Theory of Computation

Subject : 2018 Date :

1(a) Define Alphabet, string, language. Discuss basic operations of language. 3

2019-(C)

Alphabet: An alphabet is a finite set of symbols, such as $\{a, b\}$ or $\{0, 1\}$ or $\{A, B, C, \ldots, Z\}$

We will usually use the Greek letter Σ to denote the alphabet. example, $\Sigma = \{0, 1\}$ is an alphabet of binary digits

String: A String is a finite sequence of symbols selected from some alphabet. It is generally denoted as w. Eg.. example for alphahet $\Sigma = \{0, 1\}$, $w = 010101$ is a string.

A string over Σ is a finite sequence of symbol in Σ

For a string x, $|x|$ stands for the length (the no. of symbols) of x. and the length of a string denoted as $|w|$.

Language.

Σ^* is the set of all possible string (often power set (need not be unique have or we can say multiset) of string) so this implies that language is a subset of Σ^*

⇒ Empty string is the string with zero occurance of symbol represented as ε.

Subject :..... Date :

Number of Strings (of length 2) generated over alphabets {a, b} - a a, ab, ba, bb.

Length of String $|w| = 2$, Number of strings = 4.

Language: A language over Σ (alphabet) is a subset of Σ^* (the set of all strings over Σ).

This means that language L is subset of Σ^*.

An example is English language, where the collection of legal English words is a set of strings over the alphabet that consists of all the letters.

The language formed over 'Σ' can be finite or Infinite.

Example of Finite language.

$\Sigma = \{a, x, y\}$

$L_1 = \{\text{set of string of 2}\}$

$L_1 = \{xy, xx, yx, yy\}$

Example of Infinite language:

$L_1 = \{\text{set of all strings starting with 'b'}\}$

$L_1 = \{ba, baa, babb, bbb, \ldots\}$

Subject : ..... Date : [ ][ ][ ]

2018 1 (b) Define & Classify finite Automata

2019 1(b)

Finite automata is an abstract computing device. It is a mathematical model of a system with discrete inputs, outputs, states and set of transitions from state to state that occurs on input symbols from alphabet Σ.

Its representation;
- → Graphical (Transition Diagrams or Transition table)
- → Tabular (Transition table)
- → Mathematical (Transition function or Mapping function)

Formal definition of FA.

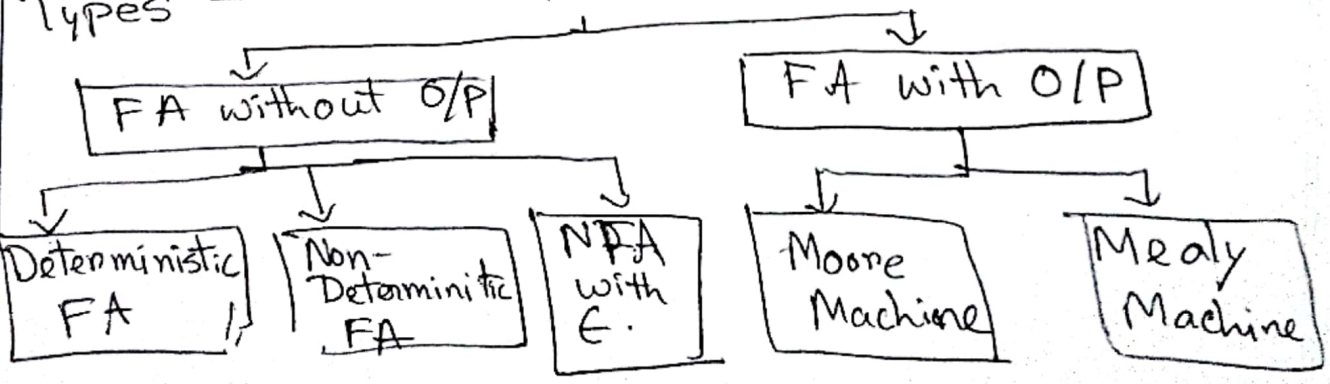
A finite automata is 5-tuples; they are -

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

Q : finite set of states.
Σ : finite set of alphabets
δ : $Q \times \Sigma \to Q$ transition function
$q_0 \in Q$: start state or initial state
$F \subseteq Q$: Set of accept states / final states.

19-1C
18-1C

Types — Finite Automata



SAT SUN MON TUE WED THU FRI

Date

1 (b) Define and Classify Finite Automata

Ans: FA is a simple idealized machine used to ~~pattern~~ recognize patterns within input taken from some character set (or alphabet). The job of an Finite Automaton is to accept or reject an input depending on whether the pattern defined by FA occurs in input.

A DFA (Deterministic) Finite Automaton consists of:

1) A finite set S of N states : Q
2) a special start state / input alpha.
3) a set of final (or accepting) state.
2) A finite input alphabet : Σ
3) The initial state: $q_0 \in Q$
4) The set of accepting states: $A \subseteq Q$
5) The transition function: $\delta : Q \times \Sigma \to Q$

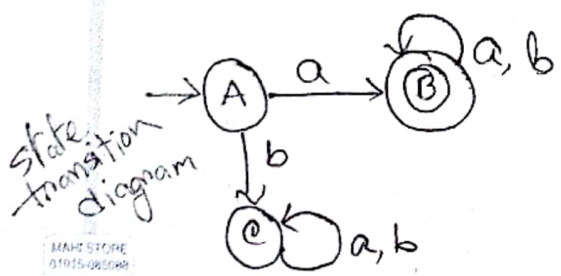
DFA

A Finite automaton (FA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$ where.

