CS E 322 Formal Languages & Automation Theory

String: Fine sequence of symbol from alphabets.

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

$\{a, b\}$

Input symbol

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

length of string ($|W|$)

$W = \{aba\}$

Here $|W| = |W^r|$

$W^r = aba$

substring:

Ex: SUBSTRING

BST ✓ STR ✓ SUB ✓ SRS ✗ SRI ✗

These are substrings — These are not substring.

~~Formal definition~~

Empty string / Null: zero occurence.

denoted by 'ϵ' i.e. Epsilon. or 'λ'.

$|\epsilon| = 0$

Unit I Finite Automata

"O" is known as states and "→" are transition.

- Finite Automata is divided into two types.

FA

without O/P

i) Determinate FA (DFA)

ii) Non Determinate FA (NDFA)

with output

i) Mealy Machine

ii) Moore Machine

Formal Defination of Finite Automaton

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

Q : Finite nonempty set of states.

Σ : finite non empty set of input alphabets.

δ : (direct) transition function that maps $Q \times \Sigma \rightarrow Q$

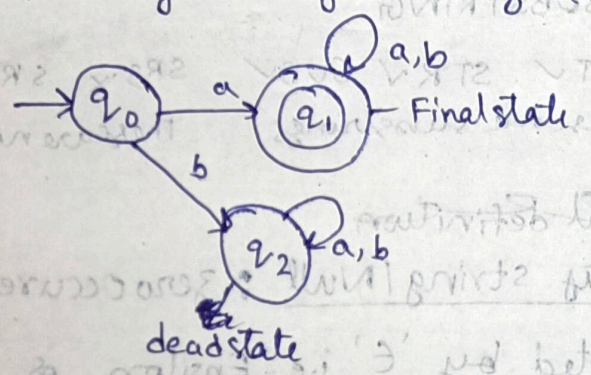
q_0 : initial state

F : set of final states.

Q) Create a DFA which will accept only string starting a.

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

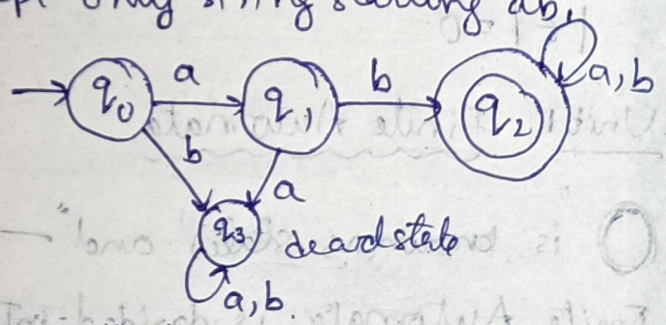
Ans:- $\{a, ab, abb, aaa, \ldots\}$



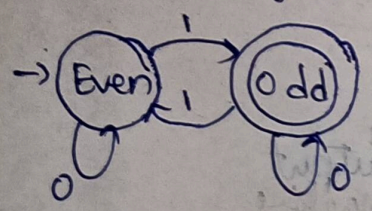
Q) Create a DFA which will accept only string starting ab.

abxx ...

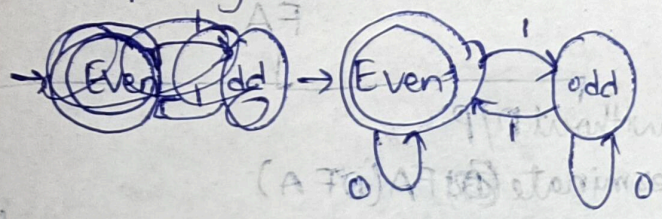
$L = \{ab, aba, abb, abab \ldots\}$



Q) odd Parity Detector



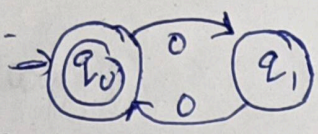
even parity detector.



Representation of Finite Automata

① Representation by Equations

Eg:-



$\Sigma = \{0\}$

Accept only even zeros

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

$Q = \{q_0, q_1\}$, $\Sigma = \{0\}$

δ is given by

$\delta(q_0, 0) = q_1$ (δ means $Q \times \Sigma \to Q$)

$\delta(q_1, 0) = q_0$

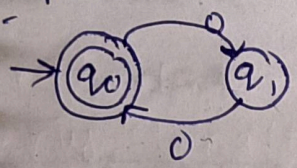
q_0 is initial state

$F = \{q_0\}$ is a final state.

② Representation by Transition diagram.

③ Representation by Transition table.

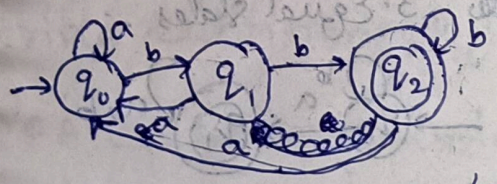
Eg:-



PS	Σ 0
$\to q_0$	q_1
q_1	q_2

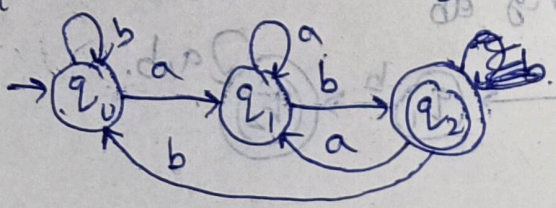
Q) Design DFA {a, b} such that string must ends with bb.

