THEORY OF COMPUTATION

By- Pathik455

DATE: / / PAGE NO.

L-1 : Syllabus:

1) Regular language and finite automata (80%)
   i) language (formal)
   ii) Grammer
   iii) Automata (machine)

2) Context Free language and Pushdown automata. (10%)
3) REC, RE and Turing Machine. (10%)
4) Closure Properties and Undecidability. ⊛
   i) Table **
   ii) language (identification)
   iii) Machine (automata)

L-2

TOC - To understand how computers work?

⊛ 3 pillars of TOC -
   → language
LAG → Automata
   → Grammar

① Language -

→ Symbol : {a, b, c, 0, 1, 2 ...}

→ Alphabet : $\Sigma(a, b)$ finite set of symbol.

→ String : Collection of alphabet, sequence.

→ language : Collection of all strings.

Q. $\Sigma(a,b)$

1) L_1 = strings of length 3.
language (finite language)
$= \{aaa, bbb, aab, baa, aba, abb, bba, bab\}$

2) L_2 = All strings start with a, end with a.

$= \{aa, aaa, aaaa, \ldots, aba, abba, \ldots \infty\}$
= Infinite language.

3) L_3 language with length 0.

$\Sigma^0 = \varepsilon$

L-3 Automata: (Mathematical model, Machine)

Language

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

Finite	Infinite
L_1 = Length 2	L_2 = Atleast 1 a
$\{aa, ab, ba, bb\}$	$\{a, aa, ba, ab, aaa, \ldots \infty\}$

→ Finite Automata
→ Pushdown Automata
→ Linear bound Automata
→ Turing Machine Automata

Does {abba} string is there in language?
Ans: we can find using automata.

(L-4)

Power of Σ : $\Sigma(a,b)$

minimum power of sigma = 0.

2^0 Σ^0 = Set of all strings with length 0.
= ε (epsilon) = NULL or λ (epsilon) NULL

2^1 Σ^1 = " " Length 1.
= $\{a, b\}$

2^2 Σ^2 = $\{aa, ab, ba, aa\}$ $\Rightarrow$ $\Sigma \cdot \Sigma = \{a,b\} \cdot \{a,b\}$

2^3 $\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma = \{a,b\} \cdot \{a,b\} \cdot \{a,b\}$
$= \{aa, ab, ba, bb\} \cdot \{a,b\}$
$= \{aaa, aba, bba, bba, aab, abb, bab, bbb\}$

2^4 Σ^4

2^n Σ^* (Kleene closure) = $(a+b)^*$ (All combinations)
= Infinite language.

$\Rightarrow$ Possible string of length $n = 2^n$.

Σ^+ (Positive closure) = $\Sigma^* - \Sigma^0$
= $\Sigma^* - \varepsilon$

(1.5) Grammar in TOC:

→ A grammar 'G' is defined as quadruple:

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

- V → Variable
- T → Terminal
- P → Production Rule
- S → Start

eg:

$$S \longrightarrow aSb / \varepsilon$$

Grammar, Language.

ans: ε, $a\varepsilon b$, $aaSbb$, $aaaSbbb$, $aaaaSbbbb$ ..... $a^n b^n$

ε, ab, a^2Sb^2, a^3Sb^3, a^4b^4

a^2b^2, a^3b^3

$n \geq 0$

eg:-

$S \longrightarrow SS$	ε, asb, bsa
$S \longrightarrow aSb$	ab, ba
$S \longrightarrow bSa$	$assb$, $basba$
$S \longrightarrow \varepsilon$	

∴ Language ⇒ $n_a(w) = n_b(w)$

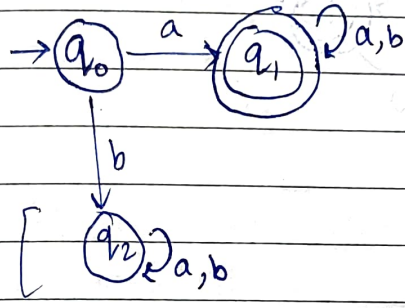
number of a = number of b in any order

L-6 DFA $(Q, \Sigma, \delta, q_0, F)$ {Deterministic finite automata}

- Q: set of finite states
- Σ: Alphabets
- δ: Transition
- q_0: Start State
- F: set of finite final states

$F \subseteq Q$ (subset)

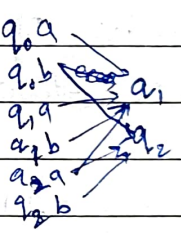
eg: Strings starting with a. {a, ab, aa, aaa ...}



$\delta: Q \times \Sigma \to Q$

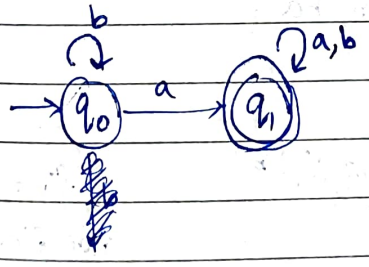
q_0	(a,b)
q_1	
q_2	

⇒



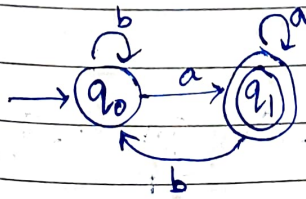
L-7 Construct a DFA which accept a language of all strings (i) → containg 'a' (ii) → end with 'a'. Σ (a, b)

(i)



L = {a, aab, aaa, ba, ...}

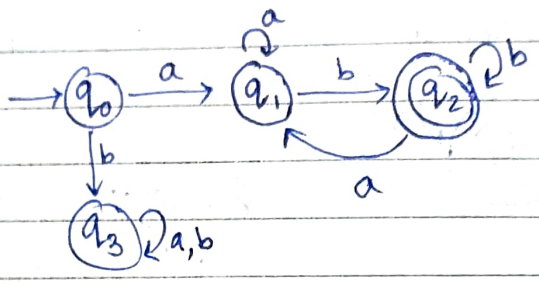
(ii)



L = {a, aa, aaa, ba, bba abba ....}

Q.) Construct a DFA which accept a language of all strings starting with 'a' and ending with 'b'.

ans:- $L = \{ ab, abab, aaab, abbbab, abbaab \}$



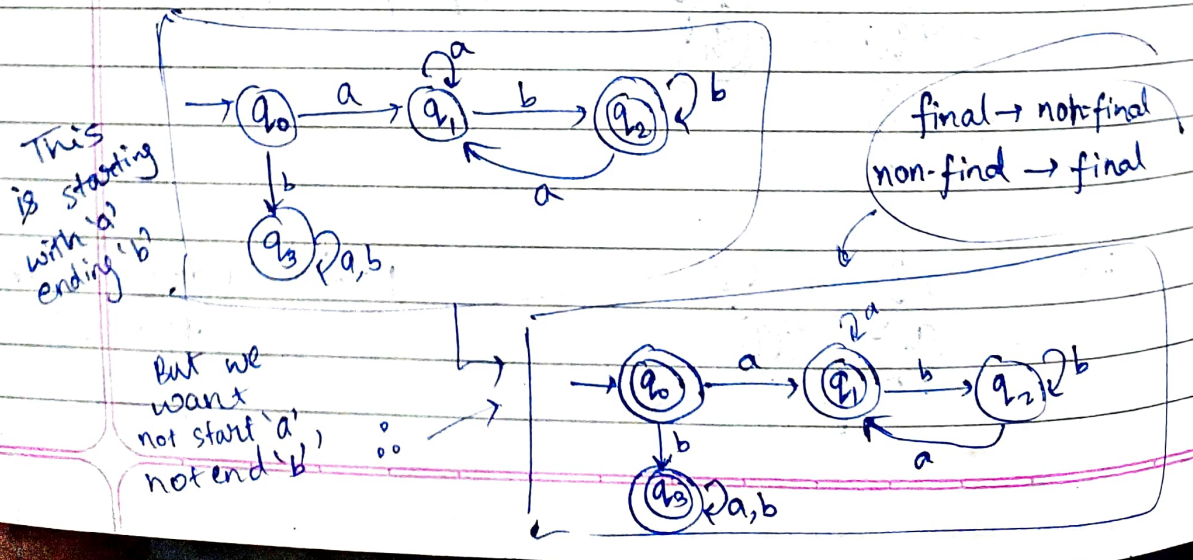
L-10 Make DFA which accept a language of all strings not starting with 'a' ~~and~~ or not ending with 'b'.

ans:- Using DeMorgan's theorem,

$(A \cup B)^c = A^c \cap B^c$

= starting with 'b' and ending with 'b'

$L = \{ \underset{\varepsilon}{\Lambda}, a, b, ba \}$

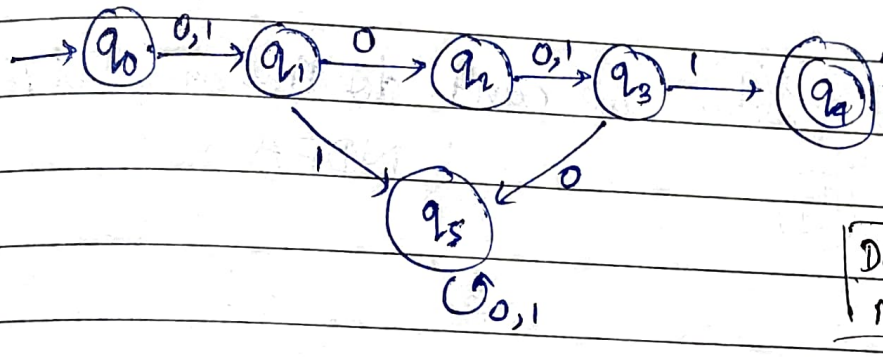


DATE: / / PAGE NO.