For any element q of Q and any symbol $\sigma \in \Sigma$, we interpret $\delta(q, \sigma)$ as the state of which the FA moves, if it is in state q and recives the input σ

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

L(M) = set of all strings start with a
= {a, ab, aab, abb, aaab, aaaa, ---}



state transition diagram

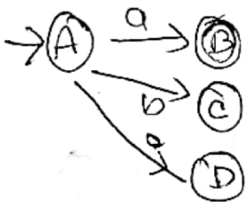
Finite automaton

String: a a b b
A → B → B → B ⇒ Yes in Language

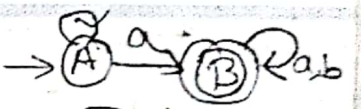
String: b a b a
A → C → C → C ⇒ Not in language

Introduction to NFA

Non-deterministic Finite Automata:



From one to 0, 1 or multiple no. of states.



Deterministic

5-Tuple $(Q, \Sigma, q_0, A, \delta)$ where

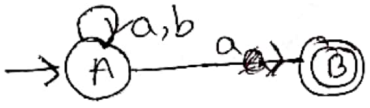
- Q = finite set of states
- Σ = finite input alphabet
- $q_0 \in Q$ = initial state
- $A \subseteq Q$ = set of accepting states.

→ different from DFA

$\delta : Q \times (\Sigma \cup \{\Lambda\}) \to 2^Q$ (subset of Q) $\cdot \delta : Q \times \Sigma \to 2^Q$

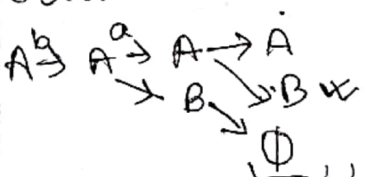
Example: $L = \{$ ends with a$\}$, $\Sigma = \{a, b\}$

$= \{a, aa, aaa, ba, \dots\}$



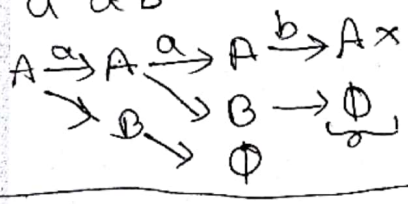
same alphabet 'a' দিয়ে 2টা state এ travel possible, & last state/final state এর transition গুলা উল্লেখ করা লাগেনি।

baa.



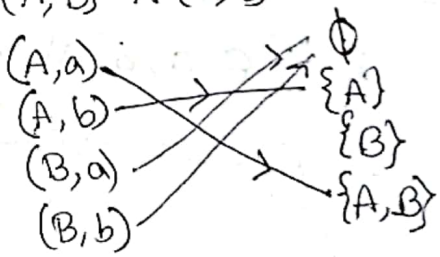
এরকম final state থাকতে হবে.

aab



$Q \times \Sigma \to 2^Q$

$\{A, B\} \times \{a, b\}$



Note: DFA : $Q \times \Sigma \to Q$

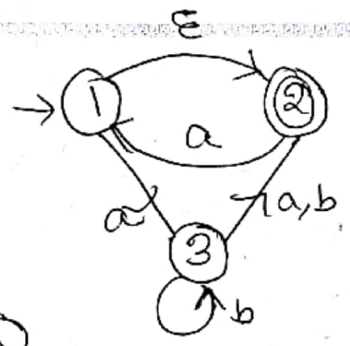
NFA : $Q \times \Sigma \to 2^Q$

All DFA are NFA.

ε-NFA to DFA conversion

SAT SUN MON TUE WED THU FRI

Date :



Rules

1) STT for ε-NFA
2) Find ε-closure for all states
3) Make STT for DFA

$$E\left(\bigcup_{r \in R} \delta(r,a)\right)$$

4) $q_0' = E(q_0)$

ε-NFA মানে একটা ε-edge থাকবে

Step-1

	a	b
→1	{3}	ϕ
*2	{1}	ϕ
3	*{2}	{2,3}

STT of ε-NFA.

Step-2

ε-closure(1) = {1, 2}
ε-closure(2) = {2}
ε-closure(3) = {3}

ε-closure মানে হলো যে state টাকে আমরা select করছি ঐটা ε পেলে মানে কোনো value না পেলে কোন কোন state এ যেতে পারে।
মানে হলো কোনো string না দেখে সেই state টা কোথায় যেতে পারে।

$q_0' = E(\{q_0\})$
$= E(1)$
$= \{1, 2\}$

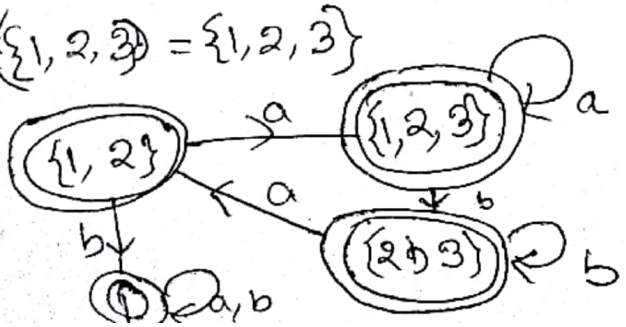
{1,2} → {1, 3} তে।
এবার {1,3} এর ε-closure বের করতে হবে।

$E(\{1, 3\}) = \{1, 2, 3\}$
ε-closure(1) ∪ ε-colos(3)
{1, 2} ∪ {3} = {1, 2, 3}

Step-3

	a	b
→*{1, 2}	*{1,2,3}	ϕ
*{1, 2, 3}	*{1,2,3}	*{2,3}
*{2, 3}	*{1,2}	*{2,3}

E{1, 2, 3} = {1, 2, 3}



LA
to Regular

SAT SUN MON TUE WED THU FRI

Date

3) $G3 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to aA \mid a, B \to bB \mid b\})$

$S \to AB \to ab$

$S \to AB \to aAbB \to aabb$

$S \to AB \to aAb \to aabb$

$S \to AB \to abB \to abb$

$L(G3) = \{ab, a^2b^2, a^3b^3, ab^2, \ldots\}$

$= \{a^m b^m \mid m \geq 1 \text{ and } n \geq 1\}$ $\quad m, n \geq 1$ $\quad m, n > 0$

Melay Machine: A melay machine is a FSM (finite state machine) whose output depends on the present state as well as the present input.