L = { bb, abb, abbb, aabb, ... }

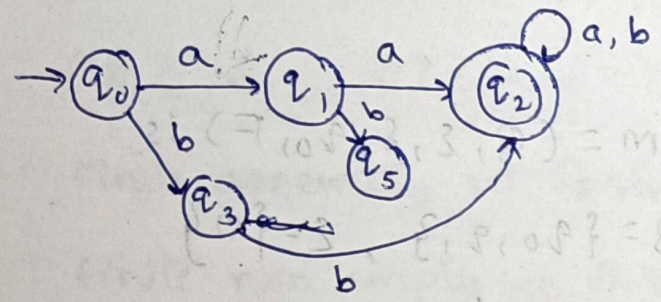


Q) Design DFA {a, b} such that string ends with ab

L = { ab, aab, bab, bbab ... }

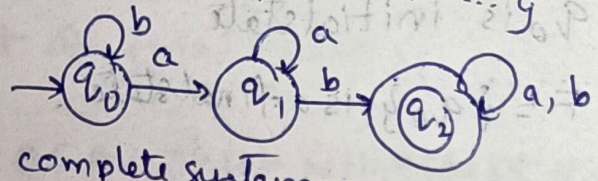


Q) Design DFA which will accept starting from either aa or bb.

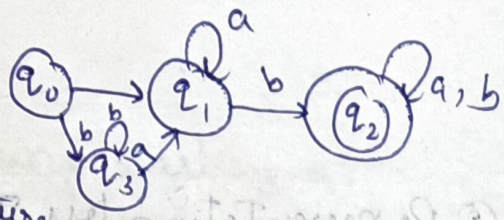


Q) Contain substring ab: i-e xxxabxxx

L = { ab, aaaba, babaa ... }



(or)



*DFA is complete system and complex structure.

Minimization of DFA

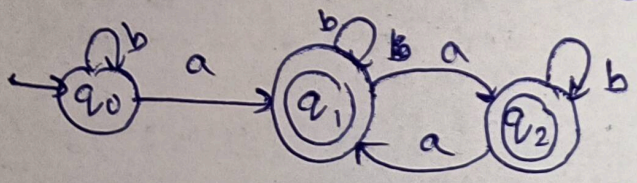
Reduce the Complexity & increase efficiency.

1. Productive state
2. Unproductive state → Dead state
3. Unreachable state (URS)

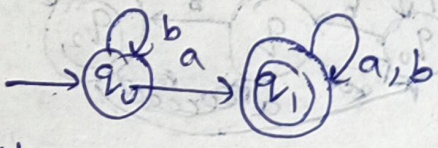
Minimization

1. Remove dead state 2. Remove the ~~dead state~~ (Unreachable State) URS 3. Equal states.

Q)

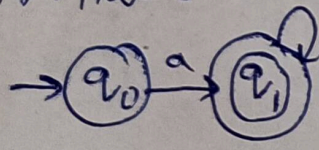


=) Minimized to

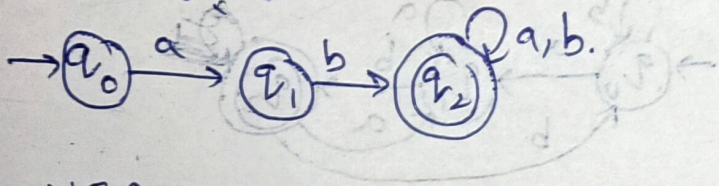


NDFA/NFA (No dead states)

Q) Initial is "a"



Q) Starting with ab



Power of DFA = Power of NFA,

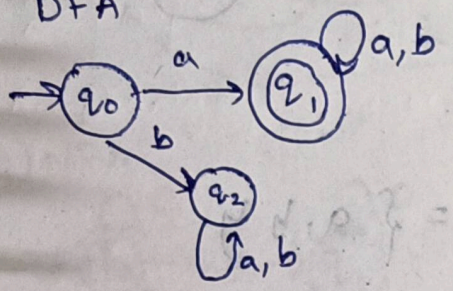
* DFA is complete system & complex structure.

* DFA having 5 tuples. $\{Q, \Sigma, q_0, F, \delta\}$

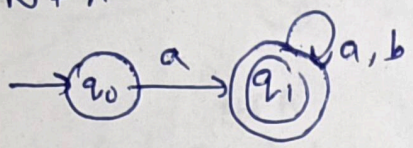
* NFA " " "

Q) Initial 'a'

DFA



NFA



* NFA is not practically implemented.

NFA:- A string 'w' is said to be accepted by an NFA if there exist atleast one transition path on which we ~~start~~ start at initial state & end at the final state.

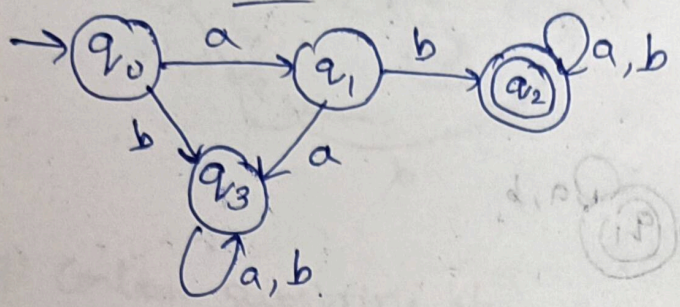
DFA	NDFA
1) For DFA each symbol has only one transition.	1) For NDFA each symbol has multiple transition.
2) DFA cannot use empty string transition (null string)	2) NFA can use empty string transition. (null string)
3) DFA can be understood as a one machine	3) NFA can be understood as multiple machine.

