10.9.22

Theory of Computation :-

→ LAG :- Language, Automata, Grammar

- Symbols :- are an entity or individual objects, which can be any letter, ~~of~~ alphabet or any picture.
  Ex :- 1, a, b, #

- Alphabets :- are ~~the~~ a finite set of Symbols. It is denoted by Σ.
  Ex :- Σ = { a, b }

- String :- It is a finite collection of Symbols from the alphabet. The String is denoted by w.
  Ex :- { ab, ba, aa, aaa } etc.

- Language :- A language is a collection of appropriate String. A language which is formed over Σ can Finite or Infinite.
  Ex :- L1 :- { Set of String of length 2 }
  = { aa, bb, ba } Finite langua

  L2 :- { Set of all Strings start with 'a' }
  = { a, aa, aaa, ab, abba } Infinite Langu

- A String with Zero occurrences of Symbols is known a an empty String. It is represented by ε. Σ^*

- Automata :- It is a model or machine to check w the given string is a part of language.

Types :- FA
PDA
LBA
TM

→ Power of Sigma :- $\Sigma = \{a, b\}$

Σ^0 = Set of all strings with length 0 = λ, ε

Σ^1 = Set of all strings with length 1 = $\{a, b\}$

Σ^2 = Set of all strings with length 2 = $\{a, b\}\{a$

Σ^3 = " =

Σ^4 = " =

Σ^* = Set of all possible string of all length possible on a and b. also known as Klee Closure. = $(a+b)^*$ = Infinite language

Σ^+ = Set of all possible string of all length ex Σ^0 or ε. also known as positive closu

$\Sigma^* - \varepsilon = \Sigma^+$

- Grammar :- Standard way of representing a language.

→ A grammar 'G' is defined as quadruple

Terminal Start
'a'

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

Variable 'V' — Production rule

{ Capital = V
small = t
Start = S
ε = Null }

$S \to aSb / \varepsilon$

$\varepsilon, aSb, aaSbb, aaaSbbb \ldots$

$\varepsilon, ab, a^2Sb^2, a^3Sb^3 \quad a^4b^4 \ldots a^n b^n$

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

where $n > 0$.

Ex :- abab is in the string $S \to aSb / \varepsilon$

$= aaSbb, a\varepsilon b$

a^2b^2, ab

∴ abab is not in the grammar

Ex :- $S \to SS$
$S \to aSb$
$S \to bSa$
$S \to a\varepsilon$

$\varepsilon, aSb\ bSa$
$\varepsilon, abba$

Here language = $n_a(w) = n_b(w)$

- Finite automata :-

i) Finite automata are used to recognize patterns.

ii) It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.

iii) At the time of transition, the automata can either move to the next state or stay in same state.

iv) Finite automata have two states, Accept state or Reject state. when the input string is processed successfully and the automata reached its final state then it will accept.

Types of finite automata :-

1. DFA
2. NFA

1. DFA :- Deterministic finite automata. Deterministic refers to the uniqueness of the computation. In the DFA, the machine goes to one state only for a particular input character. DFA does not accept the null move.

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

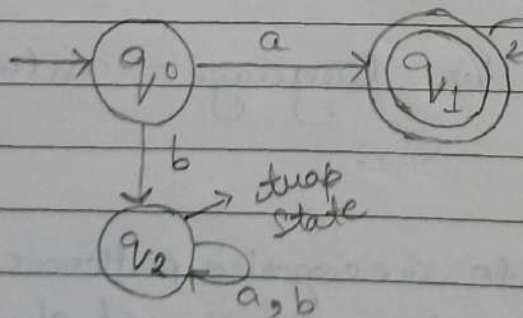
Q = Set of finite States
Σ = finite set of the input Symbol.
q_0 = initial State / Start
F = final State
δ = Transition function

$\{(F)$ Final State $\subseteq Q\}$

- Strings Starting with a

$\{a, aa, aaa, ab, bâ\}$

$\{$ In DFA, Start with minimum String $\}$



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

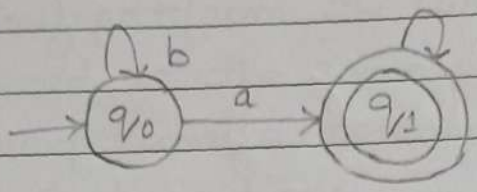
$q_0 \times (a, b)$
$q_1 \times$
$q_2 \times$

q_0 a
q_0 b
q_1 a
q_1 b
q_2 a
q_2 b

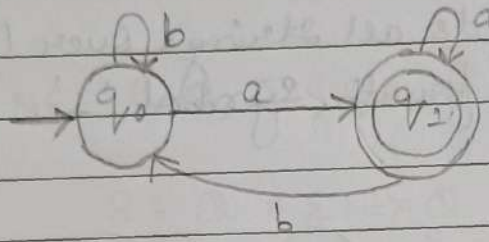
$q_0 \to q_1$ Representing
q_2 transition

Q Construct a DFA which accept a language of all
→ Containing 'a'
→ End with 'a' $\Sigma = (a, b)$ { a, aa, aaa, ba, ab, abab }

i)

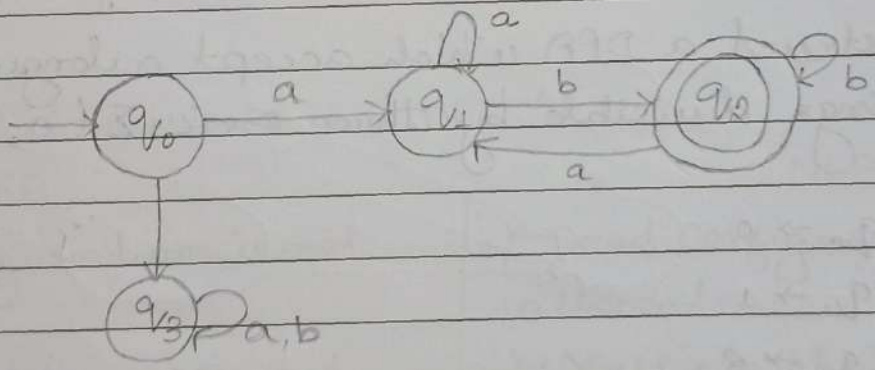


ii)



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

Ans:- $\Sigma (a, b)$ $L = \{ ab, abab, ababab, aaaabbbb \}$



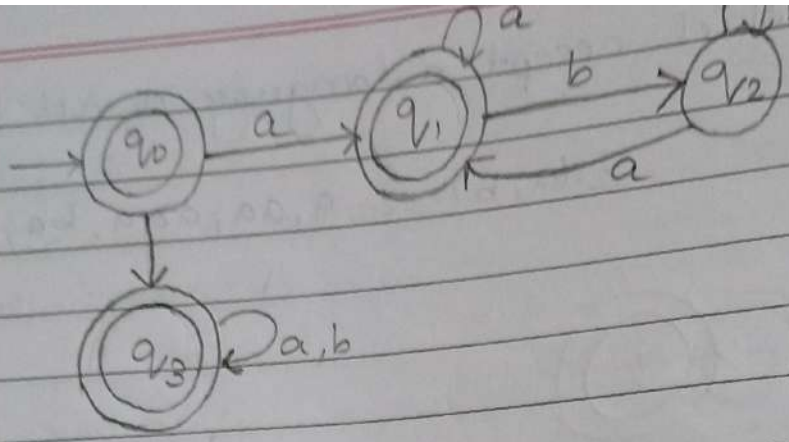
Q Construct DFA which accept a language of all Strings not starting with 'a' or not ending with 'b'.

