Date
Page

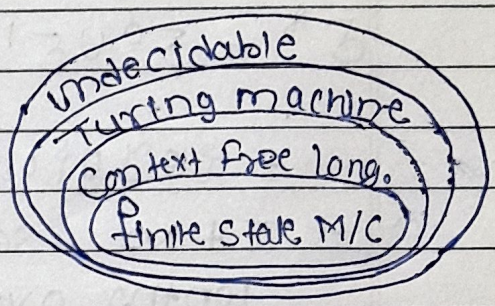
Theory of Computation → mathematical part of CSE

Step by step process to execute a given task

TOC is mainly about what kind of things can you really compute mechanically, how fast and how much space does it take to do so

FSM → finite state Machine

CFL → context free language
set of strings



Automata is abstract model of computational Devices.

finite State Machine (prerequisites)

- Symbol → a, b, c, 0, 1, 2, 3 (Smallest Building Block)
- Alphabet (Σ) → A collection of symbols. Ex:- {a, b}, {0, 1, 2}
- String → Sequence of symbols. Ex:- a, b, 0, 1, aa, bb,
- Language - Set of strings Ex:- Σ = {0, 1}

L_1 = Set of all strings of length 2

finite — L_1 = {00, 01, 10, 11}

L_2 = Set of all strings of length 3
= {000, 001, 010, 011, 100, 101, 110, 111}

L_3 = Set of all strings that begin with 0
= {0, 00, 01, 000, 001, 010, 011, 0000 ....}

infinite

Powers of Sigma (Σ): Σ = {0, 1}

Σ^0 = Set of all strings of length 0; Σ^0 = {ε}
Σ^1 = Set of all strings of length 1; Σ^1 = {0, 1}
Σ^2 = Set of all string of length 2; Σ^2 = {00, 01, 10, 11}
Σ^3 = ' ' ' ' ' ' 3; Σ^3 = {000, 001, 010, 011, 100, 101, 110, 111}

Fuzzy Logic → for Realife average accuracy
Propositional Logic → for Ideal 99.99% accuracy

Σ^n = Set of all strings of length n.

Cardinality - No. of Elements in a Set.

↳ $\Sigma^n = 2^n$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots$ (Kleen Star)
or
Σ closure

$= \{\epsilon\} \cup \{0,1\} \cup \{00,01,10,11\} \cup \ldots\ldots$

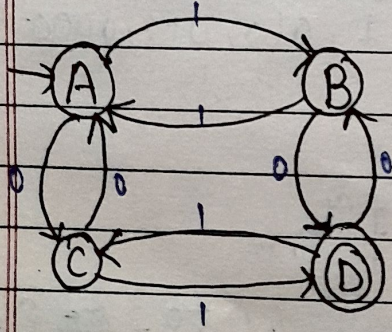
= It is a set of all possible strings of all lengths over $\{0,1\}$ → (Infinite set)

Finite State Machines

also known as. **Finite Automata**

- With o/p
  - Moore m/c
  - Mealy m/c
- Without o/p
  - DFA
  - NFA
  - ∈NFA

- DFA → Deterministic Finite Automata
  - → It is the simplest model of computation
  - → It has very limited memory



① States :- These circles represented by A, B, C, D are known as states

② Edge :- These 0, 1 lines (Edges) are transition,

When I in state (A) gets input of (1), So my state (A) goes from (A) to (B)

- Memory of FSM is very limited
- It cannot store or count strings

Date
Page

③ Initial State:- Whenever you see arrow coming from nowhere pointing to a state, i.e., the Initial state or the starting state

④ final state:- Whenever you see a double circle i.e., the terminating means final state

Every DFA Defined using these follows Tuples

a) Q = Set of all states.
b) Σ = inputs.
c) q_0 = Start State / Initial state
d) F = set of final states.
e) δ = Transition func^n from $Q \times \Sigma \to Q$

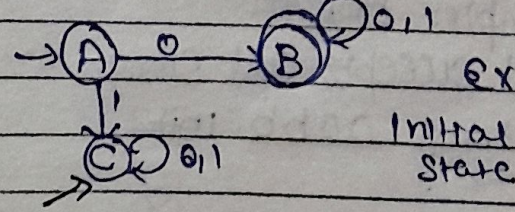
• Tuples values for above DFA.

Q = {A, B, C, D}
Σ = {0, 1}
q_0 = A
F = {D}
δ =

	0	1
A	C	B
B	D	A
C	A	D
D	B	C

Deterministic finite Automata Ex:-

L1 = Set of all strings that start with '0'
= { 0, 00, 01, 000, 010, ... }



Dead State or Trap State

Ex:- 001
Initial State A → B → B → B final state
0 0 1

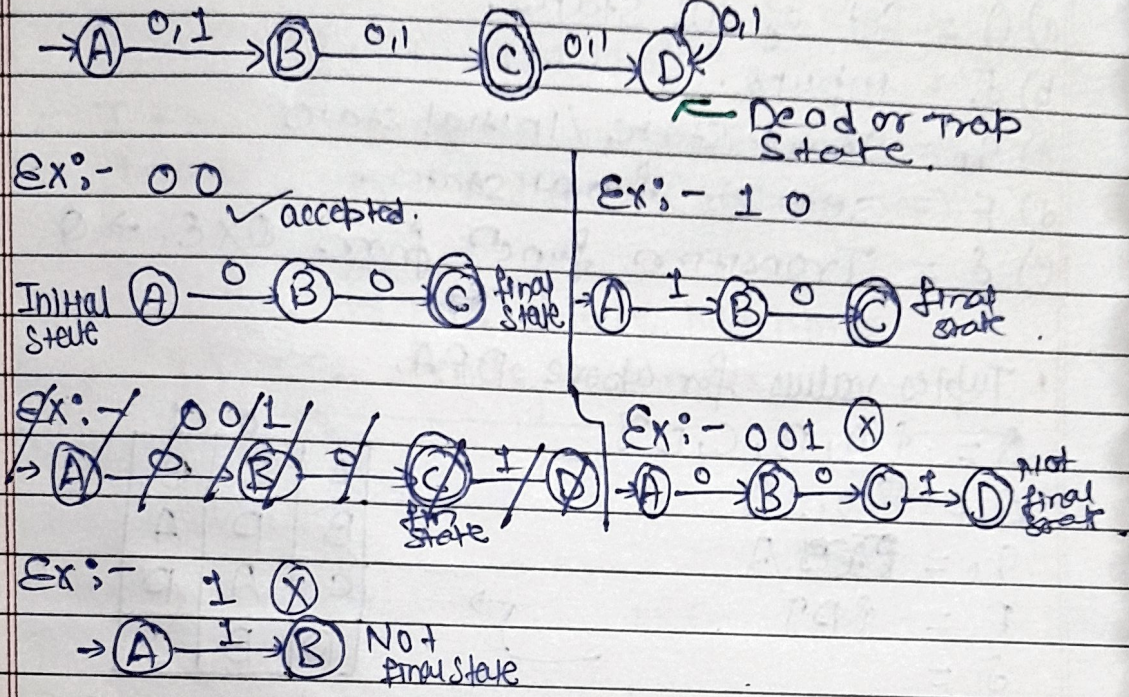
Ex:- 10 1 ✗
Initial State A → C → C → C
1 0 1
Not final state.

Date
Page

Q Construct a DFA that accepts sets of all strings over {0,1} of length 2.