(1-11) Design DFA which accepts all strings over (0,1) in which second symbol is '0' and fourth symbol '1'

ans:

0/1 0 0/1 1 0/1 ....

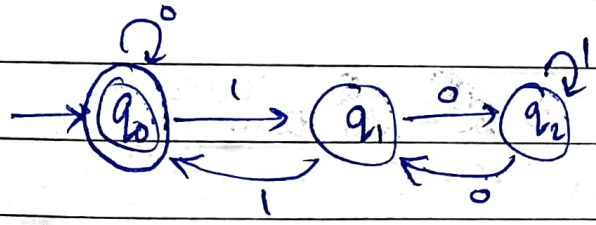


Don't require Multiple trap state

(1-12) Construct a DFA which accept a language of all binary string divisible by three over $\Sigma(0,1)$

remainder (1-3).

$q_0 = 0$
$q_1 = 1$
$q_2 = 2$



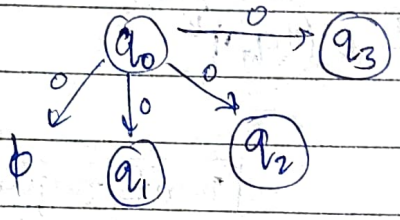
000 - 0
001 - 1
010 - 2
011 - 3
100 - 4
101 - 5
110 - 6
111 - 7
1000 - 8
1001 - 9
1010 - 10
1011 - 11
1100 - 12
1101 - 13
1110 - 14
1111 - 15
10000 - 16

L-13

NDFA $(Q, \Sigma, q_0, F, \delta)$ Non-deterministic finite automata

- Q: set of finite states
- Σ: Alphabet
- q_0: start state
- F: Final state
- δ: Transitions

$F \subseteq Q$

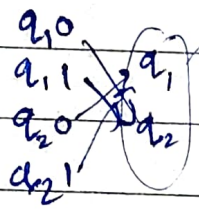


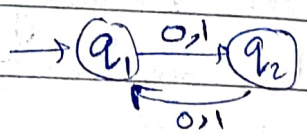
we can't do this in a DFA
∴ NDFA ✓

✱ In DFA:

$\delta : Q \times \Sigma \Rightarrow Q$

(transitions) (q_1, q_2) $(0, 1)$

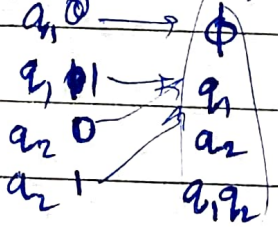


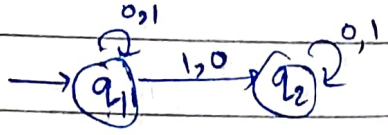


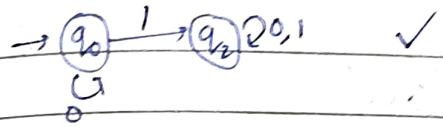
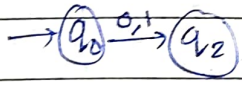
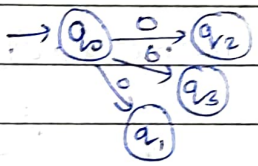
✱ In NDFA:

$\delta : Q \times \Sigma \Rightarrow 2^Q$ states

(transition) (q_1, q_2) $(0, 1)$





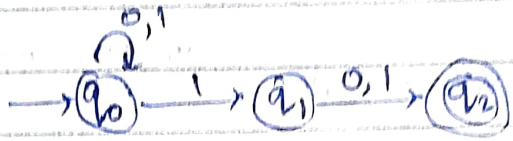
	DFA	NDFA
1)	Dead configuration is NOT allowed.	Dead configuration is allowed.
	Eg: (0,1 is to be used in every state for transition)	Eg: It can leave any transition.
	 ✓	 ✓
2)	Multiple choices are NOT available corresponding to an input.	Multiple choices are available corresponding to an input.
		
3)	ε-move is not allowed	ε-Move is allowed.
		
4)	Digital computers are deterministic	Non-deterministic feature is not associated with real computers.
5)	Designing and understanding is difficult.	Designing and understanding is easy.
	more states ↑	Less states ↓
	Difficulty ↑	Difficulty ↓

To convert NDFA to DFA

max^m 2^n states are possible in DFA, if 'n' in NFA

L-15 NFA of all binary strings in which 2nd last bit is 1.

$L = \{100, 0110, \ldots 1\}$



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

$(0+1) == (0/1)$

$(0+1)^*$ == infinite times 1,0 in any order

L-16 convert above NFA to DFA:

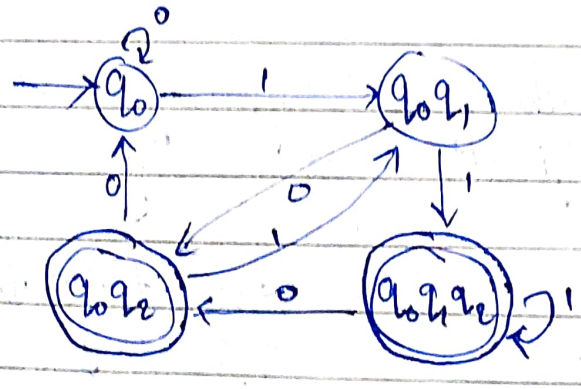
ans: First make transition table:

NFA

	0	1
→ q_0	q_0	$q_0 q_1$
q_1	q_2	q_2
(*) q_2	–	–

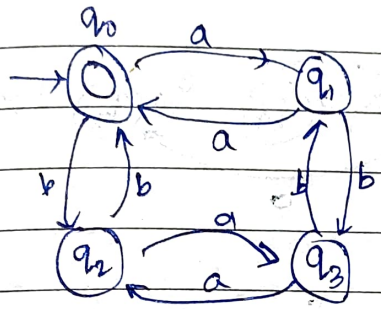
DFA

	0	1
→ q_0	q_0	$q_0 q_1$
$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
$q_0 q_2$	q_0	$q_0 q_1$
$q_0 q_1 q_2$	$q_0 q_2$	$q_0 q_1 q_2$



(L-17) Let $L = \{ w | w$ has even no. of a's and even no. of b's $\}$ Make DFA

$L = \{ \varepsilon, aa, bb, \cancel{ab}, abab, aabb, bbaa, baab, abba \}$



If 'x' stage was final:

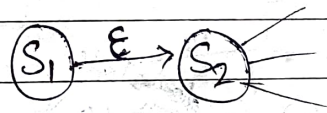
x =

q_1 = odd 'a' & even 'b'

q_2 = even 'a' & odd 'b'

q_3 = odd 'a' & odd 'b'

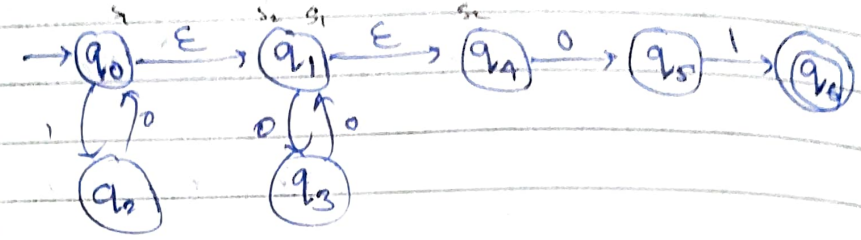
(L-18) ε-NFA (Epsilon NFA) → Eliminating ε-moves



steps:

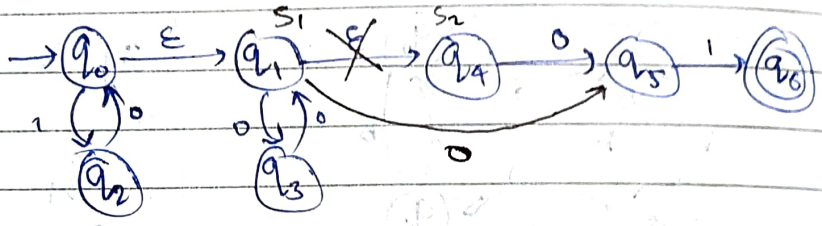
1) Find all edges starting from S_2
2) Duplicate all edges to S_1 without changing edge labels
3) If S_1 is initial state, make S_2 initial.
4) If S_2 is final state, make S_1 final.
5) Remove dead state.