[o/p depends on → present state + i/p]

6 tuples: $(Q, \Sigma, O, \delta, X, q_0)$

$Q = Q$ is finite set of states

$\Sigma = \Sigma$ is finite set of symbols called i/p alphabet

$O = O$ is set of symbols called o/p alphabet

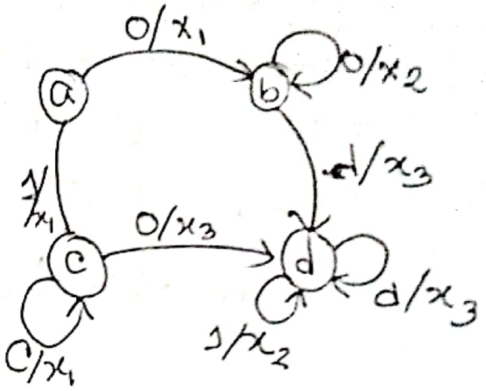
$\delta = \delta$ is i/p transition fun $\delta : Q \times \Sigma \to Q$

$X = X$ is o/p transition function where $X : Q \times \Sigma \to O$

q_0 = initial state from where any input is processed $(q_0 \in Q)$

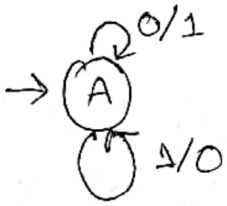
Example: State table of Mealy Machine

Present State	Next State			
	Input=0		input=1	
	State	O/P	State	O/P
→a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2

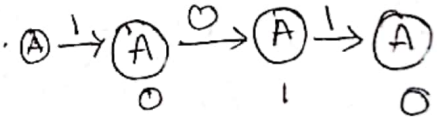


Example: Construct Mealy M/c that produce is complement of any binary input string.

Sol^n: $\Sigma = \{0, 1\}$



⇒ 101



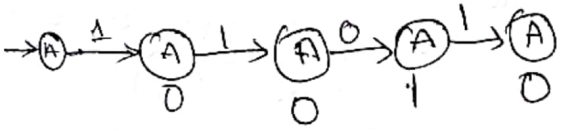
O/P-010

⇒ 01001 - I/P

1's Comp

10110 → O/P

1101 ⇒ i/p

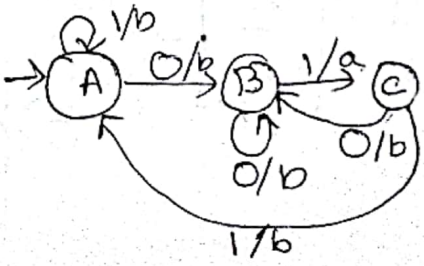


0010 ⇒ O/P

example: Construct a mealy m/c that points 'a' whenever the sequence '01' is encountered in any i/p binary string.

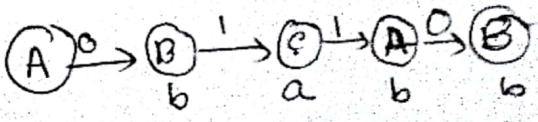
Sol^n: $\Sigma = \{0, 1\}$ i/p

O/P $O = \{a, b\}$



01 0 1 00 1
a a a

0110..



Mealy m/c accepting lang. consisting of strings from Σ^*, where $\Sigma = (a, b)$ & the strings should end with either aa & bb.

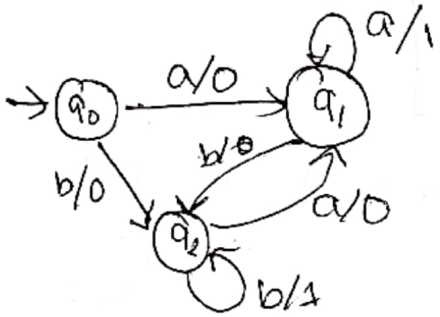
Soln: $\Sigma = \{a, b\}$ O/P $O = \{0, 1\}$

String should end with either aa or bb.

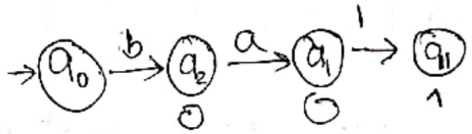
Assume that the string ends with aa or bb.

generates O/P = 1

else O/P = 0.



baa



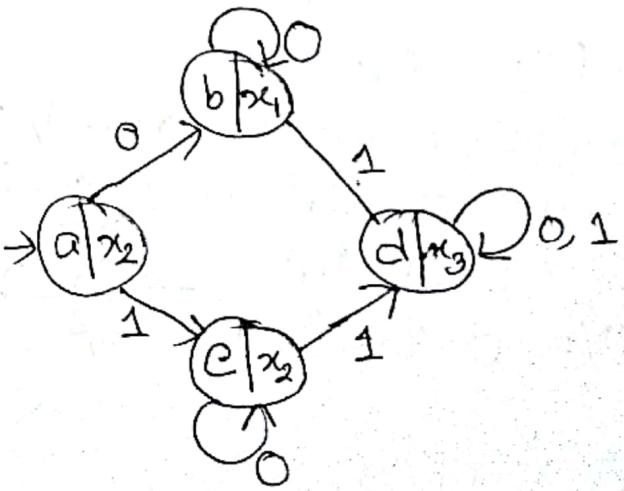
Moore: Is a FSM whose output depends on only the present state. A moore Machine can be described by a 6 tuple $(Q, \Sigma, O, \delta, x, q_0)$