~~Design DFA~~

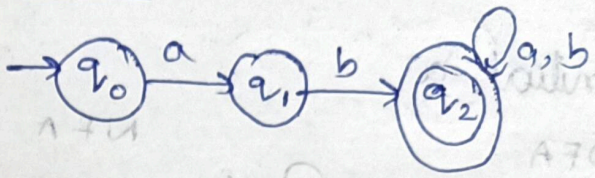
Q) DFA starting with ab. $L = \{a, b\}$

$L = \{ab, aba, abab, \ldots\}$

DFA



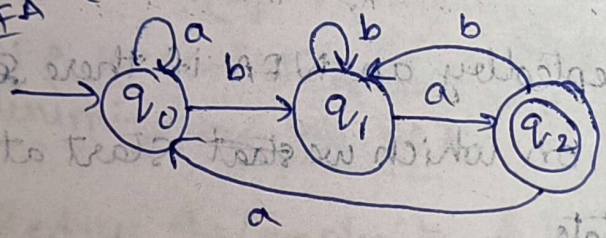
NFA



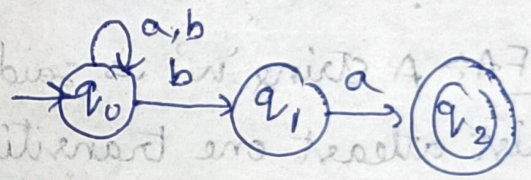
Q) Design DFA & NFA ending with ba. $L = \{a, b\}$

$L = \{ba, aba, bba, abba, \ldots\}$

DFA



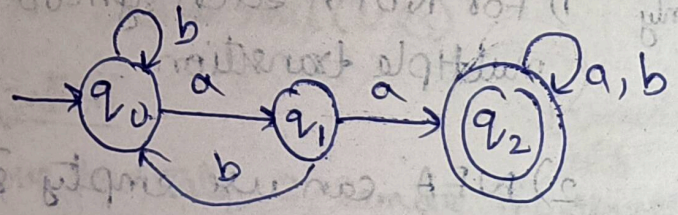
NFA



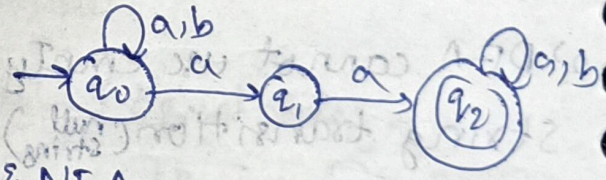
Q) Containing substring aa $L = \{a, b\}$

$L = \{aa, baa, aab, baab \ldots\}$

DFA

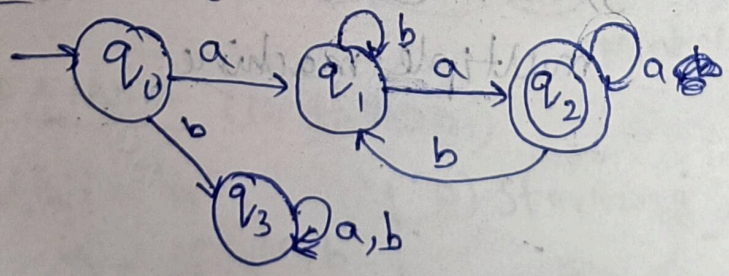


NFA

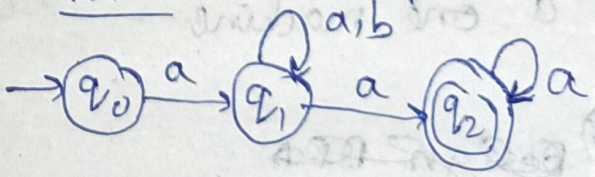


Q) Starting & Ending with a. DFA & NFA

DFA $L = \{a, aa, aba, aabba \ldots\}$

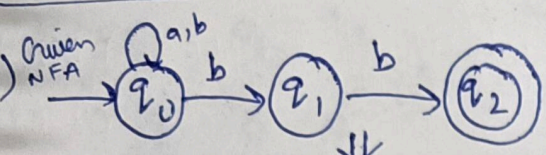


NFA

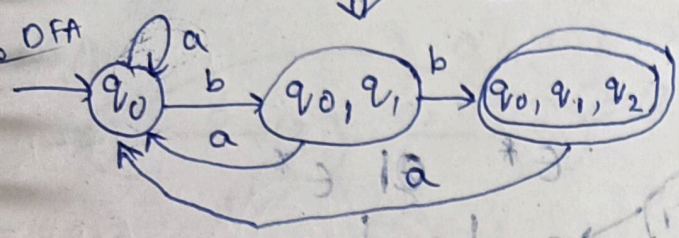


Conversion of NFA to DFA

Q) Given NFA



So DFA



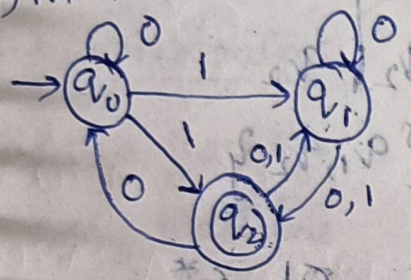
NFA

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

DFA

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

Q) NFA



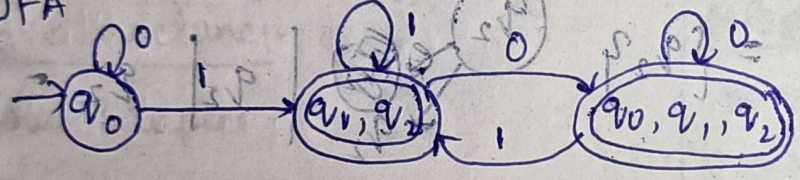
NFA

δ	0	1
$\rightarrow q_0$	q_0	q_1, q_2
q_1	q_1, q_2	q_2
q_2	q_0, q_1	q_1

DFA

δ'	0	1
$\rightarrow q_0$	q_0	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_2, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$