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

$L = \{00, 01, 10, 11\}$



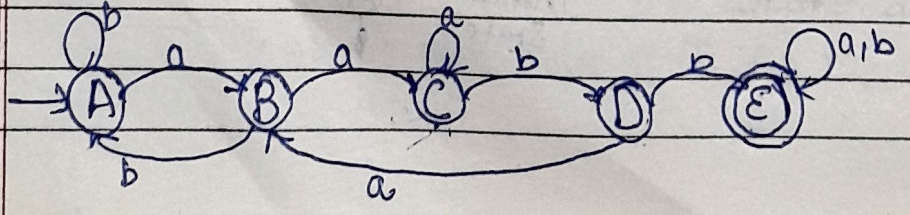
Q. Construct a DFA that accepts any strings over {a,b} that does not contain the string aabb

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

① Try to design a simpler problem

Let us construct DFA that accepts all strings over {a,b} that contains the string aabb init

$L_1 = \{aabb, baabb, baabaabb \cdots\}$

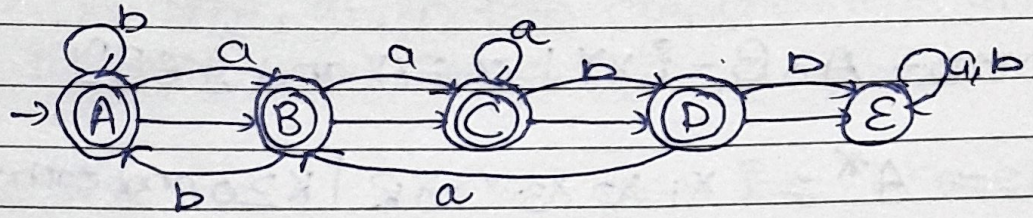


Date
Page

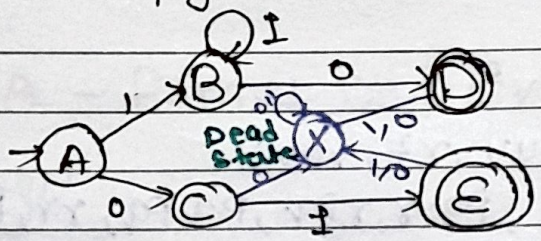
Now flip the States

① => make the final state into non-final state

② => Make the non-final states into final states.



How to figure out what a DFA recognizes?



10 ✓

1 1 1 1 ... 0 ✓
A B D

01 ✓

L = {Accepts the string 01 or a string of at least '1' followed by a '0'}

Ex:- 001, 010, 011, 1101, 1100 (All falls to Dead state in end)

x x x x x

x → Dead state

Regular Languages

A language is said to be a Regular language if and only if some finite state Machine recognizes it

The languages
» Which are not recognized by any FSM
» Which require memory

Ex:- abababbabbabb ; $0^N 1^N$

FSM can't keep this in memory. FSM can't keep count of no. of 0's

Date
Page

operations on Regular Languages

Union – $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$ (such that)

Concatenation – $A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$

STAR – $A^* = \{ x_1 x_2 x_3 \cdots x_k \mid k \geq 0 \text{ and each } x \in A \}$

Ex:- $A = \{ p, q, r \}$, $B = \{ t, uv \}$.

$A \cup B = \{ p, q, r, t, uv \}$

$A \circ B = \{ pt, puv, rt, ruv \}$

$A^* = \{ \epsilon, p, q, r, pq, rr, rp, pqr, pqpq, rr, pqpq, pq, rr \ldots \}$

Theorem 1: The class of Regular languages is closed under UNION.

Theorem 2: The class of Regular languages is closed under concatenation.

NFA – Non-deterministic Finite Automata

Deterministic Finite Automata

Determinism

→ In DFA, given the current state we know what the next state will be.

~~→ It has~~

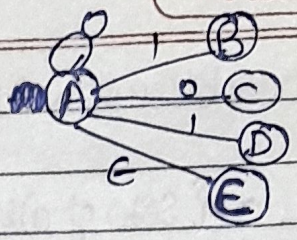
→ It has only one unique next state.

→ It has no choice or randomness.

→ It is simple and easy to design

Date
Page

Non-deterministic Finite Automata

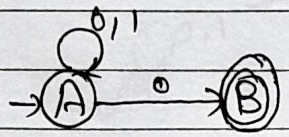


Non Determinism

→ In NFA, given the current state there could be multiple next states.

→ The next state may be chosen at random
or
→ All the next states may be chosen in parallel

NFA - Formal Definition



L = {set of all strings that end with 0}

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

- Q = Set of all States — {A, B}
- Σ = Inputs — {0, 1}
- q_0 = Start State / Initial State — A
- F = Set of final States — {B}
- $\delta = Q \times \Sigma \rightarrow 2^Q$

another Ex:-
3 States A, B, C
A^1 = ϕ, A, B, C
AB, AC, BC, ABC

from the above given ex:-

A x 0 → A
A x 0 → B
A x 1 → A
B x 0 → ϕ
B x 1 → ϕ

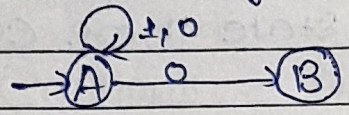
∴ A can go to ⇒ A, B, AB, ϕ
Same goes for B

2^2

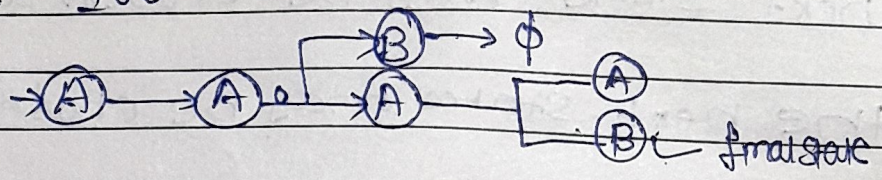
So we can say that:
When we have
2 states ⇒ 4 possibilities.
3 states ⇒ 8 possibilities
∴ 2^{states} ⇒ no. of possibilities.

Q. Create an NFA whose all strings end with 0

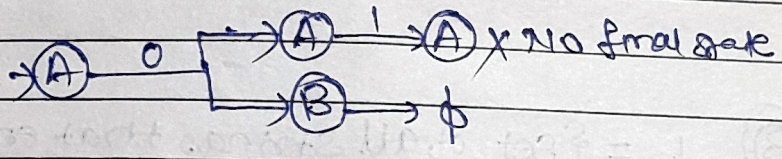
L = {Set of all strings that end with 0}



Ex:- 100



Ex:- 01

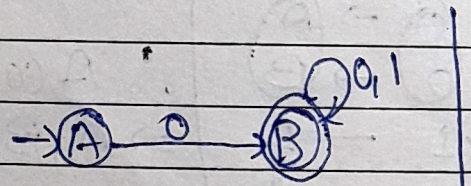


How can we formally write down when a string will be accepted

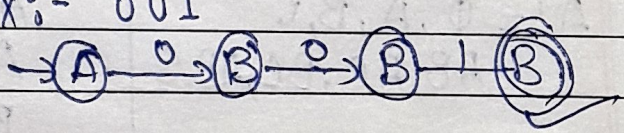
If there is any way to run the machine that ends in any set of states out of which atleast one state is a final state, then the NFA accepts

Q. L = {Set of all strings that starts with 0}

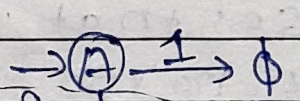
{0, 00, 01, 000...}



Ex:- 001

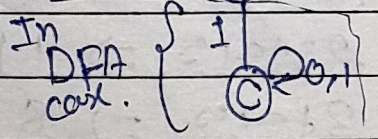


Ex:- 101



Dead configuration

In DFA case.

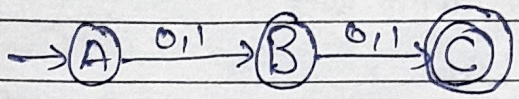


Date
Page

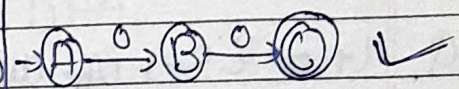
Q Construct a NFA that accepts sets of all strings over {0,1} of length 2.