Q: finite set of states, Σ: finite set of symbols i/p alphabet

O: finite set of o/p alphabet, δ: i/p transition fun^c $\delta: Q \times \Sigma \to Q$

x: o/p transition fun^c $x: Q \to O$, q_0: initial state from where any i/p is processed ($q_0 \in Q$)

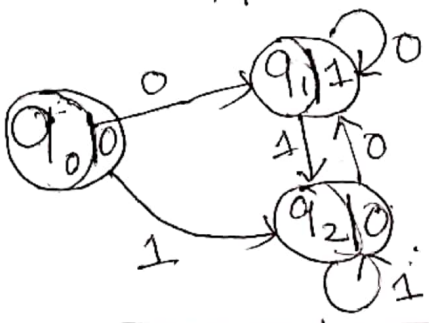
Present table	Next State		Output
	Input=0	Input=1	
→ a	b	c	x_2
b	b	d	x_1
c	c	d	x_2
d	d	d	x_3



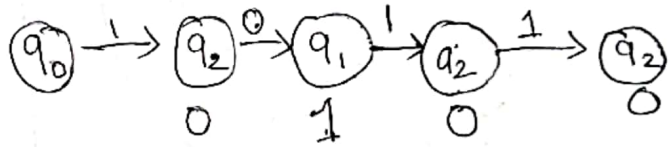
Example: Design Moole M/c to generate 1's complement of a given binary no.

Soln: If i/p 0 → O/P 1
1 → 0

3 states are required → Start state, Second for tracing 0's as i/p & produce 1 as O/p, third is tracing 1's as i/p & produce 0 as O/p.



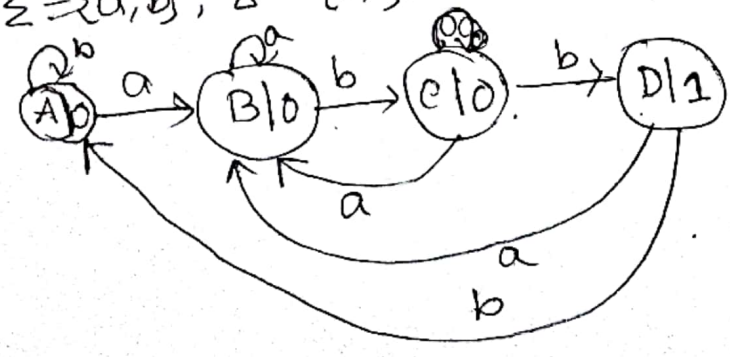
⇒ 1011 (i/p)



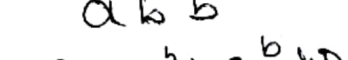
Current State	next state		O/P
	0	1	
→q_0	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

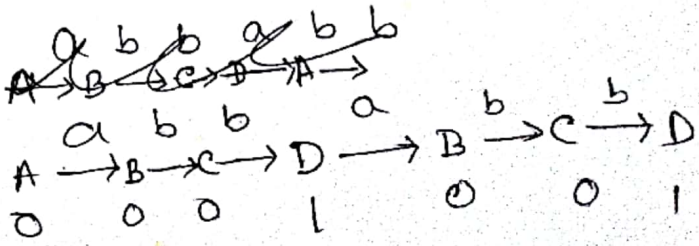
*Construct a Moore Machine that counts the occurance of sequence 'abb' in any input strings over {a, b}

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



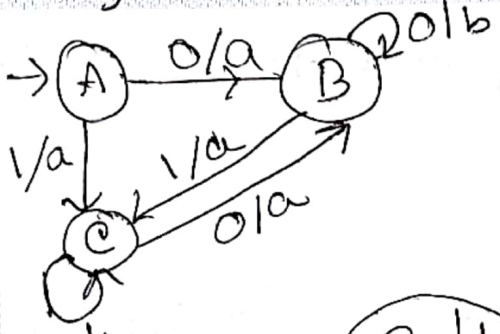
a b b

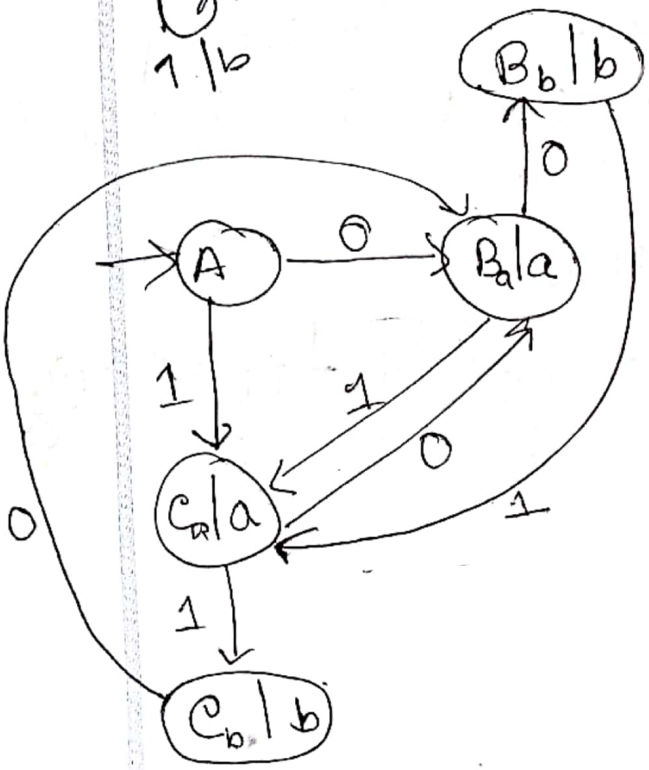




Conversion

Mealy to Moore





Mealy to Moore ⇒
no. of states increased

Moore → Mealy ⇒ no. of states ~~were~~ are same.