So DFA



ϵ-NFA

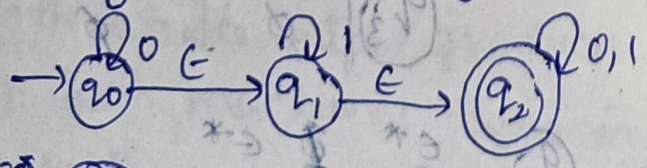
RL $\rightarrow$ ϵ-NFA $\rightarrow$ NFA $\rightarrow$ DFA $\rightarrow$ MDFA

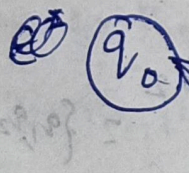
ϵ-NFA (Epsilon NFA)

ϵ^* (Epsilon closure) :- All the states that can be reached from a particular state only by seeing this 'ϵ' symbol.

- Any state that can be reached only by seeing 'ϵ' can be a final state.
- Every state only 'ϵ' goes to itself.

Q)





	ϵ^*	0	ϵ^*
	q_0	q_0	q_0, q_1, q_2
	q_1	ϕ	ϕ
	q_2	q_2	q_2

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

	ϵ^*	1	ϵ^*
q_0	q_0	ϕ	ϕ
	q_1	q_1	q_1, q_2
	q_2	q_2	q_2

$= \{q_1, q_2\}$

	ϵ^*	0	ϵ^*
q_1	q_1	ϕ	ϕ
	q_2	q_2	q_2

$= \{q_2\}$

	ϵ^*	1	ϵ^*
q_1	q_1	q_1	q_1, q_2
	q_2	q_2	q_2

$= \{q_1, q_2\}$

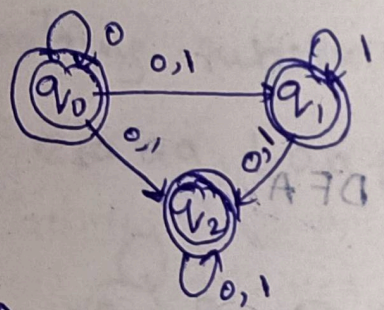
	ϵ^*	0	ϵ^*
q_2	q_2	q_2	q_2

$= \{q_2\}$

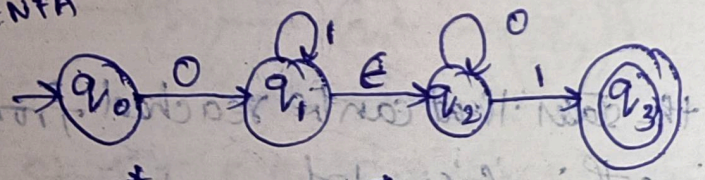
	ϵ^*	1	ϵ^*
q_2	q_2	q_2	q_2

$= \{q_2\}$

So NFA



Q) ENFA



	ϵ^*	0	ϵ^*
q_0	q_0	q_1, q_2	q_1, q_2

$= \{q_1, q_2\}$

	ϵ^*	1	ϵ^*
q_0	ϕ	ϕ	ϕ

$= \{\phi\}$

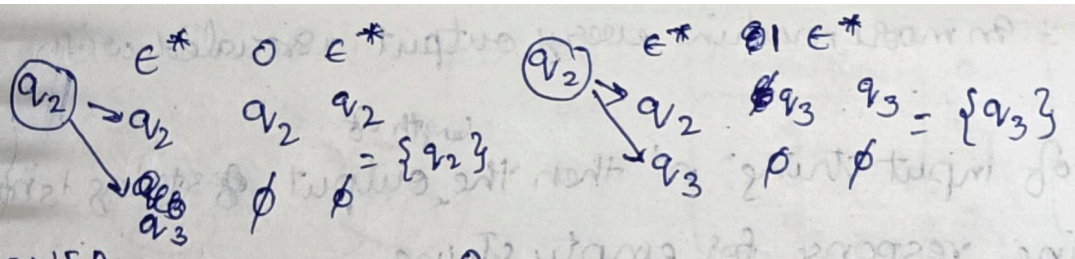
	ϵ^*	0	ϵ^*
q_1	q_1	ϕ	ϕ
	q_2	q_2	q_2

$= \{q_2\}$

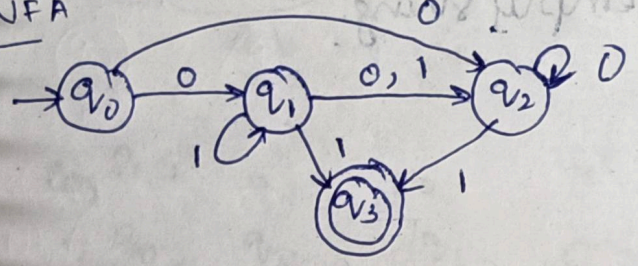
	ϵ^*	1	ϵ^*
q_1	q_1	q_1	q_1, q_2
	q_2	q_3	q_3

$= \{q_1, q_2, q_3\}$

δ	0	1
$\rightarrow q_0$	q_1, q_2	ϕ
q_1	q_2	q_1, q_2, q_3
q_2	q_2	q_3
q_3	ϕ	ϕ



SONFA



FA with outputs

- Moore Machine
- Mealy Machine.