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

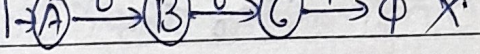
$L = \{00, 01, 10, 11\}$



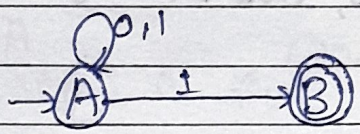
Ex:- 00



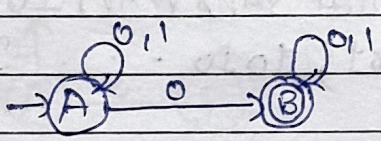
Ex:- 001



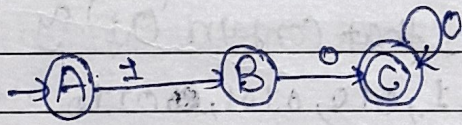
Q. L_1 = {set of all strings that ends with '1'}



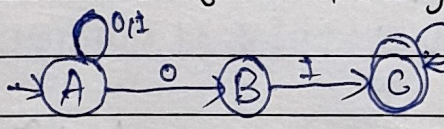
Q. L_2 = {set of all strings that contain '0'}



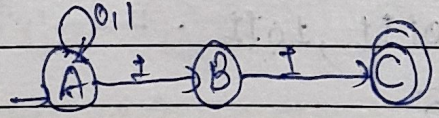
Q. L_3 = {set of all strings that starts with '10'}



Q L_4 = {set of all strings that contain '01'}



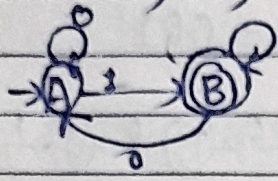
Q. L_5 = {set of all strings that ends with '11'}



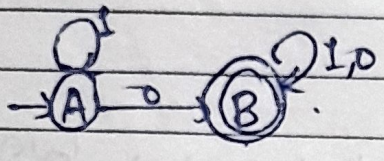
HW

Construct Equivalent DFAs for the L1 to L5 Then tell how many minimum no. of states you would use for the construction of each of DFAs.

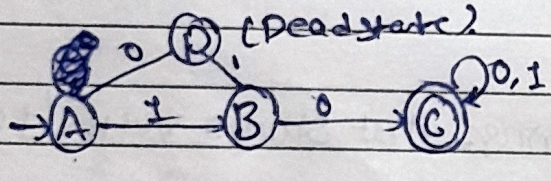
Q. L1 = {Set of all Strings that ends with '1'}
= {1, 01, 11, 001, 0101 ... }.



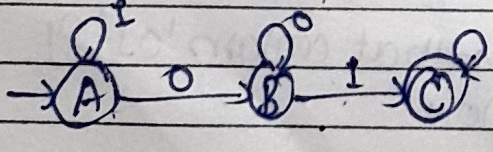
Q. L2 = {set of all Strings that contain '0'}
= { 0, 01, 10, 110, 101 ... }



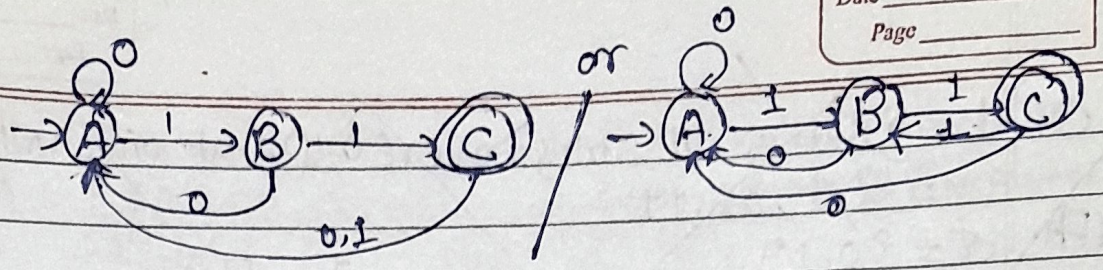
Q. L3 = {Set of all strings that starts with '10'}
= {10, 100, 101, 1001, 1010 ... }



Q. L4 = {Set of all strings that contain '01'}
= {01, 101, 001, 011, 010, 0101, 0110 ... }



Q. L5 = {Set of all Strings that ends with '11'}
= {11, 011, 111, 0111, 1011 ...

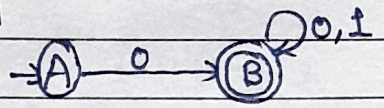


Conversion of NFA to DFA

Every DFA is an NFA, but not vice versa
But there is an equivalent DFA for every NFA

L = { set of all strings over (0,1) that starts with '0' }
= { 0, 01, 001, 010 ... }
Σ = {0,1}

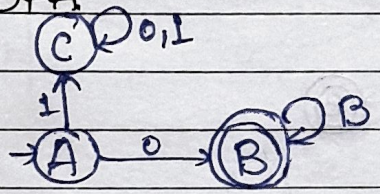
NFA



Transition Table

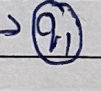
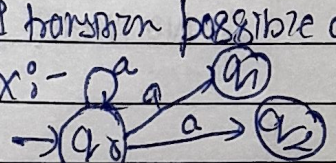
	0	1
A	B	ϕ
B	B	B

DFA



	0	1
A	B	C
B	B	B
C	C	C

C - Dead state

DFA	NFA
a) It has 5 components they are Q, Σ, δ, q_0, F	It also has 5 components they are Q, Σ, δ, q_0, F
b) Transition func^n (δ) = Q × Σ ⇒ Q	Transition func^n (δ) = Q × Σ ⇒ 2^Q
c) All states must be defined at all I/p	Not compulsory to define all I/p
d) Only 1 Transition possible on single I/p. Ex:- 	more than 1 transition possible on single I/p. Ex:- 
e) Null Transition (ϵ) not allowed	Null Transition (ϵ) allowed

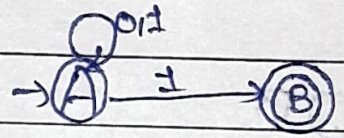
Q. L = {set of all strings over (0,1) that ends with '1'}.

L = {1, 01, 11 ...

Σ = 0 1

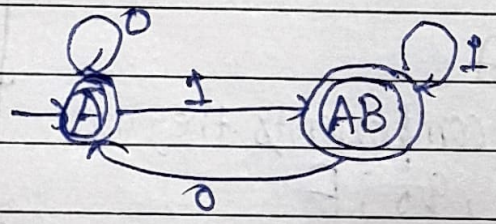
NFA.

	0	1
A	{A}	{AB}
B	ϕ	ϕ



DFA.

	0	1
A	{A}	{AB}
AB	{A}	{AB}



Note:-

AB - is now single state

∴ we can't see B state in making transition table

∴ No need of B state.

Above method is called as Subset construction method

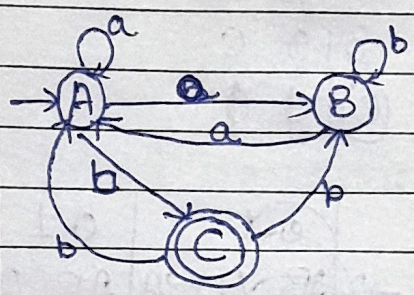
Date
Page

Q Find the equivalent DFA for the NFA given by $M = [\{A,B,C\}, \{a,b\}, \delta, A, [C]]$ where δ is given by:

Q (States) Σ (I/P) Transition Table Initial state Final state

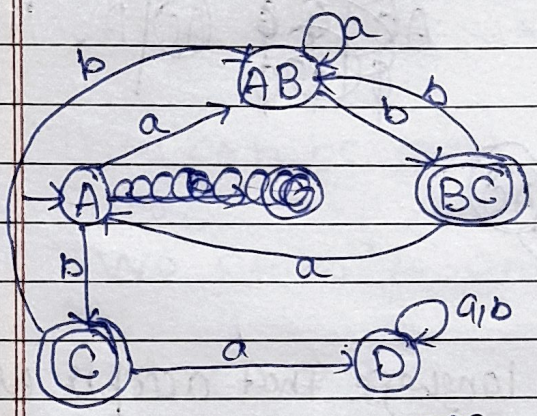
	a	b
→ A	A,B	C
B	A	B
(C)	–	A,B

NFA Transition diag.