Ans:- $L = \{ \lambda, a, b, \varepsilon, ba \}$

→ Starting with a and ending with b

{ If there is 'not' in statement simply Compliment it. }

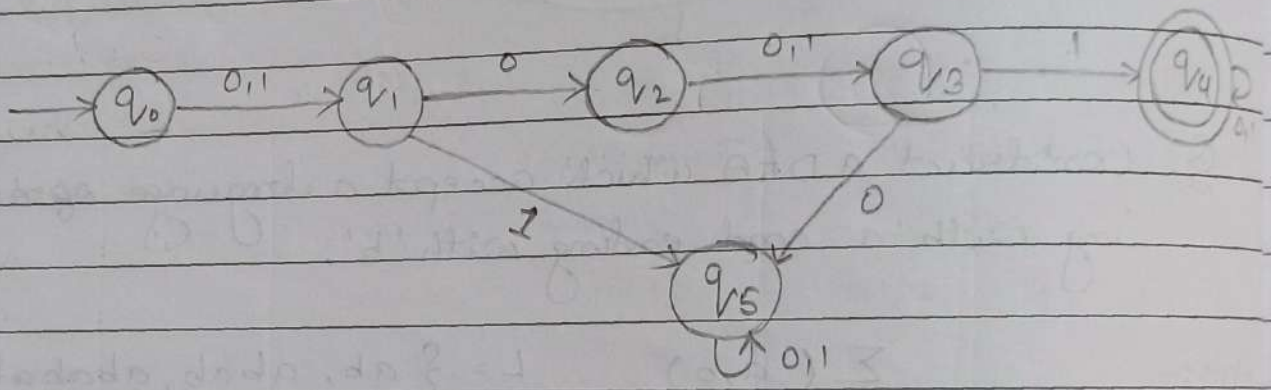
* Same as above question * $(A \cup B)^c = A^c \cap B^c$



Change the final → non-final / non-final → final

Q Design a DFA which accepts all strings over (0,1) in which Second Symbol '0' and fourth Symbol '1'.

Ans:-



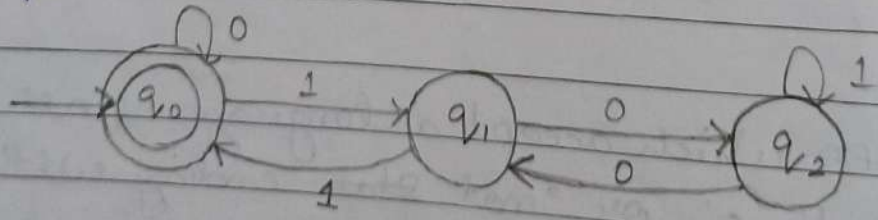
Q Construct a DFA which accept a language of all binary strings divisible by three over Σ(0,1)

Ans:-

$q_0 \rightarrow 0$

$q_1 \rightarrow 1$

$q_2 \rightarrow 2$



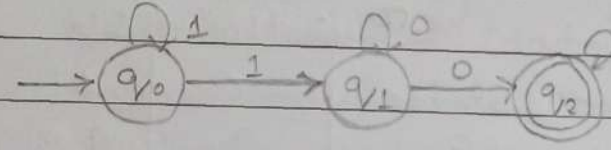
2. NFA :- Non-deterministic finite automata. It is easy to construct an NFA and DFA for a given regular language. The finite automata are called NFA when there exist many paths for specific input from the current state to the next state. It defined as the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ε transitions.

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

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

Q = finite set of states
Σ = finite set of the input symbol.
q_0 = initial state
F = final state
δ = Transition function
$\{F \subseteq Q\}$

NFA =



→ Difference between NFA and DFA :-

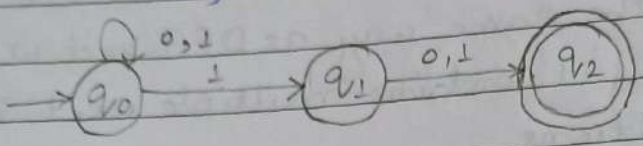
DFA	NFA
i) Dead configuration is not allowed.	i) Dead configuration is not allowed.
ii) Multiple choices are not available corresponding to an input.	ii) Multiple choices are available corresponding to an input
iii) ε-move is not allowed	iii) ε-move is allowed.
iv) Digital computers are deterministic.	iv) Non deterministic feature is not associated with real Computers.
v) Designing and understanding is difficult.	v) Designing and understanding

Q NFA of all binary strings in which 2nd last bit is 1.

Ans:- $(0+1)^* \; 1 \; (0+1)$

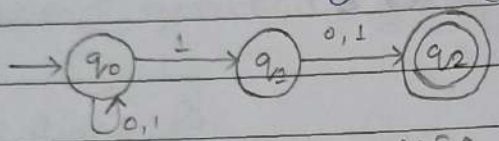
$\Sigma = 0, 1$

a, b



Q NFA of all binary strings in which 2nd last bit is 1.

Ans:-



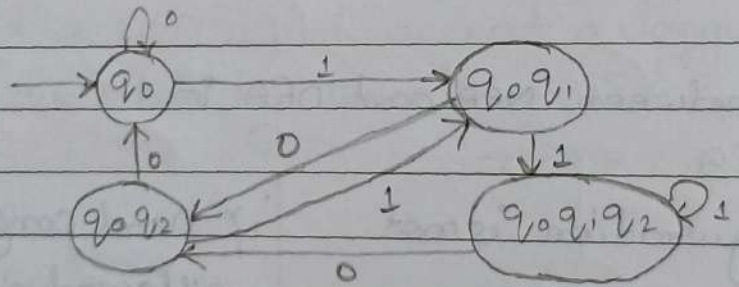
NFA

Transition table

States	0	1
$\rightarrow q_0$	q_0	$q_0 q_1$
q_1	q_2	q_2
(q_2)	-	-

Transition table

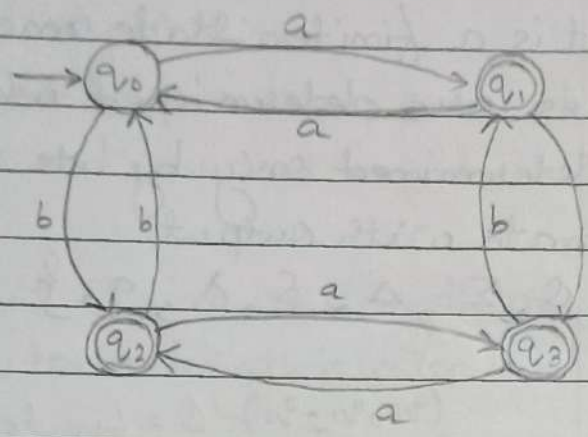
States	0	1
$\rightarrow q_0$	q_0	$q_0 q_1$
$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
$(q_0 q_2)$	q_0	$q_0 q_1$
$(q_0 q_1 q_2)$	$q_0 q_2$	$q_0 q_1 q_2$



