__/__/____

Theory of Computation

Unit 1 -

* Automata Theory.

Automata Theory is the study of abstract computing devices or "machines"

The study of abstract 'mathematical' machines or systems and the computational problems that can be solved using these machines.

→

	Formal Language	Natural Language.
1]		1]
2]		2]
3]		3]

* Finite Automata.

Finite no. of states.

No memory to store intermediate results.

Limited Power.

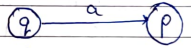
Notations - 1] Transition Table

2] Transition Diagram.

→ Transition Diagram

FA (Q, Σ, q0, F, δ)

- For each state there is a node.
- For each state q in Q & for input a ∈ Σ, let δ(q,a) = p



- There is arrow in the start state (q0)
- Final states marked by double circle (qf)

→ Transition Table

- Transition table is a conventional tabular representation of a function like δ(delta) and takes two arguments and return a state

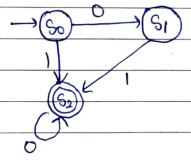
Eg

	0	1
q0	q2	q0
q1	q1	q1
q2	q2	q1

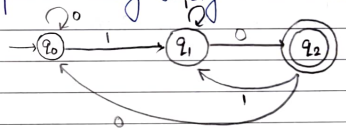
* There are 2 types of Finite Automata.
1] Deterministic FA
2] Non Deterministic FA.

→ Deterministic FA [Q × Σ → Q]

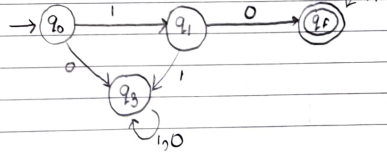
A FA is said to be deterministic, if for every state there is a unique input symbol, which takes the state to reqd next unique state.



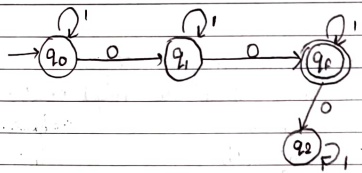
Q] Accept a binary string that ends with '10'



Q] Construct a FA to recognise language containing strings starting with '10'



Q] Construct a FA to recognize lang of strings with exactly two 0's anywhere.

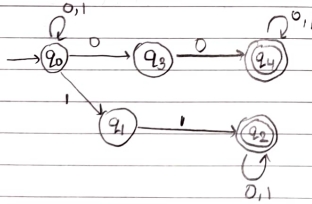


→ Non Deterministic FA $[Q \times \Sigma \longrightarrow 2^Q]$

NFA has the power to be in several states at once.
Each NFA accepts a language that is also accepted by some DFA
NFAs are easier than DFAs.

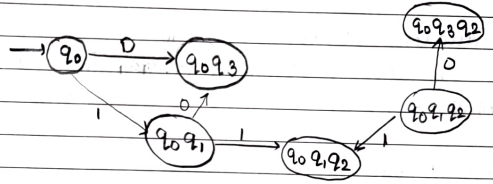
Q. $Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{0, 1\}$
$F = \{q_2, q_4\}$.

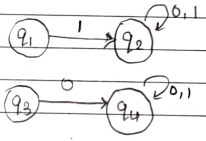
	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\{\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\{\}$
q_4	$\{q_4\}$	$\{q_4\}$



NFA

Conversion to DFA.





* DFA Minimization.

1] Equivalent States -
go to same next state on reading the same input symbol.

2] Unreachable States -
Cannot be reached from initial state on reading any input symbol.

3] Dead (or Trap) States -
Have no outgoing transitions, except to themselves.

Eg-

	0	1
p	pq	p
q	r	r
r	s	-
s	s	s

→

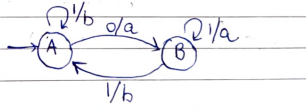
		0	1
1]	p	pq (5)	p (1)
2]	q	r (3)	r (3)
3]	r	s (4)	-
* 4]	s	s (4)	s (4)
5]	pq	pqr (11)	pr (6)
6]	pr	pqs (12)	p (1)
* 7]	ps	pqs (12)	ps (7)
8]	qr	rs (10)	r (3)
* 9]	qs	rs (10)	rs (10)
* 10]	rs	s (4)	s (4)
11]	pqr	pqrs (15)	pr (6)
* 12]	pqs	pqrs (15)	prs (13)
* 13]	prs	pqrs (15)	ps (7)
* 14]	qrs	pqrs (15)	rs (10)
* 15]	pqrs	pqrs (15)	prs (13)

Minimized DFA.