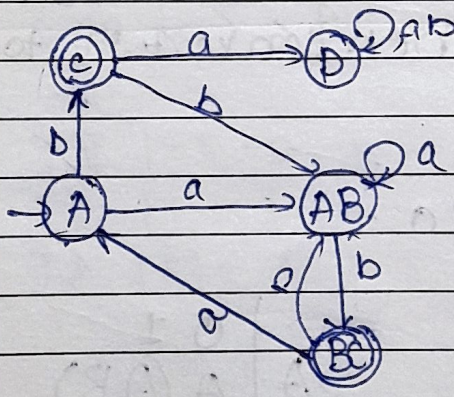


DFA

	a	b.
→ A	{AB}	C
AB	{AB}	{BC}
(BC)	A	{AB}
(C)	ϕ	{AB}
D	D	D



or.



Q. Find out what does this NFA and its equivalent DFA accept

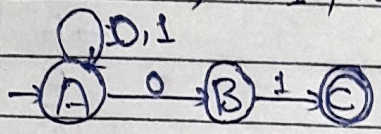
L :- Should not start with 'ba' and must have odd no. of b(s) at the end

Date
Page

Q L = {set of all strings over (0,1) that ends with '01'}. Construct its equivalent DFA.

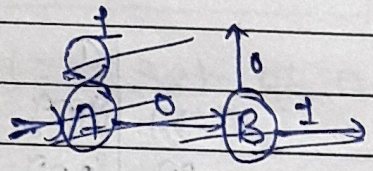
NFA.

L = {01, 101, 001}



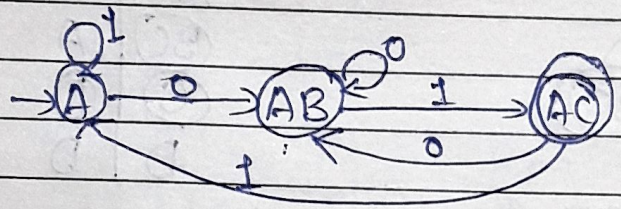
	0	1
→A	AB	A
B	ϕ	C
(C)	ϕ	ϕ

DFA.



	0	1
→A	AB	A
AB	AB	AC
AC		
ϕ	ϕ	ϕ

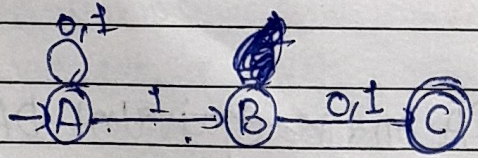
	0	1
→A	AB	A
AB	AB	AC
(AC)	AB	A



Q Design an NFA for a language that accepts all strings over {0,1} in which the second last symbol is always '1'. Then convert it to its equivalent DFA.

Ans NFA

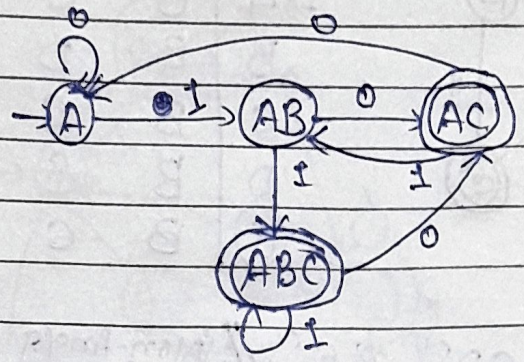
L = {10, 010, 011, 110}



	0	1
→A	A	(AB)
B	C	C
(C)	ϕ	ϕ

Date
Page

DFA.



	0	1
→ A	A	AB
AB	AC	ABC
(AC)	A	AB
(ABC)	AC	ABC

Minimization of DFA

Minimization of DFA is required to obtain the minimal version of any DFA which consists of the minimum no. of states possible

This can be done by combination of equivalent states.

Two states 'A' and 'B' are said to be equivalent if.

$\delta(A,x) \to f$ and $\delta(B,x) \to f$ OR $\delta(A,x) \not\to F$ and $\delta(B,x) \not\to F$ where 'x' is any I/p string

If $|x| = 0$, then A and B are said to be 0 equivalent

If $|x| = 1$, " " " 1 equivalent

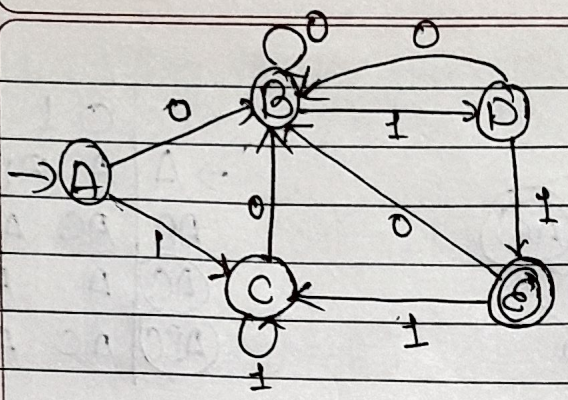
If $|x| = 2$, " " " 2 "

:

If $|x| = n$, " " " n "

If n+same I/p goes check set that in set then not Equivalent

vid 24



	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

0 Equivalence: {ABCD} {E} // {Non-finals} {finals} set set

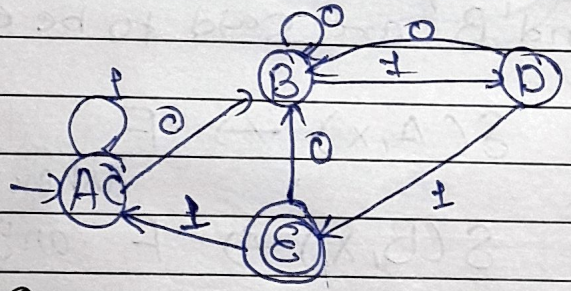
1 Equivalence {ABC} {D} {E}

Same { 2 " {AC} {B} {D} {E}

3 " {AC} {B} {D} {E}

When 2 & 3 are giving consecutively the same result so end the process of equivalence

{A, C} now as single state.



Q. Construct a minimum DFA equivalent to the DFA described by.

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

0 Equi :- ~~{$q_0 q_1 q_3 q_4 q_5 q_6 q_7$} {$q_2$}~~

1 Equi :- ~~{$q_0 q_4 q_5$}~~ {q_1} {q_3} {q_6} {q_2}

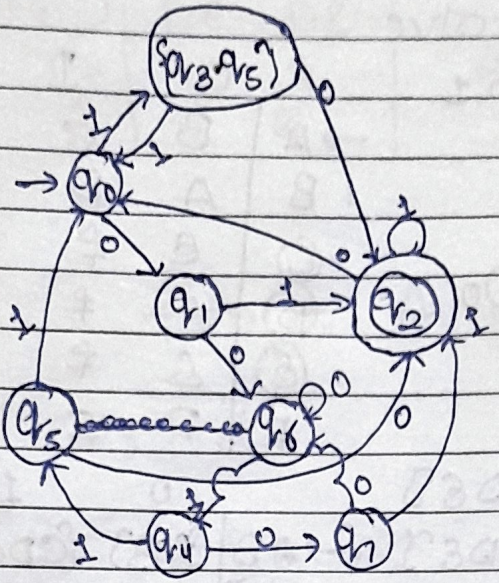
2 Equi :- {q_6} {q_3 ~~q_5~~} {q_0} {q_1} {q_2} {q_4} {q_7}

3 Equi :- {q_0} {q_1} {q_2} {$q_3 q_5$} {q_4} {q_6} {q_7}

Here 2 Equi and 3 Equi are giving same result so end the process

Date
Page

$\{q_3 q_5\}$ is now a single state.



