Non-Deterministic Finite Automata: (NFA)

The Finite automata which has 0 or 1 or more transistion for any i/p Symbol for any state is called as NFA.

(or)

NFA is a 5 tuple

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

Where Q is a finite set of states

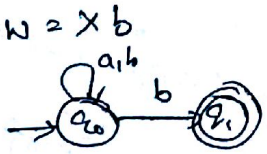
Σ is a finite alphabet

$\delta : Q \times \Sigma \longrightarrow P(Q)$ is the transistion function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accepting states

Ex: $\Sigma = \{a, b\}$ Construct NFA for Language L over an alphabet $\{a, b\}$ where every string ends with b



$Q = \{q_0, q_1\}$

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

Note:

1. The Expressive power of NFA and DFA is same

$$\in(NFA) = \in(DFA)$$

2. Construction of NFA is easier than DFA
3. DFA is more efficient than NFA
4. DFA takes care of both valid as well as invalid strings, but NFA take care of only valid i/ps.
5. No concept of dead states is there in NFA
6. In NFA, no need to define the transistion for each & every i/p Symbol at each and every state.
7. NFA can move to more than one state after taking i/p Symbol

(8) In NFA a string can have multiple transition paths.

Acceptance by NFA:

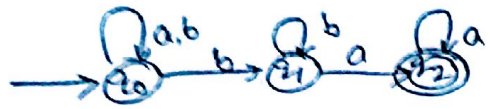
Let w be any string from the alphabet Σ, corresponding to w there can be multiple transition path in NFA

→ If atleast one path starts in initial state and ends at final state then the string w is accepted by NFA.

→ The set of all strings which are accepted by NFA is called Language of NFA.

$$\therefore L(NFA) = \{x \in \Sigma^* \mid \delta(q_0, x) = F\text{-}S\}$$

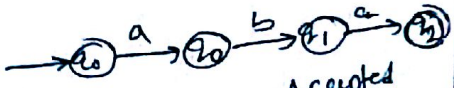
Ex:



w = aba

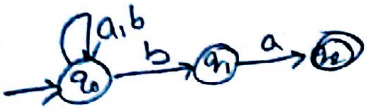
Transition path:

Transition path

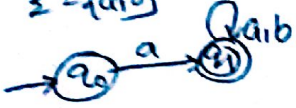


Example: Construct an NFA that accepts all the strings of a's & b's where every string

① starts with a ② ends with ba ③ Contain substring ab



① $\Sigma = \{a, b\}$



Ex 2: Construct the NFA that accepts all the strings of a's & b's where every string

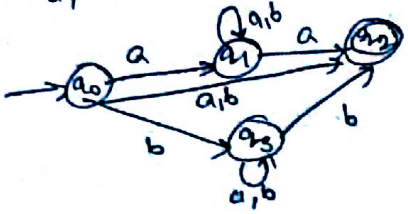
(a) starts and ends with a (b) starts and ends with same symbol

(c) starts and ends with diff symbol

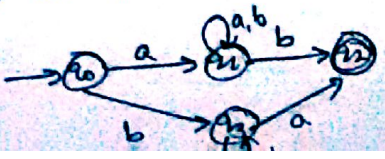
(a)

a, axa

(b) a, b axa, bxb



(c) ab axb, bxa



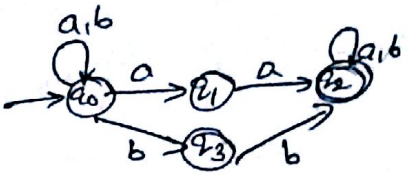
Construct the NFA that accepts all strings of a's & b's where every string

① starts with aa or bb

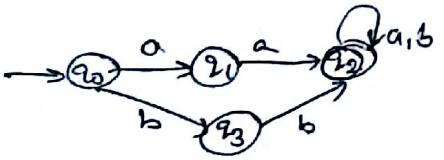
② ends with aa or bb

③ contain substring aa or bb

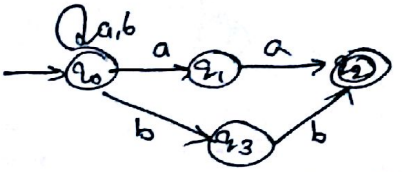
x aax, xbbx



w = aax
bbx



w = xaa
xbb

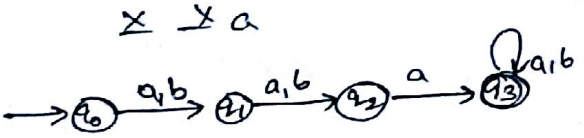


Q4 Construct the NFA that accepts all strings of a's & b's where

① The 3rd symbol from left hand side is a

② The 4th symbol from right hand side is b

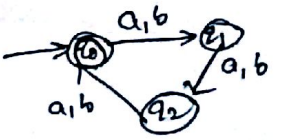
① x x a

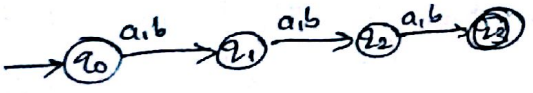


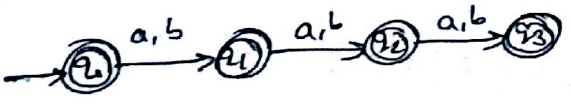
② x b x x x

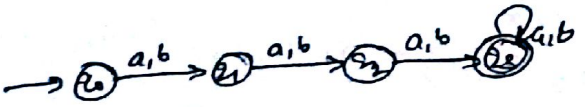
Q5 Length of the string is $\Sigma = \{a, b\}$