DFA

Q Let $L = \{w / w$ has even no of a's and even no of b's$\}$

$\Sigma = a, b$



If q_0 is final or q_1 or q_2 or

q_0 = even a, even b

q_1 = odd a, even b

q_2 = even a, odd b

q_3 = odd a, odd b

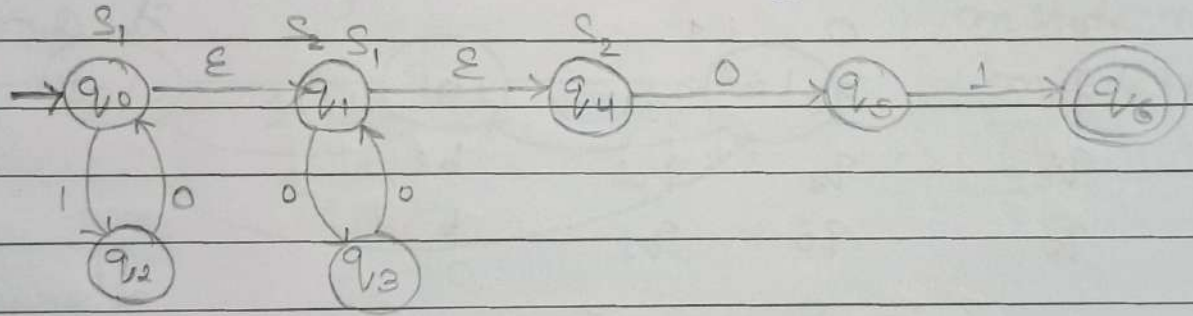
Q ε - NFA → Eliminating ε-moves

i) find all edges starting from S_2.

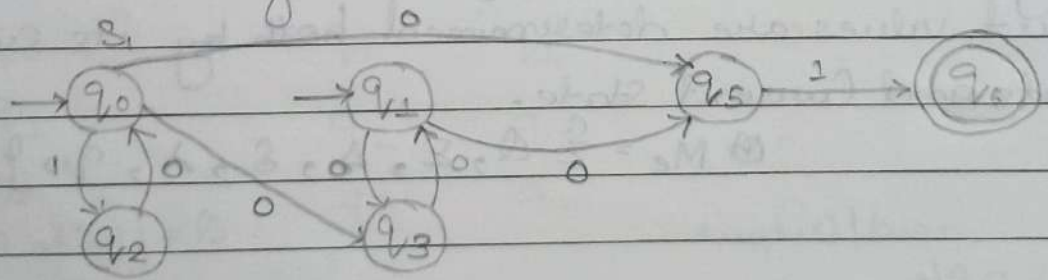
ii) Duplicate all edges to S_1 without changing edge l

iii) if S_1 is initial state, make S_2 initial.

iv) if S_2 is final state, make S_1 final.

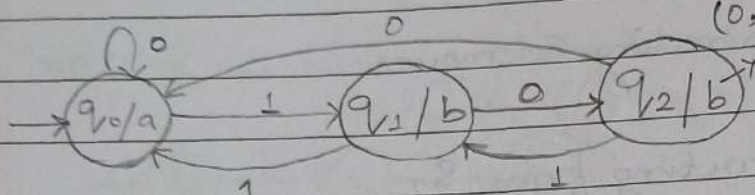


After Eliminating ε :-



→ Moore Machine :- It is a finite-state machine whose current output values are determined ~~whose current output values are determined~~ only by its current state. It is finite automata with output.

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



If m no. of input then there will be m+1 no. of output.

(q_0, q_1, q_2) Q = finite set of states

Σ = input Symbol

δ = Transition function

$Q \times \Sigma$

q_0 = Start State

(a,b) Δ = Output Symbol

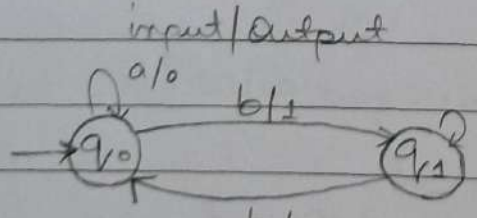
λ = Output function

$\lambda : Q \to \Delta$

Current	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

→ Mealy Machine :- It is a finite-state machine whose output values are determined both by its current ~~value~~ inputs and current state.

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



Q = finite set of states

Σ = input alphabet

δ = Transition function

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

q_0 = initial state

Δ = Output Symbol

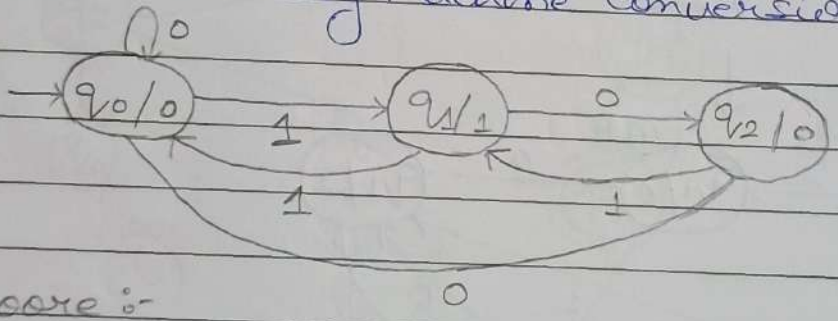
λ = Output function

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

Current State	Next State			
	Input 'a'		Input 'b'	
	S	Output	S	Output
q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

Mealy Machine	Moore Machine
i) Output depends on present State and present input	i) Output only depends on present state not on input
ii) Input string of length 'n' → Output length is also 'n'	ii) Input string length 'n' → Output string 'n+1'
iii) $\lambda : Q \times \Sigma \to \Delta$	iii) $\lambda : Q \to \Delta$
iv) Asynchronous	iv) Syncharonus.

→ Moore to Mealy Machine Conversion :-

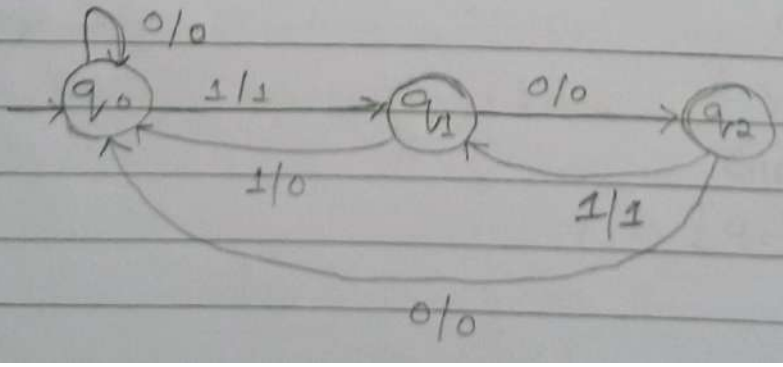


m State, n output
'm' States

Moore :-

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

Mealy :-

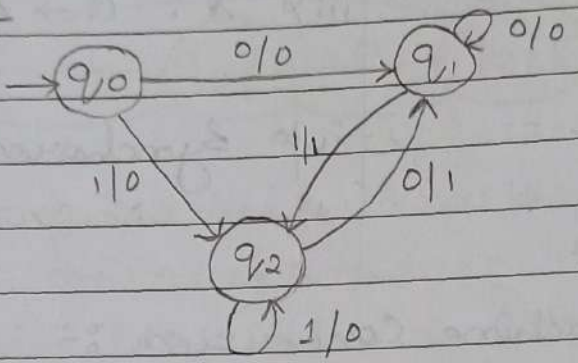


Mealy :-

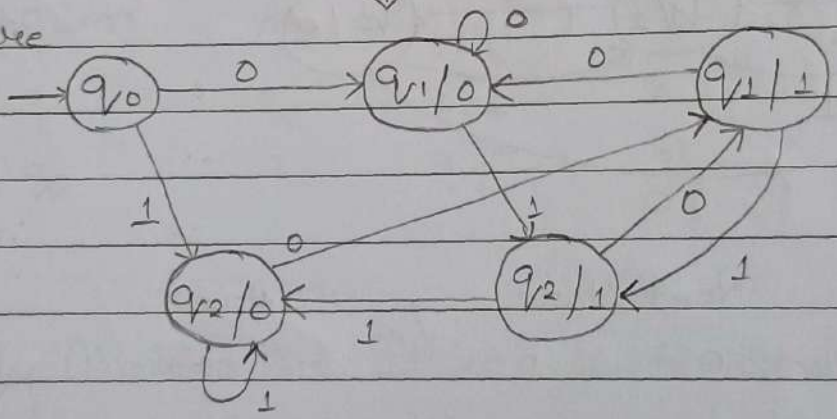
Current	0	1
q_0	$q_0, 0$	$q_1, 1$
q_1	$q_2, 0$	$q_0, 0$
q_2	$q_0, 0$	$q_1, 1$