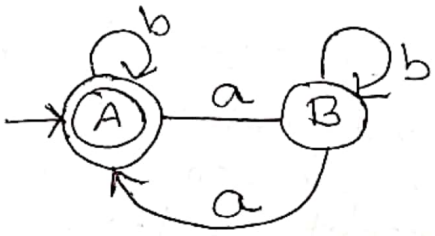
2018

Subject :..... Date :

2(a) Construct an DFA accepting the following languages:

i) $\{w \in \{a,b\}^* \mid w$ has even number of a's$\}$

$L(M) = \{ \varepsilon, bb, aab, aba, aabaa, abab, \ldots \}$

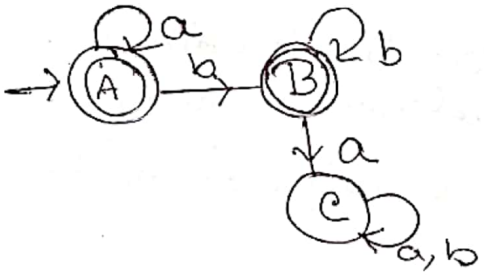


ii) $\{a^n b^m \mid m,n \geq 0\}$ [এটায় sequence matter করে, a এর পরে must b আসবে।

$L(M) = \{ \varepsilon, a, b, ab, aab, abb, aabb, \ldots \}$

[b এর পরে a accepted না]

bax
(ab ✓
aaa ..b ✓
(b--ax
ba bax
b এর



3(a) Define regular expression. Write a regular for a language containing the strings starting & ending with same symbol over alphabet $\{a,b\}$

Regular language expression: If Σ is an alphabet, the set R of regular languages over Σ is defined as follows.

1. The language $\emptyset$ is an element of R, and for every $a \in \Sigma$, the language $\{a\}$ is in R

2018 2(b) Construct an NFA that accepts a binary language ending with 001. Convert it to a DFA using subset construction method.

Solution: Given, $\Sigma = \{0, 1\}$.

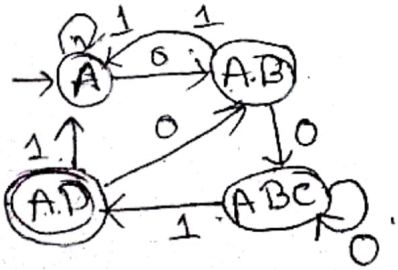
The $L(M) = \{w/w$ ends with $001\}$

The state transition table of the NFA.

	0	1
→A	{A, B}	{A}
B	{C}	{-}
C	{ }	*{D}
*D	{ }	{ }

The STT for DFA using STT of NFA.

	0	1
→ [A]	[AB]	[A]
[AB]	[ABC]	[A]
[ABC]	[ABC]	*[AD]
*[AD]	[AB]	[A]



DFA.

Subject :..... Date :

(2) For any two language L_1 and L_2 in R, the three languages $L_1 \cup L_2$, $L_1 L_2$ and L_1^* are elements of R.

~~The lang. thre~~

Regular expression: The language accepted by finite automata can be easily described by simple expression called Regular expression. It is the most efficient way to represent any language

The languages accepted by some regular expression are reffed to as Regular string.

A regular expression can also be described as a sequence of pattern that defines a string.

Regular expression are used to match character combinations in string.

Example:

In a regular expression, x^* means Zero or more occurrence of x. It can generate

$\{\varepsilon, x, xx, xxx, xxxx, \ldots\}$

In a regular expression x^+ means One or more occurrence of x. It can generate

$\{x, xx, xxx, \ldots\}$

Subject : ..... Date :

The various operations on regular language are –

Union: If L & M are two regular language then their union, L ∪ M is also a union.

1. $L \cup M = \{ s \mid s \text{ is in } L \text{ or } s \text{ is in } M \}$

Intersection: If L and M are two regular language then their intersection is also an intersection.

1. $L \cap M = \{ st \mid s \text{ is in } L \text{ and } t \text{ is in } M \}$

Kleen closure: If L is a regular language then its Kleen closure $L1^*$ will also be a regular language.

1. L^* = Zero or more occurrence of language L.

Example 1: The regular expression of language accepting all combination of a's over the set $\Sigma = \{a\}$.

Sol^n: All combination of "a" means a may be zero, single double, and so on. If a is appearing zero times, that is null string.

R.E or $R = a^*$

Example-2: ..... all ..... of a's except null string over the set $\Sigma = \{a\}$

Sol^n Regular expression for language $L = \{a, aa, aaa, \ldots\}$

This set indicates that there is no null string.

$R = a+$

3. All string containing any number of a's and b's.

$r.e = (a+b)^*$

$L = \{\varepsilon, a, aa, bb, b, ab, ba, \ldots\}$

Subject :..... Date : [ ][ ][ ]

* Write regular expression for a language containing the string starting and end ending with same symbol over alphabet {a,b}.

$\underline{a} \cup \underline{b} \cup \underline{b\Sigma^*b} \cup \underline{a\Sigma^*a}$

$a + b + b(a+b)^*b + a(a+b)^*a$

a
b
a---a
b---b

L = {w | w contains at least two 1's} $\Sigma = \{0, 1\}$

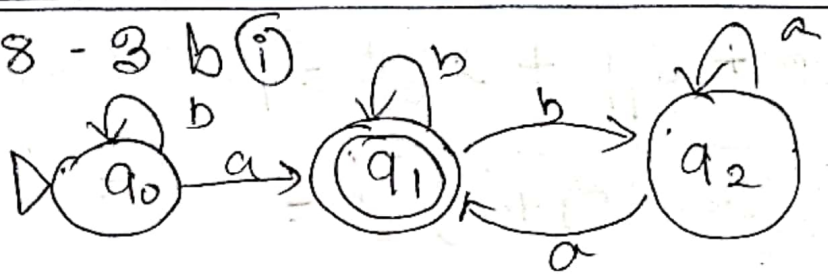
$\Sigma^*1\ \Sigma^*\ 1\Sigma^*$

$(0+1)^* + 1 + (0+1)^* + 1 + (0+1)^*$

L = {11, ...

