsilon + R^* R$

(vi) $(PQ)^* P = P(QP)^*$

(vii) $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

Q. Prove $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

Sol. L.H.S. $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$\Rightarrow (1+00^*1)(\wedge+(0+10^*1)^*(0+10^*1))$

$\Rightarrow (1+00^*1)(\wedge+(0+10^*1)^+)$

$\Rightarrow (1+00^*1)(0+10^*1)^*$

$\Rightarrow 1(\wedge+00^*)(0+10^*1)^*$

$\Rightarrow 1(\wedge+0^+)(0+10^*1)^*$

$\Rightarrow 10^*(0+10^*1)^*$

$\Rightarrow 0^*1(0+10^*1)^* =$ R.H.S.

Finite Automata to Regular Expression (Arden's Theorem & Arden's Algebraic Method)

ARDEN's Theorem :- Let P and Q be two regular expressions, if P does not contain ϵ, then following equation in $R = Q + RP$ has unique solution given by, $R = QP^*$

Proof:

$R = Q + RP$; $R = QP^*$

We assume that arden's theorem is true $\Rightarrow Q + RP$

$\Rightarrow Q + QP^*P$

$\Rightarrow Q(\wedge + P^*P)$

$\Rightarrow QP^*$.

$R = Q + RP$

$= Q + (Q + RP)P$

$= Q + (QP + RPP)$

$= Q + QP + RP^2 + \dots$

$\Rightarrow Q(\wedge + P + P^2 + P^3 + \dots + P^i) + RP^{(i+1)}$

$\Rightarrow Q(\wedge + P + P^2 + P^3 + \dots + P^i) + QP^*P^{i+1}$

$\Rightarrow QP^*$

$\Rightarrow R$

Arden's Algebraic Method –

V_1 is initial state

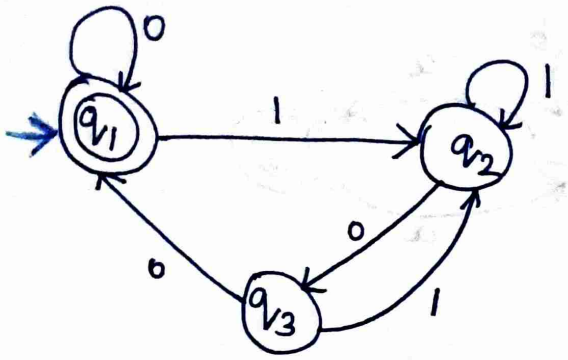
$\rightarrow V_1 = V_1\alpha_{11} + V_2\alpha_{21} + V_3\alpha_{31} + \dots + \epsilon$

$V_2 = V_1\alpha_{12} + V_2\alpha_{22} + V_3\alpha_{32} + \dots$

$\vdots$

$V_n = V_1\alpha_{1n} + V_2\alpha_{2n} + V_3\alpha_{3n} + \dots$

Q.



Sol:

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

$\Rightarrow$ $q_2 = q_1 1 + q_2 1 + q_2 01$

$$\underbrace{q_2}_{R} = \underbrace{q_1 1}_{Q} + \underbrace{q_2}_{R}\underbrace{(1+01)}_{P}$$

$$q_2 = q_1 1 (1+01)^*$$

$\Rightarrow$ $q_3 = q_2 0$

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

$\Rightarrow$ $q_1 = q_1 0 + q_3 0 + \epsilon$

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

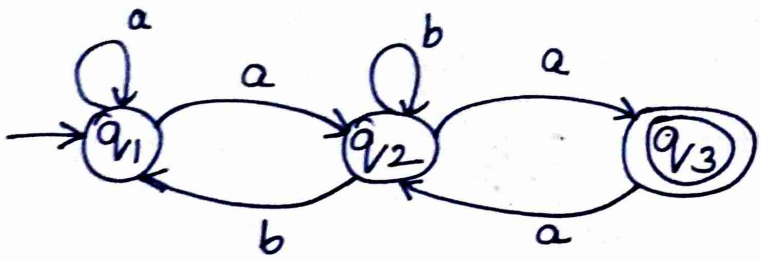
$= q_1 (0 + 1(1+01)^* 00) + \epsilon$

$$\underbrace{q_1}_{R} = \underbrace{\epsilon}_{Q} + \underbrace{q_1}_{R}\underbrace{(0+1(1+01)^*00)}_{P}$$