→ Mealy to Moore Conversion :-

Mealy :-



Moore



m state, n output Mealy

↓

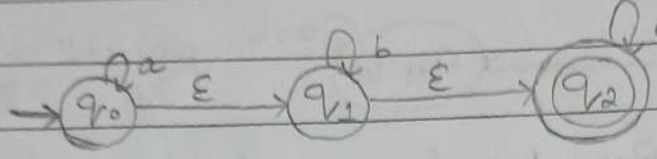
$m \times n$

→ ε - NFA :-

ε matlab bakachoda

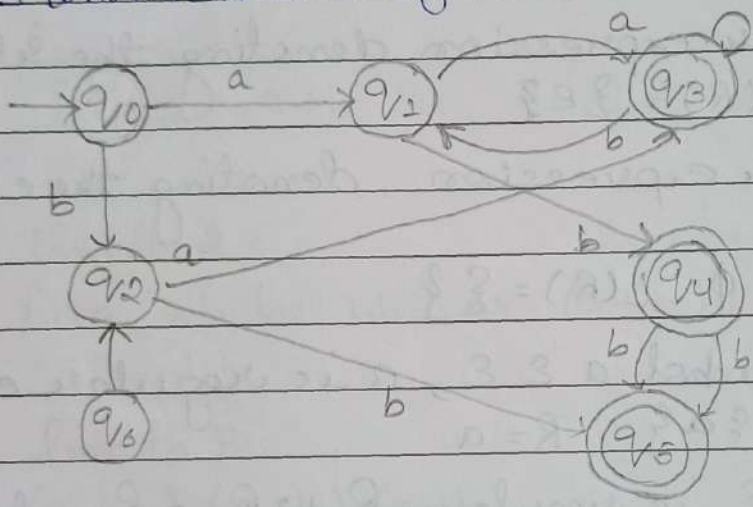
$\delta : Q \times (\Sigma \cup \varepsilon) \to 2^Q$

Q : finite set of states
Σ : input Symbol
q_0 : initial State
F : Set of final states
δ : Transition function



$(a+b+c)^*$

→ Minimization of DFA :-



1) Unreachable
2) classes

↓ Accepting State (final)

↘ Non-Accepting State

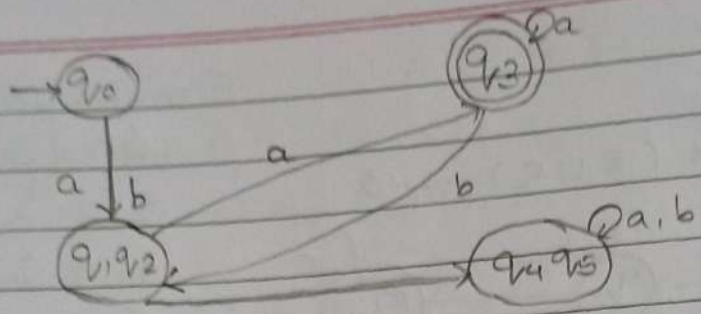
	a	b
→ q_0	q_1	q
q_1	q_3	q
q_2	q_3	q
final q_3	q_3	q
final q_4	q_4	q
final q_5	q_5	q
q_6		

$\pi_0 = \{ q_0 q_1 q_2 \} \{ q_3 q_4 q$
G_{11} G_2

$\pi_1 = \{ q_0 \} \{ q_1 q_2 \}$
G_{11} G_{12}

$\{ q_3 \} \{ q_4 q_5 \}$
G_{21} G_{22}

$\pi_2 = \{ q_0 \} \{ q_1 q_2 \} \{ q_3 \} \{$



→ Regular Expression (Regular language)

- Method to represent a language.
- Let 'R' be a regular expression over alphabet Σ. If R is:

i) 'ε' is regular expression denoting the set {ε}.
R = ε L(R) = {ε}

ii) ϕ is regular expression, denoting the empty set {}
R = ϕ L(R) = {}

iii) For each Symbol a ε Σ, a is regular expression denoting set {a} R = a

iv) Union of RE is regular. $R_1 \cup R_2$ = Regular

v) Concatenation of two RE is also ~~known as~~ regular $R_1 . R_2$ = Regular

vi) Kleene closure of RE is also regular. a∪b = {a, b}

a* = Regular

vii) If R is regular, (R) is also ~~regular~~ regular
$(RE)^*$ = Regular

viii) Nothing else, Repeat 1 to 7 Recursively

$(a+b)^*$ = {λ, a, ab, ba, b, aa, bb}

RE = language

→ Regular Expression (Finite)
eg: $\Sigma(a,b)$

language
↙ ↘
Finite Infinite
↳ FA
↳ RE
↳ Regular

1) No string
$\{\} = \phi$

2) length 0
$\{\varepsilon\} = \varepsilon, \lambda$

3) length 1
$\{a,b\}$ $a+b$

4) length 2
$\{aa, ab, ba, bb\}$ or $(a+b)(a+b)$

5) length 3
$(a+b)(a+b)(a+b)$

6) Atmost 1 (length 0/1)
$\{\varepsilon, a, b\}$ $(\varepsilon + a + b)$

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

8) Not more than 2 b's and a
$\{\varepsilon, a, b, ab, ba, abb, bab, bba \ldots\}$

→ Regular Expression (Infinite)

Consider Alphabet

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

1. All Strings having single 'b'. $a^* b a^*$
2. All Strings having atleast one 'b'. $(a+b)^* b (a+b)^*$
3. All strings having bbbb as Substring. $a\,bbbb\,a$ or $bbbb$
4. All Strings end with 'ab' $(a+b)^* ab$
5. All strings Start with 'a' $ba(a+b)^*$
6. All Strings Starting and ending with a $a(a+b)^* a$
7. All strings Containing 'a' $(a+b)^* a (a+b)^*$
8. All Strings starting & ending with b $a(a+b)^* b + b(a+b)^*$
9. All strings Containing 2b's $a^* b a^* b a^*$

$(a^*)^*$ → Most used Symbol

$(a+)+$

$+ = * - \varepsilon$

$a^+ = a^* - \varepsilon$

→ Pumping Lemma :
Why it is used ?
→ To check whether the language is regular or not.