① exactly 3 ② atmost 3 ③ atleast 3 ④ divisible by 3









Q6 Construct the NFA for the following

① $L = \{a^n b \mid n \geq 0\}$

② $L = \{ab^n \mid n \geq 0\}$

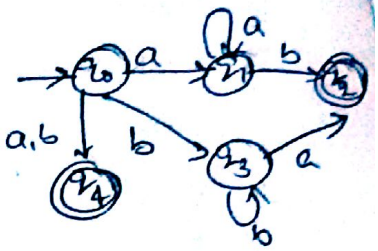
③ $L = \{a^n b \cup b^m a \mid m, n \geq 0\}$

④ $L = \{a^m b^n / m, n \geq 1\}$

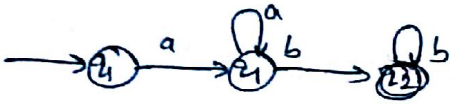
⑤ $L = \{a^m b^n / m \geq 0, n \geq 1\}$

① 

② 

③ 

④ $L = \{a^m b^n / m, n \geq 1\}$



⑤ $L = \{a^m b^n / m \geq 0, n \geq 1\}$

Conversion from NFA to DFA:

The process of conversion of NFA to DFA is called subset Construction.

NFA $\Longrightarrow$ DFA

↓ n states ↓ m states $\quad 1 \leq m \leq 2^n$

Alg:

let $M = (Q, \Sigma, \delta, q_0, F)$ be the NFA

$M' = (Q', \Sigma', \delta', q_0', F')$ be the DFA

① Initial state

No change in the initial state

$$\boxed{q_0' = q_0}$$

② Construction of δ'

$$\delta'(q_0, x) = \delta(q_0, x)$$

$$\delta'(q_0 \cup q_1, x) = \delta(q_0, x) \cup \delta(q_1, x)$$

$$\delta'(q_0, q_1, q_2 \cdots q_n, x) = \bigcup_{i=0}^{n} \delta(q_i, x)$$

③ Start the construction of δ' with the initial state and continue for every new state.

④ **Final state:** Every subset which contain the final state of NFA is a Final state in DFA.

Note: The DFA which is obtained from NFA need not be minimal & contain at most 2^n states, where n is the no. of states in NFA.

Ex: ①

δ	a	b
→ q_0	$\{q_0, q_1\}$	ϕ
q_1	$\{q_2\}$	$\{q_0, q_2\}$
* q_2	ϕ	$\{q_1, q_2\}$

⇒

δ'	a	b
→ q_0	q_0q_1	ϕ
q_0q_1	$\{q_0, q_1\}$	q_0q_2
* q_0q_2	q_0q_1	q_1q_2
* q_1q_2	q_1	$q_0q_1q_2$
* $q_0q_1q_2$	q_0q_1	$q_0q_1q_2$
q_1	q_1	q_0q_2
ϕ	ϕ	ϕ

②

δ	a	b
→ q_0	$\{q_0q_1\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$
* q_2	ϕ	$\{q_2\}$

⇒

δ	a	b
→ q_0	q_0q_1	q_1
q_0q_1	q_0q_1	q_1q_2
q_1	q_1	q_1q_2
* q_1q_2	q_1	q_1q_2

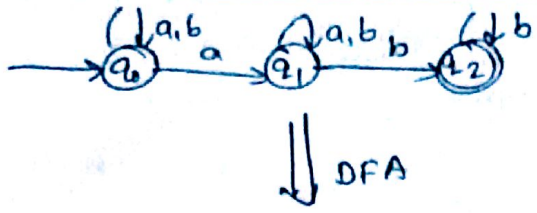
③

δ	a	b
→ q_0	$\{q_0, q_2\}$	$\{q_2\}$
* q_1	$\{q_1\}$	$\{q_1, q_2\}$
q_2	$\{q_2\}$	$\{q_1\}$

⇒

δ	a	b
→ q_0	q_0q_2	q_2
q_0q_2	q_0q_2	q_1q_2
q_2	q_2	q_1
* q_1q_2	q_1q_2	q_1q_2

④



DFA

NFA with ∈-moves or ∈-NFA:

The NFA which has a transition even for empty string ∈ is called as ∈-NFA.