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

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

Q



sol.

$$q_1 = q_1a + q_2b + \epsilon$$

$$q_2 = q_1a + q_2b + q_3a$$

$$q_3 = q_2a$$

$$\Rightarrow \quad q_2 = q_1a + q_2b + q_2aa$$

$$\underbrace{q_2}_{R} = \underbrace{q_1}_{q}\underbrace{a}_{} + \underbrace{q_2}_{R}\underbrace{(b+aa)}_{P}$$

$$q_2 = q_1a(b+aa)^*$$

$$\Rightarrow \quad q_1 = q_1a + q_1a(b+aa)^*b + \epsilon$$

$$= q_1(a + a(b+aa)^*b) + \epsilon$$

$$\underbrace{q_1}_{R} = \underbrace{\epsilon}_{q} + \underbrace{q_1}_{R}\underbrace{(a+a(b+aa)^*b)}_{P}$$

$$q_1 = \epsilon(a+a(b+aa)^*b)^*$$

$$q_1 = (a+a(b+aa)^*b)^* \quad .$$

$$q_2 = (a+a(b+aa)^*b)^*a(b+aa)^* \quad .$$

$$q_3 = q_2a$$

$$= (a+a(b+aa)^*b)^*a(b+aa)^*a \quad .$$

Proof of Arden's Theorem

$R = QP^*$ (To prove unique solution)