- Negative Test :-

L → PLT —Pass→ Undecidable
L → PLT —Fail→ Not Regular

If L is an infinite language there exist some positive integer 'n' "Pumping length" such that any string $w \in L$ length greater than equal to 'n' ie $|w| \geq n$ then can be divided into three parts, $w = xyz$ satisfy following Condition :

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

$w = x\ y\ z$

$x\ y^i\ z$ ∴ i = pump Increase

↓ Pump

$x\ yyyy\ z$ ← check whether it belongs to Lang. or not.

eg : $a^n b^{2n}$ $n=2$

$\underset{x}{aa}, \underset{y}{bb}, \underset{z}{bb}$

$\underset{x}{aa}\ \underset{y}{bbbb}\ \underset{z}{bb}$

∴ It is not Regular Language.

→ Closure / Closed properties of Regular language :-

i) Union ($L_1 \cup L_2$)

ii) Concatenation ($L_1 . L_2$)

iii) Closure (*) L^*

iv) Complement $\bar{L} = \Sigma^* - L$

v) Intersection $L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$

vi) Difference $L_1 - L_2 = L_1 \cap \bar{L}_2$

vii) Reversal $(L)^R$

viii) Homomorphism

ix) Reverse Homomorphism

x) Quotient Operation

xi) INIT

xii) Substitution

xiii) Infinite Union

Integer Z

1 + 2 = 3

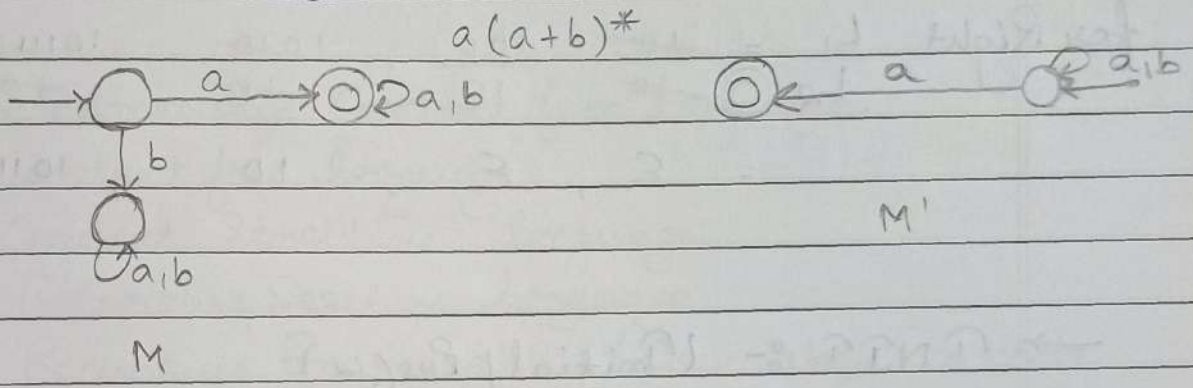
* Pick Set
* Operation bet them
* Check the output is belongs to set, if yes then closed else not closed.

→ Reversal (L^R):

Methods for making reverse language :-

1) Make initial state of M as final state of M'.
2) Final State of M became initial state of M'.
3) Reverse the direction of edges of M to make M'.
4) No change in loop and remove unnecessary states.

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



→ Quotient Operation :

↙ Left Quotient (Cut Prefix) ↘ Right Quotient (Cut Suffix)

Language. Σ 0, 1

~~$\frac{L_1}{L_2} = \{x \mid xy \in L_1$ for some $y \in L_2\}$ Right Quotient (Cutting in Right~~

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

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

Eg: Σ 0,1

$L_1 = \{10, 100, 1010, 101110\}$

$L_2 = \{10\}$

for left $\frac{L_1}{L_2} = \frac{10}{10}, \frac{100}{10}, \frac{1010}{10}, \frac{101110}{10}$

$= \varepsilon, 0, 10, 1110$

for Right $\frac{L_1}{L_2} = \frac{10}{10}, \frac{100}{10}, \frac{1010}{10}, \frac{101110}{10}$

$= \varepsilon, \varepsilon, 10, 1011$

→ INIT :- (Initial / Prefix)

{ Set of all Prefix of $w \in L$ }

Let $L = \{ab, ba\}$ [Prefix = ab = ε, a, ab]

ab = ε, a, ab

ba = ε, b, ba

Init = $\{\varepsilon, a, ab, b, ba\}$

→ Infinite Union :-

$L_1 = \{a^1 b^1\}$

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

⋮

$= a^n b^n \quad n \geq 1$

→ 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	Yes	No	Yes	Yes	No
$\cdot$	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 Sesitive language
RE = Recursive Enumerable / Countably Infinite

2^{Σ^*} Languages

Grammar			Machines
Unrestr-icted	RE	to Accept	TM
	REC	"	HTM
CSG	CSL	"	LBA
CFG	CFL	"	PDA
	DCFL	"	DPDA
RG	RFG	"	FA

Type 0
Type 1
Type 2
Type 3

FA < DPDA < PDA < LBA < TM

NTM ≡ DTM
DPDA < NPDA
DFA ≡ NFA

→ Homomorphism $h(L)$ Substitution function

Regular languages are closed under regular lang.

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

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

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

$h(0) = aa$, $h(1) = bb$

if $L = \{00, 101\}$

$h(L) = ?$

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

Homomorphism image

λ,

→ Inverse Homomorphism :-

let, $h(0) = a$ $h(1) = b$ $h(2) = ab$

$\Sigma \{0,1,2\}$. $\Delta = \{a,b\}$

let, $L = \{abab\}$

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

eg:-2 $L = \{aa, aabb, baab, ababa\}$

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

$h^-(h^-(L))$ is subset of $h^-(L)$.

Decidability & Undecidability table :-

↓

If algo. exist

	REG	DCFL	CFL	CSL	REC
Membership problem $w \in \Sigma^*$	✓	✓	✓	✓	✓
Infinitness problem L = Infinite or finite	✓	✓	✓	✗	✗
Emptiness problem $L = \phi$	✓	✓	✓	✗	✗
Equality problem $L_1 = L_2$	✓	✓	✗	✗	✗
L is Ambiguous?	✓	✓	✗	✗	✗
Completeness $L = \Sigma^*$	✓	✓	✗	✗	✗
$L_1 \cap L_2 = \phi$	✓	✗	✗	✗	✗
if $L_1 \subseteq L_2$ Subset problem	✓	✗	✗	✗	✗

→ Context Free Language & Context Free Grammar

CFG (V, T, P, S)

V: Finite set of Variable (non-terminal)

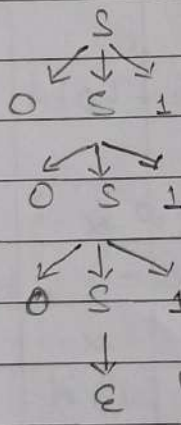
T: Finite set of Terminals ($V \cap T = \phi$)

P: Production Rules (Substitution Rules)

S: Start Variables

$\alpha \rightarrow \beta$ (only one Variable)

$S \rightarrow 0S1 / \varepsilon$



Language → $0^m 1^m$

→ Closure property of CFL

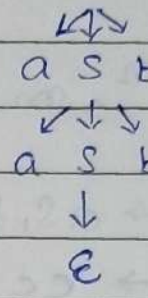
Union ✓	$a^m b^m$	$L_1 \rightarrow G_1$	$S_1 \rightarrow aS_1b / \lambda$
Concatenation ✓	$b^m c^n$	$L_2 \rightarrow G_2$	$S_2 \rightarrow bS_2c / \lambda$
Kleen Closure ✓			$S \rightarrow S_1 / S_2$
Intersection ✗			Union
Complement ✗			