(or)

∈-NFA is a 5tuple

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

where

$Q \rightarrow$ set of all states

$\Sigma \rightarrow$ I/p alphabet

$\delta \rightarrow Q \times \Sigma \cup \{\in\} \rightarrow P(Q)$ transition function

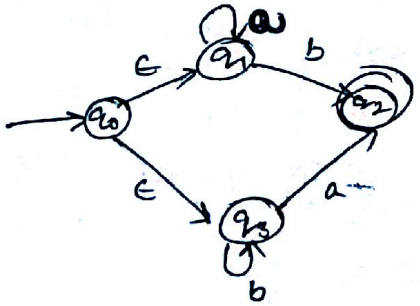
$q_0 \rightarrow$ Initial state

$F \rightarrow$ set of all final states.

Ex: ① $L = \{a^n b \mid n \geq 0\}$

② $L = \{a^m b^n \mid m, n \geq 0\}$

③ $L = \{a^n b \cup b^m a \mid m, n \geq 0\}$



Note:

① The Computing power of ∈-NFA, NFA, DFA is same.

$$\therefore E(\in\text{-NFA}) = E(\text{NFA}) = E(\text{DFA})$$

② The inclusion or exclusion of ∈-transition will not effect the language of NFA.

ϵ-closure:

Let q is any state in ϵ-NFA, then the set of all the states which are at distance zero from the state q is called ϵ-closure of q.

(or)

The set of all the states that can be reached from a state on the i/p symbol ϵ is called ϵ-closure of q.

Ex: 1

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

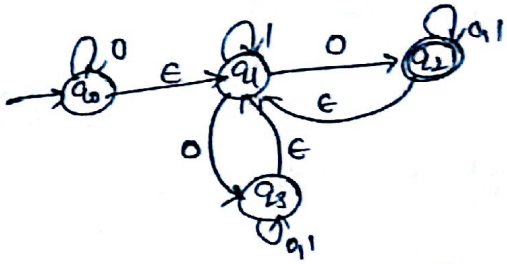
ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

Note: Every state is at 0 distance from itself.

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

Ex 2:



ϵ-closure $(q_0) = \{q_0, q_1\}$

ϵ-closure $(q_1) = \{q_1\}$

ϵ-closure $(q_2) = \{q_2, q_1\}$

ϵ-closure $(q_3) = \{q_3, q_1\}$

Conversion of ϵ-NFA to NFA: (Thomson's Algorithm)

→ The process of conversion of ϵ-NFA to NFA is called as Thomson's Construction

→ No change in total no. of states.

→ No change in the initial state.

→ May be change in the final states.

Algorithm: let $M = (Q, \Sigma, \delta, q_0, F)$ is a ϵ-NFA

let $M' = (Q', \Sigma', \delta', q_0', F')$ be the NFA.

① **Initial state:** No change in initial state.

$$q_0' = q_0$$

② **Construction of δ':**

$$\delta'(q, x) = \epsilon\text{-closure}\{\delta(\epsilon\text{-closure}(q), x)\}$$

③ **Final state:**

Every state whose ϵ-closure contain the final state of ϵ-NFA is a Final state in NFA.

Ex:

ϵ-closure $(q_0) = \{q_0, q_1\}$

ϵ-closure $(q_1) = \{q_1\}$

$$\delta'(q_0, a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), a)$$

$$= \epsilon\text{-closure}(\delta(q_0, q_1), a)$$

$$= \epsilon\text{-closure}(q_0 \cup \emptyset)$$

$$= \epsilon\text{-closure}(q_0)$$

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

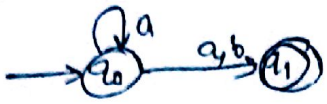
$$\delta'(q_0, b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), b)$$

$$= \epsilon\text{-closure}(\delta(q_0, q_1), b)$$

$$= \epsilon\text{-closure}(\emptyset \cup q_1)$$

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

$$= \{q_1\}$$



(2)

ϵ-closure $(q_0) = \{q_0, q_1\}$

ϵ-closure $(q_1) = \{q_1\}$

ϵ-closure $(q_2) = \{q_2, q_1\}$

$\delta'(q_0, a) = \epsilon$-closure $(\delta(\epsilon\text{-closure}(q_0)), a)$

$= \epsilon$-closure $(\delta(q_0, q_1), a)$

$= \epsilon$-closure (q_0, q_1)

$= \{q_0, q_1\}$

$\delta'(q_0, b) = \epsilon$-closure $(\delta(\epsilon\text{-closure}(q_0), b)$

$= \epsilon$-closure $(\delta(q_0, q_1), b)$

$= \epsilon$-closure $(\phi \cup q_2)$

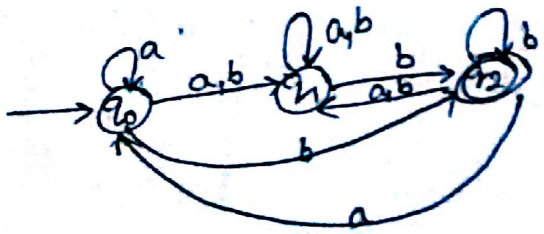
$= \epsilon$-closure (q_2)