X

0 Equi :- $\{q_0 q_1 q_3 q_4 q_5 q_6 q_7\}$ $\{q_2\}$

1 Equi :- $\{q_0 q_4, q_6\}$ $\{q_1, q_7\}$ $\{q_3 q_5\}$ $\{q_2\}$

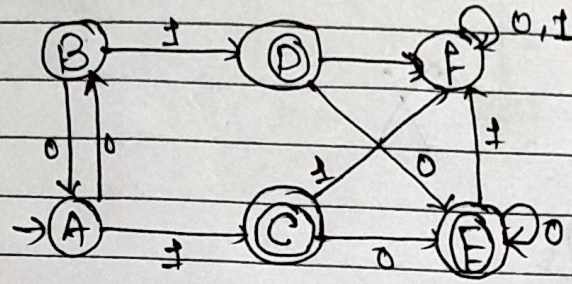
2 Equi :- $\{q_0, q_4\}$ $\{q_6\}$ $\{q_1, q_7\}$ $\{q_3, q_5\}$ $\{q_2\}$

3 Equi :- $\{q_0, q_4\}$ $\{q_6\}$ $\{q_1 q_7\}$ $\{q_3 q_5\}$ $\{q_2\}$

Stop as 2 consecutive equivalents same

	0	1
→ $\{q_0, q_4\}$	$q_1 q_7$	$\{q_3 q_5\}$
$\{q_6\}$	q_6	$q_0 q_4$
$\{q_1 q_7\}$	q_6	q_2
$\{q_3 q_5\}$	q_2	q_6
$\{q_2\}$	$q_0 q_4$	q_2

Q. minimize DFA when there are more than one final states (Diagram).



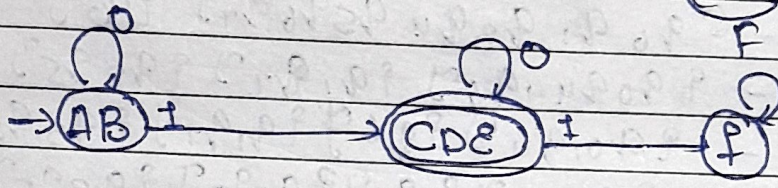
	0	1
→A	B	C
B	A	D
C	E	F
D	E	F
(E)	E	F
F	F	F

0 Equi = {ABF} {CDE}

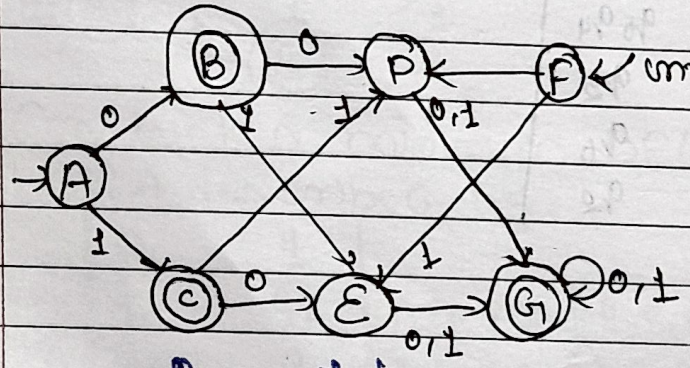
1 Equi = {AB} {F} {CDE}

2 Equi = {AB} {F} {CDE}

	0	1
→AB	{AB}	{CDE}
(CDE)	CDE	F
F	F	F



Unreachable State :- A state is said to be unreachable if there is no way it can be reached from the initial state.



unreachable & only outgoing state. { transition. No in coming }

Just remove it for minimization

	0	1
→A	B	C
(B)	D	E
(C)	E	D
D	q	q
E	q	q
(G)	q	q

0 Equi :- {ADE} {BCq}

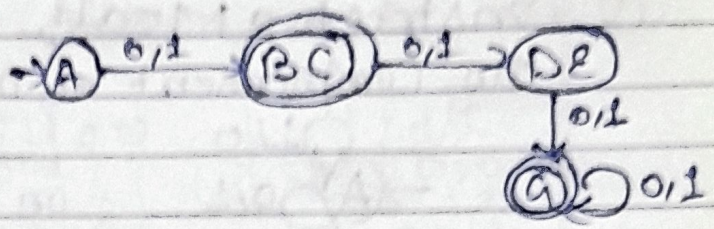
1 Equi :- {ADE} {BC} {q}

2 Equi :- {A} {DE} {BC} {q}

3 Equi :- {A} {DE} {BC} {q}

Stop for same consequtive Equivalence.

	0	1
→A	BC	BC
(BC)	DE	DE
DE	Q	Q
Q	Q	Q



Finite Automata with output

Mealy M/C

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

Where

Q = Finite set of states

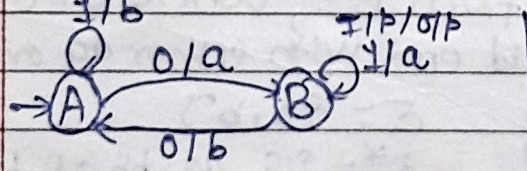
Σ = Finite non-empty set of I/Ps alphabets.

Δ = The set of output alphabets.

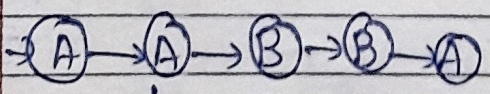
δ = Transition function $Q \times \Sigma \Rightarrow Q$

λ = Output function: $\Sigma \times Q \to \Delta$

q_0 = Initial state / Start state



Eg. 1 0 1 0



∴

Length of I/P String = O/P String

$n = n$

I/P O/P.

Moore M/C

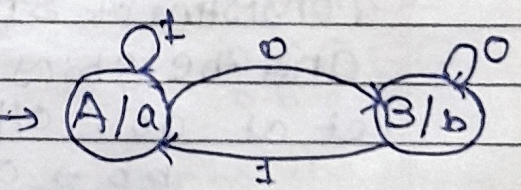
$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Where.

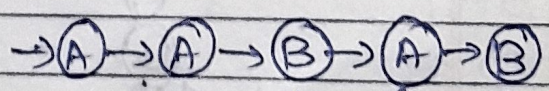
→ Same

λ = O/P func^n : $Q \to \Delta$

→ Same.



Ex:- 1 0 1 0



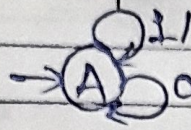
Length of I/P String = length of I/P + 1 of O/P

$n \Rightarrow n+1$

I/P O/P

Date
Page

Q Construct a Mealy machine that produces 1's complement of any binary I/P string.

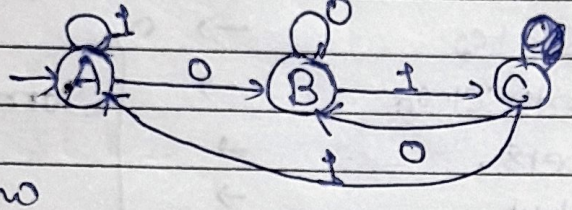


on I/P 0 ⇒ gets O/P 1
on I/P 1 ⇒ gets O/P 0

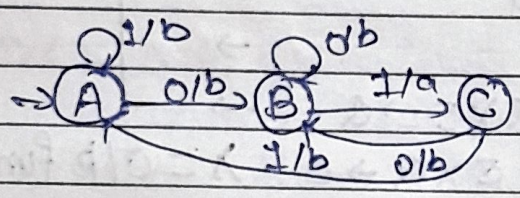
Q Construct a mealy machine that prints 'a' whenever the sequence '01' is encountered in any I/P binary string.

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

DFA. Try.