Q:)

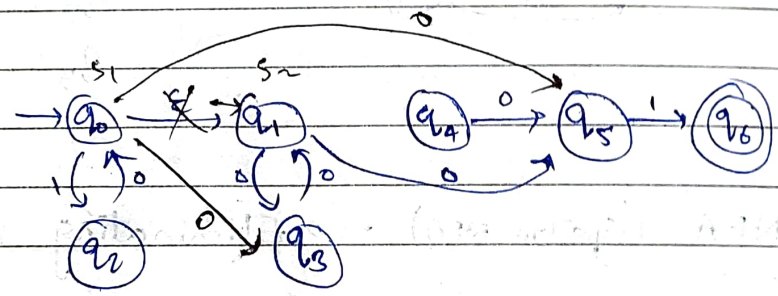


ans:-

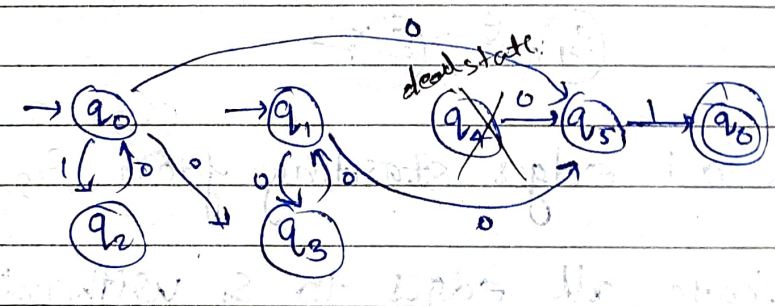
1)



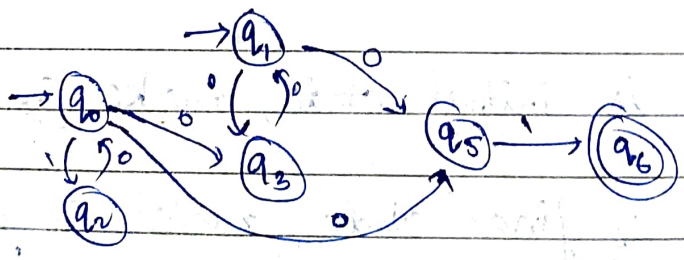
2)



3)



✓ 4)



ans.

1.11 Limitation of FSA (Finite state automata)
(→ NFA, DFA, ε-NFA, mealy machine, moore machine)

→ Limited Memory. (comparision will use memory which is limited)
→ Strings without comparision.
→ Linear Power.

a^n
even no. of a
, odd no. of a
FA ✓

$a^n b^n$ $n \geq 0$
FA ✓

$a^m b^n$ $m=n$ or $m!=n$ or $m<n$ or $m>n$
FA ✗

a^{2^n}, a^i i=prime, a^{n^2}
FA ✗

Applications of FSA:

→ Word processor program (Dictionary, etc)
→ Digital logic design — Moore's machine
— Mealy machine
→ Lexical Analyzer *
→ Text editor
→ Switching circuit design
→ Software for scanning large bodies of text such as webpages to find occurance of words, phrases, patterns.
→ Software with finite states like vending Machines weigh Machine, traffic lights, TOll Machine etc.
→ Game design. (Pac man, Treasure Hunt, etc.)

DATE: / / PAGE NO.

1.20 Moore Machine:

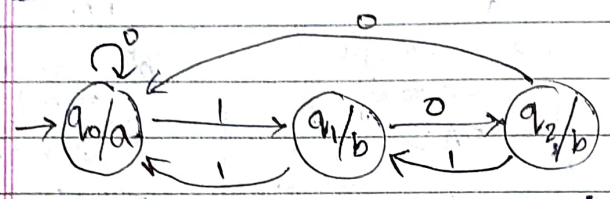
→ FA with output..
(Finite automata)

→ No final state concept.

$M_0 = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

Q = finite set of states
Σ = input symbol (Alphabet)
δ = Transition function
$Q \times \Sigma \to Q$
q_0 = start state
Δ = Output symbol
λ = Output function
$\lambda : Q \to \Delta$

Moore's Machine:



here, $Q = q_0, q_1, q_2$

$\Sigma = 0, 1$

$\delta =$ q_0, q_1, q_2 $\times (0,1) \Rightarrow$

$q_0 0$, $q_0 1$, $q_1 0$, $q_1 1$, $q_2 0$, $q_2 1$ → q_0, q_1, q_2

$q_0 = q_0$

$\Delta = a, b$

$\lambda =$

$q_0 \to a$
$q_1 \to b$
$q_2 \to b$

Ex: ⇒ input: 00110

ans: Output: aaabba

(∵ if "n" input then "n+1" will be the output length.)

∵ we count ε, start state. $(n \to n+1)$

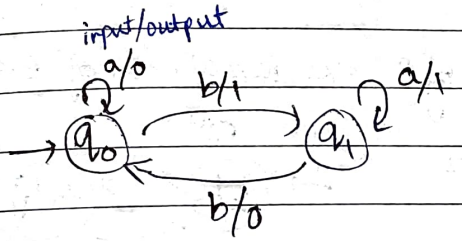
DATE: / / PAGE NO.

Transition table:

Current state	Next state		output
	0	1	
q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

(1-21) Mealy Machine: (FA with output)
→ No final state concept.

$M_e = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$



$Q = q_0, q_1$

$\Sigma = a, b$

$\Delta = 0, 1$

$\delta =$ $q_0 a \to q_0$, $q_0 b \to q_1$, $q_1 a \to q_1$, $q_1 b$

$q_0, a \to 0$
$q_0, b \to 1$
$q_1, a \to 1$
$q_1, b \to 0$

$\lambda =$ $q_0 \to 0$, $q_1 \to 1$ $\quad q_0 = q_0$

$\lambda : Q \times \Sigma \to \Delta$

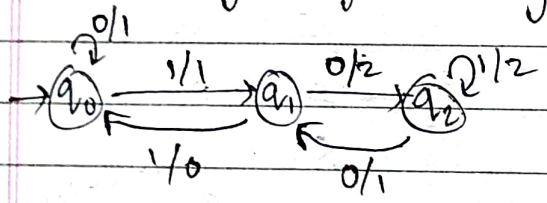
Transition table:

Current state	Next state a	Next state b	Output a	Output b
q_0	q_0	q_1	0	1
q_1	q_1	q_0	1	0

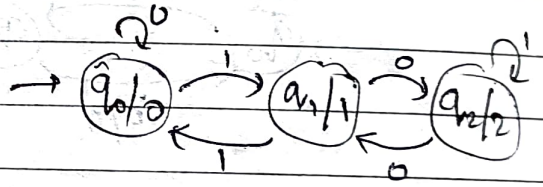
Ex: input : abba
output: 0100 (if "n" input → "n" output)
length
n→n

	Mealy Machine	Moore's Machine
1) →	Output depends on present state and present input	→ Output only depends on present state. Independent on input
2)	input string length 'n' → Ouput string length 'n'	input string length 'n' → output string length 'n+1'
3)	$\lambda : Q \times \Sigma \rightarrow \Delta$	$\lambda : Q \rightarrow \Delta$

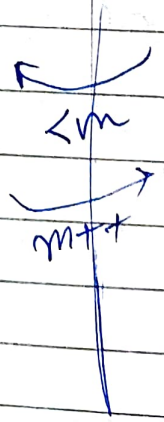
Ex: Binary string divided by 3 (Remainder)



Ex: Remainder of B.S. divided by 3.



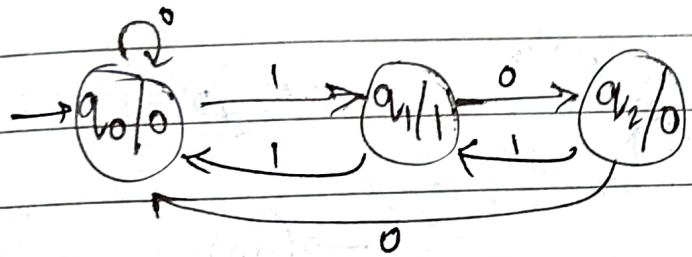
4)	Mealy output is asynchronous. ↓ (faster than Moore)	4) Output is synchronous with clock.



