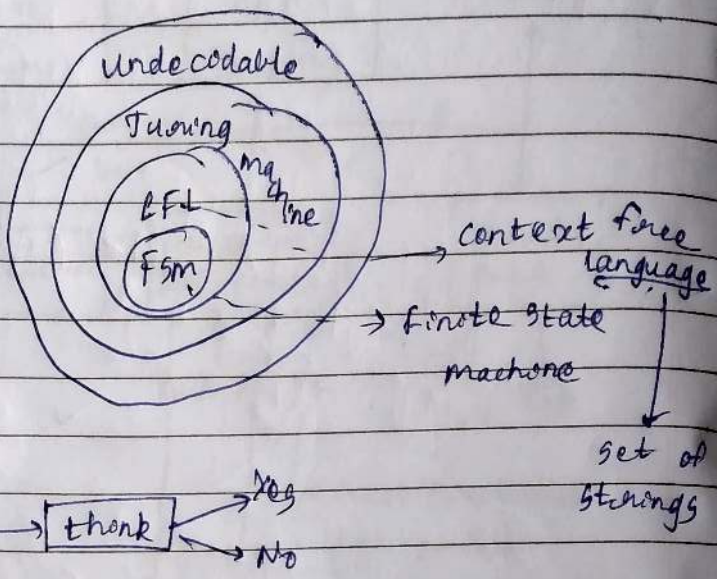
SPPU-TE-COMP-CONTENT – KSKA Git

Date 08-07-24
Page

TOC

U-I Finite State Machines



→ Symbol - a, b, c, 0, 1, 2, 3

→ Alphabet - Σ - collection of symbols. eg: {a, b}, {d, e, f, g}, {0, 1, 2}

→ String : Sequence of symbols
Eg: a, b, 0, 1, aa, bb, ab, 01,

→ Language :- Set of strings
eg: $\Sigma = \{0, 1\}$

L_1 = Set of all strings of length 2
= {00, 01, 10, 11} → finite

L_2 = Set of all strings that begin with 0
= {0, 011, 0001, ...} → infinite

* Powers of Σ $\quad \Sigma = \{0, 1\}$

Σ^0 = Set of all strings of length 0 (empty string) ! $\Sigma^0 = \{\epsilon\}$

Σ^1 = " " " " " " 1 ! $\Sigma^1 = \{0, 1\}$

Σ^2 = " " " " " " 2 ! $\Sigma^2 = \{00, 01, 10, 11\}$

SPPU-TE-COMP-CONTENT – KSKA Git

Date
Page

* Cardinality - no. of elements in a set

$\Sigma^n = 2^n$

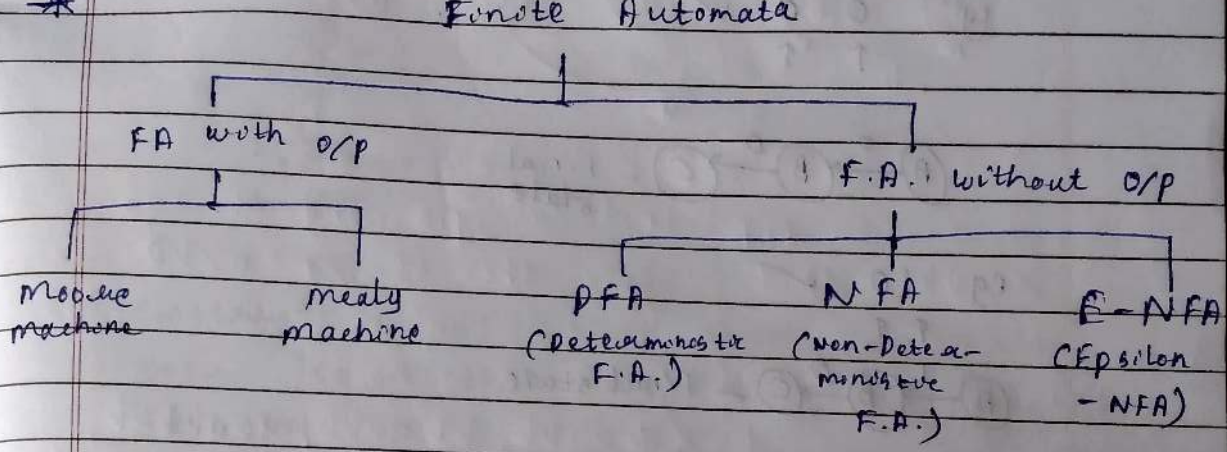
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots$

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

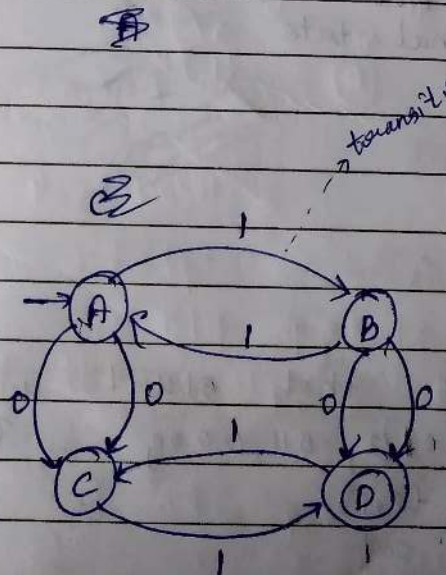
= set of all possible strings of all lengths over $\{0, 1\}$

↳ infinite

*



→ F.A.



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

Q = set of all states

Σ = I/P's

q_0 = start state / initial state

F = set of final states

δ = transition function

from $Q \times \Sigma \to Q$

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

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

$q_0 = \{A\}$

$F = \{D\}$

SPPU-TE-COMP-CONTENT – KSKA Git

Date 10-7-24
Page 1

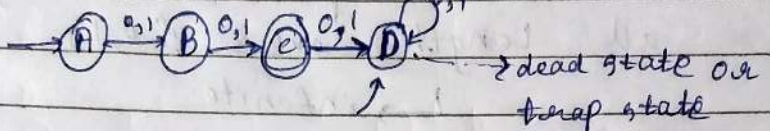
DFA (ex)

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

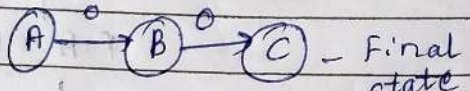
Soln:

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

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

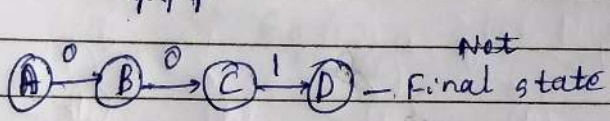


eg: 0 0 ✓

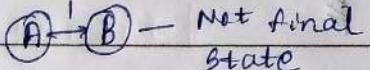


eg: 1 0 ✓

eg: 0 0 1 ✗



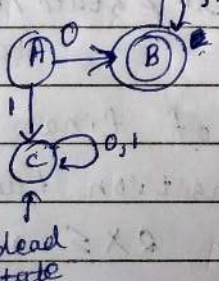
eg: 1



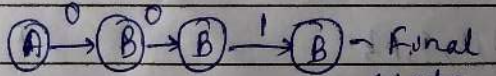
Q. L = set of all strings that start with '0'

Soln: $= \{0, 00, 01, 000, 010, 011, 0000, \ldots\}$

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



eg: 001



SPPU-TE-COMP-CONTENT – KSKA Git

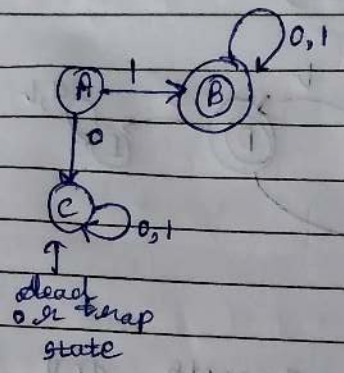
Date Page 11-7-24

Q. ~~L2~~. Design a DFA that accepts the language over I/P 0,1 and contains set of all strings that starts with 1.

Soln:

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

$L_2 = \{100, 1111, \ldots\}$

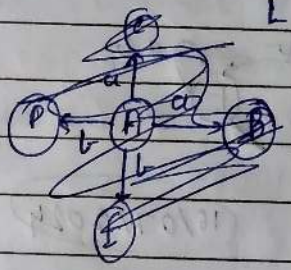


Q. Construct a DFA that accepts any strings over {a,b} that does not contain the string aabb in it.

Soln:

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

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

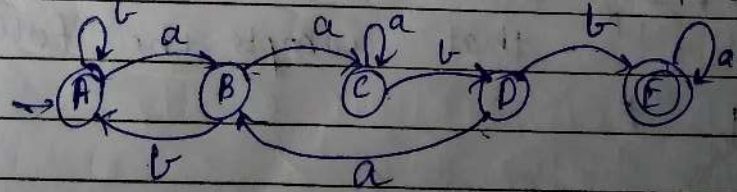


~~Step 1:~~

~~Let us cons~~

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

Step 1:



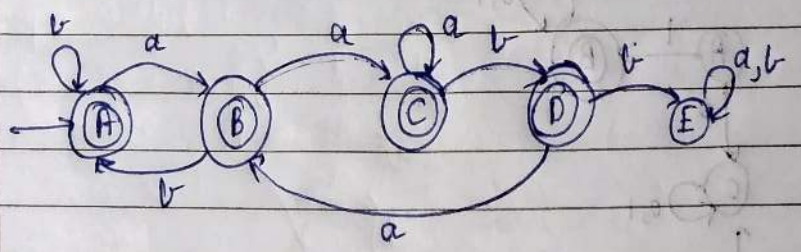
$L = \{baabb, baaabb, aabb, aaabbab, baabaabb, babaabb, bbabaabb, aabbabba, \ldots\}$

SPPU-TE-COMP-CONTENT – KSKA Git

Date
Page 15/07/2024

eg: ~~aaabb~~

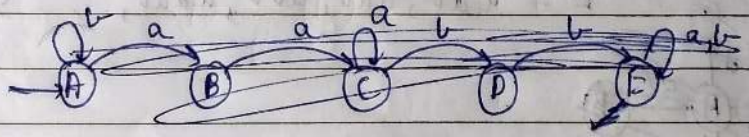
→ Flip the states
- make the final states into non final state and
- make the non-final states into final states



Q. Construct DFA that accepts any strings over {a,b} that accepts the string aabb.

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

$L = \{$ baabb, bbaabb, aaabbab, aabbbaabbbb ... $\}$



HW assignment | 16/07/2024

Q. Design a DFA over the alphabet {0,1} that contains all the strings 110

Q. " " " that accepts all the strings that ends with 11.

Q. ~~Design~~ " " that accepts the following string

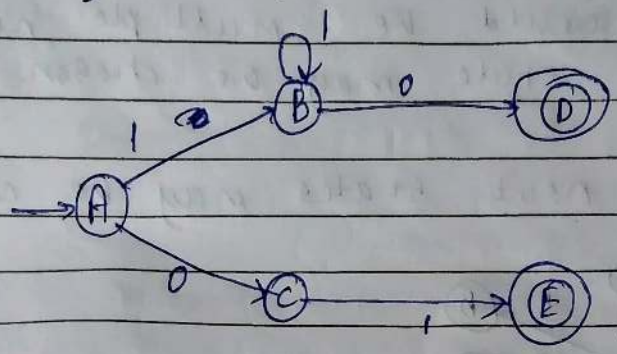
1) 1010
2) 000111
3) That accepts odd no. of 0's
4) and even number of 1's

SPPU-TE-COMP-CONTENT – KSKA Git

Assignment
Date
Page

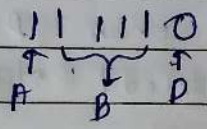
* How to figure out what a DFA recognizes ?

Q.



Soln: L = { accepts the strings of 0s or a string of atleast one '1' followed by a '0'}

10 ✓ one binary digit '1'

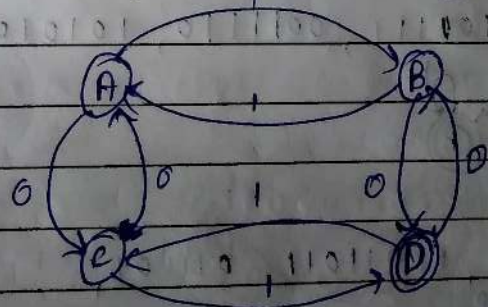


16/07/2024

* NFA (Non-Deterministic FA)
(* NFA and DFA (4 mks))

→ * Determinism (DFA)

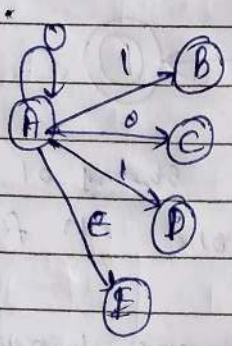
- In DFA, given current state we know what the next state will be
- It has only one unique next state
- It has no choices or randomness
- It is simple and easy to design



SPPU-TE-COMP-CONTENT – KSKA Git

→ NON-DETERMINISM

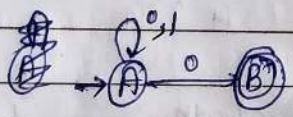
- In NFA, given the current state there are could be multiple next states
- The next state may be chosen at random
- All the next states may be chosen in parallel.