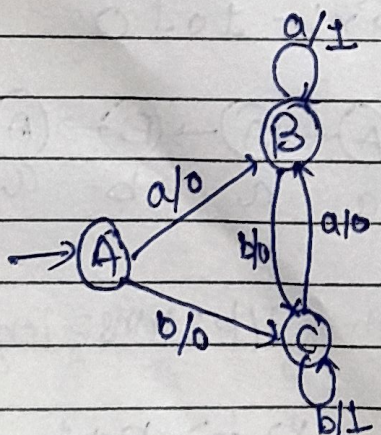
Now



Q Design a Mealy M/C accepting the language consisting of strings from Σ^*, where $\Sigma = \{a,b\}$ and the string should end with either aa or bb

Let at aa → O/P is 1 $\Sigma = \{a,b\}$

bb → O/P is 1 $\Sigma^* = \{\varepsilon, aa, bb, ab, ba$

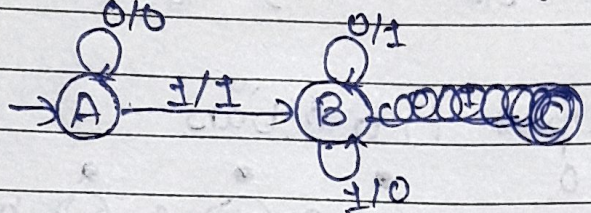


Date
Page

Construct a Mealy M/C that gives 2's complement of any binary I/p (Assume that the last Carry bit is neglected)

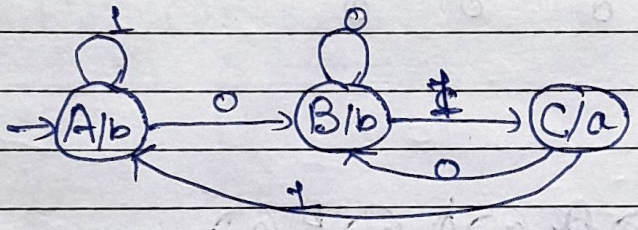
Ex:- 1 0 1 0 0 | 1 1 0 0 | 1 1 1 1
0 1 0 1 1 | 0 0 1 1 | 0 0 0 0
+ 1 | + 1 | + 1
0 1 1 0 0 | 0 0 1 0 0 | 0 0 0 1

So we set that upon 2's complement up till we get 1 we have same complement.



1 0 1 0 0
B B B A A
0 1 1 0 0 ✓

Construct a Moore M/C that prints b whenever the sequence '0 1' is encountered in any I/p binary string.



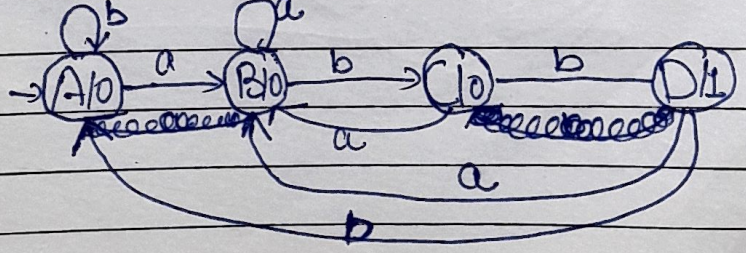
$\Sigma = \{0, 1\}$
$\Delta = \{a, b\}$

Ex:- 0 1 0 1
A B C B C
b b a b a

Q Construct a Moore M/C that counts the occurrences of the sequence 'abb' in any input strings over $\{a, b\}$

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

Let wherever we encounter abb we will get o/p 1.



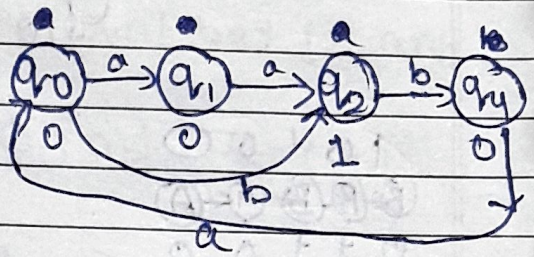
Date
Page

Q. For the ~~construction~~ following Moore M/C the i/p alphabet is $\Sigma = \{a, b\}$ and the o/p alphabet is $\Delta = \{0, 1\}$. Run the following input sequences and find the respective outputs.

i) aabab ii) abbb iii) ababb

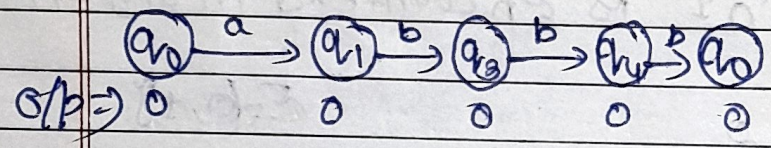
States	a	b	outputs
→ q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

i) a a b a b

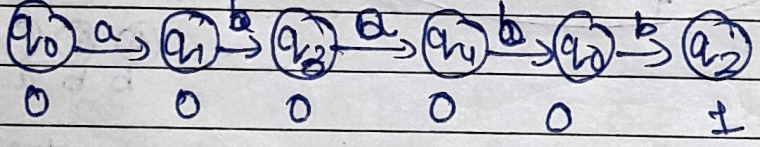


o/p = 0 0 1 0 0 1

ii) a b b b



iii) a b a b b



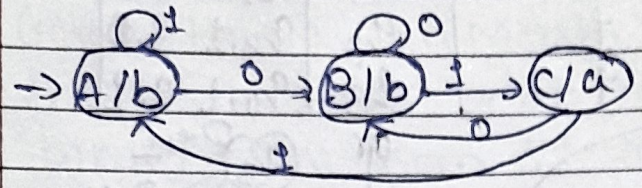
Conversion of Moore M/C to Mealy M/C

Q Construct a Moore M/C that prints a whenever the sequence '01' is encountered in any I/P binary string and then convert it to its Equivalent Mealy m/c.

Moore M/C ⇌ Mealy M/C equally powerfull.

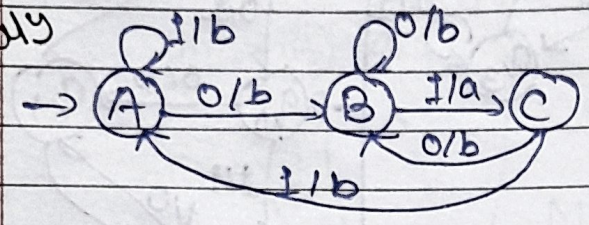
Moore

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



State	0	1	output
→A	B	A	b
B	B	C	b
C	B	A	a

Mealy

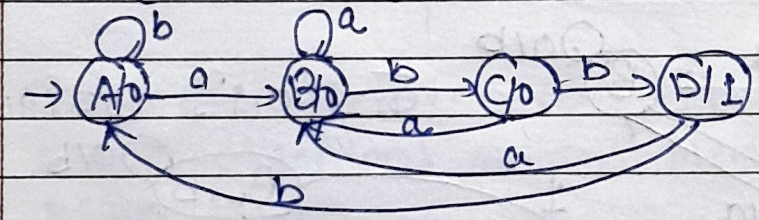


State		
→A	B,b	A,b
B	B,b	C,a
C	B,b	A,b

Q The given Moore M/C counts the occurences of the sequence 'abb' in any I/P binary string over $\{a,b\}$. Convert it to its Equivalent Mealy M/C

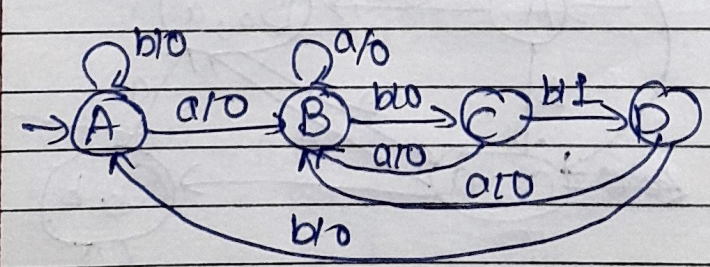
$\Sigma = \{a,b\}$ $\Delta = \{0,1\}$

Moore.



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

Mealy.



Mealy.