Moore Machine:-

It has 6 tuples.

Q

Σ

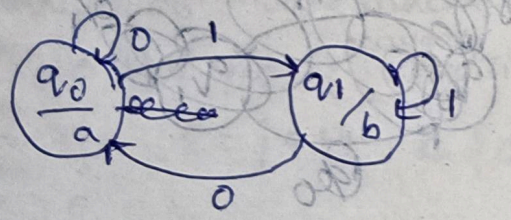
q_0

δ

$\Delta \rightarrow$ Output alphabet

$\lambda \rightarrow$ Output function $\lambda: Q \rightarrow \Delta$

Ex:-



Mealy Machine

It has 6 tuples

Q

Σ

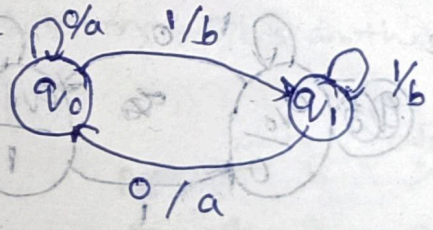
q_0

δ

Δ

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

Ex:-

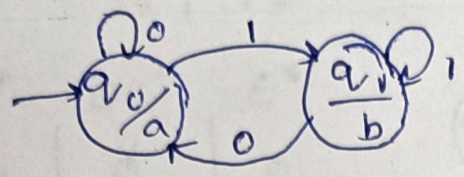


- Both Moore and Mealy machines are special case of DFA.
- Both acts like output producer. rather than language acceptor.
- In both machines there is no need to define final state & also there is no concept of dead state.
- Both Moore & Mealy are machines are equivalent.

Moore Machine :- In moore machine every output associated with the states.

- If the length of input string 'n' then the length of output string is "n+1"
- Moore machine response for empty string.

Eg: 1010



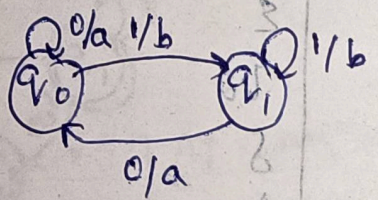
q_0 q_1 q_0 q_1 q_0

a b a b a

Mealy Machine :-

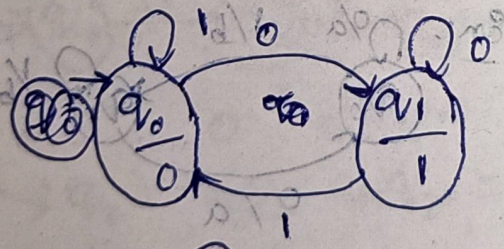
- ~~If the length of~~
- If the length of input string 'n' then the length of output string is "n"
- It doesnot response for empty string.

Eg:- 1010

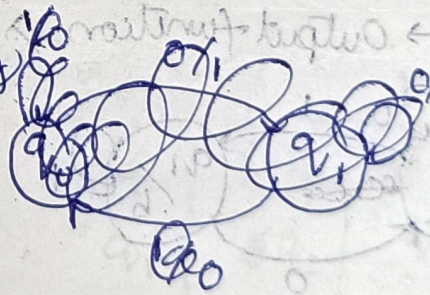


Q) Produce 1's complement

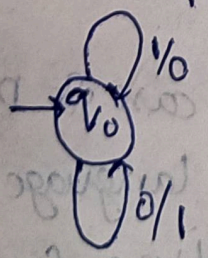
Moore



Mealy

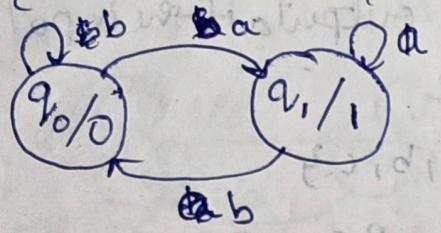


Mealy



Q) Construct a moore machine that take all the strings as a & b as input and count the no. of a's in the output string interm of 1.

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

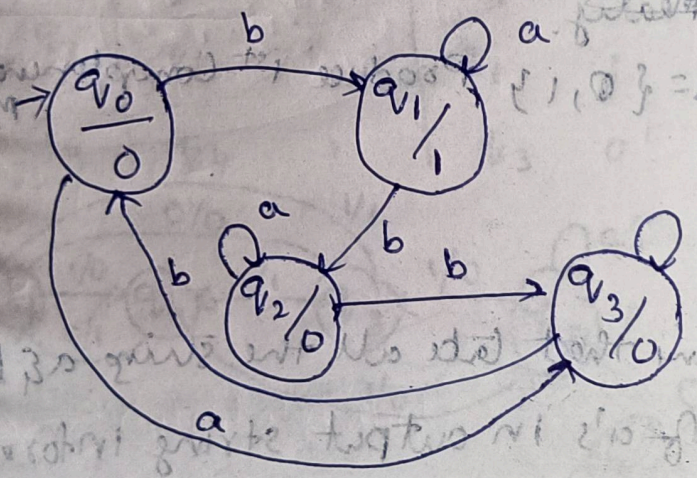


a = 1
b = 0

Present state	a	b	λ
q_0	q_1	q_0	0
q_1	q_1	q_0	1

Q)