- NFA is more powerful than DFA since we can move to the next state in parallel

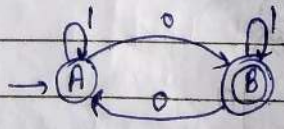
eg: ~~set of~~ L = { set of all strings that ends with 0 }

sol:

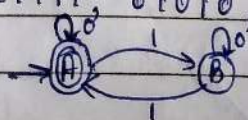


~~DFA~~ NFA since on '0' it is going on both A and B (multiple states)

Q3. 3) L = { 00011010, 1101111, 0011110, 101010, ... }



4) L = { 1111, 01010, 11011011, 01101, ... }

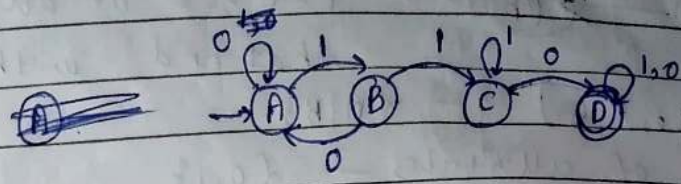


SPPU-TE-COMP-CONTENT – KSKA Git

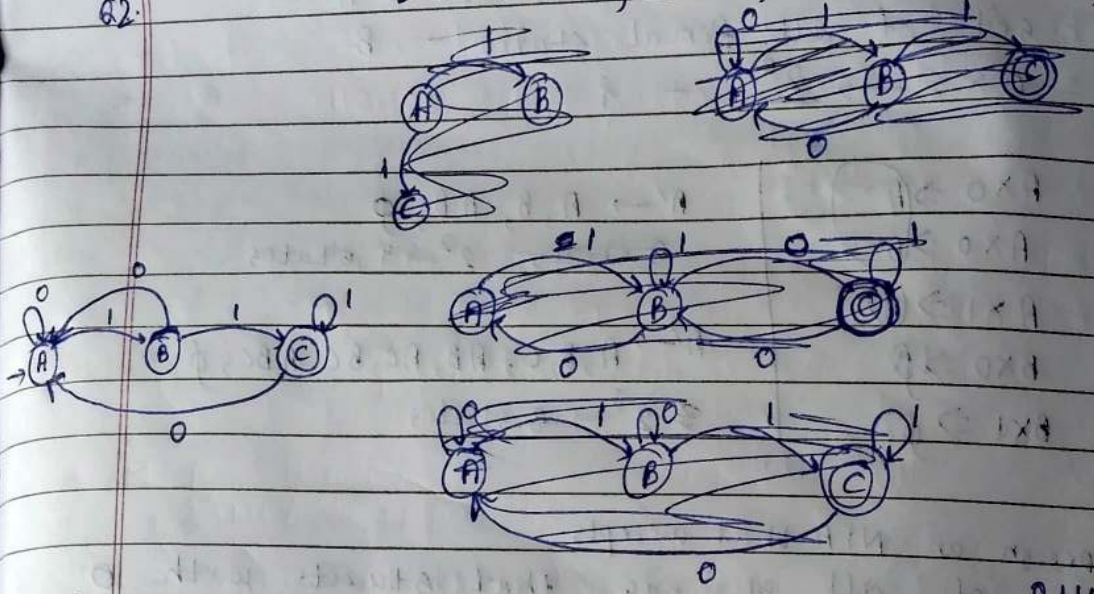
Assignment 1

Q1. $L = \{ 110, 00110, 1110, 1011011, 0111011 \ldots \}$

Soln:

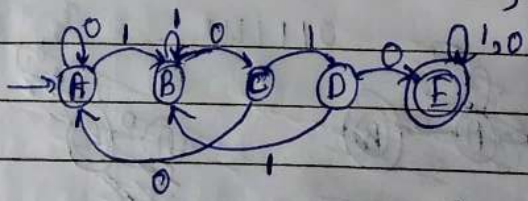


Q2. $L = \{ 0111, 1011, 0011, 11111 \ldots \}$

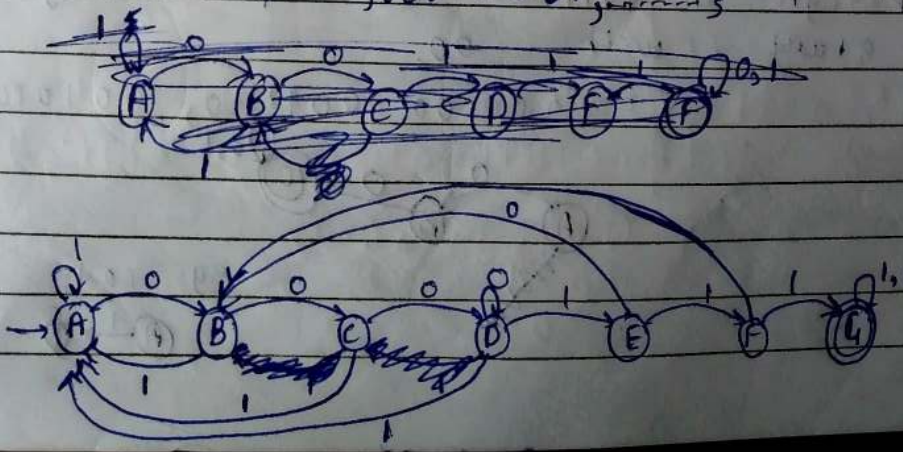


0111010

Q3. 1) $L = \{ 01010, 11010, 11010110, 0001010111 \ldots \}$



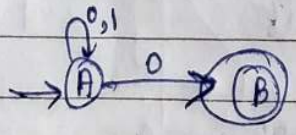
2) $L = \{ 1110\ 00111, 011000111, 011001100111, 000111111110, 00011100111 \ldots \}$



SPPU-TE-COMP-CONTENT – KSKA Git

Date
Page 19/09/24

NFA – Formal defination



L = set of all strings that end with 0 }

$[Q, \Sigma, q_0, F, \delta]$ – {A, B}

Q = set of all states – {0, 1}

Σ = inputs – A

q_0 = start state/initial state – A

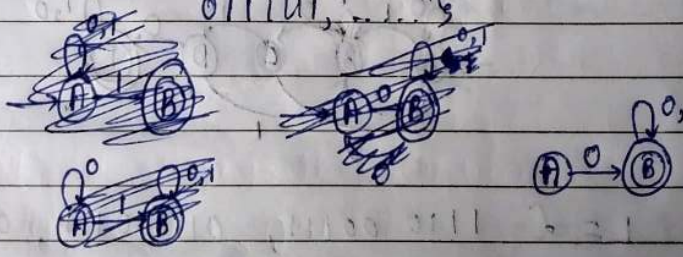
F = set of all final states – B

$\delta = Q \times \Sigma \rightarrow 2^Q$ – }

A×0 ⇒ A, B	A' → A, B, AB, ϕ
A×0 ⇒ B	~~B states~~ 2^2 → 4 states
A×1 ⇒ A	
B×0 ⇒ ϕ	A' → A, B, C, AB, AC, BC, ABC, ϕ
B×1 ⇒ ϕ	2^3 → 8 states