	0	1
1	5	1
2	3	3
3	4	-
* 4	4	4
5	11	6
6	7	1
* 7	7	7
8	4	3
11	7	6

* Finite Automata Machines.

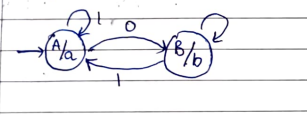
Mealy Machine
$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$



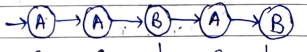
Eg 1 0 1 0

inp out.
n = n

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



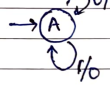
Eg 1 0 1 0



inp. out
n = n+1

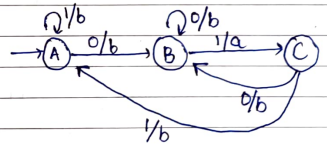
* Construction of Mealy Machine.

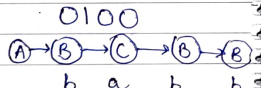
Q] Construct a Mealy Machine that produces the 1's complement of any binary inp. string.



Q] Construct a Mealy Machine prints 'a' whenever the sequence '01' is encountered in any binary string.

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

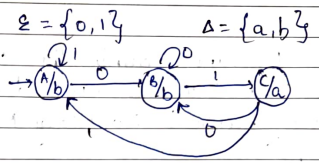


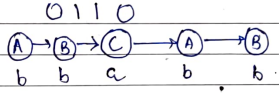


* Construction of Moore Machine.

Q] Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any binary string.

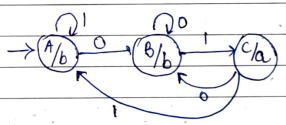
$\Sigma = \{0,1\}$ $\Delta = \{a,b\}$





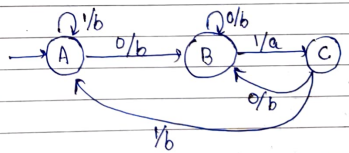
* Conversion of Moore Machine to Mealy Machine.

Moore



	0	1	out
A	B	A	b
B	B	C	b
C	B	A	a

Mealy

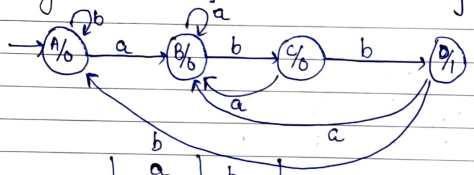


	0	1	
A	B, b	A, b	b
B	B, b	C, a	b
C	B, b	A, b	a

Moore to Mealy → same no. states
Mealy to Moore → no fix no.

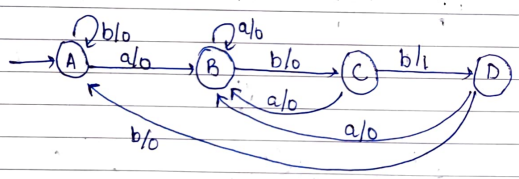
Q] The given Moore Machine counts the occurrences of the sequence 'abb' in any input binary string over {a,b}. Convert to Mealy Machine.