$= \{q_1, q_2\}$

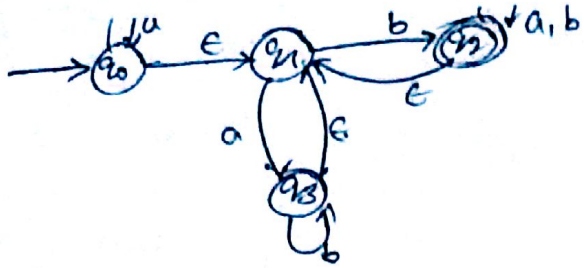
q_1 $\epsilon\rightarrow$	a	b
	$\delta(q_1, a)$	$\delta(q_1, b)$
	$\downarrow$	$\downarrow$
	$\epsilon(q_1)$	$\epsilon(q_2)$
	$\downarrow$	$\downarrow$
	q_1	$\{q_1, q_2\}$

q_2 $\epsilon\rightarrow$	a	b
	$q_1 q_2$	$q_1 q_2$
	$\downarrow$	$\downarrow$
	$\delta(q_1 q_2), a$	$\delta(q_1 q_2), b$
	$\downarrow$	$\downarrow$
	$\epsilon(q_1)$	$\epsilon(q_2)$
	$\downarrow$	$\downarrow$
	$\{q_0 q_1\}$	$\{q_1, q_2\}$

δ'	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$
$*q_2$	$\{q_0, q_1\}$	$\{q_1, q_2\}$



(3)



ϵ-closure(q_0) = $\{q_0, q_1\}$

ϵ-closure(q_1) = $\{q_1\}$

ϵ-closure(q_2) = $\{q_1, q_2\}$

ϵ-closure(q_3) = $\{q_1, q_3\}$

$\delta'(q_0, a)$ = ϵ-closure($\delta(\epsilon\text{-closure}(q_0), a)$)

= ϵ-closure($\delta(q_0, q_1), a$)

= ϵ-closure(q_0, q_3)

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

$\delta'(q_0, b)$ = ϵ-closure($\delta(\epsilon\text{-closure}(q_0), b$)

= ϵ-closure($\delta(q_0, q_1), b$)

= ϵ-closure($\emptyset \cup q_2$)

= ϵ-closure(q_2)

= $\{q_1, q_2\}$

q_1:

δ'	a	b
	$\epsilon(q_1)$	$\epsilon(q_1)$
	$\delta(q_1, a)$	$\delta(q_1, b)$
	$\epsilon(q_3)$	$\epsilon(q_2)$
	$\{q_1, q_3\}$	$\{q_1, q_2\}$

q_2:

δ'	a	b
	$\epsilon(q_2)$	$\epsilon(q_2)$
	$\delta(q_1, q_2, a)$	$\delta(q_1, q_2, b)$
	$\epsilon(q_3, q_2)$	$\epsilon(q_2, q_2)$
	$\{q_1, q_3, q_2\}$	$\{q_1, q_2\}$

q_3:

δ'	a	b
	$\epsilon(q_3)$	$\epsilon(q_3)$
	$\delta(q_1, q_3, a)$	$\delta(q_1, q_3, b)$
	$\epsilon(q_3)$	$\epsilon(q_2, q_3)$
	$\{q_1, q_3\}$	$\{q_1, q_2, q_3\}$

Minimisation of FSM:

Equal State: Two states P, q are said to be equal

if $\delta(P,x) = \delta(q,x) \quad \forall x \in \Sigma^*$

$\therefore P \leftrightarrow q$ iff both $\delta(P,x)$ & $\delta(q,x)$ goes to final or Non-Final state for every $x \in \Sigma^*$

* Equal states are also called as Non-distinguishable states.

Unequal State: Two states P, q are said to be unequal

iff $\delta(P,x) \neq \delta(q,x)$ for atleast one string $x \in \Sigma^*$

Ex:

	a	b
→A	A	B
*B	C	B

	a	b
→A	A	B
C	C	B

Note: ① Every state is equal to itself

② Final state is not equal to Non-Final state ⇒ FS ≠ NFS

③ The equality relation exists b/w two final states or two Non-Final states.

k-equivalence class:

① Two states (P, q) said to be in k-equivalence class if $\delta(P,x) = \delta(q,x)$ $\forall |x| \leq k$

② P, q are equal iff P, q ∈ k-equivalence class $\forall$ k ≥ 0.

Minimization of FA:

The process of elimination of the states whose presence or absence won't affect the Language as well as the structure of FA is called as minimization of FA.

① The presence or absence of unreachable or equal states will not effect Language and structure.

② If we remove Dead state from DFA then there won't be any change in the Language, but the structure. so dead state cannot be removed in the process of minimization.

Minimal FA: The FA which is free from unreachable & equal states is called as MFA.

(or)

The FA with 'n' states is said to be MFA for the Language L, if it is not possible to construct any other FA with less than N states to accept same Language L.

Note: DO DFA is not unique, but MDFA is unique.

Process of Minimization:

① Find all the equal states and remove them.

There are 2 methods to find out equal states.

→ State Equivalence Method.

→ Table filling Method.

② Find all the unreachable states & remove them.

③ The DFA which is free from unreachable & equal states is minimal.