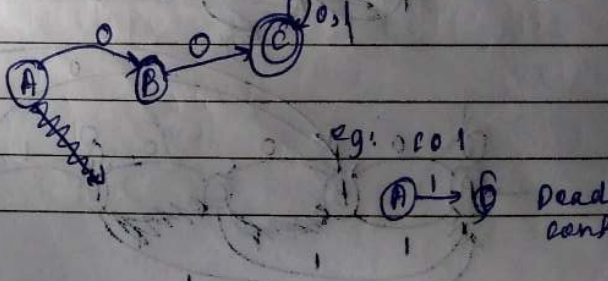
Q. Design an NFA that accepts set of all strings that starts with 0.

Soln: L = {01000, 011100, 00010, 00010, ... }



Q. Design an NFA that accepts strings that start with 00.

Soln: L = {00, 000, 00111110, 001010, 000000, 0111, }

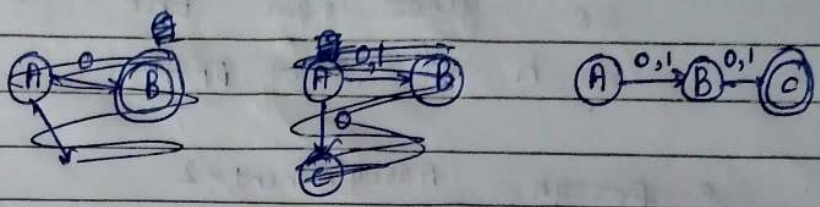


SPPU-TE-COMP-CONTENT – KSKA Git

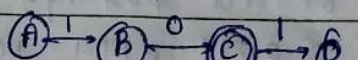
Q. Construct NFA that accepts ~~set~~ sets of all strings {0,1} of length 2

Soln:

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



eg: 101

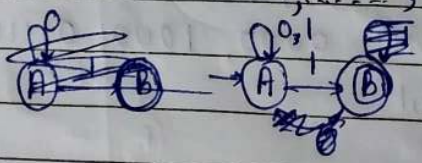


Q. Design DFA for all ~~ab~~ below 5 examples → Assgn 2

Q. L1 = { Set of all strings that ends with 1 }

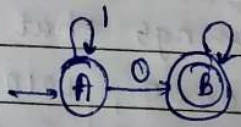
Soln:

$L1 = \{001111, 010111, 00001, 111111, 11101, \ldots\}$



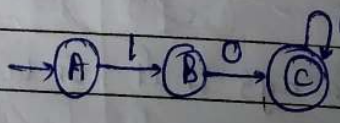
Q. L2 = { Set of all strings that contain 0 }

Soln:



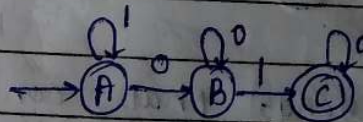
Q. L3 = { Set of all strings that starts with 10 }

Soln:



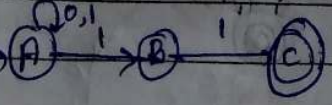
Q. L4 = { Set of all strings that contain 01 }

Soln:



Q. L5 = { Set of all strings that contain 11 }

Soln:



SPPU-TE-COMP-CONTENT – KSKA Git

Assignment:

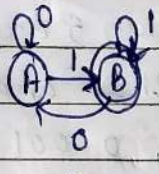
If you were to construct the equivalent DFA's for the above NFA's then ~~tell~~ state how many min number of states would we use for the construction of each of the DFA's.

~~a. Design~~ Assignment-2

Q. Design DFA

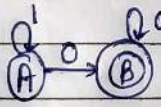
1) L1 = {set of all strings that ends with 1}

Soln: L1 = {0011, 0111, 10111, 10110111, ...}



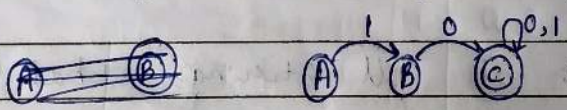
2) L2 = {set of all strings that ~~accepts~~ contains 0}

Soln: L2 = {1011, 0000, 1000, 01111, 101010, ...}



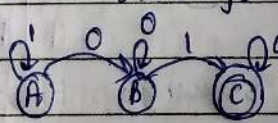
3) L3 = {set of all strings that starts with 10}

Soln: L3 = {10, 1011, 10000, 101000, 101111, 1001, ...}



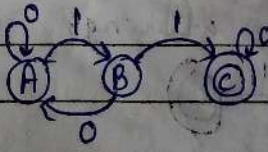
4) L4 = {set of all strings that contain 0}

Soln:



5) L5 = {set of all strings that contain 11}

Soln: L5 = {11111, 01111, 00110011, 110000, 111000, 001001011, ...}



SPPU-TE-COMP-CONTENT – KSKA Git

Date
Page

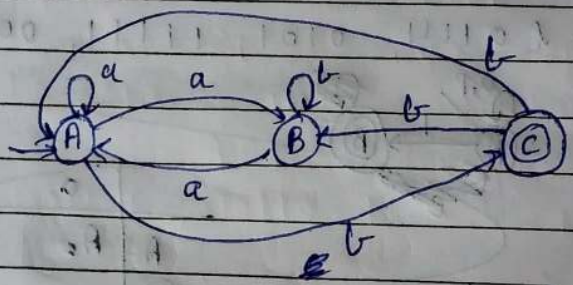
* NFA to DFA conversion:

→ Subset construction method

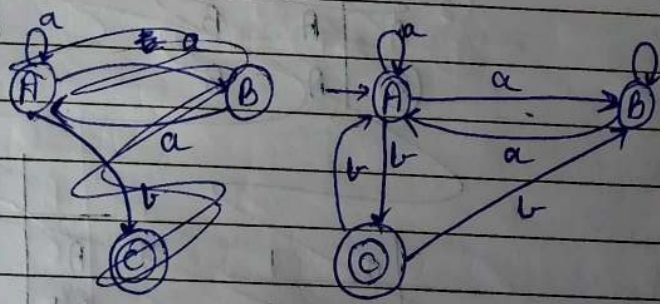
Q. Find the equivalent DFA for the given NFA given by M = [{A, B, C}, {a, b}, δ, A, {C}] where δ is given by:

	a	b
→A	A, B	C
B	A	B
(C)	-	A, B

Soln:



OR



	a	b
→A	AB	C
AB	AB	BC
BC	A	AB
C	D	AB
D	D	D

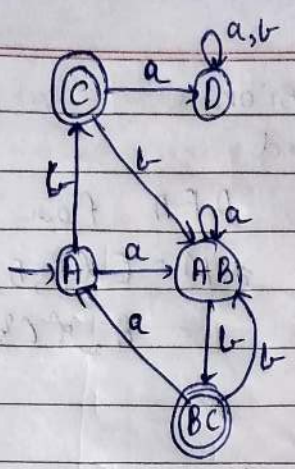
dead/trap state

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

DFA:-

M = [{A, AB, BC, C, D}, {a, b}, Q × Σ → Q, A, {BC, C}]

SPPU-TE-COMP-CONTENT – KSKA.Git



29/07/2024

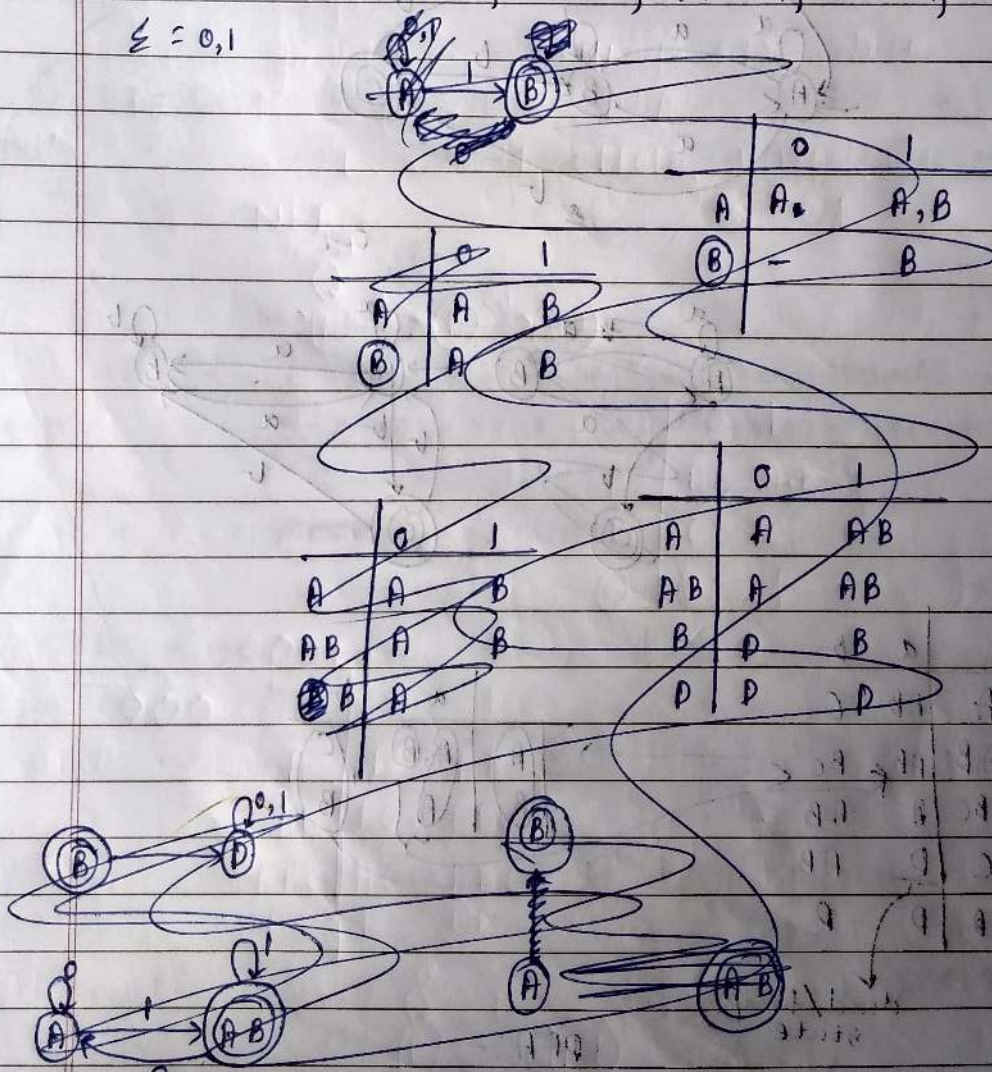
Q. ~~Set~~ NFA to DFA

L = {set of all strings over {0,1} that ends with 0 or 1}

SOLN:

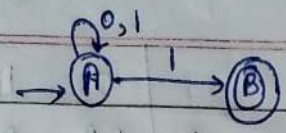
L = {0, 11, 01, 101, 111, 1111, 0000, 1101, ...}

Σ = 0,1



SPPU-TE-COMP-CONTENT – KSKA Git

NFA :-

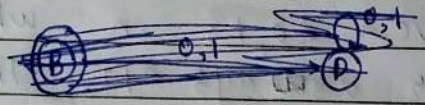


	0	1
A	{A}	{A, B}
(B)	ϕ	ϕ

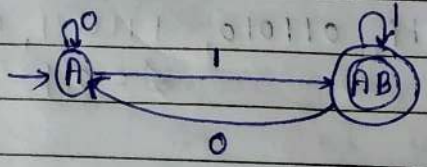
M = [{A, B}, (0,1), δ, A, {B}]

	0	1
A	{A}	{A, B}
AB	{A}	{AB}
B	D	D
D	D	D

M = [{A, AB, B, D}, (0,1), Q × Σ → Q, A, {AB, B}]



DFA :-



Q. NFA to DFA

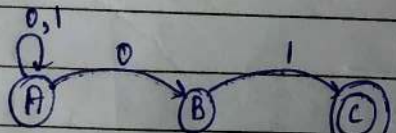
L = {set of all strings over (0,1) that ends with '001'}.

Soln.

L = {001, 10011101, 10001, 00001, 01101, ...}

Σ = 0,1

NFA :-



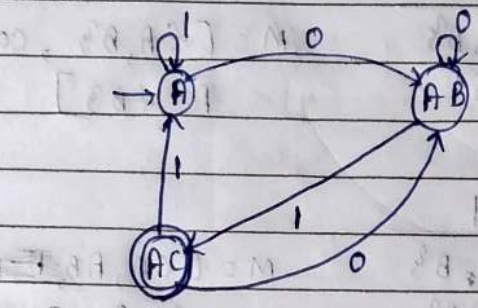
	0	1
A	A, B	A
B	ϕ	C
(C)	ϕ	ϕ

	0	1
A	AB	A
AB	AB	AC
BC	D	C
AC	AB	A
D	D	D

SPPU-TE-COMP-CONTENT – KSKA Git

	0	1
A	AB	A
AB	AB	AC
(AC)	AB	A

DFA:-



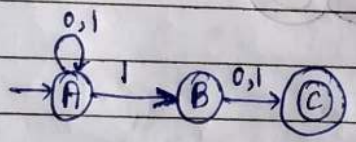
Q Design an NFA for a lang that accepts all strings over {0,1} in which the 2nd last symbol is always '1'. Then convert it to its equivalent DFA.