P.S	a	b	λ
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

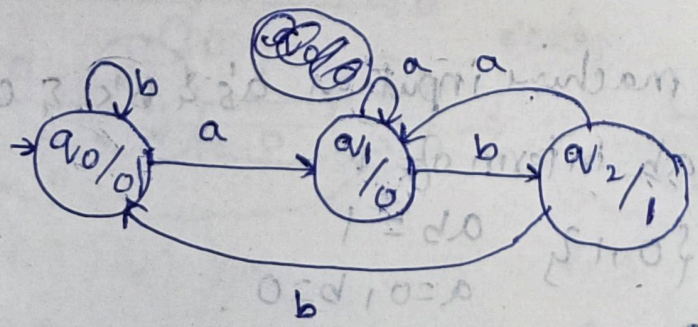


Q) Construct of Moore machine taking all the string a's & b's as the input & count the no. of occurance substring ab in term of 1.

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

a = 0
b = 0

P.S	Next state a	Next state b	output λ
q_0	q_1	q_0	0
q_1	q_1	q_2	0
q_2	q_1	q_0	1

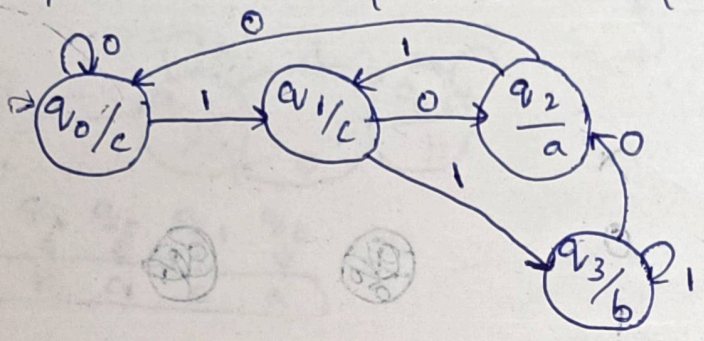


a a b a
0 0 0 1 0

b a b b a
0 0 0 1 0 0 0

Q) Construct a moore machine input are {0,1} and output alphabet are Δ = {a,b,c}. Machine should give output a if the input string ending with 10 ~~ab~~, 11 – b otherwise ~~b~~ c

10 – a , 11 – b Σ = {0,1} Δ = {a,b,c}

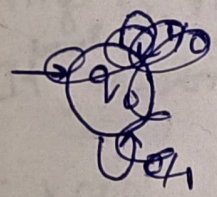


P.S	0	1	λ
q_0	q_0	q_1	c
q_1	q_2	q_3	c
q_2	q_0	q_1	a
q_3	q_2	q_3	b

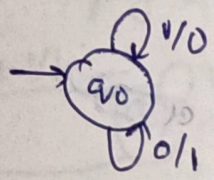
<u>Mealy Machine</u>:- output is assosiated on transision ~~state~~.

- If length of input string is 'n' the o/p string also 'n'.
- It does not respond to empty string.

~~Q) Produce 1st complement in Mealy~~

Q) 

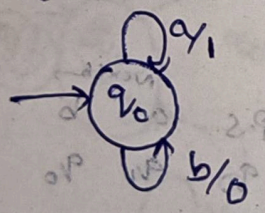
Σ = {0,1} Δ = {0,1} Produce 1st Complement in Mealy



Q) Construct a mealy machine that take all the string a & b as input and count the no. of a's in output string in form of 1.

Σ = {a,b} Δ = {0,1} a = 1, b = 0

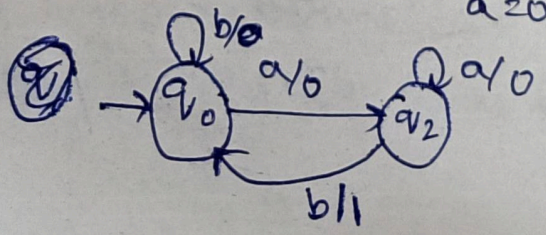
Sol:-



Q) Construct a mealy machine input as a's & b's & count no. of occurance substring ab in term of 1

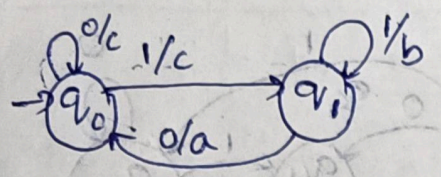
Σ = {a,b} Δ = {0,1} ab = 1

a = 0, b = 0



Q) Construct a mealy machine input $\Sigma = \{0,1\}$ & $\Delta = \{a,b,c\}$

10 = a , 11 – b otherwise c.