State Equivalence Method:

Step1: We will create a set Π_0 as $\Pi_0 = \{Q_1^0, Q_2^0\}$ where Q_1^0 is set of all final states and $Q_2^0 = Q - Q_1^0$ where Q is a set of all states in DFA.

Step 2: Now we will construct Π_{k+1} from Π_k. Let Q_i^k be any subset in Π_k.

If q_1 and q_2 are in Q_i^k they are (k+1) equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k equivalent. Findout whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are residing in the same equivalence class Π_k. Then it is said that q_1 and q_2 are (k+1) equivalent. Thus Q_i^k is further divided into (k+1) equivalence classes. Repeat step 2 for every Q_i^k in Π_k and obtain all the elements of Π_{k+1}.

Step 3: Construct Π_n for $n = 1, 2, \ldots$ untill $\Pi_n = \Pi_{n+1}$

Step 4: Then replace all the equivalent states in one equivalence class by representative state. This helps in minimizing the given DFA.

Table Filling Method:

This method is based on distinguishable and indistinguishable states.

Distinguishable states: If for some input string w - $\delta(p, w)$ gives accepting state and $\delta(q, w)$ gives non-accepting state then state p and q are called distinguishable states of non-equivalent states.

Indistinguishable states: If for some input string w - $\delta(p, w)$ and $\delta(q, w)$ both produce either accepting states (or) non-accepting states then states p and q are called indistinguishable (or) equivalent states

Ex:1

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

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_2

State Equivalence Method:

0-equivalence $\pi_0 = \{q_0, q_2\}\ \{q_1\}$ (G_1, G_2)

1-equivalence $\pi_1 = \{q_0, q_2\}\ \{q_1\}$

	a	b
q_0	q_1	q_0
q_2	q_1	q_2
	G_2	G_1

$\therefore q_0 = q_2$

2-equivalence $\pi_2 = \{q_0, q_2\}, \{q_1\}$

$\therefore \pi_1 = \pi_2$

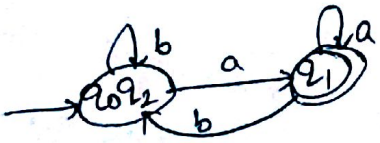
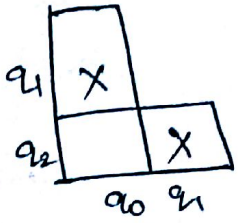
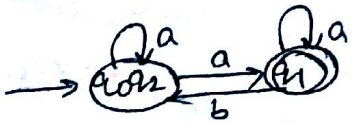


Table Filling Method:

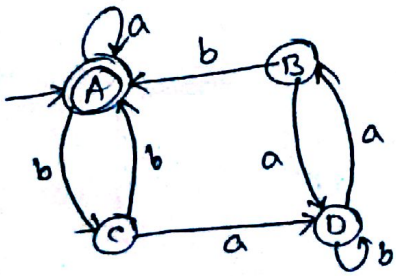


	a	b
q_0	q_1	q_0
q_2	q_1	q_2
	F	NF

$\therefore q_0 = q_2$



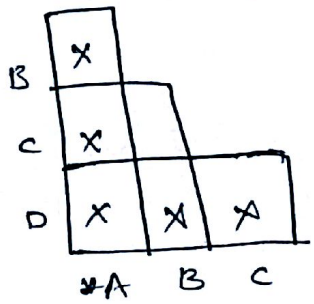
Ex2: Construct the minimal FA for the following DFA.



Q = { A, B, C, D }

δ	a	b
A	A	C
B	D	A
C	D	A
D	B	D

Table Filling Method:



	a	b
B	D	A
C	D	A

NF F

	a	b
B	D	A → F
D	B	D → NF

	a	b
C	D	A → F
D	B	D → NF

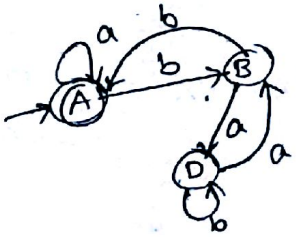
NF

	a	b
B	D	A
C	D	A

G2 G1

∴ B = C

δ	a	b
*A	A	B
B	D	A
D	B	D



State Equivalence Method:

Π_0 : {B, C, D} {A}
(0)

0-equi Π_0 = {A} {B, C, D}
G_1 G_2

1-equi Π_1 = {A} {B, C} {D}

	a	b
B	D	A
C	D	A

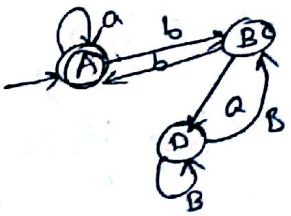
G_2 G_1

	a	b
B	D	A → G
D	B	B → G_2

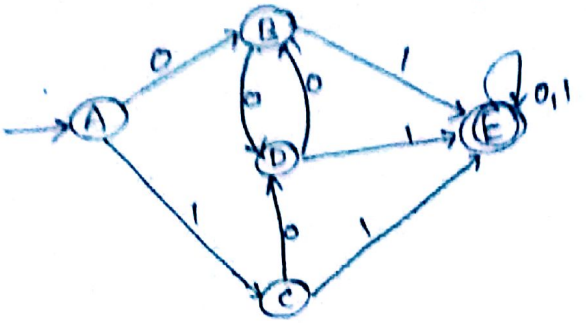
G_{12}

2-equi Π_2 = {A} {B, C} {D}

∴ $\Pi_1 = \Pi_2$



Ex 3:



	0	1
→A	B	C
B	D	E
C	D	E
D	B	E
*E	E	E

Table Filling Method:

B	X			
C	X			
D	X			
*E	X	X	X	X
	A	B	C	D

	0	1	
A	B	C	→ NF
D	B	E	→ F
NF			

	0	1	
A	B	C	→ NF
C	D	E	→ F
NF			

	0	1	
A	B	C	→ NF
B	D	E	→ F
NF			

	0	1	
B	D	E	
D	B	E	
NF		F	

B = D

	0	1	
B	D	E	
C	D	E	
NF		F	

B = C

	0	1	
C	D	E	
D	B	E	
NF		F	

C = D

∴ B = C = D

State Equivalence Method:

0-equi $\Rightarrow \pi_0 = \{A, B, C, D\}, \{E\}$

1-equi $\Rightarrow \pi_1 = \{A\} \quad \{B, C, D\} \quad \{E\}$

	0	1
A	B	C $\rightarrow G_1$
B	D	E $\rightarrow G_2$

G_1

	0	1
A	B	C $\rightarrow G_1$
C	B	E $\rightarrow G_2$

G_1

	0	1
A	B C	$\rightarrow G_1$
D	B	E $\rightarrow G_2$

G_1

	0	1
B	D	E
C	D	E

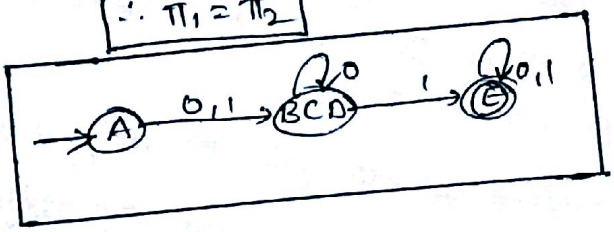
	0	1
B	D	E
D	B	E

G_1 G_2

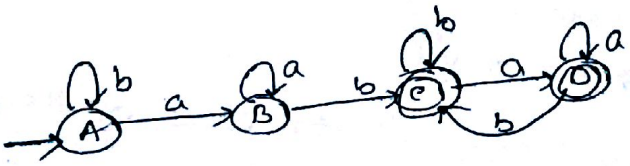
2-equi $\Rightarrow \pi_2 = \{A\} \quad \{BCD\} \quad \{E\}$

$\therefore \pi_1 = \pi_2$

Minimal FA



Ex4:



$\pi_0 = \{A, B\} \{C, D\}$

$\pi_1 = \{A\} \{B\} \{C, D\}$

	a	b
A	B	A $\rightarrow G_1$
B	B	C $\rightarrow G_2$

G_1

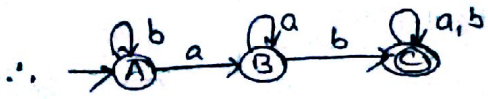
	a	b
B	D	C
D	D	C

G_2 G_2

B	X	
C	X	X
D	X	X
	A	B C

$\pi_2 = \{A\} \{B\} \{C, D\}$

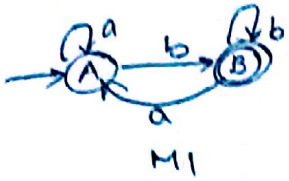
$\therefore$



Equivalence b/w Two Finite State M/cs:

Two FAs M_1 and M_2 are said to be equal if both of them accepts the same language.

Ex:



M_1

M_2

$w = xb$ $\quad L(M_1) = L(M_2)$

$\therefore M_1$ & M_2 are equal

Comparision Algorithm:

Let M_1, M_2 - two FSM

① Start the construction of the transistion table, that contains pair of state (P, q) where $P \in M_1$ and $q \in M_2$

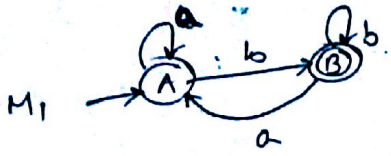
② Start with pair of initial state.

③ While constructing the table if we get any pair of the form (Final, Non-Final) (or) (Final, Non-Final) then stop the construction of the table and declare that two m/c are not equal.

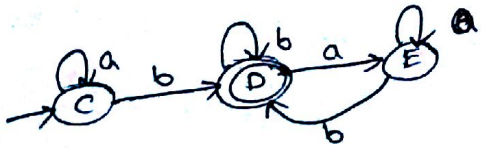
④ Continue the construction of the table for every new pair of the form (Final, Final) or (Non-Final, Non-Final) and stop the construction whenever no such new pair occurs.