1.23

Moore to Mealy Conversion:

Moore:



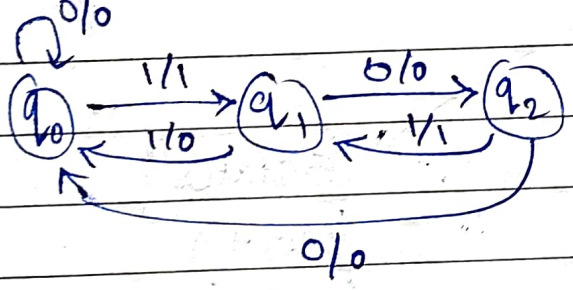
(Transition diagram)

Transition table:

Current state	Next state 0	Next state 1	Output
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

moore ↓ mealy

Mealy:

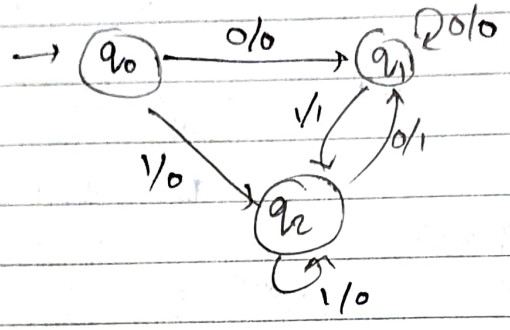


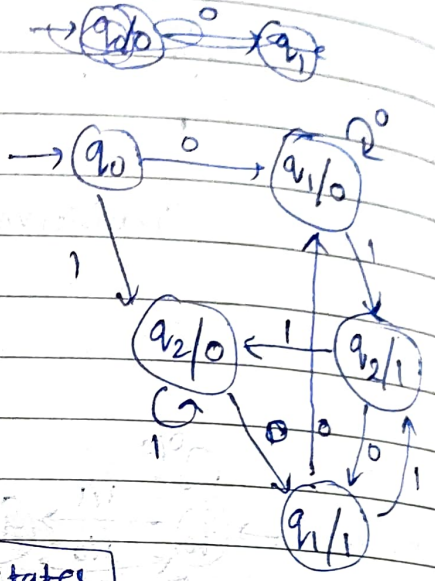
Current state	0	output	1	output
q_0	q_0	0	q_1	1
q_1	q_2	0	q_0	0
q_2	q_0	0	q_1	1

(Moore) (Mealy)

'm' states, n output ⟶ 'm' state (or less but not more)

L-24 Mealy to Moore conversion:





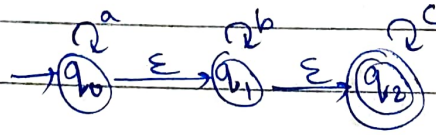
3 states
(m)
n output

5 states
(m+t)
(m x n) max^m state.

L-25 ε-NFA (Epsilon NFA)

Q = finite set of states
Σ = input symbols
q_0 = initial state
F = set of final set
δ = Transition function

$$\delta : Q \times (\Sigma \cup \varepsilon) \rightarrow 2^Q$$



Q = 3
∴ Transiton (δ) = $2^3 = 8$

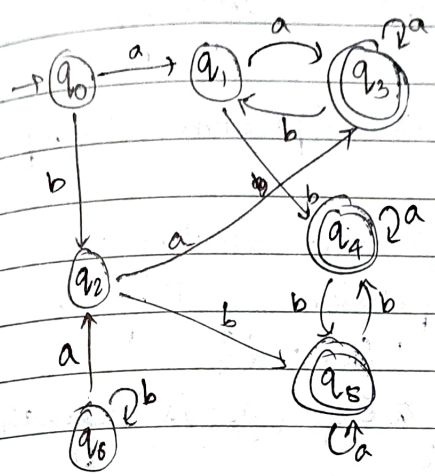
(a+b+c)*

NFA → DFA

(States after transition): n → 2^n

Minimization of DFA:

Q)



	a	b
→ q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
q_3	q_3	q_1
q_4	q_4	q_5
q_5	q_5	q_4
unreachable ~~q_6~~	~~q_2~~	~~q_6~~

Step 1: Remove unreachable states.

→ q_6 is unreachable from q_0.

∴ Remove it. as it is not useful.

Step 2: Convert in equivalent classes:

Accepting (final) → Non-Accepting (non-final)

$\Pi_0 = \{q_0\ q_1\ q_2\}\ \{q_3\ q_4\ q_5\}$

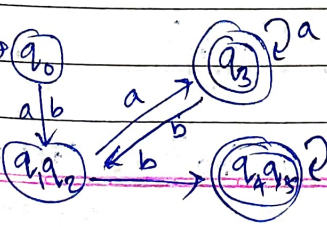
G_1 G_2

(check if they are is same grp of not)

$\Pi_1 = \{q_0\}\ \{q_1\ q_2\}\ \{q_3\}\ \{q_4\ q_5\}$

G_{11} G_{12} G_{21} G_{22}

$\Pi_2 = \{q_0\}\ \{q_1 q_2\}\ \{q_3\}\ \{q_4 q_5\}$



DATE / / PAGE NO.

Regular lang.
Finite Infinite
Finite automata

L-27 Regular Expressions:

→ Method to represent a language. (L)

→ Let 'R' be a Regular Expression over Alphabet Σ if R is:

1) 'ε' is Regular expression denoting the set {ε}
2) ϕ, is Regular expression denoting the set { }
   $R = \phi$, $L(R) = \{\}$
3) For each symbol $a \in \Sigma$, a is regular expression denoting set {a}.

4) Union of two RE is also Regular.
5) Concatenation of two RE is also Regular
6) Kleene closure * of RE is also Regular.
7) if R is regular, (R) is also Regular.
8) Nothing else, Repeat 1 to 7 Recursively.

L-28 Regular Expression: Example 1:

finite lang.
↳ Regular lang. ✓
↳ Finite Automata ✓
↳ Regular Expression ✓

$\Sigma(a, b)$

	L(R)	R.E.
1)	No string	{ } ϕ
2)	length 0	{ε} ε, λ
3)	length 1	{a, b} (a+b)
4)	length 2	{aa, bb, ab, ba} (aa+ab+bb+ba) = (a+b)(a+b)

DATE: / / PAGE NO.

5) Length 3 $(a+b)(a+b)(a+b)$ — R

6) Atmost 1 0,1 $\{\varepsilon, a, b\}$ $(\varepsilon+a+b)$ — R

7) Atmost 2 $(\varepsilon+a+b)(\varepsilon+a+b)$ — R

⋮

Ex: Not more than 2b's and 1a : $\{\varepsilon, a, b, ab, ba, abb, bab, bba\}$

$(\varepsilon+a+b+ab+ba+abb+bab+bba)$ — R

L-29 Regular Expression: Example 2: (infinite lang.)

Consider Alphabet $\Sigma = (a, b)$

1) All strings having a single 'b' a^*ba^*

2) " atleast one 'b' $(a+b)^*b(a+b)^*$

3) " bbbb as substring $(a+b)^*bbbb(a+b)^*$

4) " All strings end with 'ab' $(a+b)^*ab$

5) " Start with 'ba' $ba(a+b)^*$

6) " begining & ending with 'a' $a(a+b)^*a$

7) " containg 'a' $(a+b)^*a(a+b)^*$

8) " Starting & ending with diff. symbol $(a(a+b)^*b + b(a+b)^*a)$

9) " have only 2 'b' ~~$(a+b)^*b(a+b)^*b(a+b)^*$~~ $a^*ba^*ba^*$

DATE: / / PAGE NO.

L-30

Which two of following is equivalent?

(i) $(00)^*(\varepsilon+0)$
(ii) $(00)^*$
(iii) 0^*
(iv) $0(00)^*$

a) (i) & (ii)
b) (ii) & (iii)
c) (i) & (iii)
d) (iii) & (iv)

ans:

all zero $0^* = \varepsilon, 0, 00, 000, 0000 \ldots$

even no. of 0 $(00)^* = \varepsilon, 00, 0000, 000000 \ldots$

odd no. of 0 $0(00)^* = (00)^*0 = 0, 000, 00000, \ldots$

$(00)^*(\varepsilon+0) = (00)^*\varepsilon + (00)^*0$
$= \text{even} + \text{odd}.$

$\therefore 0^* = (00)^*(\varepsilon+0)$

$\therefore$ c) (i) & (iii) are equivalent

L-31 Pumping Lemma: If L is an infinite language then there exists some positive integer 'n' (Pumping length) such that any string $w \in L$ has length greater than equal to 'n' i.e. $|w| \geq n$ then w can be ~~divisible by~~ divided into three parts, $w = xyz$ satisfy following conditions,

(i) for each $i \geq 0$, $xy^iz \in L$
(ii) $|y| > 0$ and
(iii) $|xy| \leq n$

L → PLT → Pass → Maybe R or Non-R
PLT → fail → Non-Regular

eg: (i) $a^n b^{2n}$, $n \geq 0$

$\therefore$ aabbbb $\in L$
$x \quad y \quad z$

$y \times 2 \Rightarrow$ aa bb bb bb $\notin L$
(pump)

$\therefore$ $a^n b^{2n}$ is Not Regular Lang.

(ii) $\therefore$ aabbbb $\in L$
$x \quad y \quad z$

$y \times 2 \Rightarrow$ a abab bbb $\notin L$
(pump)

$\therefore$ $a^n b^{2n}$ is not-Regular L.