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

Mealy → o/ps are associated with transition
Moore → o/p are associated with states

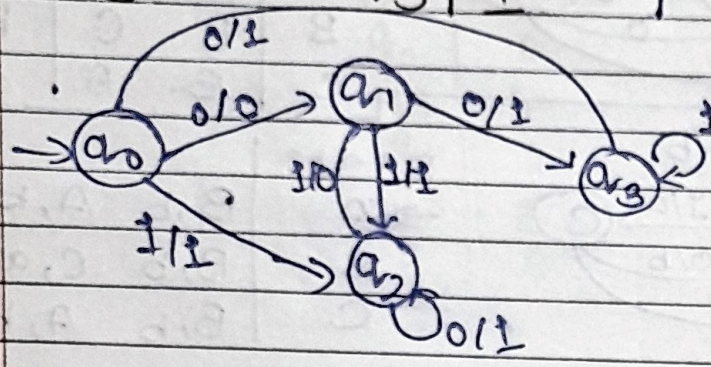
Q. Convert the given Moore M/C to its equivalent Mealy M/C.

Moore.

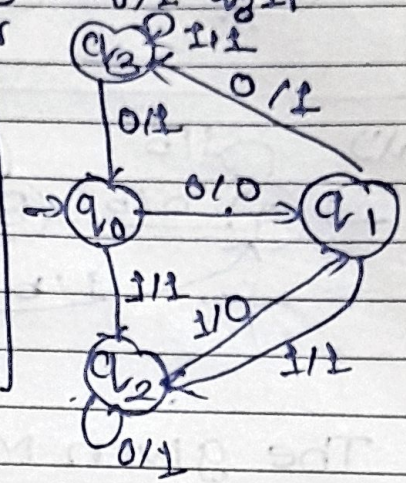
State	0	1	Output
→ q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_0	q_1	1
q_3	q_0	q_3	1

Mealy.

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



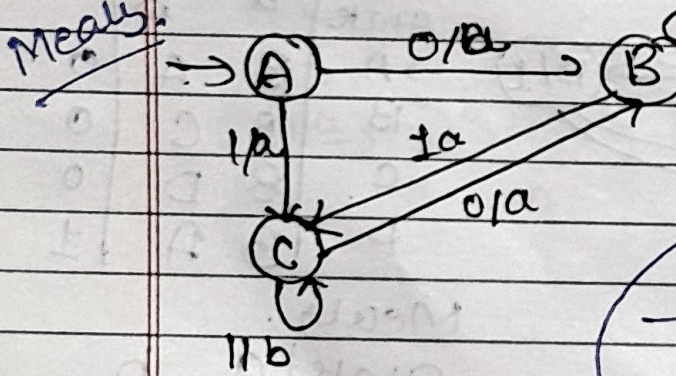
or



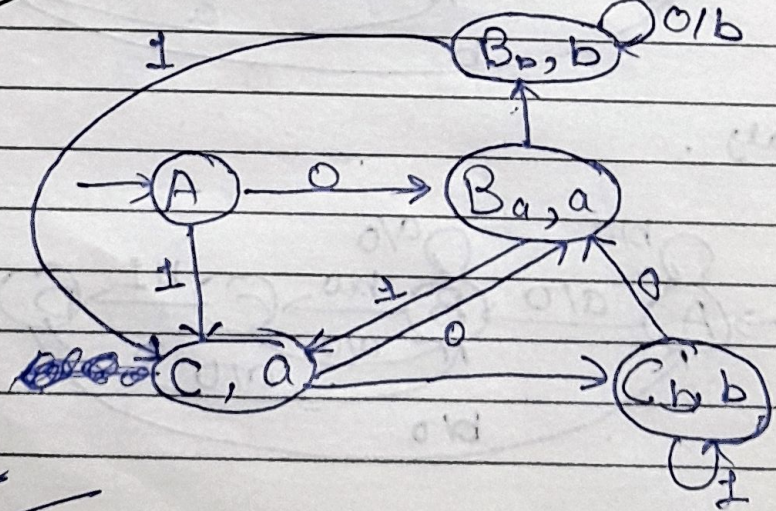
Mealy to Moore

Q Convert the following Mealy m/c to its equivalent Moore M/C

Mealy:



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

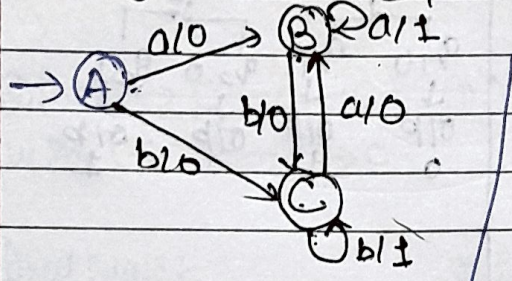


Moore

Date
Page

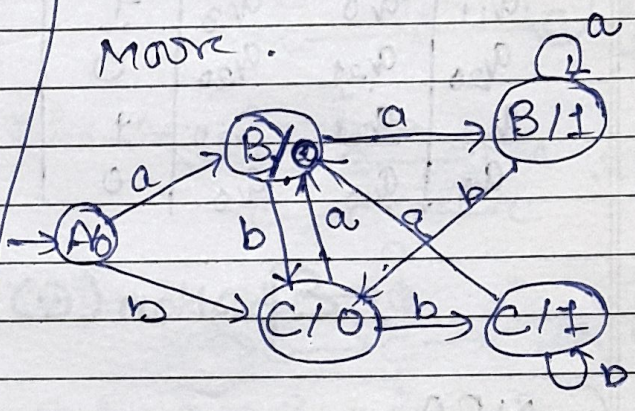
Moore → Mealy ⇒ No. of States were Same
Mealy → Moore ⇒ No. of States increased
↓ (n x y) no. of states at max.
n and y
States outputs.

Q Given below is a Mealy M/C that prints '1' whenever the sequence 'aa' or 'bb' is encountered in any i/p binary string from Σ^* where $\Sigma = \{a, b\}$. Design the equivalent Moore M/C for it.



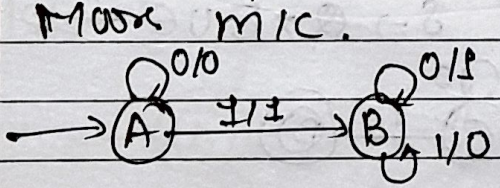
$\Delta = \{0, 1\}$ $\Sigma = \{a, b\}$

Moore.



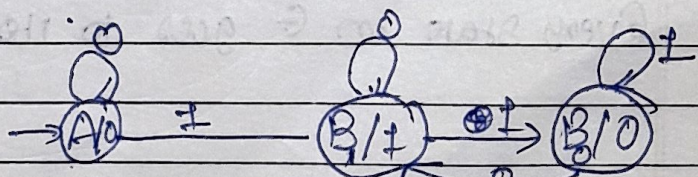
Test Run.
aa' ab abba
001 000 00001

Q. Convert the given mealy m/c that gives the 2's complement of any binary i/p to its equivalent Moore m/c.



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

Test run. R to L Run
10100
01011
+1
01100



Mealy
10100
BBBAA

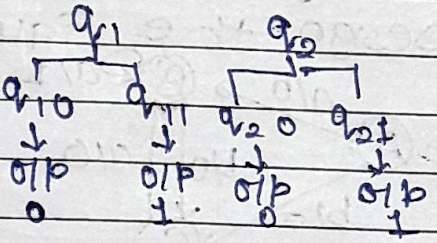
Moore
10100
BBBAA

Q. Convert the given Mealy M/C to its Equivalent more m/c.

State	a	b
$\rightarrow q_0$	$q_3, 0$	$q_1, 1$
q_1	$q_0, 1$	$q_3, 0$
q_2	$q_2, 1$	$q_2, 0$
q_3	$q_1, 0$	$q_0, 1$

more m/c.

State	a	b	output
$\rightarrow q_0$	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

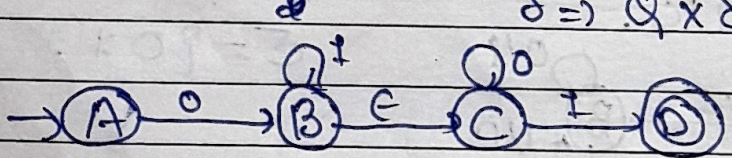


Epsilon (ϵ) NFA

ϵ-NFA
↳ Empty Symbols.