→ Convert CFL → C.F.G

1. $a^n b^n$, $n \geq 1$

$S \rightarrow aSb \mid ab$

eg $S \rightarrow aSb \mid \lambda$



aaabbb
→ $a^n b^n$

2. $a^n b^{n+2}$, $n \geq 0$

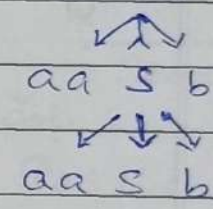
$S \rightarrow aSb \mid bb$

3. $a^{2n} b^n$, $n \geq 0$

$S \rightarrow aaSb \mid \lambda$

$a^{2n} b^n$, $n \geq 1$ $S \rightarrow aaSb$

$S \rightarrow aaSb \mid aab$



aaaaaa bbb

4. $a^{2n+3} b^n$, $n \geq 0$

$S \rightarrow aaSb \mid aaa$

5. $a^m b^n$, $m > n$

$n \geq 0$

$S_1 \rightarrow aS_1b \mid \lambda$

Extra $A \rightarrow A \rightarrow aA \mid a$

Concatenate

$S \rightarrow AS_1$

$a^m b^n$, $m \geq n$

$n \geq 0$

$S_1 \rightarrow aS_1b \mid \lambda$

$A \rightarrow aA \mid \lambda$

6. $\{ w \mid n_a(w) = n_b(w) \}$ n_a = ~~no~~ no. of a

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

7. $ww^R \cup w(a+b)^{w^R}$

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

8. $a^m b^m c^n, m, n \geq 0$

$S_1 \to aS_1 b \mid \lambda$

$C \to cC \mid \lambda$ $\quad S \to S_1 C$

→ Left most & Right most Derivation :-

Generation of grammar from a string

$S \to AB$

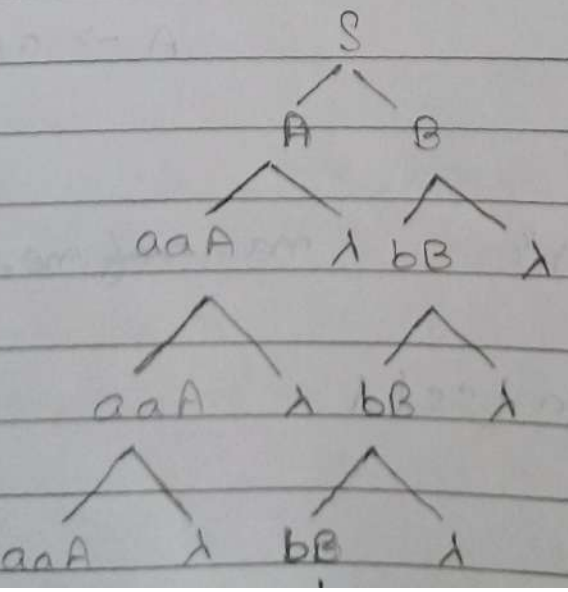
$A \to aaA \mid \lambda$

$B \to bB \mid \lambda$

check whether

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

↳ Yes, the above string belongs to the language.

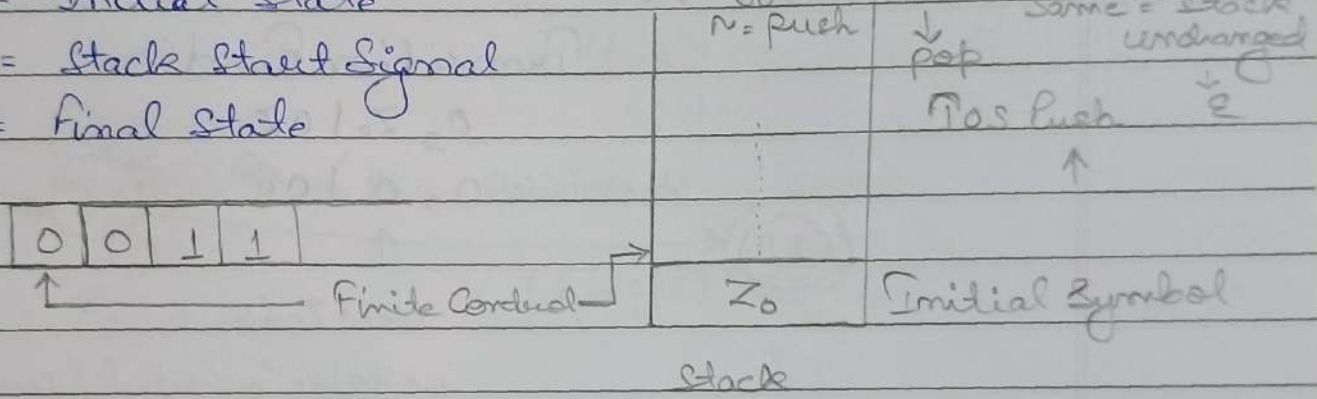


→ Push Down Automata (PDA) :-

FA + 1 Stack = PDA More powerful than regular

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

Q = Finite set of states like FA
Σ = Input Symbol like FA
Γ = Stack Alphabet
δ = Transition function $\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \times \Gamma^*$
q_0 = Initial state
Z_0 = Stack start Signal
F = Final state

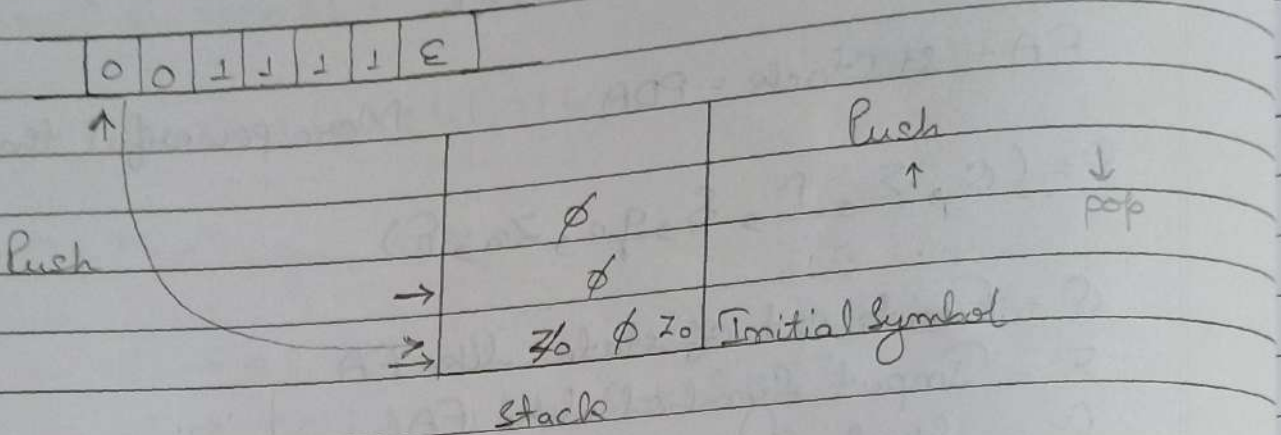


→ PDA Design :-

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

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

String = 011, 001111, 000111111 ....