Moore



	a	b	
A	B	A	0
B	B	C	0
C	B	D	0
D	B	A	1

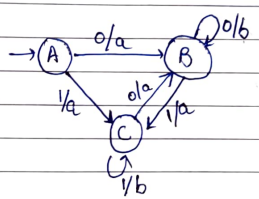
Mealy



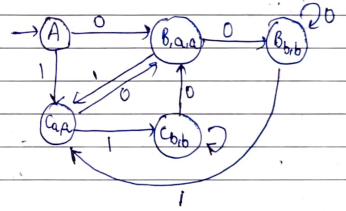
	a	b	
A	B,0	A,0	0
B	B,0	C,0	0
C	B,0	D,1	0
D	B,0	A,0	1

* Conversion of Mealy to Moore Machine (Using transition graph)

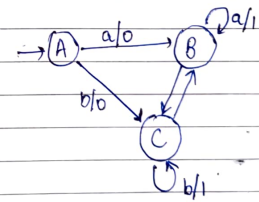
Q]

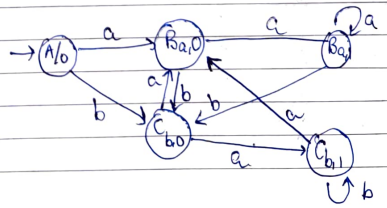


$\Sigma = \{0,1\}$
$\Delta = \{a,b\}$.

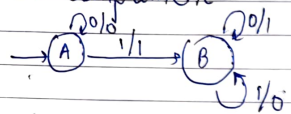


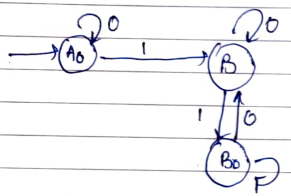
Q]





3] 2's compliment





(Using transition table)

Mealy

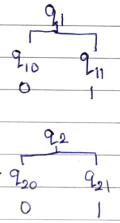
	a	b	
q_0	$q_3, 0$	$q_1, 1$	1
q_1	$q_0, 1$	$q_3, 0$	
q_2	$q_2, 1$	$q_2, 0$	
q_3	$q_1, 0$	$q_0, 1$	0

Moore

	a	b	
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

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

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

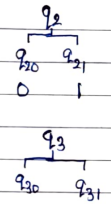


Questions

* Mealy to Moore Machine Problems

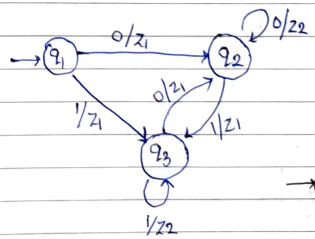
Prob2

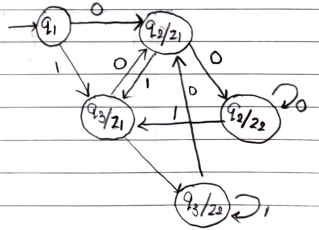
	0	1
q_1	$q_1, 1$	$q_2, 0$
q_2	$q_4, 1$	$q_4, 1$
q_3	$q_2, 1$	$q_3, 1$
q_4	$q_3, 0$	$q_1, 1$



	0	1	
q_1	$q_1, 1$	$q_{20}, 0$	1
q_{20}	$q_{41}, 1$	$q_{41}, 1$	0
q_{21}	$q_{41}, 1$	$q_{41}, 1$	1
q_{30}	$q_{21}, 1$	$q_{31}, 1$	0
q_{31}	$q_{21}, 1$	$q_{31}, 1$	1
q_4	$q_{30}, 0$	$q_1, 1$	1

Prob 1 -

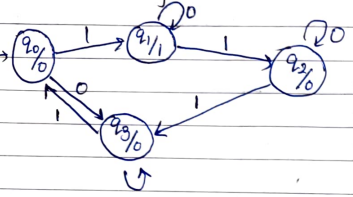




* Moore to Mealy Machine

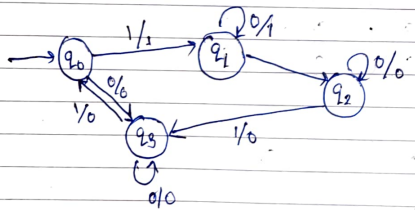
Prob-

Moore



	0	1
q_0	$q_3, 0$	$q_1, 1$
q_1	$q_1, 1$	$q_2, 0$
q_2	$q_2, 0$	$q_3, 0$
q_3	$q_3, 0$	$q_0, 0$