Subject :..... Date :

2018 - 3 b (i)



Here,

$$q_0 = \in + q_0 b \quad \text{---(I)}$$

$$q_1 = q_0 a + q_1 b + q_2 a \quad \text{---(II)}$$

$$q_2 = q_1 b + q_2 a \quad \text{---(III)}$$

From (I) $\Rightarrow$ $q_0 = \in + q_0 b$ [Arden's Theorem]

$$= \in b^*$$

$$q_0 = b^* \longrightarrow \text{(IV)}$$

$R = Q + RP$
$= QP^*$

From (III) $\Rightarrow$ $q_2 = q_1 b + a q_2$

$$= q_1 b a^* \quad \text{[Arden's Theorem]}$$

$$\therefore q_2 = q_1 b a^* \quad \text{---(V)}$$

Putting values from (IV) & (V) in (II) we get,

$$q_1 = b^* a + q_1 b + q_1 b a^* a$$

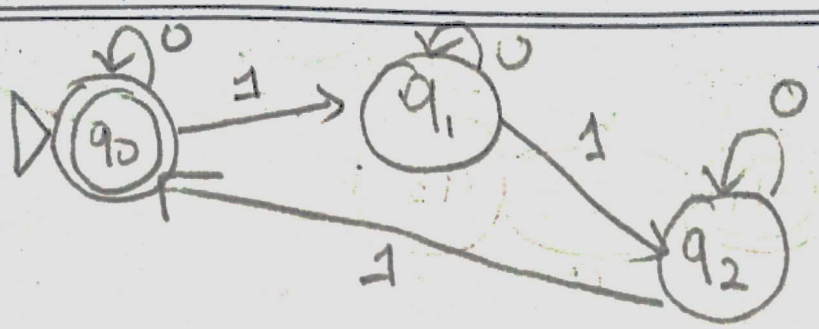
$$= b^* a + (b + b a^* a) q_1$$

$$q_1 = b^* a (b + b a^* a)^*$$

Ans.

Subject :..... Date :

3 (b) (ii)



$$q_0 = \epsilon + q_0 0 + q_2 1 \longrightarrow (i)$$

$$q_1 = q_1 0 + q_0 1 \longrightarrow (ii)$$

$$q_2 = q_2 0 + q_1 1 \longrightarrow (iii)$$

(iii) $\Rightarrow q_2 = q_1 1 + q_2 0$

$= q_1 1 0^* \longrightarrow (iv)$

(ii) $\Rightarrow q_1 = q_0 1 + q_1 0$

$= q_0 1 0^* \longrightarrow (v)$

(i) $\Rightarrow q_0 = \epsilon + q_2 1 + q_0 0$

$= \epsilon + q_1 10^* 1 + q_0 0$ [from (iv)]

$= \epsilon + q_0 10^* 10^* 1 + q_0 0$ [from (v)]

$= \epsilon + (10^* 10^* 1 + 0) q_0$

$q_0 = \epsilon (10^* 10^* 1 + 0)^*$

$= (10^* 10^* 1 + 0)^*$

left a non-terminal

Subject : ..... Date : [ ][ ][ ][ ][ ][ ][ ][ ]

3 (c) What is the regular language represented by the following left-linear grammar? 2·75

$A \to Aa \mid Ab \mid Ba$, $B \to Cb$, $C \to \varepsilon$

$A \to Aa$
$\to Aba$ [A → Ab]
$\to Baba$ [A → Ba]
$\to Cbaba$ [B → Cb]
$\to \varepsilon baba$ [C → ε]
$\to baba$

$A \to Ab$
$\Rightarrow Abb$ [A → Ab]
$\to Abbb$ [A → Ab]
$\to Babbb$ [A → Ba]
$\Rightarrow Cbabbb$ [B → Cb]
$\to \varepsilon babbb$ [C → ε]
$\to babbb$

$A \to Ba$
$\Rightarrow Cba$ [B → Cb]
$\to \varepsilon ba$ [C → ε]
$\to ba$

$A \to Aa$
$\to Aaa$ [A → Aa]
$\to Aaaa$ [A → Aa]
$\to Baaaa$ [A → Ba]
$\Rightarrow Cbaaaa$ [B → Cb]
$\Rightarrow \varepsilon baaaa$ [C → ε]
$\to baaaa$

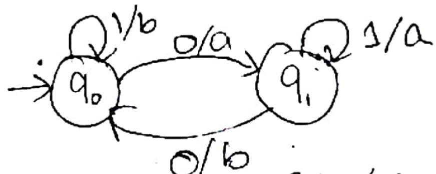
$A \to Aa$
$\to Aba$
$\to Abba$
$\to Babba$
$\to Cbabba$
$\to babba$