⑤ If the transistion table contains all the pairs of the form (F, F) or (NF, NF) then the two machine are equal.

Eg1:



M_1

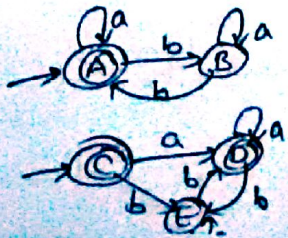


M_2

δ	a	b
(A,C)	(A,C)	(B,D)
(B,D)	(A,E)	(B,D)
(A,E)	(A,E)	(B,D)

$\therefore M_1$ & M_2 are equal

Ex2:



	a	b
(A,C)	(A,D)	(B,E)
(B,E)	(B,E)	(A,D)
(A,D)	(A,D)	(B,E)

$\therefore M_1$ & M_2 are equal

FA with output: Moore Machine and Mealy Machine.

① Moore & Mealy machine are special case of DFA.

② Moore & Mealy machine are output producing rather than language acceptors.

Moore Machine: FA where the o/p is associated with the state is called Moore M/c.

(or)

Moore Machine is a 6-tuple.

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

Where $Q \rightarrow$ set of all states

$\Sigma \rightarrow$ i/p alphabet

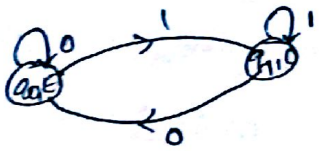
$\delta: Q \times \Sigma \rightarrow Q$ is a transition function

Δ: o/p alphabet

$\lambda \rightarrow (Q \rightarrow \Delta)$ is an o/p function

$q_0 \rightarrow$ Initial state

Ex:



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

$\Delta = \{E, O\}$

E → Even

O → odd

Representation of Moore M/c:

Moore M/c can be represented in two ways

① Transition diagram

② Transition table.

	0	1	Δ
→q_0	q_0	q_1	E
q_1	q_0	q_1	O

$\lambda(q_0) = E$

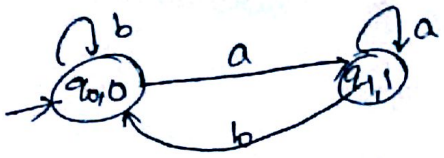
$\lambda(q_1) = O$

Ex: 010

q_0 (E) $\xrightarrow{0}$ q_0 (E) $\xrightarrow{1}$ q_1 (O) $\xrightarrow{0}$ q_0 (E)

Ex: ① Construct the Moore M/c that takes all the strings of a's & b's as i/p and count the no. of occurancy of a's in the i/p string.

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

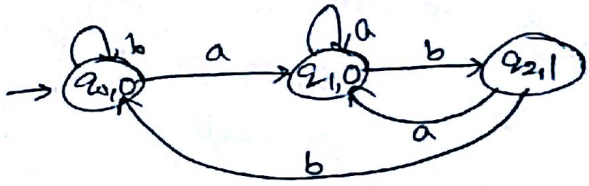


$\lambda(abaa) = 1011 = ③$

② Construct the Moore M/c that takes all the strings of a's & b's and count the no. of occurancy of the substring ab.

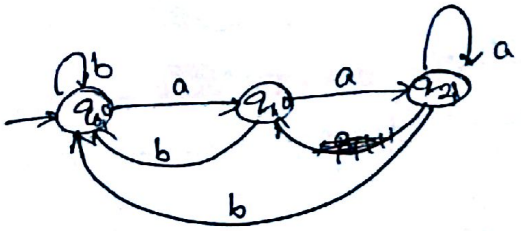
$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

ab = 1

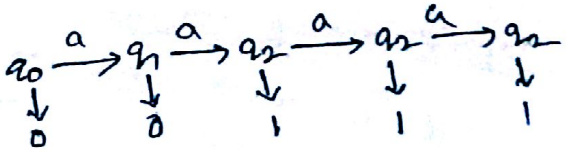


③ Construct the Moore M/c that takes all the strings of a's & b's as i/p and count the no. of occurance of 2 consecutive a's in the i/p string

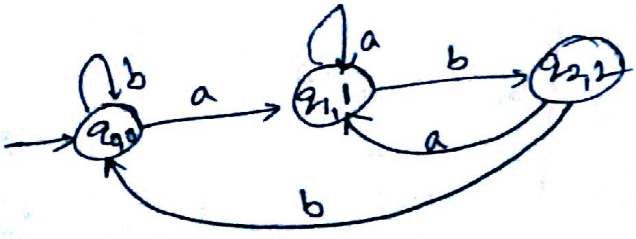
$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



$\lambda(aaaa)$

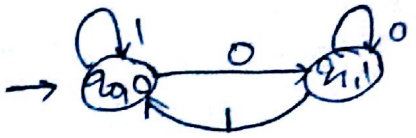


④ Construct the Moore M/c that takes all the strings of a's & b's where the o/p is 1, if the i/p ends with a, o/p is 2 if the i/p ends with ab, otherwise o/p is 0.