L-32 Closure Properties of Regular languages:

1.) Union $(L_1 \cup L_2)$
2.) Concatenation $(L_1 . L_2)$
3.) Closure (*) L^*
4.) Complementation $\overline{L} = \Sigma^* - L$
5.) Intersection $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
6.) Reversal $(L)^R$
7.) Difference $(L_1 - L_2) = L_1 \cap \overline{L_2}$
8.) Homomorphism
9.) Reverse Homomorphism
10.) Quotient operation
11.) INIT
12.) Substitution
13.) Infinite Union.

(L-33) Regular languages are closed under Reversal: (L^R)

1) Make initial state of M as final state of M'

2) Final state of M become initial state of M'

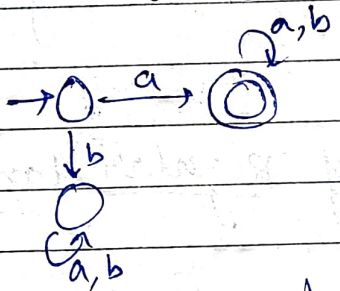
3) Reverse the direction of edges of M to make M'

4) No change in loop and Remove unnecessary state

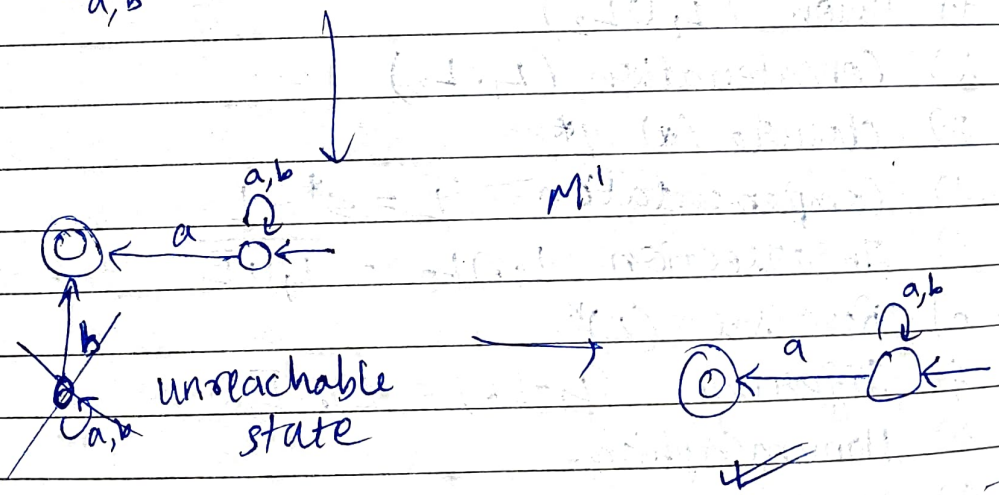
Ex: L_1 = { set of all strings over (a,b) starting with a }

$= a(a+b)^*$

DFA :



M



(Reversal of R.L.)

L-39

Quotient Operation: → Left Quotient (cut prefix)
→ Right Quotient (cut suffix)

$\frac{L_1}{L_2} = \{ x \mid xy \in L_1, \text{ for some } y \in L_2 \}$ Right Quotient (cutting in Right)

$\frac{L_1}{L_2} = \{ y \mid xy \in L_1 \text{ for some } x \in L_2 \}$ Left Quotient (cutting in Left)

Ex: $L_1 = \{10, 100, 1010, 10110\}$
$L_2 = \{10\}$

ans: $\frac{L_1}{L_2}$ (left) $= \{\varepsilon, 0, 10, 110\}$ ✓
($\frac{10}{10}$, $\frac{100}{10}$, $\frac{1010}{10}$)

$\frac{L_1}{L_2}$ (right) $= \{\varepsilon, \varepsilon, 10, 101\}$ ✓
($\frac{10}{10}$, $\frac{100}{10}$ ✗, $\frac{1010}{10}$, $\frac{10110}{10}$)

Ex: $L_1 = a^*b = \{a, aab, aaab, aaaab, \ldots\}$
$L_2 = ab^* = \{ab, abb, abbb \ldots\}$

ans: $\frac{L_1}{L_2}$ (left) $= \{\varepsilon, \varepsilon, \varepsilon \ldots, \varepsilon, \varepsilon, \ldots \varepsilon, ab, aab \ldots\}$
($\frac{a}{ab}$, $\frac{aab}{ab}$, $\frac{aaab}{ab}$, $\frac{a}{abb}$, $\frac{aab}{abb}$, $\frac{aab}{a}$, $\frac{aaab}{a}$)

$\frac{L_1}{L_2}$ (Right) $= \{\varepsilon, a, aa, aaa \ldots\}$
($\frac{a}{a}$, $\frac{ab}{ab}$)

L_1 → Regular
L_2 → Regular

∴ Regular Language are closed under quotient operation.

(L-35) INIT Operation: (Initial/Prefix)

{set of all prefix of $w \in L$}

$\Sigma = (a, b)$

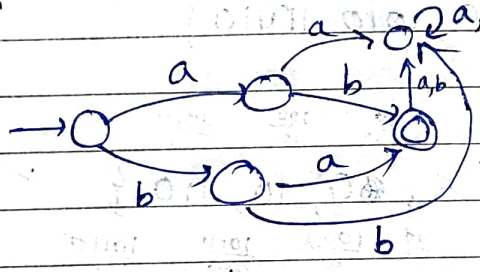
→ Let $L = \{ab, ba\}$

$ab = \varepsilon, a, ab$

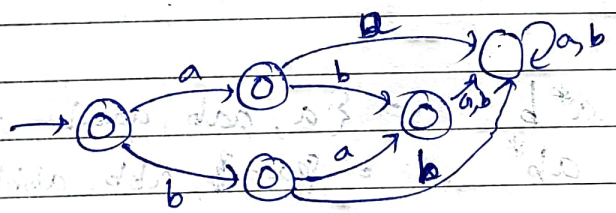
$ba = \varepsilon, b, ba$

⇒ Lang. After INIT Operation: $\{\varepsilon, a, ab, b, ba\}$

⇒ DFA of L



DFA after INIT of L



{Make all final state except trap state}

(L-36) Regular Language are NOT closed under infinite Union:

$L_1 = \{ab\}$

$\cup$

$L_2 = \{a^2b^2\}$ $\cup$

$L_3 = \{a^3b^3\}$ $\cup$

$L_4 = \{a^4b^4\}$

$\vdots$

$\{a^n b^n \quad n \geq 1\}$ ⇒ Non-Regular Language (∵ comparison)

$L_1 \cup L_2$ ⇒ Regular lang.

∴ Not closed under ∞ Union.

$a^1 \cup a^2 \cup a^3 \cup \dots a^n$

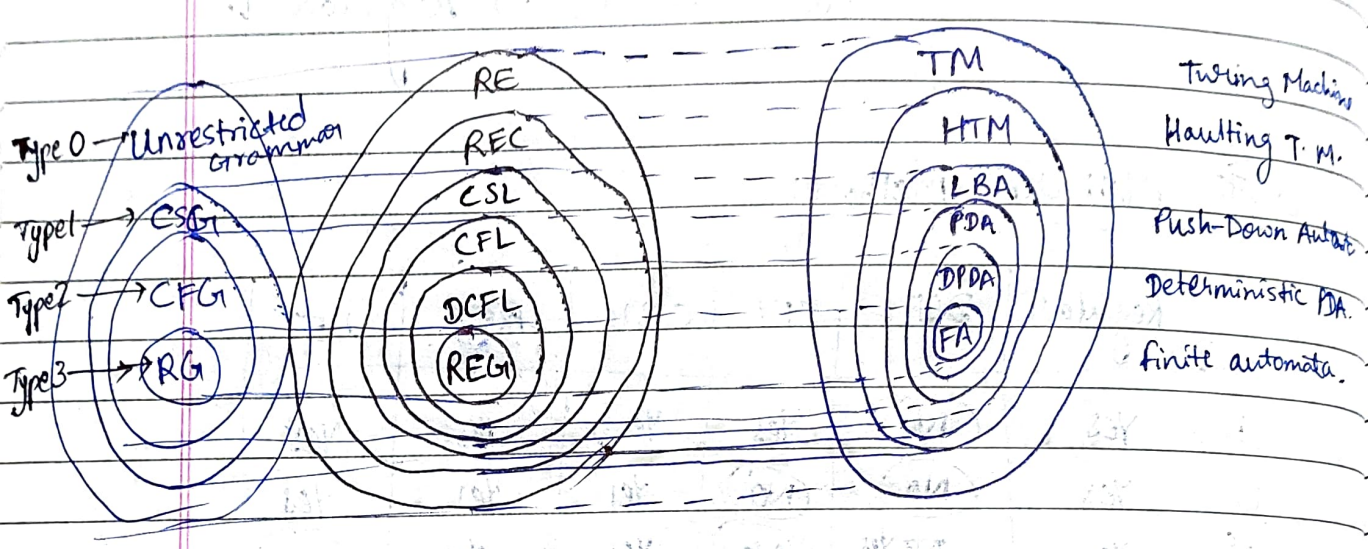
a^n $n \geq 1$ ⇒ Regular L.