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

$0,1 \quad 0, Z_0$

Transition function

$0, Z_0 | 0 Z_0$

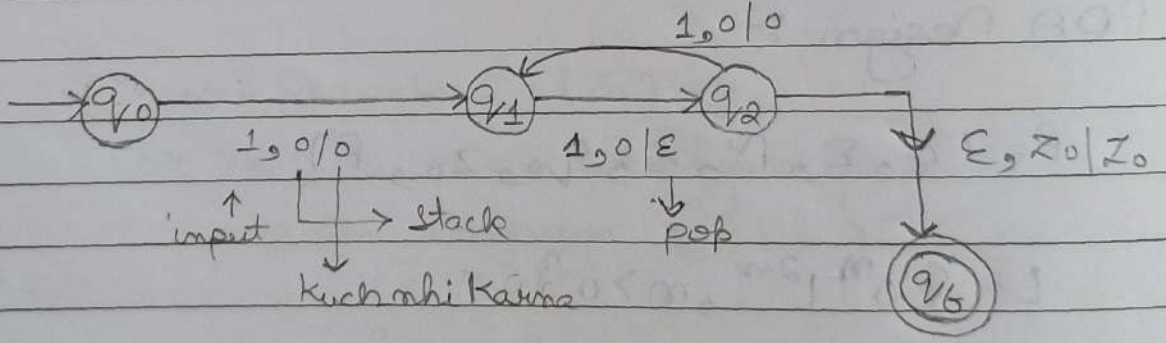
Stackwala → $0, 0 | 00$

↑

New Zero

i.e to be inserted / pushed

in stack



Transition State :-

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

$q_0, 0, 0 \rightarrow q_0 00$

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

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

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

$q_2, \varepsilon, Z_0 \rightarrow q_6, Z_0$

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

String : 01, 10, ~~00~~, ~~11~~, 0011, 1100, 0101, 1010....

0	1	0	0	1	1	ε

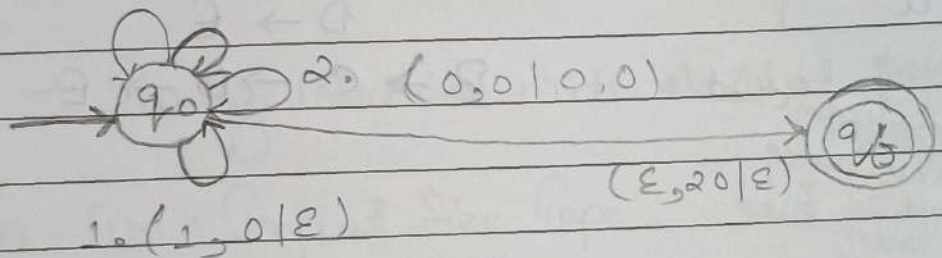
↑ finite control ⟶

	TOS
0	
∅	
Z0	Initial Symbol

Stack

Push ↑ Pop ↓

3. $\{1, 0, \varepsilon\}$



→ Elimination of Null (ε) production :

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

$S \to ABC \mid BC \mid AC \mid C$

$S \to ABC$

$aA \mid a$ $A \to aA \mid \varepsilon$

$aB \mid b$ $B \to bB \mid \varepsilon$,

$C \to C$

Nullable :
$\{A, B$

Nullable Variable :
$\{A, S\}$
↑ Direct ↳ Indirect

* If ε is the part of La
uage. It cannot be use
ed.

Step :
i) Find ε

→ How to remove unit production?

Unit production

$\alpha \to \beta$

$\alpha, \beta \in V$

Step:
① Recognize
② Replace
③ Remove the unutilized Variables

$S \to AB$
$A \to a$
$B \to \varepsilon / b$
$C \to Da$
$D \to Ea$
$E \to a$

Not utilized that's why can be removed.

Unit production :-

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

Unit production

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

Unit production
$S \to B$
$B \to A$
$S \to B \to A$

B A

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

No utilization
∴ Can be removed

B
A bba
ba bb
A
ba bb

→ Turing Machine :- { ① Read & Write ② Left & Right }

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

B ↓ Blank

R/W Head ↑ Finite Control

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

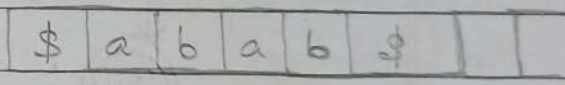
$(q_0, a) \to (q_1, \alpha, \underset{L}{R})$

$2^{Q \times \Gamma \times (L,R)}$

→ Linear Bound Automata (LBA) → Accepts { Context Sensitive Language }

→ Turing Machine with limited size.

LBA = TM + Input Size Tape



Power :

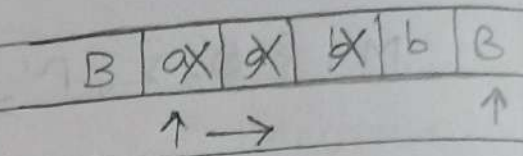
FA < PDA < LBA < TM

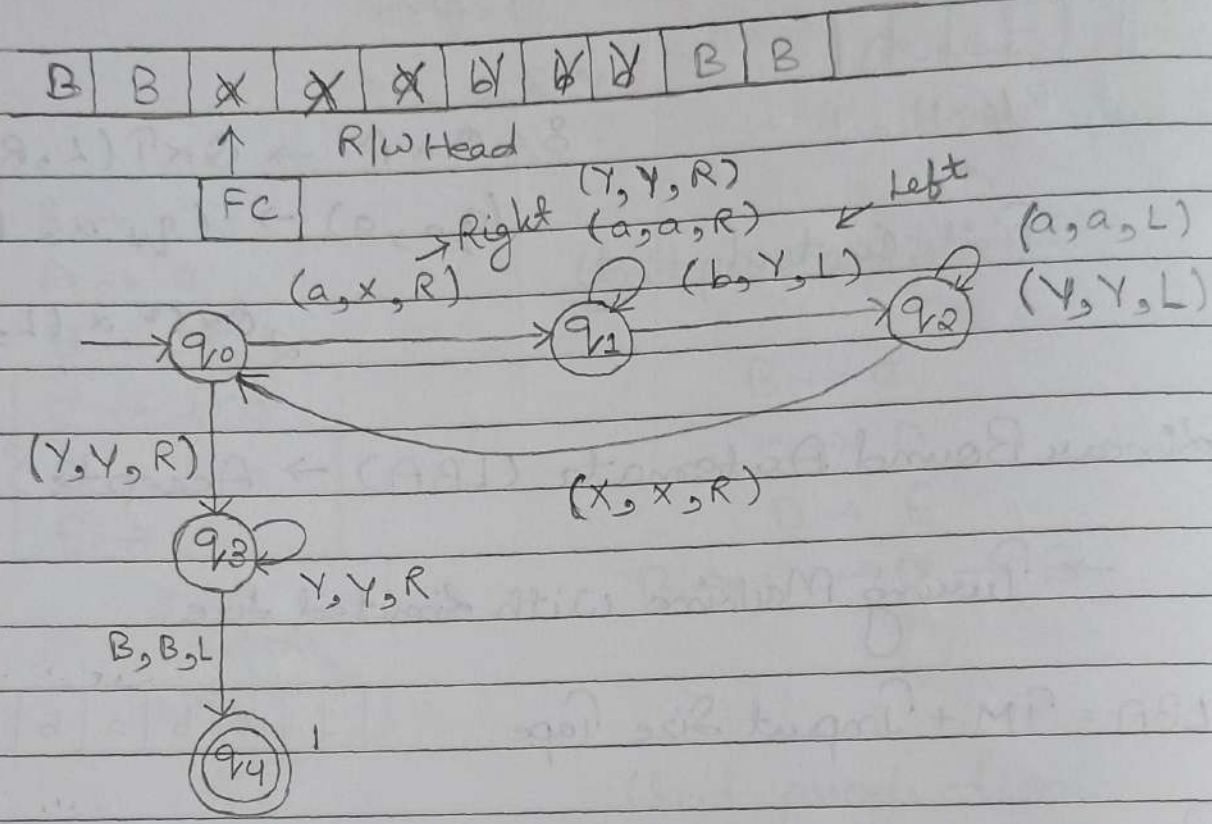
Standard Examples :- (Accepted by LBA not by PDA)