$L(G_1) = \{ba, baba, babbb, baaaa, \ldots\}$

$= \{ba b^m a^n \mid m, n \geq 1\}$

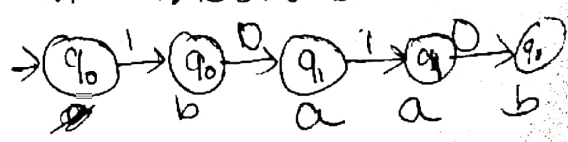
$ba b^* a^*$

4(a) Contrast between Moore machine & Mealy machine

Moore Machine	Difference	Mealy Machine
Output depends only on current state.	1 Definition.	Output depends on the present state as well as on the present input.
More states than Mealy machine	2. States	Fewer states than Moore machine.
Synchronous output and state generation	3. Output generation	Asynchronous output generation.
The design of Moore model is easy	4. Design	The design of Mealy model is complex
Output is a set of state	5. Output type.	Output is a set of transition
To design, more hardware is necessary	6. Hardware	It's easier to design with less hardware.
Can refer a counter as a Moore machine	7 Counter	Can not refer a counter as Mealy machine
	8. Example.	

i/p seq 1010 - O/P seq aabab

If sequence 1010
o/p -> baab



Moore মেশিনে 0 আসলে আগের state output এ দেখাবে

Subject : ..... Date :

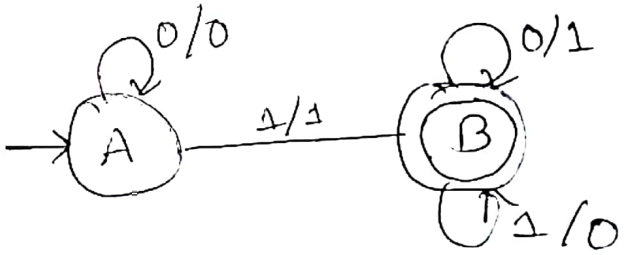
4 (b) Construct a Mealy machine that takes binary numbers as input and produces 2's complement of that number as output. Assume that the string is read ~~LSM~~ LSB to MSB and end carry is discarded.

Ans: 2's Complement logic:- First calculate 1's complement of binary number, Convert 1 to 0 and 0 to 1 and then add 1 to it. For example: if binary number is 1011 then 1's complement is 0100 and it's 2's complement 0101

1's 0 1 1
2's +1
1 1 0 0

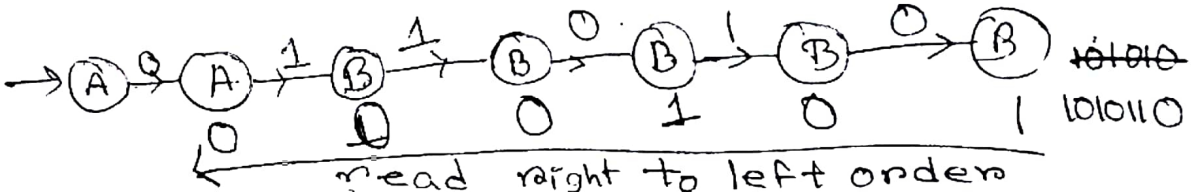
Design Mealy machine:

1. Take initial state A.
2. If there are n number of zeros at initial state, it will remain at initial state.
3. Whenever 1st input 1 is found then it gives output 1 and goes to state B.
4. In state B, if input is zero, output will be 0. if input is 1 then output will be 1.
5. And then set state B as final state.



The approach:
1. Start from right to left
2. Ignore all 0's
3. When 1 comes ignore it and then take 1's complement of every digit

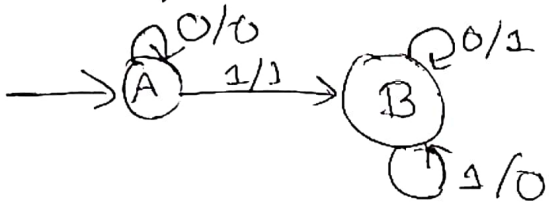
example: 010110 Read right to left.

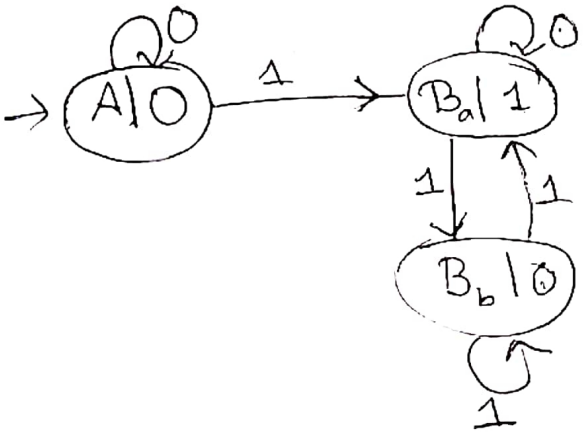


Subject :..... Date :

4 (c) Convert above Mealy machine of 4(b) into Moore machine..

Sol^n: The Mealy machine for 2's Complement.





Digital Produ...

Subject : 2018 5-b Date :

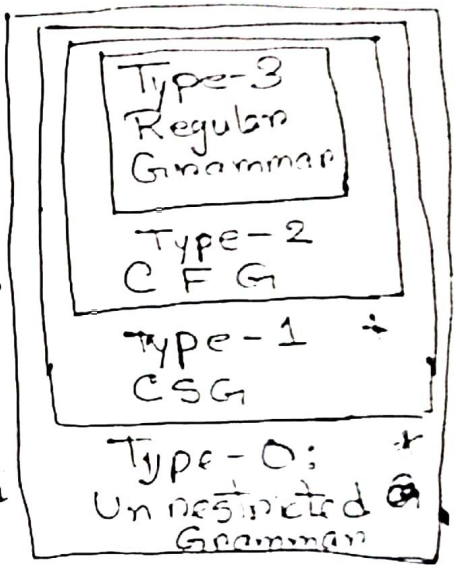
Chomsky Hierarchy of Grammar with respect to formal language.