P.S	0	λ	1	λ
q_0	q_0	c	q_1	c
q_1	q_0	a	q_1	b

0 1 1 0 0

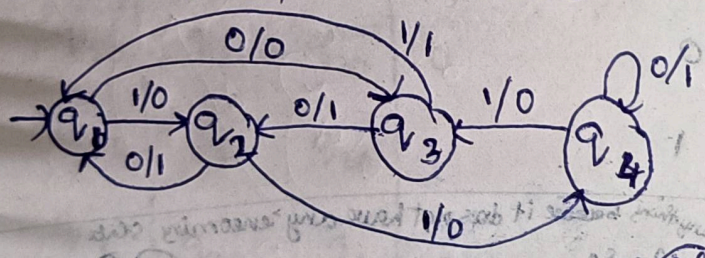
q_0 q_0 q_2 q_2 q_0 q_0

c c b a c

P.S	Next state 0	λ	Next state 1	λ
q_0	q_0	c	q_1	c
q_1	q_0	a	q_1	b

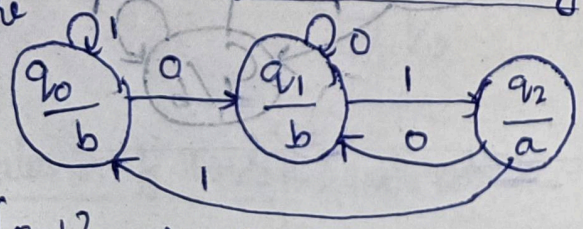
Q) Next state

P.S	a=0 state	O/P	a=1 state	O/P
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

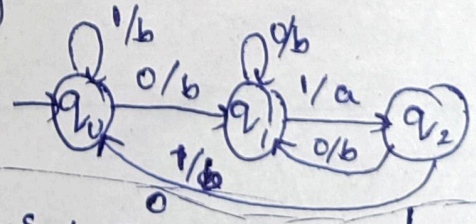


Conversion of Moore → Mealy.

Q) Moore



→ Mealy



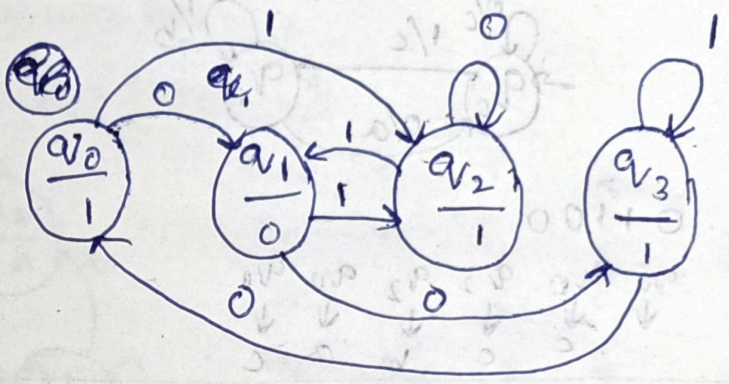
$\Sigma = \{0,1\}$ $\Delta = \{a,b\}$.

P.S	0	1	λ
q_0	q_1	q_0	b
q_1	q_1	q_2	b
q_2	q_1	q_0	a

P.S	0	λ	1	λ
q_0	q_1	b	q_0	b
q_1	q_1	b	q_1	a
q_2	q_1	b	q_0	b

Q) Moore Table

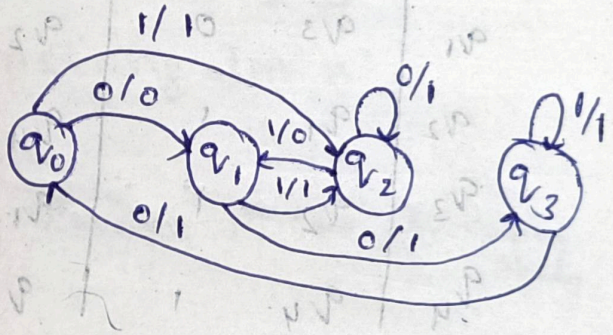
P.S	0	1	λ
q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1



(we can convert to moore table to mealy table without drawing transition diagram)

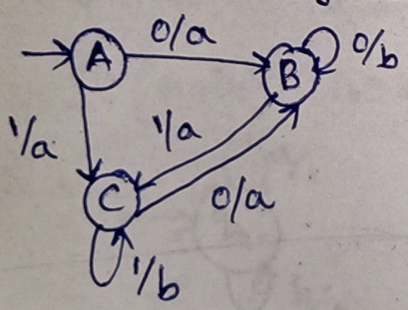
Mealy Table

P.S	0	λ	1	λ
q_0	q_1	0	q_2	1
q_1	q_3	1	q_2	1
q_2	q_2	1	q_1	0
q_3	q_0	1	q_3	1



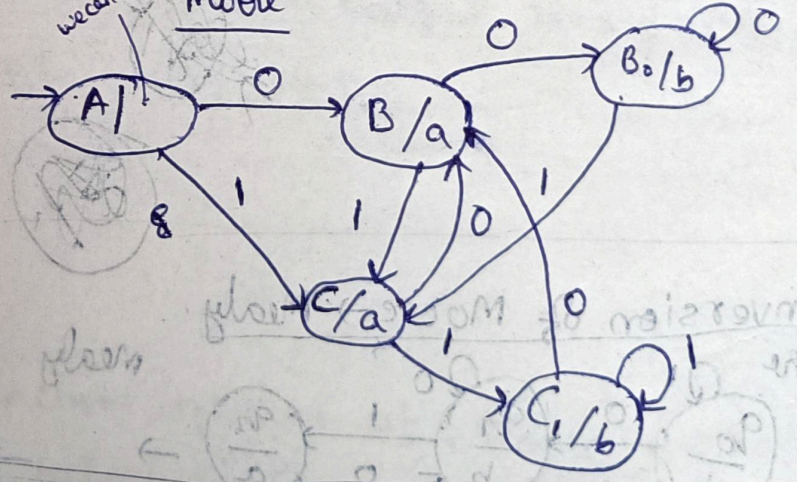
Conversion Mealy to Moore.