Mealy

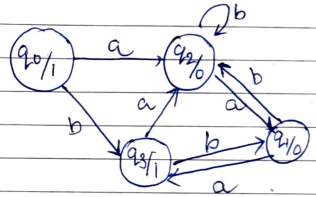


	0	1	
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

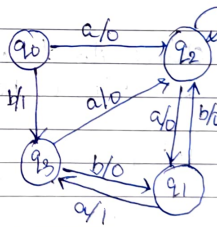
Prob-

	a	b	
q_0	q_2	q_3	1
q_1	q_3	q_2	0
q_2	q_1	q_2	0
q_3	q_2	q	1

Moore

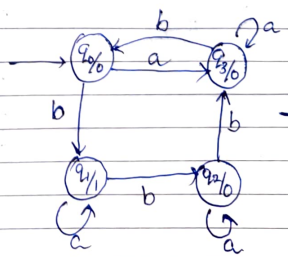


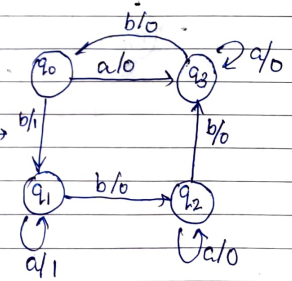
Mealy



	a	b
q_0	$q_2, 0$	$q_3, 1$
q_1	$q_3, 1$	$q_2, 0$
q_2	$q_1, 0$	$q_2, 0$
q_3	$q_2, 0$	$q_1, 0$

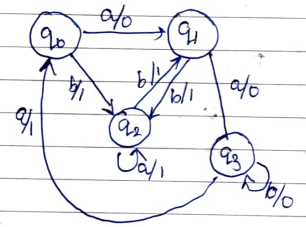
Prob -





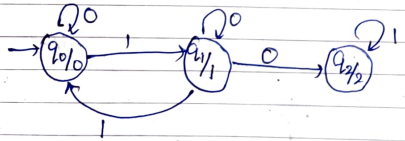
Prob -

	a	b	
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	0



Prob- Design a moore machine to design residue modulo 3 for binary no.

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



0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

* NFA with ε- Transitions

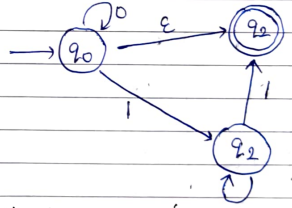
5 tuple:

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

Q : Finite set of states
Σ : Finite input alphabet
δ - STF that maps $Q \times (\Sigma \cup \{\varepsilon\})$ to 2^Q

$Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q$

eg

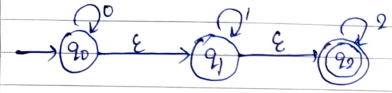


Indirect conversion method

Prob- Convert the NFA with ε- Transitions to its equivalent DFA.

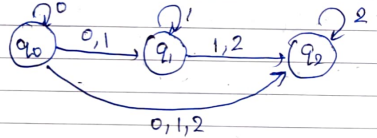
	0	1	2	ε
q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

Direct conversion method.

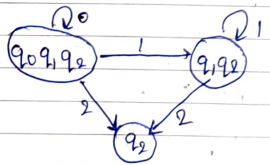


NFA

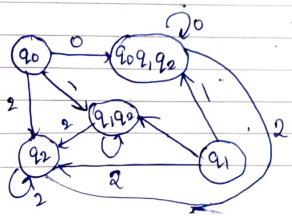
	0	1	2
q_0	$\{q_0 q_1 q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$.



	0	1	2
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$	q_2
$q_1 q_2$	ϕ	$q_1 q_2$	q_2



DFA



__/__/____

* Applications of FA

1] Software for designing and checking behaviour of digital circuits
2] The "lexical Analyzer" of a compiler.
3] Software for scanning large bodies of Text
4] Used in Text Editor and Spell checkers

* limitations of FA.

1] Cannot recognize Palindrom of string.
2] Set of strings over "(" and ")" of have balanced parenthesis.
3] No memory, Only memory it has is state of State.