L-37 Closure Method:

	Regular	DCFL	CFL	CSL	REC	RE
$\cup$	Yes	No	Yes	Yes	Yes	Yes
$\cap$	Yes	No	No	Yes	Yes	Yes
L^c	Yes	~~No~~ Yes	No	Yes	Yes	No
.	Yes	No	Yes	Yes	Yes	Yes
*	Yes	No	Yes	Yes	Yes	Yes
+	Yes	No	Yes	Yes	Yes	Yes

DCFL: Deterministic context free language
CFL: Context free language
CSL: Context sensitive language
REC = Recursive
RE = Recursive Enumerable

L-38 Lang., Automata, Grammar:



FA < DPDA < PDA < LBA < TM

L-39 Which of the following is/are false?

(I) $\{a^n b^n\} \cup \{a^n b^{2n}\}$ is CFL but not DCFL

(II) $a^m b^m c^n d^n$ is CFL but NOT DCFL

(III) $\{a^m b^n \mid m,n \geq 0\}$ is DCFL but not Regular

ans: (II) and (III).

$a^m b^m c^n d^n$ is determinent CFL

$a^m b^n$ is regular for $m,n \geq 0$ (∵ no comparison required)

{if $m > n$ or $m < n$ or $m = n$ was given then it is not Regular but DCFL ✓}

L-43 Homomorphism: $h(L)$ (Substitution function)

$$h(L) = \{h(\omega) \mid \omega \in L\}$$

↳ homomorphic image.

where $h: \Sigma$ to Γ^* is called homomorphism.

Q) $\Sigma = \{0, 1\}$ $\Gamma = \{a, b\}$
$h(0) = aa$, $h(1) = bb$
if $L = \{00, 101\}$
Find $h(L) = ?$

$h(L) = \{aaaa, bbaabb\}$

R.L → R.E → $(0+1)^* 1^*$
$= (aa+bb)^* bb^*$

L-44 Inverse homomorphism $h^{-1}(L)$

Let $h(0) = a$, $h(1) = b$, $h(2) = ab$
$\Sigma = \{0, 1, 2\}$, $\Gamma = \{a, b\}$

Let $L = \{abab\}$
$h^{-1}(L) = ?$

ans: $h^{-1}(L) = \{0101, 22, 201, 012\}$

Q) $h(0) = aa$, $h(1) = bb$
$\Sigma = \{0, 1\}$, $\Gamma = \{a, b\}$ $L = \{aa, aabb, baab, ababa\}$

ans:- $h^{-1}(L) = \{0, 01\}$

$h(h^{-1}(L)) = \{aa, aabb\}$ $h(h^{-1}(L)) \subset L$

(L-45) Decidability & Undecidability table in toc for all L.

	REG	DCFL	CFL	CSL	REC	RE
* Membership Problem $w \in \Sigma^*$	✓ (FA)	✓	✓ (CYK)	✓	✓	✗
Infiniteness Problem L = infinite or finite	✓	✓	✓	✗	✗	✗
Emptiness problem $L = \phi$	✓	✓	✓	✗	✗	✗
• Equality problem $L_1 = L_2$	✓	✓	✗	✗	✗	✗
L is ambigious?	✓	✓	✗	✗	✗	✗
Completeness $L = \Sigma^*$	✓	✓	✗	✗	✗	✗
$L_1 \cap L_2 = \phi$	✓	✗	✗	✗	✗	✗
$L_1 \subseteq L_2$ subset Problem	✓	✗	✗	✗	✗	✗

L-46 CFL and CFG Introduction & Syllabus:

1) Standard CFL's → Grammar & properties

2) Derivation , Derivation Tree , Ambiguity.
   ↙ ↘
   LMD RMD

3) Parsing , Membership algo.
   ↓ ↓
   Brute force CYK

4) Machines (DPDA, NPDA)

5) DCFL Vs. CFL

6) Algorithm → Removal of λ Production
   → " unit "
   → " Left Recursion
   → " Left factoring

7) Normal form → Chomsky
   → Greibech

L-47 CFG: (V, T, P, S)

V: finite set of variables (Non Terminals)
T: finite set of terminals ($V \cap T = \phi$)
P: Production Rules (substitution Rule)
S: Start variable.

Eg: $\alpha \rightarrow \beta$

α : only one variable

β : $(V \cup T)^*$

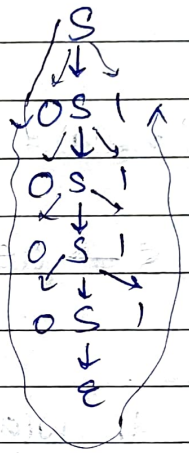
CFG: $S \rightarrow OS1$ OR $S \rightarrow OS1 \mid \varepsilon$

$S \rightarrow \varepsilon$

$\{S\}$

$\{S\}$

$\therefore \{0000\,1111\}$



CFL: $\{0^n 1^n : n \geq 0\}$

(-48)

Q.) Convert CFL → CFG:

(i) $a^n b^n$, $n \geq 0$

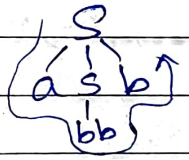
ans: $S \rightarrow aSb \mid \lambda$

(ii) $a^n b^n$, $n \geq 1$

$S \rightarrow asb \mid ab$

(iii) $a^n b^{n+2}$, $n \geq 0$

$S \rightarrow asb \mid bb$



aabbb

(iv) $a^{2n} b^n$, $n \geq 0$

$S \rightarrow aasb \mid \varepsilon$

(v) $a^{2n+3} b^n$, $n \geq 0$

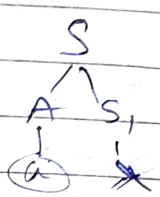
$S \rightarrow aaSb \mid aaa$

(vi) $a^m b^n$, $m > n$, $n \geq 0$

$S \rightarrow AS_1$

$S_1 \rightarrow aS_1b \mid \lambda$

$A \rightarrow aA \mid a$



(vii) $\{ w \mid n_a(w) = n_b(w) \}$

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$

(viii) WW^R U $W(a+b)W^R$

(Even Palindrome) (odd Palindrome)

$S \rightarrow aSa \mid bSb \mid \lambda$ $S \rightarrow asa \mid bsb \mid a \mid b \mid \lambda$

U

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$

(ix) $a^m b^m c^n$, $m, n \geq 0$

$S \rightarrow S_1C_1$

$S_1 \rightarrow aS_1b \mid \lambda$

$C_1 \rightarrow cC_1 \mid \lambda$

L-49 Left most & Right most Derivation in CFG.

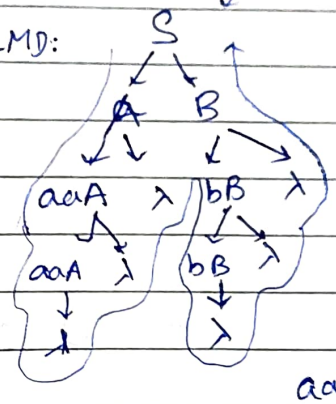
Derivation
LMD RMD

Derivation tree

Eg: $S \rightarrow AB$
$A \rightarrow aaA \mid \lambda$
$B \rightarrow bB \mid \lambda$

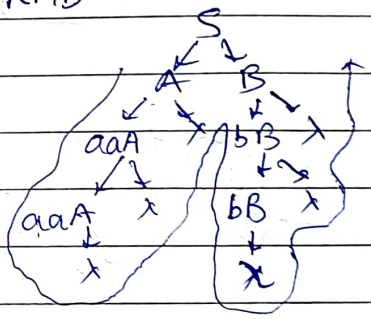
Check Weather
$\omega = aaaabb \in L(G_1)$

ans: LMD:



aaaabb

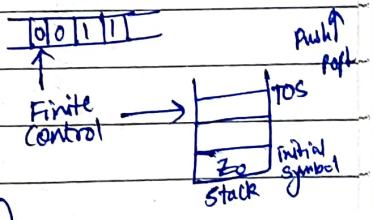
RMD:



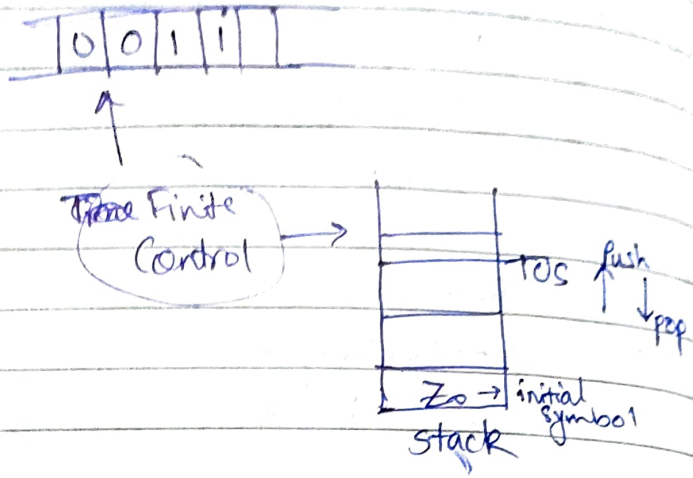
aaaabb

L-50 PDA: (FA + Stack)