1. $L = \{ a^n b^n c^n : n \geq 1 \}$
2. $L = \{ a^n : n \text{ is prime} \}$
3. $L = \{ a^n : n \text{ is non-prime} \}$
4. $L = \{ a^{n!}, n \geq 0 \}$
5. $L = \{ ww : w \in (a,b)^+ \}$
6. $L = \{ www^R : w \in (a,b)^+ \}$
7. $L = \{ w^n : w \in (a,b)^+ \; n \geq 1 \}$
8. $L = \{ a^n : n = m^2, m \geq 1 \}$

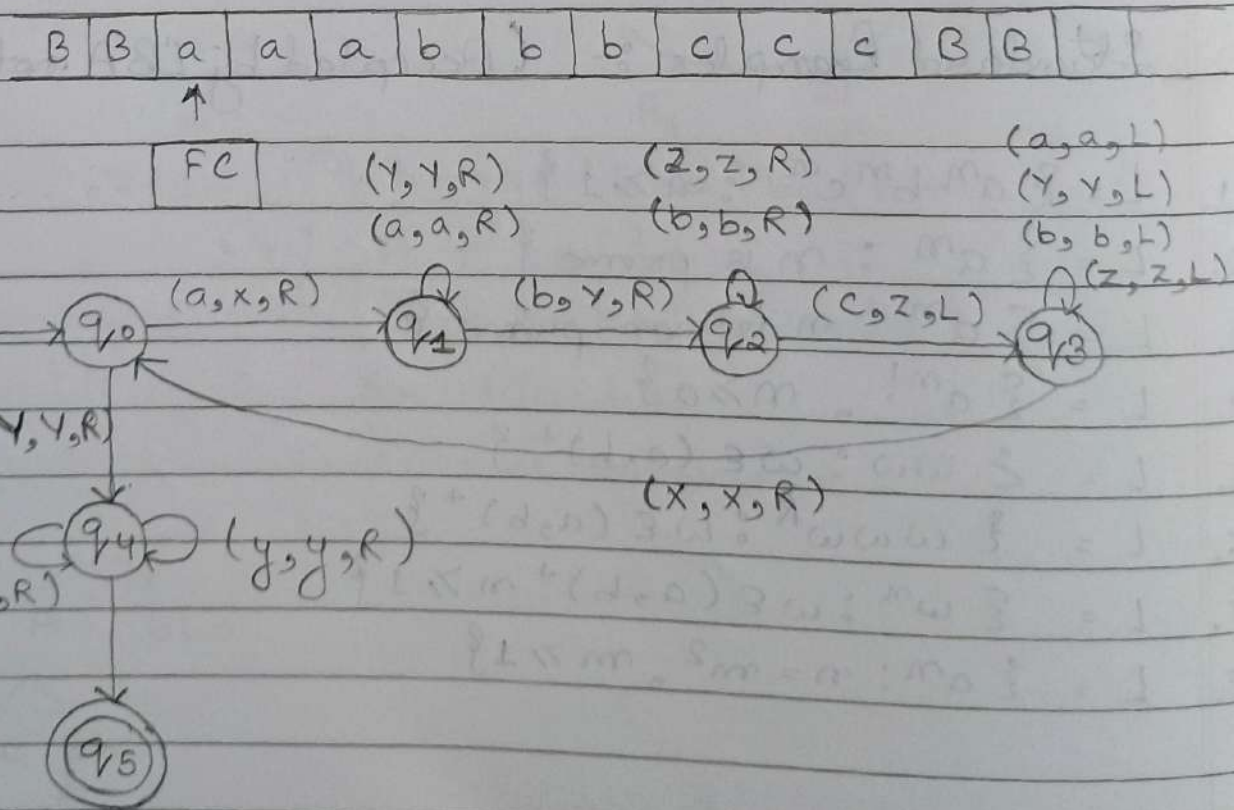
→ Design of Turing Machine :-

for $\{ a^n b^n \mid n \geq 1 \}$





→ Design of Turing Machine :-



→ Recursive Enumerable Language

- L is RE if there is Turing Machine

Three States :
i) Halt & Accept
ii) Halt & Reject
iii) Never Halt

Closed under all except Set difference, Complement

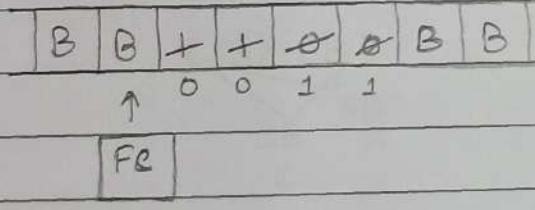
Recursive Language

- L is recursive if there is Halting / Total TM.

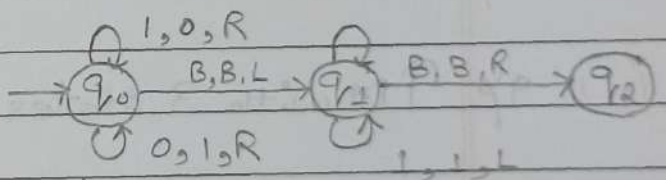
Two States :
i) Halt & Accept
ii) Halt & Reject

Closed under all except Homomorphism and Substitution.

→ Turing Machine for 1's Complement :-



for pointing the FC in initial pt.



States	0	1	B
q_0	1 R q_0	0 R q_0	B L q_1
q_1	0 L q_1	1 L q_1	B R q_2
q_2	-	-	-

Change
↑
∝ B R → changed state
↳ direction

→ Turing Machine with Modification :-

1) TM with Stay option.
2) TM with Semifinite tape.
   → Multistuack TM
3) Offline TM Tape → Input Read
   → Input Read/Write
4) Multidimensional TM
   Left-Right-Up-down
5) NDTM
6) UTM
7) Multitape TM
8) Multihead TM
9) TM with 3 States
10) Jumping TM
11) Non-Erasing TM
12) Always writing TM

FA
1) TM without writing Capacity
LBA
2) TM with Input size tape
PDA
3) TM is with Tape used as stack (NDTM) → PDA
NPDA
FA
4) TM with finite tape

→ CYK Algorithm :-
Check whether a string 'abbb' is a valid ~~mem~~ member of following CFG.

→ Membership Algo :-

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

abbb
1234

* CYK applicable on CNF only
* CNF :- $A \to BC$ OR $A \to a$
* It is Universal (applicable on all CNF)
* Time Complexity $O(n^3)$
* Space Complexity $O(n^2)$

11, 22 → Concatenate

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

(12)(33) → (SB)(B) → SB, BB, A, ϕ

(12)(34) → B B → BB

→ CNF (Chomsky Normal form) GNF (Greibach Normal form)

1. $A \to BC$ or $A \to a$
where $A, B, C \to$ variable, $a \to$ Terminal

1. $A \to a, x$
where $x \in V^*$, $a \to$ terminal

2. No. of steps required to generate a string of length n is "2|n|-1"
2. 'n'

3. Derivation or parse tree obtained 3. Not always from always binary ~~free~~ tree

4. Length of production restricted 4. Not

5. Used by Membership Algo. 5. Used to Convert CFG to PDA.