$R = Q + RP$

$= Q + (Q + RP)P$

$= Q + QP + RP^2$

$= Q + QP + (Q + RP)P^2$

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

$= Q + QP + QP^2 + \ldots + QP^i + RP^{i+1}$

$\Rightarrow Q(\Lambda + P + P^2 + \ldots + P^i) + QP^*P^{i+1}$

$\Rightarrow Q(\Lambda + P + P^2 + \ldots P^i + P^*)$ $\{P^*P^{i+1} = P^*\}$

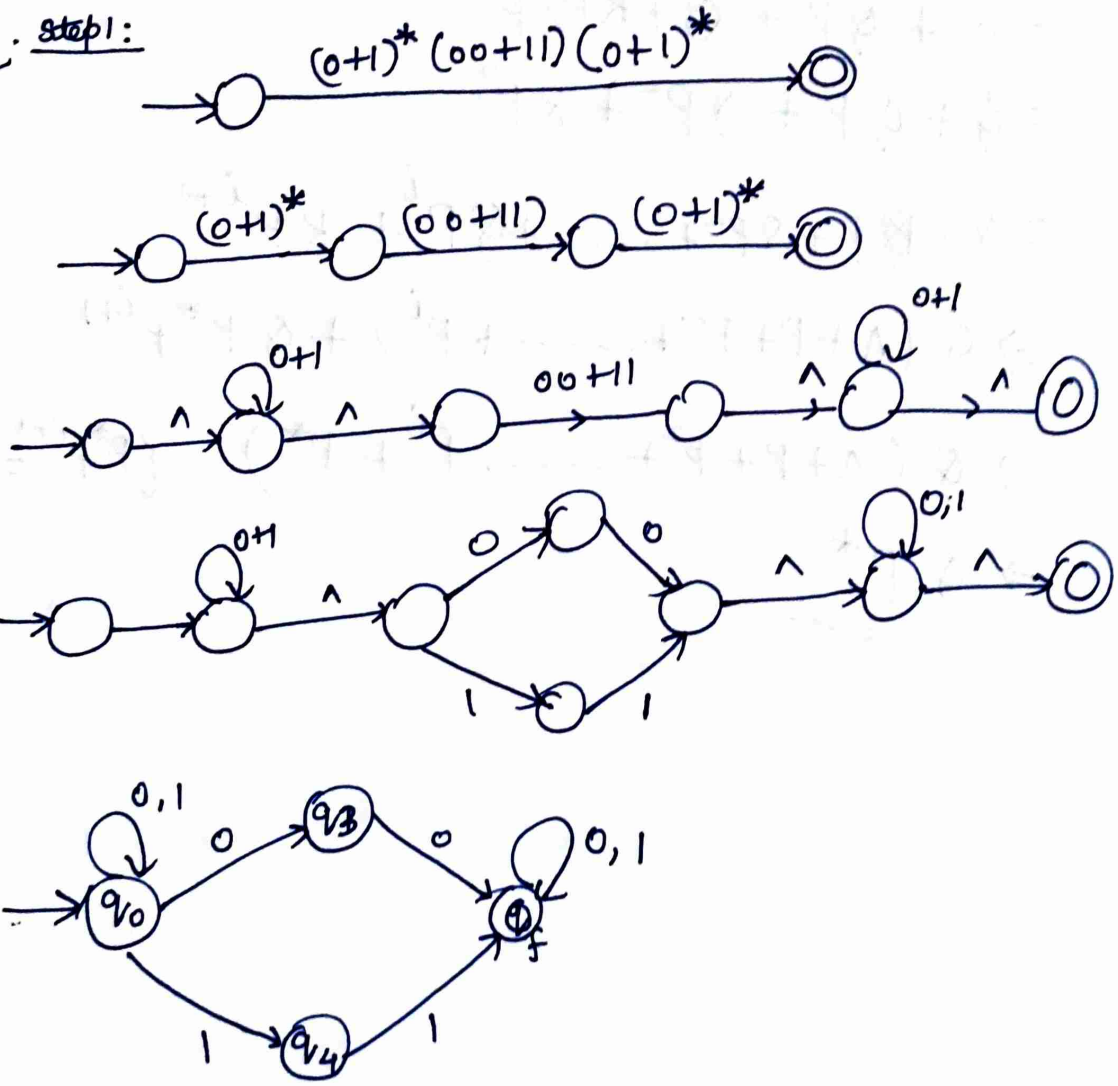
$\Rightarrow QP^*$

Regular Expression to Finite Automata

Q. Construct the finite automaton equivalent to the regular expression.

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

Sol: Step 1:



Step 2: Construction of DFA

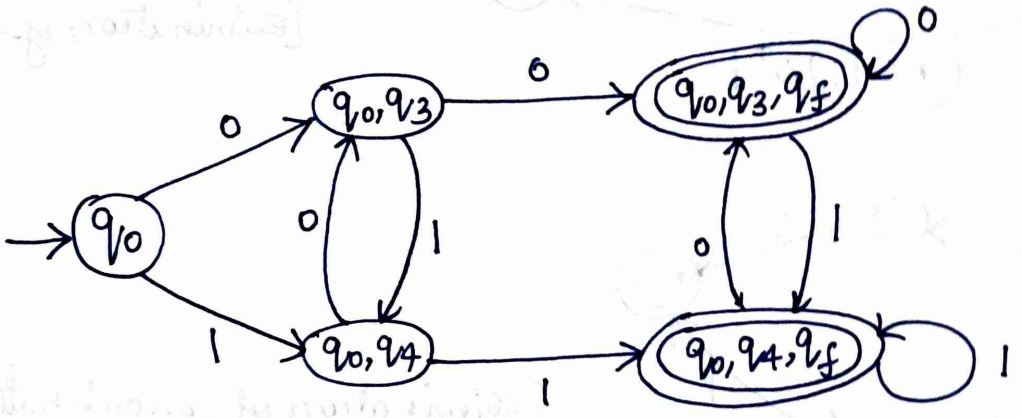
NFA

State	0	1
→q_0	$\{q_0, q_3\}$	$\{q_0, q_4\}$
q_3	$\{q_f\}$	$\{-\}$
q_4	$\{-\}$	$\{q_f\}$
(q_f)	$\{q_f\}$	$\{q_f\}$