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

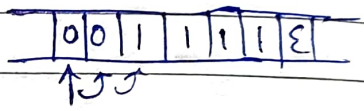


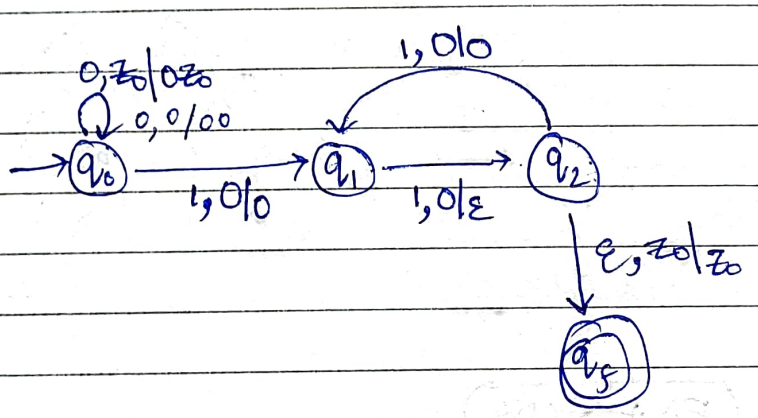
Q = finite set of states like FA
Σ = Input Symbol like FA 0, 1
Γ = Stack Alphabet (Push in stack)
δ = Transition function, $\delta: Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \rightarrow \Gamma^*$
Z_0 = Stack start symbol
F = final state.

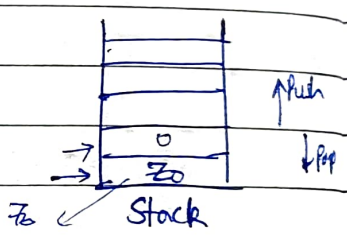


L-5. $L = \{0^n 1^{2n}, n>0\}$

{011, 001111, 000111111}







$q_0, 0, Z_0 \rightarrow q_0 0 Z_0$

$q_0, 0, 0 \rightarrow q_0 00$

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

$q_1, 1, 0 \rightarrow q_2, \varepsilon$

$q_2, 1, 0 \rightarrow q, 0$

$q_2, \varepsilon, Z_0 \rightarrow q_f, Z_0$

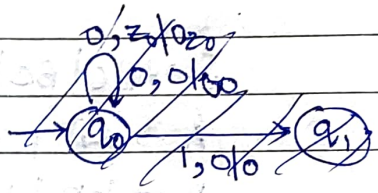
L-52

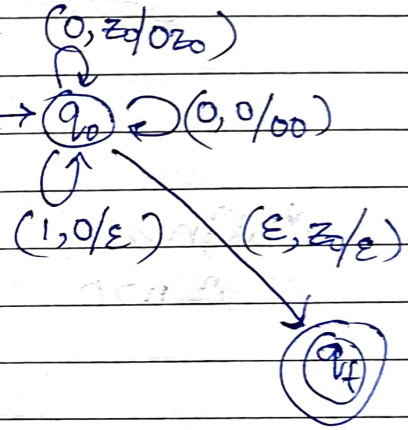
$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$\delta = Q \times (\Sigma \cup \varepsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$

$\Rightarrow L = \{ w \in (0,1)^* \mid n_0(w) = n_1(w) \}$

{01, 10, 1100, 0101, 1010}





L-53

Closure Properties of CFL:

Union ✓
Concatenation ✓
Kleene closure ✓
Intersection ✗
Complementation ✗

lang. → Grammer

$L_1 \rightarrow G_1 \quad S_1 \rightarrow aS_1b \mid \lambda$

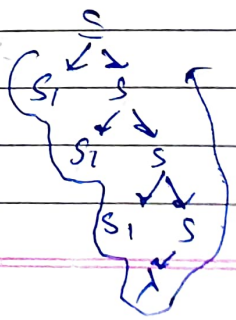
$L_2 \rightarrow G_2 \quad S_2 \rightarrow bS_2c \mid \lambda$

$S \rightarrow S_1 \mid S_2$ ✓
$(S_1 \cup S_2)$

$S \rightarrow S_1 \cdot S_2$ ✓

$L_1^* \rightarrow G_1$

$S \rightarrow S_1 S \mid \lambda$



(54) Removal of Null production from CFG:-

$S \to aS \mid A$
$A \to \varepsilon$

$\{A, S\}$

$S \to aS \mid a \mid \varepsilon$

$L(G) = \varepsilon$

$a^* \; a^n; n \geq 0$
$a^+ \; a^n \; n > 0$

$S \to ABC$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$
$C \to c$

Nullable

ε removal

$S \to ABC \mid BC \mid AC \mid C$
$A \to aA \mid a$
$B \to bB \mid b$
$C \to c$

(55) Elimination of unit Production:

$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E$
$E \to a$

unit production

$B \to C$
$C \to D$
$D \to E$

Eliminate

$S \to AB$
$A \to a$
$B \to a \mid b$
$C \to a$
$D \to a$
$E \to a$

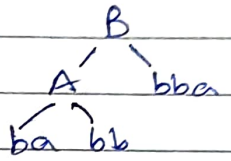
{C, D, E are not been used in S ∴ remove from the grammar}

Final Grammar:

$S \to AB$
$A \to a$
$B \to a \mid b$

Eg: $S \to aA \mid B$
$A \to ba \mid bb$
$B \to A \mid bba$

unit Production
$S \to B$
$B \to A$



~~Final Grammar~~ Elimination

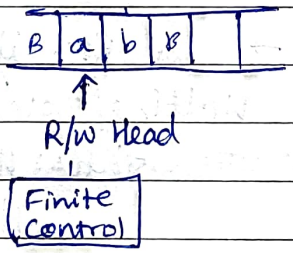
$S \to aA \mid bba \mid ba \mid bb$
$A \to ba \mid bb$
$B \to ba \mid bb \mid bba$

Final Grammar: (B is not in use)
$S \to aA \mid bba \mid ba \mid bb$
$A \to ba \mid bb$

(56) Turing Machine:

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

$\delta = Q \times \Gamma \to Q \times \Gamma \times (L, R)$ (Left, Right)



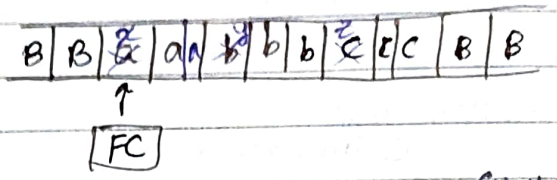
PDA ✓ $a^n b^n$ $n \geq 1$
PDA ✗ TM ✓ $a^n b^n c^n$ $n \geq 1$

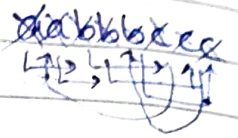
1) Read, write allowed
2) Left, Right

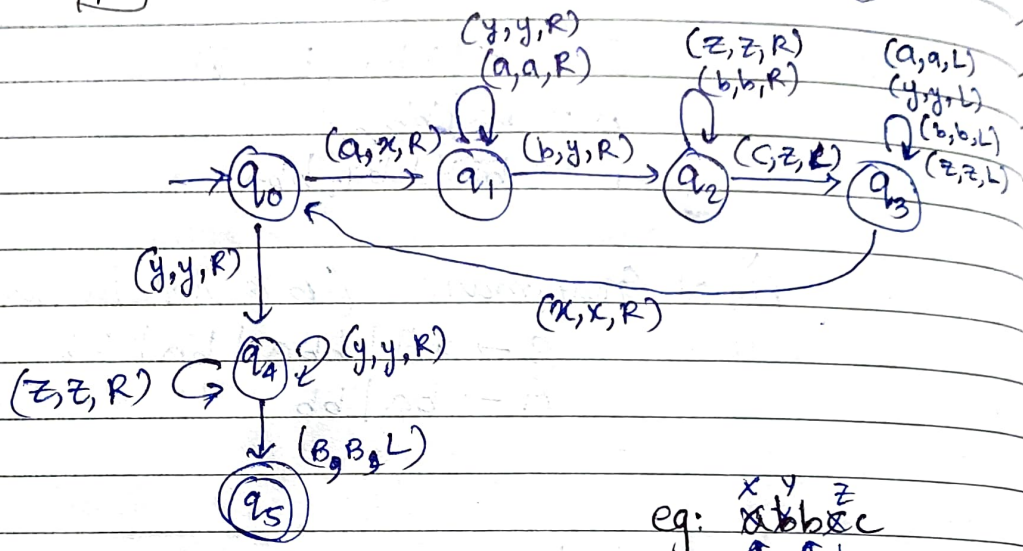
$(q_0, a) \to (q, x, R)$ (Deterministic)
$Q \times \Gamma \times (L, R)$