Soln:

L = { 0011, 011010, 111111, 0 1111, ... }

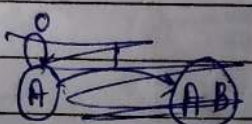
Σ = {0,1}

NFA:-



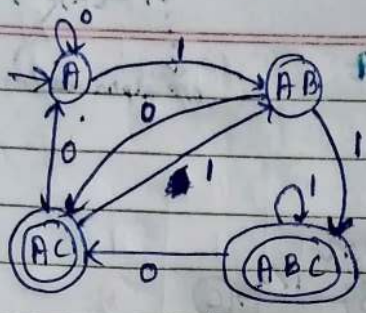
	0	1
A	A	A,B
B	C	C
(C)	ϕ	ϕ

	0	1
A	A	AB
AB	AC	ABC
ABC	AC	ABC
AC	A	AB



SPPU-TE-COMP-CONTENT – KSKA Git

DFA:-



Q. Given below is the NFA for a language

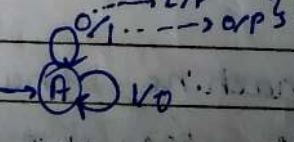
Q. Construct a minimum DFA equivalent to the DFA described by

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

* Mealy and Moore state machines

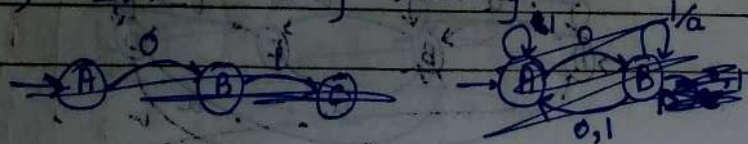
Q. Construct a mealy machine that produces the 1's complement of any binary input string.

Ans



Q. Construct a Mealy machine that prints 'a' whenever the sequence '001' is encountered in any I/P binary string.

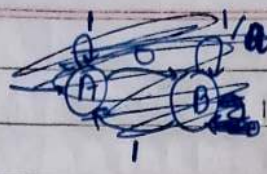
Soln:



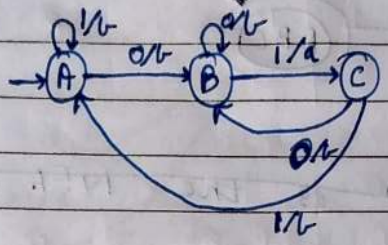
SPPU-TE-COMP-CONTENT – KSKA Git

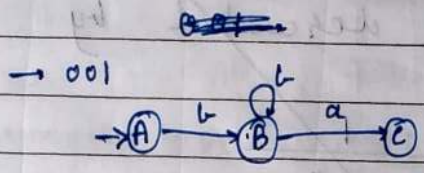
$\Sigma = \{0,1\}$

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



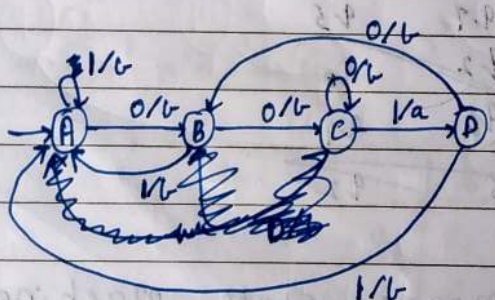
o/p alphabets





bba

Q. Same qn as above but '001'



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

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

$\Sigma = \{1001, 001001, 1110011110, 001000, \ldots\}$

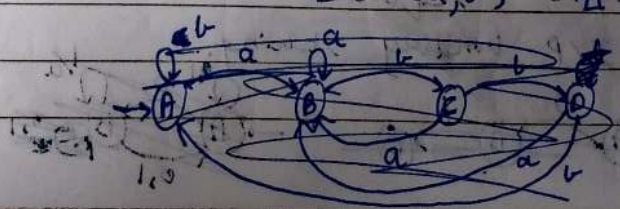
0001

~~000~~

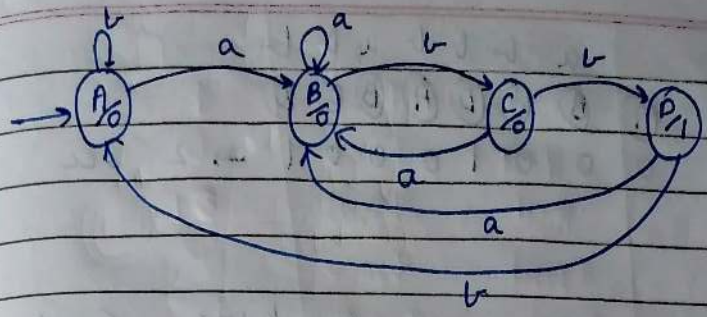
* Moore machine

Q. Construct a moore machine that counts the occurences of the sequence 'abb' in any I/p strings over $\{a,b\}$

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



SPPU-TE-COMP-CONTENT – KSKA Git



a b b
A B C D
0 0 0 1
↳ o/p is one more than I/p

- when the substring abb is encounded I will be printed by moore machine and in all other cases the machine will O/P 0.

→ IMP NOTE:-

1). In moore machine, O/P is associated with the state
~~In other words~~ : OR
. In the moore machine, o/p is a functn of current state.

2). In the mealy machine, O/P is a functn of current state and current I/P.
OR
. In the mealy machine, o/p is associated with the current state and current transition.