A grammar G can be defined as $G = (V, T, P, S)$

V = Finite set of variables, T = Finite set of terminals

P = Production , S = Start Production

The production rule must be in the form $\alpha \to \beta$

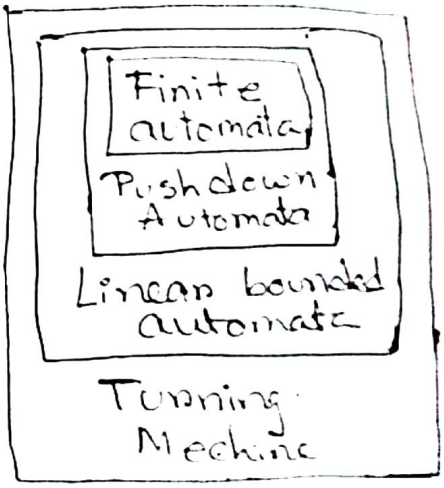


Type-0: Unrestricted Grammar: There is no restriction on the grammar rules of these types of language. These language can be efficiently modeled by turing machine. Example: $bAa \to aa$, $S \to s$

Type-1: Context Sensitive Grammar: Repesents context sensitive language. The rules:

1. The context s.G has more than one Symbol on left hand side of their production rules.
2. No. of symbols on L.H.S must not exceed no. of symbols on R.H.S.
3. The rule $A \to \varepsilon$ is not allowed unless A is start symbol. It doesn't occur in R.H.S of any rule.
4. Type-1 should be in type-0. $V \to T$. Count symbol of V. is less than or equal to T.

$S \to AT$, $T \to xy$, $A \to a$.



Type-2: Context free grammar: Regular language represented by CFG. Type-2 should be in type-1 Allows $A \to \alpha$ where A is any single non-terminal and α is any combination of terminals and non-terminals. $A \to aBb$, $A \to b$, $B \to a$

Type-3: Regular Grammar: Contains Regular language of those languages which can be described using regular expression. Can be modeled by NFA or DFA. Type-3 is most restricted form of grammar. Type-3 should be Type-2 and Type-1.

Type-3 ~~shold~~ should be in form $V \to T^*V / T^*$

example: $A \to xy$

Digital Products

Subject :..... Date :

5 (a) What do you mean by derivation? What is the function of it? Define sentence & sentential form. (1·75)

⇒ Derivation is a sequence of production rules. It is used to get input string through these production rules. During parsing, we have to take two decisions. These are:

① We have to decide the non-terminal which is to be replaced.

② We have to decide the production rules by the non-terminal will be replaced.

Sentential form: Is the start symbol S of a grammar or any string in $(V \cup T)^*$ that can be derived from S. Consider linear grammar $(\{S, B\}, \{a, b\}, S, \{S \to aS, S \to B, B \to bB, B \to \lambda\})$. A derivation using this grammar might look like $S \Rightarrow aS \Rightarrow aB \Rightarrow abB \Rightarrow abbB \Rightarrow abb$. Each of $\{S, aS, aB, abB, abbB, abb\}$ is a sentential form, Because this grammar is linear, each sentential form has at most one variable. Hence there is never any choice about which variable to expand next.

b) Construct context-free grammar that generates the following languages: (4)

i) $\{w \in \{a,$ with same symbols

$L = \{\varepsilon, 0, 1, 00, 11, 010, 000, 1001, 0101010, \ldots\}$

$0(0+1)^*0 \mid 1(0+1)^*1 \mid \varepsilon \mid 1 \mid 0$

Let, Start variable, $S \to 0S_1 0 \mid 1S_1 1 \mid 1 \mid 0 \mid \varepsilon$

$S \to 0X0 \mid 1X1 \mid 1 \mid 0 \quad S_1 \to S_1 0 \mid S_1 1 \mid \varepsilon$

$X \to 0X \mid 1X \mid \varepsilon$

$CFG, G = \{V, T, P, S\}$

Subject : ..... Date : [ ][ ][ ]

6(a) Construct Context-free grammar separately ~~for~~

i) $\{w \in \{a,b\}^* \mid w$ starts & end with different symbol$\}$

Soln:

P: $S \to aAb \mid bAa$

$A \to aA \mid bA \mid \in$

$L(G) = \{ab, ba, abab, aaab, baaa, baba, \ldots\}$

$CFG, G = (\{S, A\}, \{a, b\}, P, S)$

ii) $w \in \{a,b\}^* \mid w$ is palindrome" [even length & odd length both]

Soln: $L = \{aba, baab, b, bab, \ldots\}$

$V = \{S\}$

$T = \{a, b\}$

$S \to aSa$

$S \to bSb$

$S \to a$

$S \to b$

$S \to \in$

অথবা

$S \to aSa \mid bSb \mid a \mid b \mid \in$

odd length

$S \to aSa$

$\to abSba$

$\to abbba$

even length

$S \to bSb$

$\to baSab$

$\to baaab$

$CFG, G = (\{S\}, \{a, b\}, \in\}, P, S)$

Subject :.....

Date :

(iii) $\{a^{2n}b^n \mid n \geq 1\}$

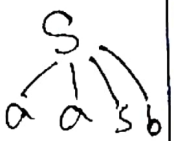
$L = \{aab, aaaabb, aaaaaabbb, \ldots\}$

$S \rightarrow aaSb$

$S \rightarrow aab$

$P: S \rightarrow aaSb \mid aab$

CFG, $G = (\{S\}, \{a, b\}, P, S)$



S

6. b) Define Chomsky normal form. Convert the following CFG to Chomsky normal form (CNF):

$S \to aX \mid Yb$, $X \to S \mid \varepsilon$, $Y \to bY \mid b$

A context free grammar is in CNF if all the production rules satisfy one of the following Condition:

1. Start symbol generating ε. For example $A \to \varepsilon$
2. A non-terminal generating two non-terminal example, $S \to AB$
3. A non-terminal generating a terminal. example: $S \to a$