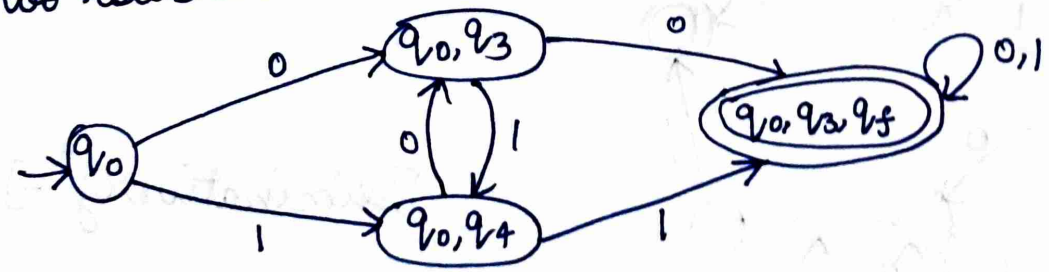
DFA

State	0	1
→ $[q_0]$	$[q_0, q_3]$	$[q_0, q_4]$
$[q_0, q_3]$	$[q_0, q_3, q_f]$	$[q_0, q_4]$
$[q_0, q_4]$	$[q_0, q_3]$	$[q_0, q_4, q_f]$
$[q_0, q_3, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$
$[q_0, q_4, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$

Identical Rows



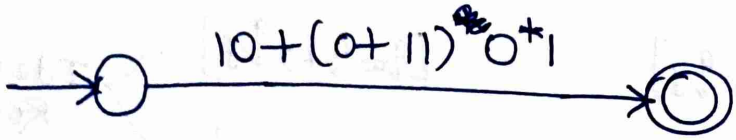
we try to reduce the states (This is possible when two rows are identical in the successor table.

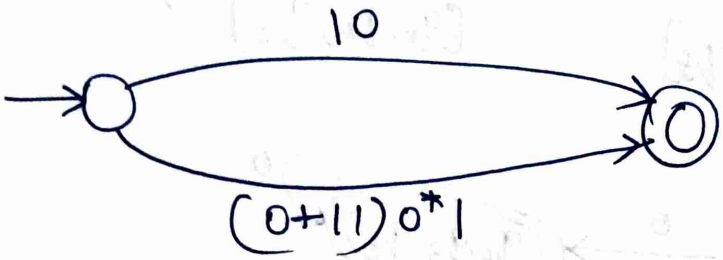


Q. Construct a DFA with reduced states equivalent to the R.E. $10+(0+11)0^*1$

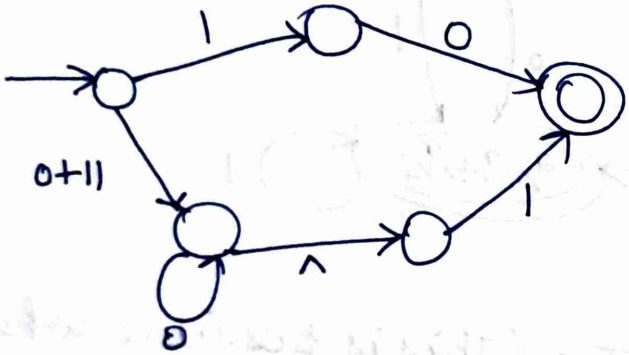
Sol:

Step 1: construction of NDFA.

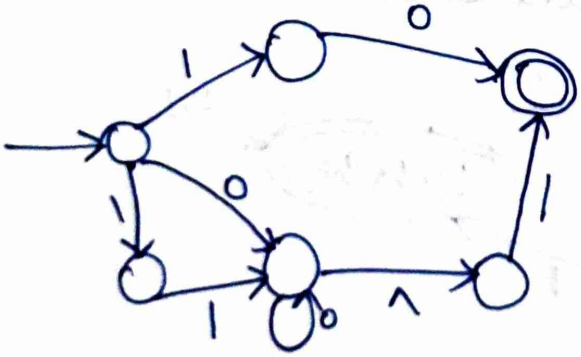




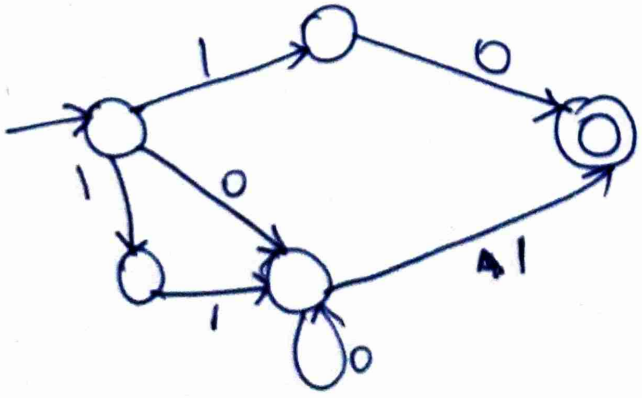
[Elimination of +]