3) n I/P bits 3 then n+1 O/P bits in moore machine. eg: I/P bits = 3 ; O/P bits = 3+1 = 4

→ case 1:

~~a b a b b~~ a b a b b
~~A B C E~~ A B B C D
C

0 0 0 0 0 1 → 1 occ

→ Case 2:

a b b a b b
A B D B C D
0 0 0 1 0 0 1 → 2 occ

SPPU-TE-COMP-CONTENT – KSKA GIL

→ Case 3: a b b b a b b

A B C D A B C D

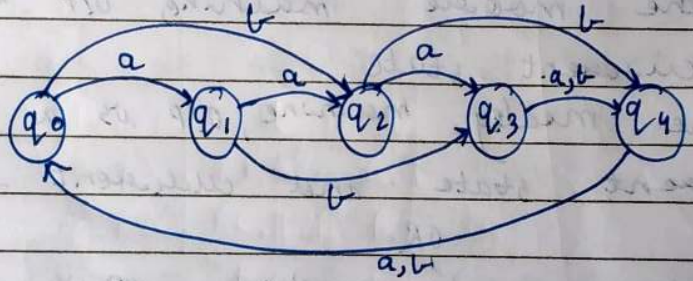
0 0 0 1 0 0 0 1 → 2 occ

Q. Construct moore machine $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$ for following:-

1) aabab 2) abbb 3) ababb

States	a	b	O/P's
→ 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

1) aabab



a a b a b

q_0 q_1 q_2 q_4 q_0 q_2

0 0 0 1 0 0

2) abbb

a b b b

q_0 q_1 q_3 q_4 q_0

0 0 0 0 0

3) ababb

a b a b b

q_0 q_1 q_3 q_4 q_0 q_2

0 0 0 1 0 1

SPPU-TE-COMP-CONTENT – KSKA Git

Q. Construct a mealy machine that gives 2's complement of any binary I/P. (Assume that the last carry bit is neglected).

Soln:

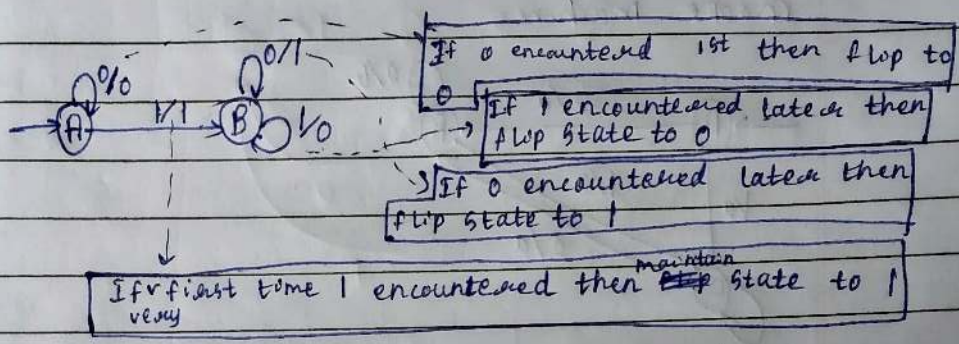
2's complement = 1's complement + 1

Eg: 10100 | Eg: 11100 | Eg: 1111

1's: 01011 | 1's: 00011 | 1's: 0000

\+ 1 | \+ 1 | \+ 1

2's: 01100 | 2's: 00100 | 2's: 0001



* Conversion of Mealy machine to Moore machine (using Transition table)

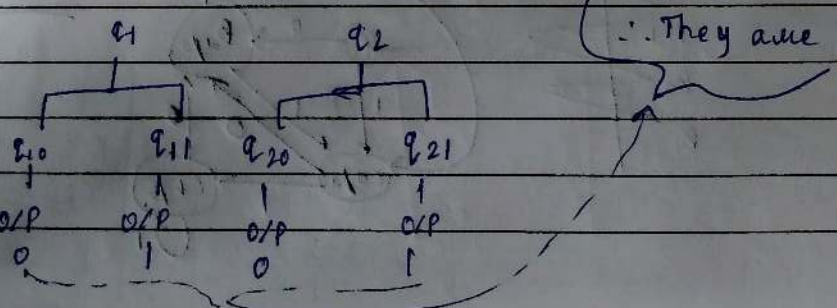
Q. Convert the given Mealy machine to its equivalent Moore machine.

State	a	b
→ 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	$q_1, 0$	$q_0, 1$

q_0 has only one O/P = 1 ∴ It is not split

q_1, q_2 both have two O/P's = 0,1 ∴ They are split

Sol:



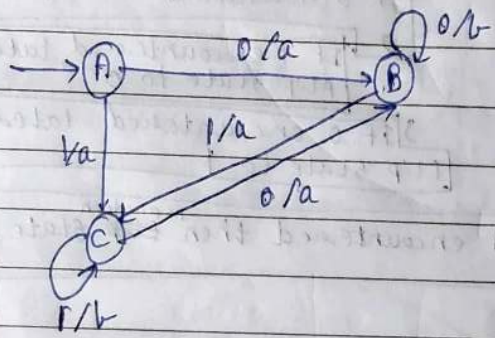
SPPU-TE-COMP-CONTENT – KSKA Git

State	a	b	O/P
→ 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

→ in table q_0 has only one o/p = 1

→ in table q_{10} has only 1 o/p = 0

Q. Convert the mealy machine to its equivalent moore machine.

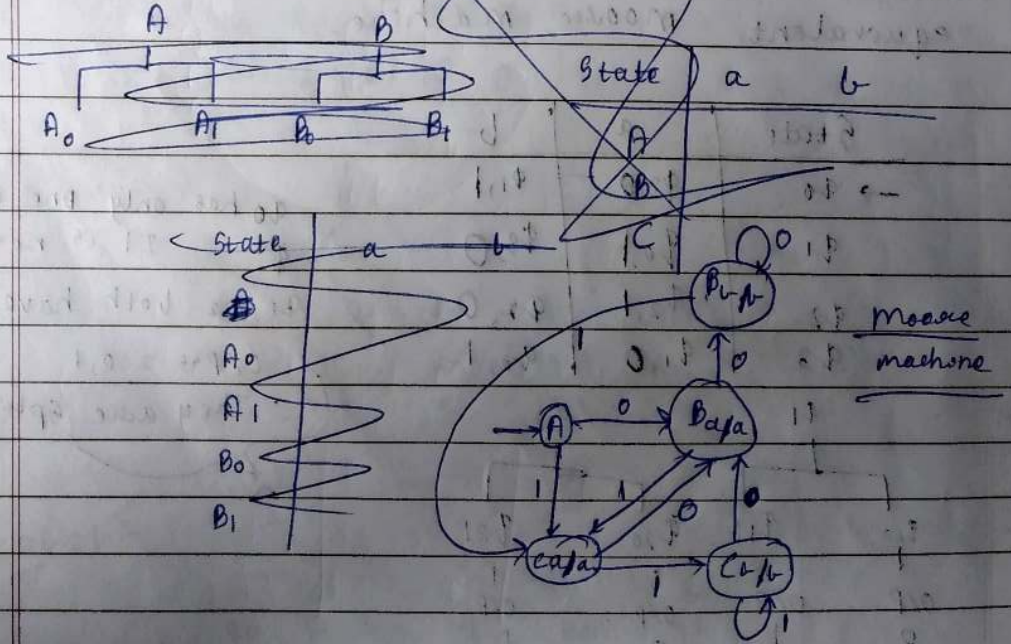


Soln:

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

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

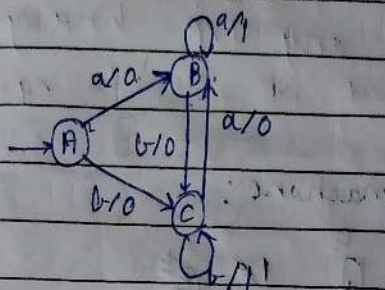
$\Delta = \{a, b\}$



SPPU-TE-COMP-CONTENT – KSKA Git

Date
Page

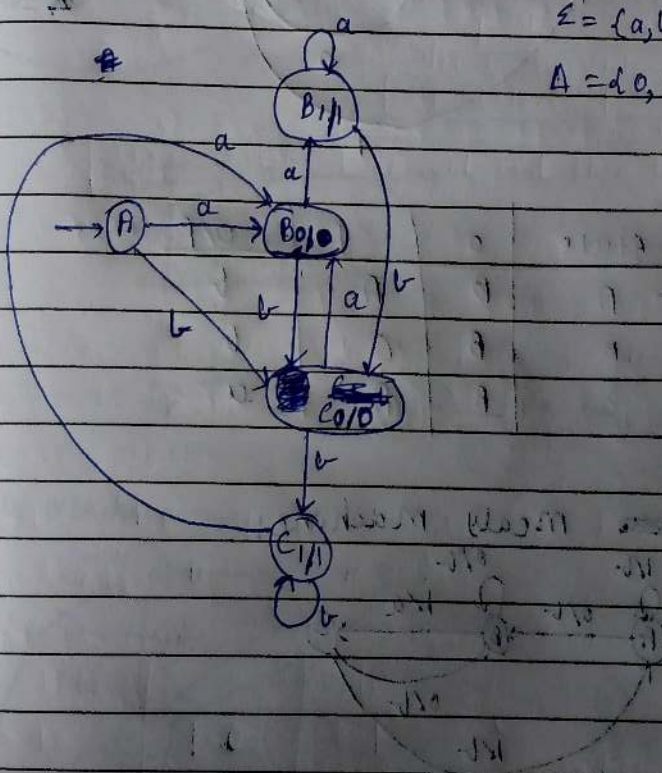
8. Given below is a mealy machine that prints '1' whenever the sequence 'aa' or 'bb' is encountered in any input binary string from Σ^* where $\Sigma = \{a, b\}$. Design the equivalent moore machine for it



Soln:

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

$A = \{0, 1\}$



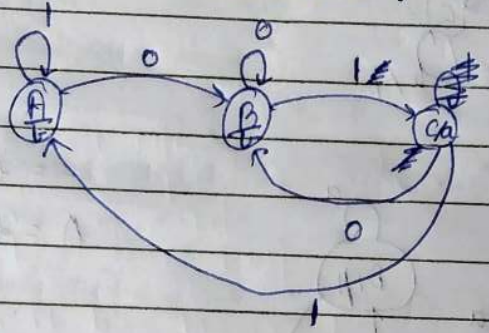
SPPU-TE-COMP-CONTENT – KSKA Git

* Conversion of Moore machine to Mealy machine

Q. Construct a Moore machine that prints 'a' whenever the sequence '01' is encountered in any I/P binary string and then convert it to its equivalent mealy machine.

Ans.

Moore machine:

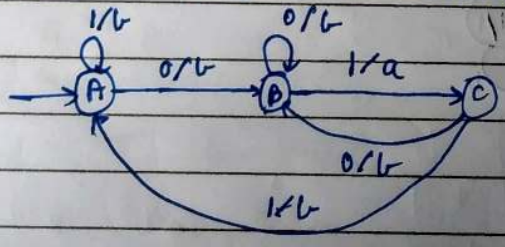


$\Sigma = \{0, 1\}$

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

State	0	1	O/P
→ A	B	A	b
B	B	C	b
C	B	A	a

Mealy machine:



Q. Convert the given Moore machine to its equivalent Mealy machine.

Moore:

State	0	1	O/P
→ q_0	q_1	q_2	1
q_1	q_3	q_2	0

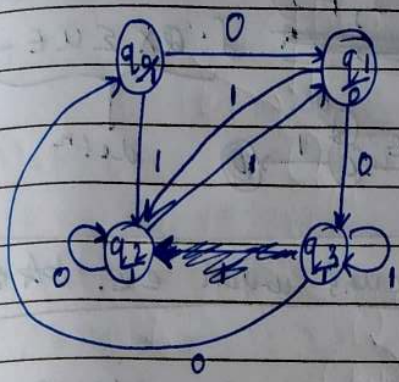
SPPU-TE-COMP-CONTENT – KSKA Git

q_2	q_2 q_1	z 1
q_3	q_0 q_3	1

Soln:

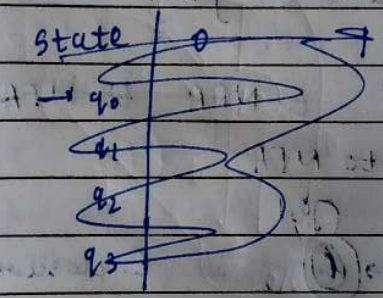
$\Sigma = \{0,1\}$ $\Delta = \{0,1\}$

Moore machine

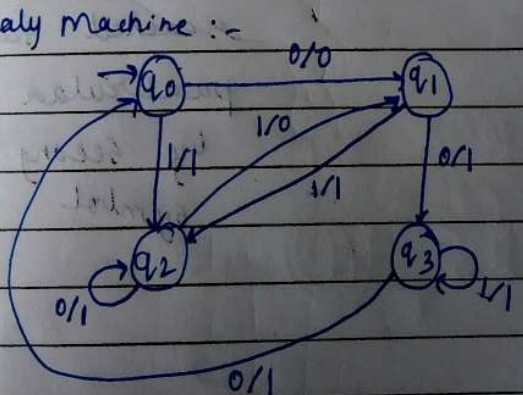


Mealy machine :-

State	0	1
→ 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



Mealy machine :-



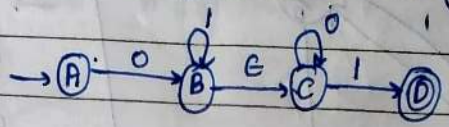
SPPU-TE-COMP-CONTENT – KSKA Git

* Epsilon (ε)-NFA

ε-NFA
↳ empty symbols

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

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



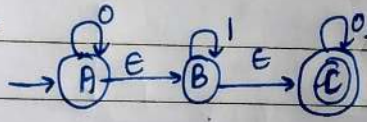
Q. What is ε-NFA. Discuss with ex. ★★★★

- Every state on ε goes to itself.

* Conversion of ε-NFA to NFA (Not likely asked)

Q. Convert ε-NFA to NFA

→



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