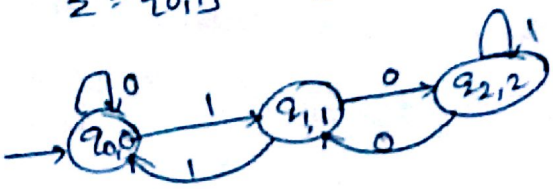
Ex5: Construct the Moore M/c to find out the 1's complement of binary no.

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



Ex6: Construct the Moore M/c to produce the remainder when an integer equivalent of binary no is divisible by 3.

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



	0	1	
→q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

Moley M/c: Finite Automata, where the o/p is associated with the transition is called as Moley Machine. (5)

Moley M/c is 6-tuple

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

where Q → set of all states

Σ → I/p alphabet

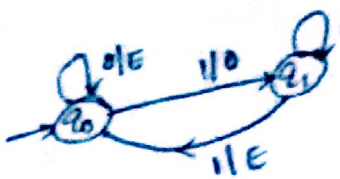
$\delta : Q \times \Sigma \to Q$ transition function

Δ → o/p alphabet

$\lambda : Q \times \Sigma \to \Delta$ is o/p function

q_0 → Initial state.

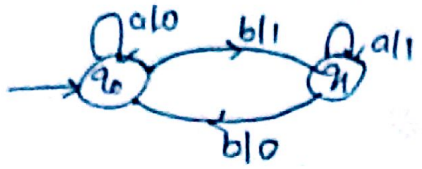
Ex:



Representation of Melay Machine:

Melay M/c can be represented in two ways.

transition diagram

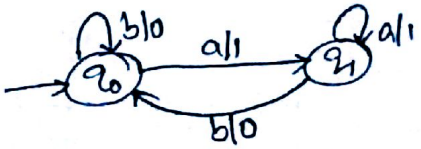


table

	a	λ	b	λ
→ q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

Ex1: Construct the Melay M/c that takes all the strings of a's & b's as i/p and count the no. of occurance of a's in the given substring.

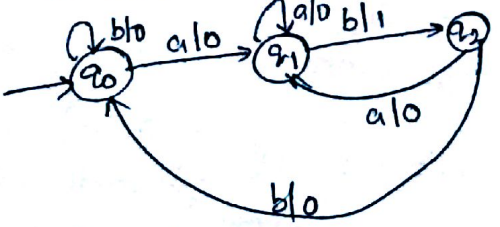
$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



(or)

Ex2: Construct the melay M/c that takes all the strings of a's & b's count the no. of occurance of substring ab.

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

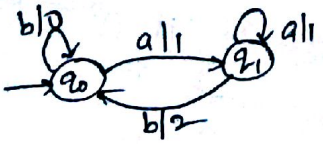


abaaab

↑

λ

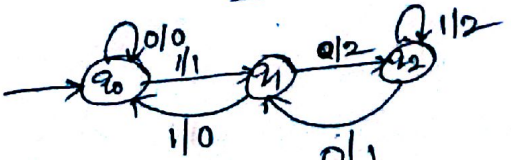
Ex3: Construct the Melay M/c which accepts all the strings of a's & b's where o/p is 1 if i/p string ends with a, o/p is 2 if string ends with ab, otherwise 0.



Ex4: 1's Complement

Ex5: Construct the Melay M/c to find out the remainder and integer equivalent of binary no is divisible by 3.

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



	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Conversions:

(1) Conversion of Moore M/c to Mealy M/c:

Ex1:

δ	a	b	λ
→q_0	q_2	q_1	0
q_1	q_0	q_2	1
q_2	q_1	q_0	0

⇒

Mealy M/c:

δ	a	λ	b	λ
q_0	q_2	0	q_1	1
q_1	q_0	0	q_2	0
q_2	q_1	1	q_0	0

Ex2:

	a	b	λ
→q_0	q_1	q_0	0
q_1	q_1	q_2	1
q_2	q_1	q_0	2

⇒

Mealy M/c:

	a	λ	b	λ
q_0	q_1	1	q_0	0
q_1	q_1	1	q_2	2
q_2	q_1	1	q_0	0

(2) Conversion of Mealy M/c to Moore M/c:

Ex1:

	a	λ	b	λ
q_0	q_1	1	q_0	0
q_1	q_1	1	q_2	2
q_2	q_1	1	q_0	0

⇒

Moore M/c

δ	a	b	λ
q_0	q_1	q_0	0
q_1	q_1	q_2	1
q_2	q_1	q_0	2

Ex2:

δ	a	λ	b	λ
→q_0	q_4	1	q_3	0
q_1	q_3	0	q_2	0
q_2	q_2	1	q_1	1
q_3	q_1	0	q_4	0
q_4	q_0	1	q_0	1

⇒

Moore M/c:

δ	a	b	λ
→q_0	q_4	q_3	1
q_{10}	q_3	q_2	0
q_{11}	q_3	q_2	1
q_{21}	q_{21}	q_{11}	1
q_{20}	q_{21}	q_{11}	0
q_3	q_{10}	q_4	0
q_{40}	q_0	q_0	0
q_{41}	q_0	q_0	1