[Elimination of concatenation and *]



[Eliminations of +]



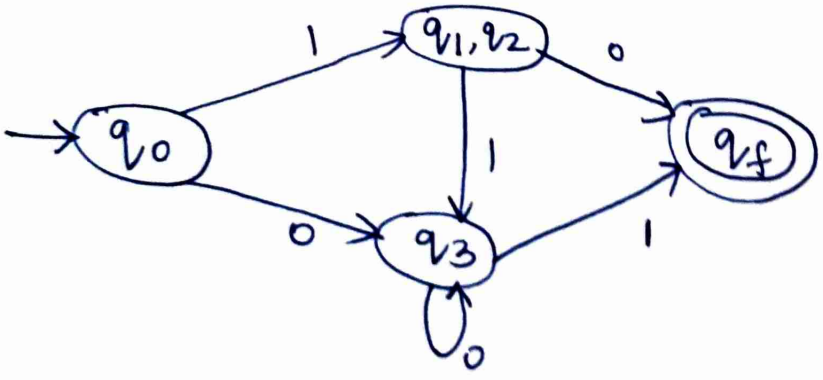
[Elimination of ^-moves]

Step 2: construction of DFA

NFA

State	0	1
→ q_0	q_3	q_1, q_2
q_1	q_f	–
q_2	–	q_3
q_3	q_3	q_f
q_f	–	–

state	0	1
→ $[q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1 q_2]$	$[q_f]$	$[q_3]$
$[q_f]$	ϕ	ϕ
ϕ	ϕ	ϕ



Reduced DFA

PUMPING LEMMA
(For Regular Languages)

Statement: If A is a Regular Language, then A has a Pumping Length 'P' such that any string 'S' where $|S| \geq P$ may be divided into 3 parts $S = xyz$ such that the following conditions must be true:

(1) $xy^iz \in A$ for every $i \geq 0$

(2) $|y| > 0$

(3) $|xy| \leq P$

To prove that a language is not Regular using PUMPING LEMMA, follow the below steps:

(we prove using contradiction)

→ Assume that A is Regular.
→ It has to have a Pumping Length (say P)
→ All strings longer than P can be pumped, $|S| \geq P$
→ Now, find a string 'S' in A such that $|S| \geq P$
→ Divide S into xyz
→ Show that $xy^iz \notin A$ for some i
→ Then consider all ways that S can be divided into xyz.

Example 1:

Using Pumping Lemma prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is not regular.

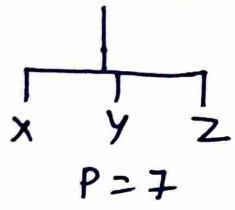
Sol.

PROOF:

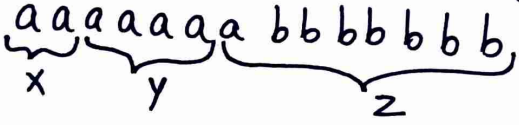
Assume A is Regular Language

Pumping length = P

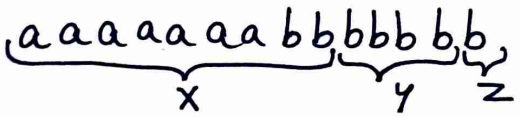
$S = a^P b^P \Rightarrow S = aaaaaaaabbbbbbbb$



Case 1: The Y is in 'a' part



Case 2 : The Y is in `b' part



Case 3 : The Y is in `a' and `b' part

Now,

Case 1: $xy^i z \Rightarrow xy^2 z$

aa aaaaaaaa abbbbbbb (Not Regular)

$11 \neq 7$

Case 2: $xy^i z \Rightarrow xy^2 z$

aaaaaaa bbbbbbbbbbbbbb, (Not Regular)

$7 \neq 11$

Case 3: $xy^i z \Rightarrow xy^2 z$

aaaaaa aabbaabb bbbbbb (Not Regular)

It does not follow the pattern $a^n b^n$