Q)



we can write anything because it does not have any incoming state

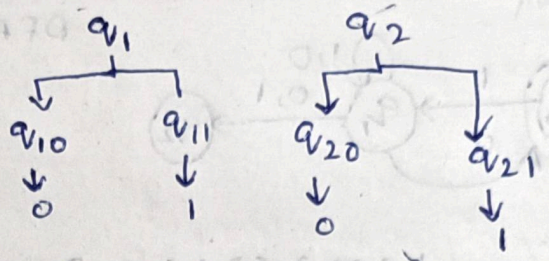
Moore



Q) Mealy table

P.S	a	λ	b	λ
→ 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

Moore table $q_3 \to 0$, $q_0 \to 1$



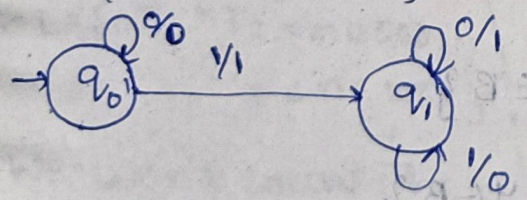
P.S	a	b	λ
→ q_0	q_3, 0	q_{11}, 1	1
q_{10}	q_0, 1	q_3, 0	0
q_{11}	q_0, 1	q_3, 0	1
q_{20}	q_{21}, 1	q_{20}, 0	0
q_{21}	q_{21}, 1	q_{20}, 0	1
q_3	q_{10}, 0	q_0, 1	0

Moore table

P.S	a	b	λ
q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

Q) Design Mealy machine ~~gives~~ 1's complement to 2's complement.

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

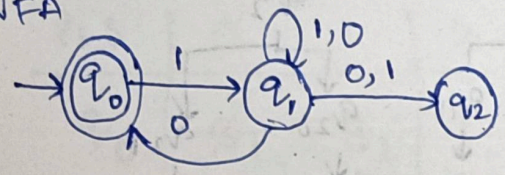


Equivalence of Finite Automata (FA):-

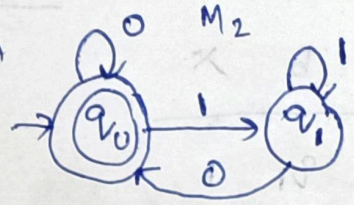
- Two FA M_1 & M_2 are said to be Equivalent if they both accept the same language.

Ex:-

Q) NFA M_1

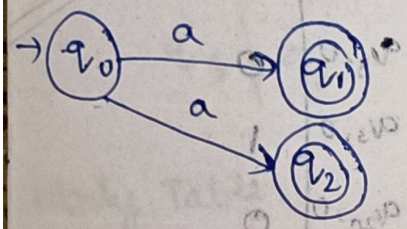


DFA M_2

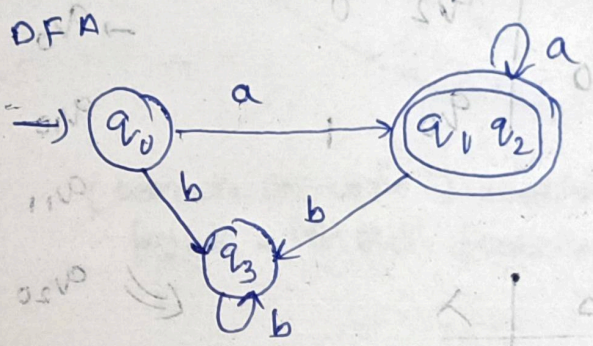


$L(M_1) \Leftrightarrow L(M_2)$

Q) NFA



DFA



Regular Language (RL):-

- A language is said to be a regular language if & only if some finite state machine recognizes it.
- Not regular language are those languages which is not recognized by any finite state machine.
- In RL it require memory.
- RL cannot store the count of the string

Operation on R.L

~~Union U For~~

Union. $\cup$ $\quad A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$

Concatenation $\quad A \cdot B = \{ xy \mid x \in A \text{ and } y \in B \}$

Star or kree closure $\quad A^* = \{ x_1, x_2, x_3, \ldots x_k \mid k \geq 0 \text{ \& each } x_i \in A \}$

i.e. ϵ, A, AA, AAA.

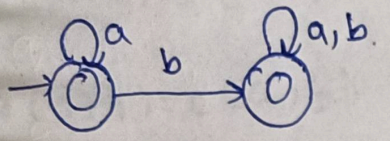
Q) $A = \{ pq, r \}$ $B = \{ t, uv \}$

$A \cup B = \{ pq, r, t, uv \}$ $\quad A \cdot B = \{ pqt, pquv, rt, ruv \}$

$A^* = \epsilon, pq, r, pqr, rpq, pqpq, r \ldots$

$B^* = \epsilon, t, uv, tuv, uvt, \ldots$

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



Language

- Empty ϕ → RL
- Non-empty $\neq \phi$
  - finite → R.L
  - Non-finite → It may or may not

RL	NR
1) FSM	1) Not FSM
2) Finite	2) May or may not infinite
3) Not require Memory	3) Memory
4) Comparision or finite Ex:- $L = \{a^m, b^m \mid 1 \leq m \leq 500, 1 \leq n \leq 1000\}$	4) Comparision & infinite Ex:- $L = \{a^m b^n \mid m > n, n \geq 0\}$
5) We won't count the string.	5) We count the string.