5 tuples $\{Q, \Sigma, q_0, \delta, F\}$

$\delta \Rightarrow Q \times \Sigma \cup \epsilon \rightarrow 2^Q$

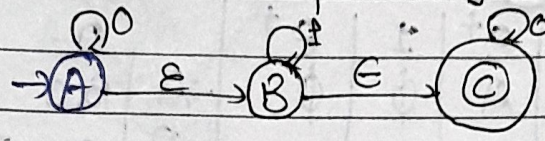


Every state on ϵ goes to itself.

Date
Page

Conversion of ϵ-NFA to NFA

Q Convert the following ϵ-NFA to its equivalent NFA.



	0	1
→A	{ABC}	{BC}
B	C	{BC}
C	C	C

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

We the following framework.

State	ϵ^+	Input	ϵ^*

(i) For A: A —0→ A —ϵ→ A; B —0→ ϕ; C —0→ C → ∴ upon I/P 0 A goes to {ABC}

(ii) For A: A —1→ ϕ; B —1→ B —ϵ→ B,C; C —1→ C —ϵ→ C → ∴ {BC}

(iii) For B: B —0→ ϕ; C —0→ C —ϵ→ C → ∴ C

(iv) For B: B —1→ B —ϵ→ B; C —1→ C —ϵ→ C → ∴ {BC}

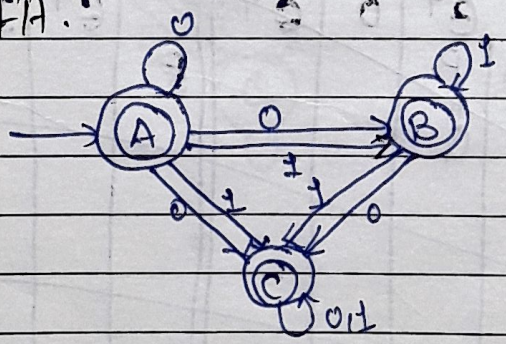
(v) For C: C —0→ C —ϵ→ C

(vi) For C: C —1→ C —ϵ→ C

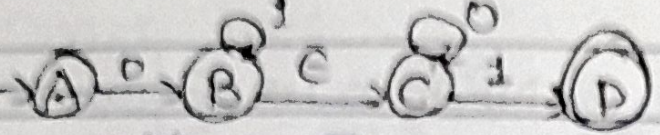
- Final State:-
  Any state that can reach the final state only by seeing ϵ

∴ ABC are final state.

- NFA:



Q. Convert the following ε NFA to its equivalent NFA.

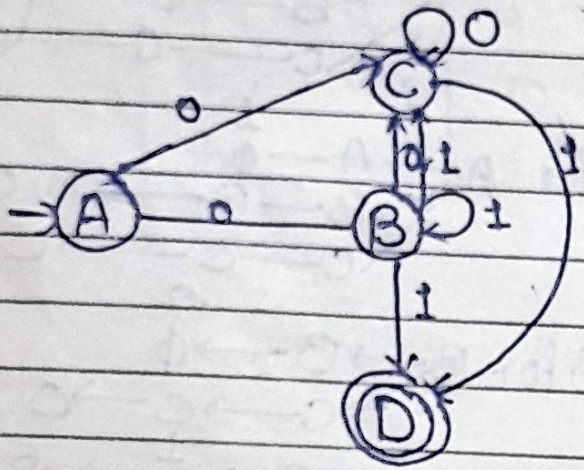


	ε*	0	ε*
A	A	B	B, C
B	B	ϕ	ϕ
	C	C	C
C	C	C	C
D	D	ϕ	ϕ

	ε*	1	ε*
A	A	ϕ	ϕ
B	B	B	B C
	C	D	D
C	C	D	D
D	D	ϕ	ϕ

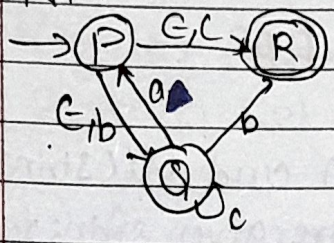
NFA

	0	1
→A	B, C	ϕ
B	C	B, C, D
C	C	D
(D)	ϕ	ϕ



Date
Page

Convert the following ∈ NFA to its Equivalent NFA.

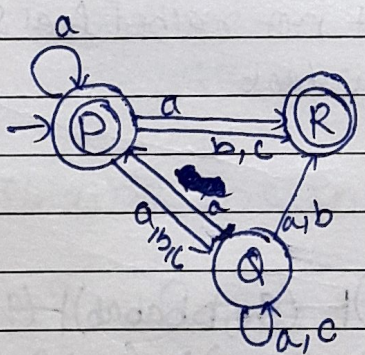


	a	b	c
→P	{P,Q,R}	{Q,R}	{Q,R}
Q	{P,Q,R}	{R}	{Q}
R	ϕ	ϕ	ϕ

	∈+	a	∈+
P	P	ϕ	
	R	ϕ	P
	Q	P	P, Q
Q	Q	P	P, Q, R
R	R	ϕ	ϕ

	∈+	b	∈+
P	P	Q	Q
	R	ϕ	ϕ
	Q	R	R
Q	Q	R	R
R	R	ϕ	ϕ

	∈+	c	∈+
P	P	R	R
	R	ϕ	ϕ
	Q	Q	Q
Q	Q	Q	Q
R	R	ϕ	ϕ



If we are reaching the final state from Initial state that string is called accepted string.

Some Gist topics

Acceptability of a string by F.A

If we have a given F.A and W (string) we have to check weather the given string will be accepted by the m/c or not by parsing the whole string.

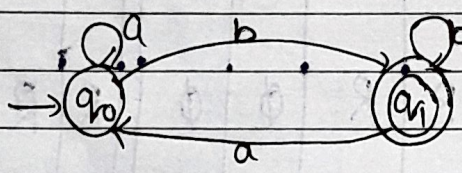
$M = \{Q, \Sigma, \delta, q_0, F\}$

$\delta[q_0, w]$ — * — F

0 or more transition

and Reaches final state

Ex:-



① $w = aa\,bbab$

$q_0\,q_0\,q_1\,q_1\,q_0\,(q_1)$ final state

∴ String (W) will be accepted by the given m/c

② $w = a\,bba$

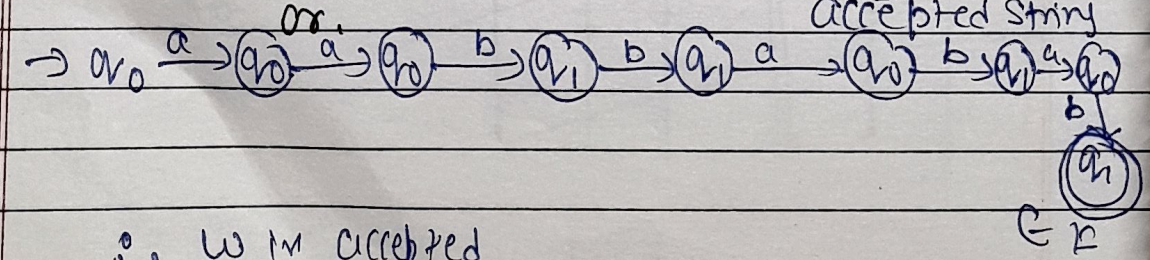
$q_0\,q_1\,q_1\,q_0$

∴ W is not acceptable as it not reached final state. in the last.

Have to represent it.

$w_3 = aa\,bbabab$

$(q_0, aabbabab) \vdash (q_0, abbabab) \vdash (q_0, bbabab) \vdash (q_1, babab) \vdash (q_1, abab) \vdash (q_0, bab) \vdash (q_1, ab) \vdash (q_0, b) \vdash (q_1) \in F$



∴ W is accepted