$(q_0, a) \to q_1, a, R$ (Non-Deterministic)
q_0, x, L

(59) Design Turing Machine for $\{a^n b^n c^n \mid n \geq 1\}$



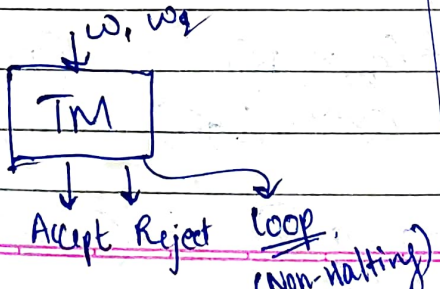




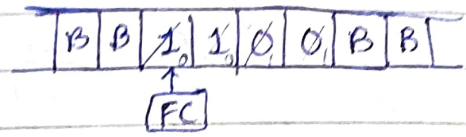
eg:

(60)

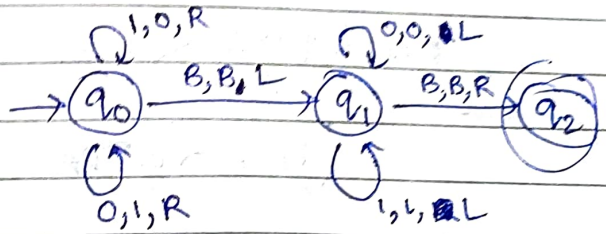
Recursive Enumerable language	Recursive language
→ L is RE if there is TM	→ L is Rec. if there is Halting/Total TM.
→ Three states: Halt & accept Halt & Reject Never Halt.	→ Two states: Halt & Accept Halt & Reject
→ Closed under all except −, complement.	→ closed under all except Homomorphism & Substitution



Q6) Turing Machine for 1's Compliment:



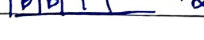
0 x 0 x B
1 0 1 0



States	0	1	B
q_0	$1Rq_0$	$0Rq_0$	BLq_1
q_1	$0Lq_1$	$1Lq_1$	BRq_2
q_2	—	—	—

α β γ
α → change
β → Direction
γ → New state

Q7) Modification of TM:

1) TM with stay option
2) TM with semi-infinite tape 
3) Offline TM
4) Multidimensional
5) Non-Deterministic TM
6) Universal TM
7) Multitape TM
8) Multihead TM
9) TM with 3 states
10) Jumping TM
11) Non-Erasing TM
12) Always writing TM.

No change in power.

1) TM without writing capacity → (FA)
2) TM with input size Tape → (LBA)
3) TM with Tape used as stack * (NDTM) (DPDA) (PDA / NPDA)
4) TM with finite tape → (FA)

FA < PDA < LBA < TM

↓

DPDA < NPDA

(63) CYK algorithm :

⇒ check weather a string 'abbb' is a valid member of following CFG.

$$\begin{cases} S \to AB \\ A \to BB \mid a \\ B \to AB \mid b \end{cases}$$

* CYK applicable on CNF only.

CNF: A → BC
A → a OR

* It is universal (applicable on all CNF)

* Time complexity $O(n^3)$

* Space complexity $O(n^2)$

abbb
1 2 3 4

	4	3	2	1
1	S,B	A	S,B	A
2	S,B	A	B	
3	A	B		
4	B			

12	23	34
(11)(22)	(22)(33)	(33)(44)
AB	B.B	BB

14	13	24
11(24) OR (12)(34) OR (13)(44)	(11)(23) OR (12)(33)	(23)(44) OR (22)(34)
A(S,B) (S,B)A AB	AA OR (S,B)(B)	AB OR BA
AB AB SA BA AB	SB, BB	

S is present in (1,4)
∴ abbb is valid CNF.

$S \begin{cases} (1,2) \Rightarrow ab \\ (2,4) \Rightarrow bbb \end{cases}$ are also valid CNF.

(69)

CNF (Chomsky Normal Form)	GNF (Greibach Normal Form)
1) $A \to BC$ OR $A \to a$ $\{A, B, C \in$ variables, $a \in$ Terminal$\}$	1) $A \longrightarrow ax$ where $\{x \in V^*\}$ $\{a \in T\}$
2) No. of steps required to generate a string of length 'n' is "$2|n| - 1$"	2) No. of steps required to generate a string of length 'n' is 'n'.
3) Used in Membership Algo.	3) Used to convert CFG to PDA.
4) Derivation or Parse tree obtained from CNF is always Binary Tree.	4) Not always.
5) Length of each production Restricted.	5) Not restricted.

Chomsky Normal Form: (CNF)

$A \to BC$
$A \to a$

Q.)

$S \to XB \mid AA$
$A \to a \mid SA$
$B \to b$
$X \to a$

Convert CNF → GNF

ans: The given grammar G is already in CNF & there is no left recursion.

→ Production rule $A \to SA$ is not GNF, so we substitute $S \to XB \mid AA$ in the production rule.

$A \to SA$ as:

$S \to XB \mid AA$
$A \to a \mid XBA \mid AAA$
$B \to b$
$X \to a$

⇒ Substitute ~~Replace~~ $X \to a$ to S and B to make GNF,

$S \to aB \mid AA$
$A \to a \mid aBA \mid AAA$ → left recursion.
$B \to b$
$X \to a$

$A \to A\alpha \mid \beta$
Eliminate LR:
$A \to \beta A'$
$A' \to \alpha A' \mid \varepsilon$

⇒ We will need to remove left recursion. (A → AAA)

{Introduce new variable 'C' to remove left recursion. One 'A' in production is replaced with new variable & without new variable}.

$S \rightarrow aB \mid AA$

$A \rightarrow a \mid aBA \mid aC \mid aBAC$

$C \rightarrow AAC \mid AA$ (∵ $C \rightarrow AAC \mid \varepsilon$ Removing ε, $C \rightarrow AAC \mid AA$)

$B \rightarrow b$

$X \rightarrow a$

→ $S \rightarrow AA$ is not GNF, so, substitute (A)A,

$S \rightarrow aB \mid aA \mid aBAA \mid aCA \mid aBACA$

$A \rightarrow a \mid aBA \mid aC \mid aBAC$

$C \rightarrow AAC \mid AA$ → Not GNF subs. A.

$B \rightarrow b$

$X \rightarrow a$

→ $S \rightarrow aB \mid aA \mid aBAA \mid aCA \mid aBACA$

$A \rightarrow a \mid aBA \mid aC \mid aBAC$

$C \rightarrow (A+AC) \mid (A+(A))$

$B \rightarrow b$

$X \rightarrow a$

∴ The above grammar is now GNF.

Left factoring:

$$A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3$$

Eliminating:

$$A \to \alpha A'$$
$$A' \to \beta_1 \mid \beta_2 \mid \beta_3$$

(Q) convert CFG to CNF:

P: $S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \varepsilon$

ans:-
1. Create new start symbol S'. $S' \to S$
2. Eliminate NULL production
3. Remove UNIT production
4. Remove more than 2 variables in RHS
5. Remove productions of the from $A \to aB$.

(1) $S' \to S$
$S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \varepsilon$

(2) $B \to b \mid \varepsilon$

$S' \to S$
$S \to ASA \mid aB \mid a$
$A \to B \mid S \mid \varepsilon$
$B \to b$

$A \to \varepsilon$ $S' \to S$ $S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \to B \mid S$
$B \to b$

unit p.

③
$S' \to S$
$S \to S$
$A \to B$
$A \to S$

$S' \to ASA \mid aB \mid a \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to ASA \mid aB \mid a \mid SA \mid AS \mid b$
$B \to b$

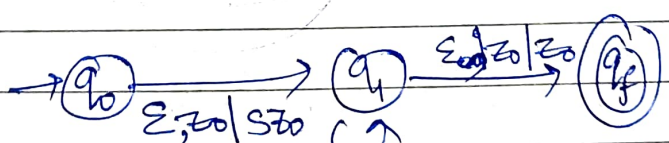
④
$S' \to AX \mid aB \mid a \mid SA \mid AS$
$S \to AX \mid aB \mid a \mid SA \mid AS$
$A \to AX \mid aB \mid a \mid SA \mid AS$
$B \to b$
$X \to AS$

$SA \to X$

⑤
$S' \to AX \mid YB \mid a \mid SA \mid AS$
$S \to$ ''
$A \to$ ''
$B \to b$
$X \to As$
$Y \to a$

GNF → PDA

$S \to aAA$
$A \to aS \mid bS \mid a$



$a, S \mid AA$
$a, A \mid S$
$b, A \mid S$
$a, A \mid \varepsilon$